SALMAN KURTULAN
İTÜ Elektrik-Elektronik Fakültesi Kontrol ve Kumanda Sistemleri Anabilim Dalı

PLC
ile
ENDÜSTRİYEL OTOMASYON

İSTANBUL – 2007

BİRSEN YAYINEVİ

Babam
İbrahim Kurtulan'ın
Anısına

SUNUŞ

Kontrol mühendisliği, belirli bir amaca uygun olarak, bir sistemin en iyi şekilde çalışmasını sağlayan bilgi ve teknolojileri üreten ve uygulayan bir mühendislik dalı olarak tanımlanır. Son yıllarda bilgisayarların, özellikle kişisel bilgisayarların (PC) ve yardımcı yazılımların yaygınlaşması ile, kontrol sistemlerinin analiz ve tasarımı gittikçe daha kolay bir hale gelmiştir. Buna paralel olarak programlanabilir kontrolörlerin (PLC) hızlı gelişimi endüstriyel uygulamalarda da benzer bir değişime neden olmuştur. Endüstri, karmaşık kuramsal yaklaşımları, sistem ve arabirim elemanlarının el verdiği oranda, artık sayısal ortamda yeterli hız ve doğrulukta, standartların öngördüğü bir şekilde gerçekleyebilmektedir. Üniversitelerde her türlü kuramsal bilgiyi edinen ve iş hayatına atılan kontrol, elektrik, elektronik ve makina mühendisleri öğrendikleri bu bilgileri mühendislik ortamına nasıl aktaracakları konusunda çok zorlanmaktadırlar. Elinizde tuttuğunuz bu kitap, geleneksel bir kontrol eğitiminde genellikle söylenmeyen ancak uygulamada kesinlikle bilinmesi gereken her türlü bilgiyi konu edinmektedir. Burada amaç nasıl gerçeklenir sorusuna en yeni teknolojik yöntemleri kullanarak yanıt bulmaktır. Bu kitap, yukarıda belirtilmiş olan bu zor görevi üstlenen ender kitaplardan biri olma özelliğine sahiptir. Ülkemizde bir ilki temsil eden bu eserin, konuya ilgi duyan tüm mühendis ve teknisyenlere son derece yararlı olacağına inanıyorum. Tüm bildiklerini uygulanabilir bilgiler halinde kullanıma sunan Doç. Dr. Salman Kurtulan'ı tebrik ederim.

Mart 2003 İstanbul

Prof. Dr. Atilla Bir
İTÜ Elektrik-Elektronik Fakültesi
Kontrol ve Kumanda Sistemleri
Anabilim Dalı Başkanı

ÖNSÖZ

Birinci basımı Mayıs 1999, endüstriyel kontrol uygulamalarına ilişkin yeni bölümler ile genişletilmiş ikinci basımı 2001, güncelleştirilmiş üçüncü basımı 2003 ve dördüncü basımı 2005 yılında yayımlanan bu kitap, kapsamı genişletilerek ve güncelleştirilerek yeniden sunulmaktadır. Endüstriyel otomasyon eğitimi konusunda sürdürmekte olduğum etkinliklere katılan kişilerin soru, öneri ve yaklaşımlarından edindiğim bilgiler doğrultusunda, kitapta işlenen konular ve içeriklerinde önemli değişiklikler yapılmıştır.

Daha önce yayımlanan baskılardaki kumanda devrelerinin tasarımına ilişkin kuramsal konuların anlatıldığı Bölüm 6 çıkartılmıştır. Kumanda devrelerinin, eşzamansız ve eşzamanlı ardışıl devre yaklaşımıyla tasarımına ilişkin konuların anlatıldığı bu bölüm, daha ayrıntılı başka yaklaşımların da incelendiği "Endüstriyel Kumanda Sistemleri" adlı kitabın kapsamına alınmıştır. İTÜ Türkçe kitap yazımı etkinliği çerçevesinde NOBEL yayın tarafından dağıtımı yapılan bu kitap Eylül 2007'de yayımlanmıştır.

SIMATIC S7-200 PLC ailesi yeni nesil CPU 22x işlemcileri, CPU 21x işlemcilerine göre birçok değişik özellik içermektedir. Bu özellikler göz önünde bulundurularak, daha önceki baskılarda CPU 21x işlemcileri ile gerçeklenen tüm uygulamalar yeni nesil CPU 22x işlemcilerine göre yeniden düzenlenmiştir.

Orta ve büyük ölçekli otomasyon sistemlerinin gerçeklenmesinde kullanılan SIMATIC S7-300/S7-400 ailesi PLC'ler birçok yönden S7-200 PLC ailesinden değişik özelliklere sahiptir. Bu kitapta, SIMATIC S7-300/S7-400 PLC ailesi ve uygulamalarına ilişkin konular da ayrıntılı olarak incelenmiştir.

Endüstriyel otomasyon sistemleri, kumanda, kontrol ve veri iletişimi işlevlerini sağlayan bölümlerden oluşur. Bu kitapta kumanda ve kontrol sistemlerinin PLC ile gerçeklenmesine ilişkin uygulama bilgileri verilmiş, kumanda ve kontrol sistemlerine ilişkin kuralların üretilmesinde yararlanılan kuramsal bilgilere değinilmemiştir. Kumanda sistemlerinin tasarımı için gerekli kuramsal bilgiler ve

yaklaşımlar yukarıda değinilen “Endüstriyel Kumanda Sistemleri” başlıklı kitapta ayrıntılı olarak açıklanmıştır. Kontrol sistemlerinin tasarımı için gerekli olan kuramsal bilgi ve bu bilgiden üretilen kontrol kurallarının PLC ile gerçeklenmesine ilişkin konular, yakın gelecekte yayımlanması düşünülen “Endüstriyel Kontrol” başlıklı kitapta incelenecektir.

Süreç, kontrol birimi ve üst denetim birimleri arasında güvenli veri akışını sağlayan iletişim sistemlerinin kurulumu ve işletimi ile ilgili teknolojik bilgiler otomasyon sisteminin gerçeklenmesi için genellikle yeterlidir. İletişim sistemlerinin geliştirilmesi konusundaki kuramsal bilgiler ve çalışmalar, otomasyon sisteminin tasarımı ve gerçeklenmesini doğrudan etkileyen konular değildir.

İTÜ Elektrik-Elektronik Fakültesi Kontrol ve Kumanda Sistemleri Anabilim Dalı ile Siemens Otomasyon ve Kontrol Sistemleri Eğitim Bölümü arasında sürdürülen bir eğitim etkinliği çerçevesinde, endüstriyel otomasyon sistemlerinin gerçeklenmesinde kullanılan donanımlar ve uygulama alanları konusundaki güncel bilgiler, hem öğrencilerimize hem de endüstriyel kuruluşlarımızdan gelen teknik elemanlara kurs, seminer ve yayın gibi etkinliklerle aktarılmaktadır. Endüstriyel Otomasyon Laboratuvarı ortamında on yıldır sürdürülen bu çalışmaların ürünü birçok yayın yapılmıştır. Yeniden güncelleştirilerek yayımlanan bu kitap da bu ürünlerden biridir.

Bu yapıtın tüm meslektaşlarıma ve öğrenci arkadaşlarıma yararlı olmasını diler, bu konudaki çalışmalarıma destek veren, öneri, eleştiri ve değerlendirmeleri ile çalışmalarımı yönlendiren meslektaşlarım ve öğrenci arkadaşlarıma; Endüstriyel Otomasyon Laboratuvarının kurulması ve geliştirilmesi konusundaki çalışmalara desteğini sürdüren Siemens Otomasyon ve Kontrol Sistemleri Eğitim Bölümü Sorumlusu Sayın Murat S. Oral ve eğitim elemanı Zeynel Erdoğan’a, kitabın basımını yapan Birsen Yayınevi çalışanlarına teşekkür ederim.

Aralık 2007 Salman Kurtulan
İstanbul

İÇİNDEKİLER

BÖLÜM 1 Sayfa

ENDÜSTRİYEL OTOMASYON ---- 1

1.1 Giriş ---- 1

1.2 Programlanabilir Lojik Kontrolör (PLC) ---- 2

1.2.1 Genel Yapı ---- 2

1.2.2 Giriş Birimi ---- 4

1.2.3 Çıkış Birimi ---- 5

1.2.4 Diğer Birimler ---- 6

1.2.5 Programlayıcı Birimi ---- 6

1.3 Çalışma Biçimi ---- 7

1.3.1 Kullanıcı Programının Yürütülmesi ---- 7

1.4 Programlama Dilleri ---- 8

1.4.1 Merdiven Mantığı ile Programlama ---- 10

1.4.2 Fonksiyon Blok Diyagramı ile Programlama ---- 11

1.4.3 Komut (STL, IL) ile Programlama ---- 13

1.5 Program Örnekleri ---- 16

1.5.1 Seri Bağlı Anahtarlama Devresi ---- 16

1.5.2 Seri-Paralel Bağlı Anahtarlama Devresi ---- 17

BÖLÜM 2

SIMATIC S7-200 ---------- 29
2.1 Giriş ---------- 29
2.2 Genel Özellikler ---------- 29
2.2.1 Bellek Alanları ve Erişim Adresleri ---------- 32
2.2.2 Özel Bellek Alanı (SMx) ---------- 34
2.2.3 Program ve Verilerin PLC'ye Aktarılması ve PLC'den Alınması ---------- 35
2.2.4 Enerji Kesilmesi Durumunda Verilerin Korunması ---------- 36
2.3 Dış Devre Bağlantıları ---------- 36
2.4 Programlama ---------- 40
2.5 Mantıksal İşlemler ---------- 47
2.5.1 Temel Mantıksal İşlemler ---------- 49
2.5.2 Kurma-Silme (SET-RESET) İşlemleri ---------- 52
2.5.3 Kurma veya Silme Baskın İki Kararlı İşlemler ---------- 52
2.5.4 Çıkan ve Düşen Kenarı Algılama İşlemleri ---------- 54
2.5.5 Doğrudan Okuma ve Yazma İşlemleri ---------- 55
2.5.6 Mantıksal Yığın İşlemleri ---------- 56
2.6 Zamanlama İşlemleri ---------- 60
2.6.1 Gecikmeli Kapatan Zamanlayıcı (TON) ---------- 61
2.6.2 Gecikmeli Kapatan Kalıcı Zamanlayıcı (TONR) ---------- 62
2.6.3 Gecikmeli Açan Zamanlayıcı (TOF) ---------- 64
2.7 Sayma İşlemleri ---------- 65
2.7.1 İleri Sayıcı ---------- 66
2.7.2 Geri Sayıcı ---------- 67
2.7.3 İleri-Geri Sayıcı ---------- 68
2.8 Karşılaştırma İşlemleri ---------- 69
2.9 IEC 1131-3 Programlama Modu ---------- 72

BÖLÜM 3

S7-200 CPU 22X TEMEL KUMANDA UYGULAMALARI ---------- 75
3.1 Giriş ---------- 75
3.2 Kısa Devre Asenkron Motorun Kumandası ---------- 75
3.3 Kısa Devre Asenkron Motorun İki Yönde Kumandası ---------- 80
3.4 Kısa Devre Asenkron Motora Yıldız-Üçgen Yol Verme ---------- 85
3.5 Rotoru Sargılı Asenkron Motorun Kumandası ---------- 92
3.6 Bir Pnömatik Sistemin Kumandası ---------- 97
3.7 Zamanlayıcı ve Sayıcı Uygulamaları ---------- 102
3.7.1 Uzun Süreli Gecikme İşareti Üretilmesi ---------- 102
3.7.2 Zamanlayıcı ve Karşılaştırma İşlemleri ile Değişik Zamanlama İşaretleri Üretilmesi ---------- 108

BÖLÜM 4

S7-200 CPU 22X PROGRAM DENETİM İŞLEMLERİ ---------- 111
4.1 Giriş ---------- 111
4.2 Atlama İşlemi ---------- 111
4.3 Program Denetim Komutları ---------- 113
4.4 Ardışıl Denetim Komutları ---------- 117
4.5 Tanılama (Diagnostic) İşareti Üretme ---------- 122
4.6 Altprogram Kullanımı ---------- 123

BÖLÜM 5

SAYISAL İŞLEMLER ---------- 129
5.1 Giriş ---------- 129
5.2 Veri Aktarma İşlemleri ---------- 130
5.2.1 8, 16, 32 Bitlik Veri Aktarma İşlemleri ---------- 130
5.2.2 Anında Okuma İşlemi ---------- 131
5.2.3 Anında Aktarma İşlemi ---------- 132
5.2.4 Yer Değiştirme İşlemi ---------- 132
5.2.5 Veri Bloğu Aktarma İşlemi ---------- 133

5.3 İşaretli Tamsayılar Üzerinde İşlemler ---------- 135
5.3.1 16 bitlik işaretli tamsayılar üzerinde işlemler ---------- 135
5.3.2 32 bit İşaretli Tamsayılar Üzerinde İşlemler---------- 140
5.4 Reel Sayılar Üzerinde İşlemler ---------- 143
5.5 Reel Sayılar Üzerinde Diğer İşlemler---------- 146
5.5.1 Karekök Alma İşlemi ---------- 146
5.5.2 Reel Sayılar Üzerinde Diğer İşlemler ---------- 147
5.6 Dönüştürme İşlemleri ---------- 151
5.6.1 Reel Sayılara Dönüştürme ---------- 151
5.6.2 İşaretli 16 bitlik Tamsayıya Dönüştürme ---------- 153
5.7 Veri Tablosu İşlemleri ---------- 156
5.8 Dolaylı Adresleme ---------- 164
5.9 Veriler Üzerinde Mantıksal İşlemler---------- 167
5.10 Veriler Üzerinde Kaydırma ve Döndürme İşlemleri ---------- 169

BÖLÜM 6

GERÇEK ZAMAN İŞLEMLERİ ---------- 175
6.1 Giriş ---------- 175
6.2 Zaman Bilgisinin Okunması ---------- 175
6.3 Uzun Süreli 1MS Artımlı Zaman Bilgisi ---------- 180

BÖLÜM 7

KESMELİ ÇALIŞMA ---------- 183
7.1 Giriş ---------- 183
7.2 Kesmeli Çalışmaya İlişkin Genel Konular ---------- 183
7.3 Olaya Bağlı Kesmeli Çalışma ---------- 185
7.4 Zamana Bağlı Kesmeli Çalışma ---------- 187
7.5. Zamanlayıcı Kaynaklı Kesmeli Çalışma---------- 190
7.6 Hızlı Sayıcı Kaynaklı Kesmeli Çalışma ---------- 192
7.6.1 Hızlı Sayıcının Programlanması---------- 192

7.7 Yüksek Hızlı Çıkış İşareti Üretilmesi---207
7.7.1 Yüksek Frekanslı Kare Dalga Üretilmesi---208
7.7.1.1 Tek Parçalı İşleme İlişkin Bir Darbe Katarı Üretilmesi---209
7.7.1.2 Çok Parçalı İşleme İlişkin Bir Darbe Katarı Üretilmesi---212
7.7.2 Darbe Genişlik Modülasyonlu Yüksek Hızlı İşaret Elde Edilmesi---215

BÖLÜM 8

KONTROL SİSTEMLERİNDE CPU 22X UYGULAMALARI---223
8.1 Giriş---223
8.2 Analog Giriş-Çıkış Birimi---225
8.2.1 Analog Verilere Erişim---226
8.2.2 Analog Veriler Üzerinde İşlemler---230
8.3 Kontrol Uygulamaları---233
8.3.1 İki Konumlu Kontrol---234
8.3.2 PID Kontrol---241
8.3.3 Bir PI Kontrolör Altprogramı Gerçekleme---254
8.3.4 Birinci Mertebeden Alçak Geçiren Filtre Gerçekleme---261
8.4 Sayısal Benzetim---265
8.4.1 Birinci Mertebeden Sistemler---265
8.4.2 İkinci Mertebeden Sistemler---274
8.4.2.1 Az Sönümlü Sistemler---275
8.4.2.2 Kritik Sönümlü Sistemler---286
8.4.3 Birinci Mertebeden İntegral Etkili Sistemler---291

BÖLÜM 9

SIMATIC S7-300/S7-400 PLC AİLESİ VE UYGULAMALARI---297
9.1 Giriş---297
9.2 Program Yapısı ve Çalışma Biçimi---297
9.3 Programlama---301
9.3.1 Temel Mantıksal İşlemler---302

9.3.2 Kurma ve Silme İşlemleri ---- 303
9.3.3 Çıkan ve Düşen Kenarı Algılama İşlemleri ---- 305
9.3.4 Zamanlama İşlemleri ---- 307
9.3.4.1 Darbe Zamanlayıcısı (S_PULSE, SP) ---- 307
9.3.4.2 Uzatılmış Darbe Zamanlayıcısı (S_PEXT, SE) ---- 309
9.3.4.3 Gecikmeli Kapatan Zamanlayıcı (S_ODT, SD) ---- 311
9.3.4.4 Kalıcı Gecikmeli Kapatan Zamanlayıcı (S_ODTS, SS) ---- 312
9.3.4.5 Gecikmeli Açan Zamanlayıcı (S_OFFDT, SF) ---- 314
9.3.5 Sayma İşlemleri ---- 321
9.3.5.1 İleri-Geri Sayıcı ---- 321
9.3.5.2 İleri Sayıcı ---- 322
9.3.5.2 Geri Sayıcı ---- 323
9.3.6 Program Denetim İşlemleri ---- 325
9.3.6.1 Atlama İşlevini Sağlayan Komutlar ---- 325
9.3.6.2 Ana Denetim İşlevini Sağlayan Komutlar ---- 326
9.3.7 Matematiksel İşlemler ---- 330
9.3.7.1 İşaretli Tamsayılar Üzerinde İşlemler ---- 330
9.3.7.2 Reel Sayılar Üzerinde İşlemler ---- 333
9.3.8 Dönüştürme İşlemleri ---- 338
9.3.8.1 BCD Türünde Sayılar ile Tamsayılar Arasında Dönüşüm İşlemleri ---- 338
9.3.8.2 İşaretli Tamsayılar ile Reel Sayılar Arasında Dönüşüm İşlemleri ---- 339
9.3.9 Veri Aktarma İşlemleri ---- 341
9.4 Veri Blokları ve Bellek Alanlarına Erişim ---- 343
9.5 STL ile Programlama ---- 346
9.5.1 Temel Mantıksal İşlemler ---- 349
9.5.2 Atlama İşlemleri ---- 356
9.5.3 Matematiksel İşlemler ---- 358
9.5.4 Dolaylı Adresleme ---- 361
9.6 Analog Verilere Erişim ve Kontrol Uygulamaları ---- 371
9.6.1 Analog Verilere Erişim ---- 374

9.6.2 Kontrol Uygulamaları ---------- 375
9.6.2.1 İki konumlu kontrol ---------- 375
9.6.2.2 Oransal Kontrol ---------- 378
9.6.2.3 PI Kontrol ---------- 380
9.6.2.4 Birinci Mertebeden Alçak Geçiren Filtre Gerçekleme ---------- 383
9.6.2.5 PID Kontrol ---------- 385
9.6.2.6 FB41 PID fonksiyon bloğu ---------- 392
9.6.3 Sayısal Benzetim Uygulamaları ---------- 396
9.6.3.1 İkinci Mertebeden Sistemler ---------- 396
9.6.3.2 Birinci Mertebeden İntegral Etkili Sistemler ---------- 401
9.6.3.3 Kontrol Sisteminin Tasarımında Benzetim Uygulamaları ---------- 404
Kaynakça ---------- 407
Ekler ---------- 409
Dizin ---------- 445

BÖLÜM 1
ENDÜSTRİYEL OTOMASYON

1.1 GİRİŞ

Endüstriyel otomasyon, bir endüstriyel üretim sisteminin istenen ve amaçlanan biçimde çalışması için gerekli olan işlemlerin kendiliğinden yapılmasını sağlayan süreç olarak tanımlanabilir. Endüstriyel otomasyon sistemleri, üretim biriminin amaca uygun çalışmasını düzenlediği gibi, üretim sistemlerinin üst düzeyde yönetim ve planlaması için gerekli olan süreç verilerinin alınması ve aktarılması işlevini de görür. Endüstriyel otomasyon sistemi en genel anlamda, kumanda, kontrol ve veri iletişimi ile ilgili işlevleri sağlayan bölümlerden oluşur. Her bölüm işlev, donanım ve dayandığı bilgi nedeniyle değişik alanlar olarak değerlendirilir ve incelenir. Endüstriyel kumanda sistemleri, üretim birimlerinin çalışma koşullarını mantıksal kurallara göre (devreye girme ve devreden çıkma gibi) düzenleyen ve gerçekleyen sistemlerdir. Bu tür devrelerde mantıksal ilişki, zamanlama ve sayma işlevleri kullanılarak amaca uygun kumanda işaretleri üretilir. Kontrol sistemleri, bir üretim sürecini her türlü bozucu etkiye karşı, istenen değerde çalışmasını sağlamak üzere kurulan sistemlerdir. Kontrol sisteminin temel görevi, herhangi bir nedenle oluşan kontrol edilen büyüklük ile istenen büyüklük arasındaki farkı belirli ölçütler altında en kısa sürede gidermektir. Veri iletişim sistemleri ise birimler arasında gerçek zamanlı bilginin güvenilir ve hızlı akışını sağlayan sistemlerdir. Veri iletişim sistemlerinin sağladığı olanaklardan yararlanılarak, hem etkileşim içinde bulunan birimlerin çalışması düzenlenir hem de SCADA (Supervisory Control and Data Acquisition) gibi özel amaçlı yazılımlar ile gerçek zamanlı süreç izleme, uzaktan kumanda ve kontrol işlemleri gerçeklenebilir. Programlanabilir lojik kontrolör (PLC), günümüz endüstriyel otomasyon sistemlerinin her üç bölümünü de gerçeklemede önemli görevler yüklenilen en önemli aygıt özelliğini taşır.

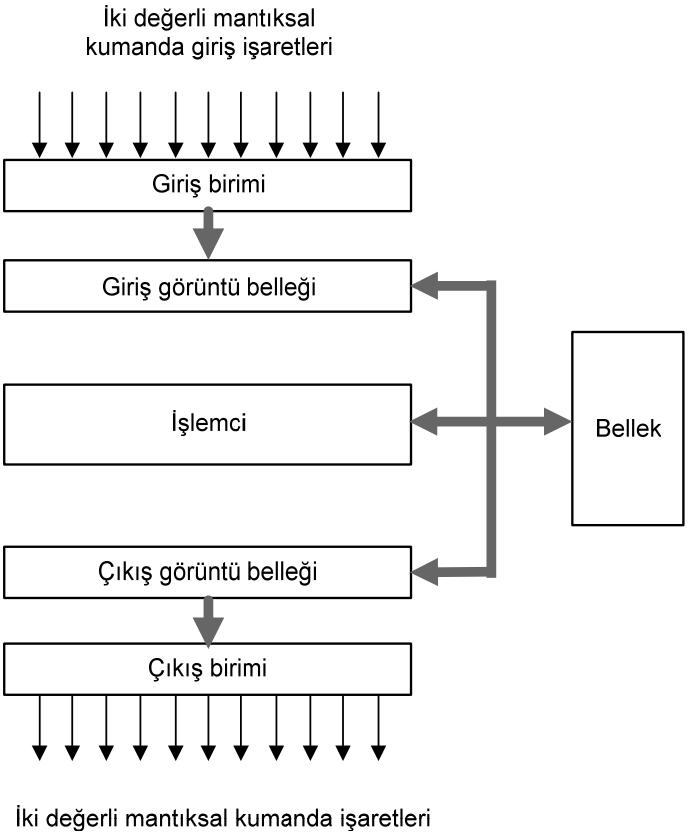
1.2 PROGRAMLANABİLİR LOJİK KONTROLÖR (PLC)

Programlanabilir lojik kontrolör (Programmable Logic Controller, PLC) endüstriyel otomasyon sistemlerinin kumanda ve kontrol devrelerini gerçeklemeye uygun yapıda giriş-çıkış birimleri ve iletişim arabirimleri ile donatılmış, kontrol yapısına uygun bir sistem programı altında çalışan bir özel sayısal işlemci veya endüstriyel bilgisayar olarak tanımlanabilir. İlk ticari PLC, 1969 yılında röleli elektriksel kumanda devrelerinin yerine kullanılmak üzere Modicon firması tarafından geliştirilmiş ve üretilmiştir. Röleli kumanda devreleri yerine kullanılmak üzere geliştirilen bu aygıt yalnız temel mantıksal işlem komutları içerdiğinden, programlanabilir lojik kontrolör (Programmable Logic Controller, PLC) adı ile sunulmuştur. İlk ticari PLC'nin endüstride başarıyla uygulanmasından sonra, Allen-Bradley, General Electric, GEC, Siemens ve Westinghouse gibi firmalar orta maliyette yüksek başarımlı PLC'ler üretmişlerdir. Mitsubishi, Omron ve Toshiba gibi firmaların ucuz maliyette yüksek performanslı PLC'ler geliştirmelerinden sonra, bu aygıtlar endüstriyel otomasyon devrelerinde yaygın olarak kullanılmaya başlanmıştır.

Günümüzde üretilen PLC'lerin giriş-çıkış sayısı, program belleği, işlem yeteneği gibi özellikleri başlangıçtaki durumu ile kıyaslanamayacak bir düzeye ulaşmıştır. Örneğin, geniş ölçekli olarak tanımlanan PLC'lerde giriş-çıkış sayısı binleri, program belleği "megabyte" boyutunu ve işlem yeteneği bir genel amaçlı kişisel bilgisayar düzeyine ulaşmıştır. Bu durum , daha karmaşık kumanda ve kontrol işlevlerinin PLC ile kolayca gerçeklenmesine olanak sağlar. Bu aygıtın geribeslemeli kontrol devrelerinde de kullanılmaya başlanması, alışılagelmiş PLC adının tartışılmasına neden olmuştur. Birçok üretici firma, bu kontrolörlerin hem mantıksal temelli kumanda devrelerinde hem de geribeslemeli kontrol sistemlerinde kullanılmaları nedeniyle, PLC yerine, programlanabilir kontrolör adını kullanmayı daha uygun bulmuş ve kişisel bilgisayarlardan ayırmak amacıyla, kısaca PCs olarak tanımlamıştır. Bazı üretici firmalar ise ilk sunulduğu andaki "Programmable Logic Controller" kelimelerinin baş harflerinden oluşan PLC adını kullanmayı sürdürmektedirler. Ülkemizde de PLC adı daha çok kullanılmaktadır.

1.2.1 GENEL YAPI

Bir PLC, en genel anlamda işlevsel üç temel birimden oluşur: Giriş birimi, işlemci birimi ve çıkış birimi. Giriş birimi çeşitli elektriksel özelliklerdeki kumanda ya da geribesleme işaretlerinin PLC'de işlenecek sayısal büyüklüklere dönüştürülmesini, çıkış birimi ise PLC'de işlenen verilere ilişkin sayısal değerlerin kontrol edilen sisteme uygun elektriksel işaretlere dönüştürülmesini sağlayan birimdir. İşlemci birimi, PLC belleğindeki sistem programına göre çalışmayı düzenleyen ve kullanıcı programını yürüten en önemli birimdir. Kumanda amaçlı kullanılan bir PLC'ye ilişkin genel yapı Şekil 1.1'deki gibi gösterilebilir.



Şekil 1.1 PLC genel yapısı

İşlemci (mikrobilgisayar veya mikrokontrolör), PLC sistem programı ve kullanıcı programını yürüten, PLC'nin çalışmasını düzenleyen birimdir.

Bellek, işletim sistemi programının bulunduğu sistem program belleği, kullanıcı programının bulunduğu program belleği ve veri belleği gibi bölümlerden oluşur. Sistem belleği ve PLC'ye ilişkin değiştirilmeyen veriler için salt okunur kalıcı bellek; program belleği ve veri belleği için rasgele erişimli bellek kullanılır. Veri belleği giriş-çıkış işaret durumları, sayıcı ve zamanlayıcı içerikleri, özel amaçlı kaydedici içerikleri, analog işaretlere ilişkin sayısal değerlerin tutulduğu çeşitli bölümlerden oluşur.

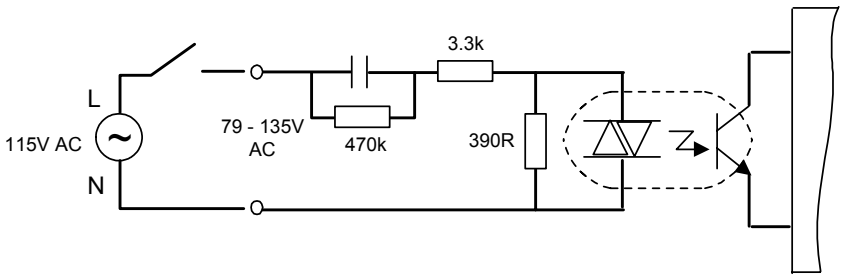
Giriş görüntü belleği, programın yürütülmesi sürecinde, giriş birimindeki iki değerli işaretlerin mantıksal durumlarının (0-1) saklandığı bellek alanı; çıkış görüntü belleği ise kullanıcı programının yürütülmesi sürecinde, çıkış noktalarına ilişkin hesaplanan mantıksal değerlerin saklandığı bellek

alanıdır. Siemens tarafından üretilen PLC'lerde giriş görüntü belleği için I0.0, I0.1, I0.2 gibi, çıkış görüntü belleği için ise Q0.0, Q0.1, Q0.2 gibi adresler kullanılır. PLC'lerde programda hesaplanan ara değerlerin saklandığı "marker", "flag", "internal output", "auxiliary relay" gibi isimler verilen bellek alanları da kullanılır. Bu adres alanları için F, M, V gibi harflerle başlayan F0.0, F0.1,..; M0.0, M0.1,…; V0.0, V0.1 gibi adresler kullanılır. Bu tür verilere genellikle 1 bit, 8 bit, 16 bit ya da 32 bit olarak erişilebilir.

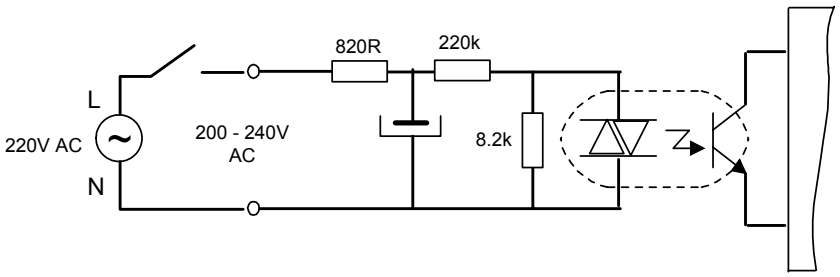
1.2.2 GİRİŞ BİRİMİ

Kumanda edilen sistemle ilgili algılama elemanlarından gelen elektriksel işaretleri mantıksal gerilim seviyelerine dönüştüren birimdir. Kumanda edilen sisteme ilişkin basınç, seviye, sıcaklık algılayıcıları, kumanda düğmeleri ve yaklaşım anahtarları gibi elemanlardan gelen iki değerli işaretler (0 veya 1) giriş birimi üzerinden alınır. Gerilim seviyesi 24V DC, 100V-120V AC veya 200V-240V AC değerlerinde olabilir. Örneğin Şekil 1.2'de 24V doğru gerilimle, Şekil 1.3'te 115 V alternatif gerilimle ve Şekil 1.4'te 220V alternatif gerilimle uyarılan giriş birimine ilişkin devreler verilmiştir. PLC giriş birimi devresine gelen bir işaretin mantıksal 1 kabul edilebildiği bir alt sınır ve mantıksal 0 kabul edilebildiği bir üst sınır değeri vardır. Giriş bilgisinin doğru olarak alınabilmesi için işaret gerilim seviyesinin bu sınır değerleri aşılmamalıdır. Örneğin Şekil 1.4'te verilen giriş birimine uygulanan bir işaretin mantıksal 1 olarak değerlendirilmesi için genliği en az 150V, mantıksal 0 olarak değerlendirilmesi için en çok 40V olmalıdır.

Şekil 1.2 24V DC giriş birimi



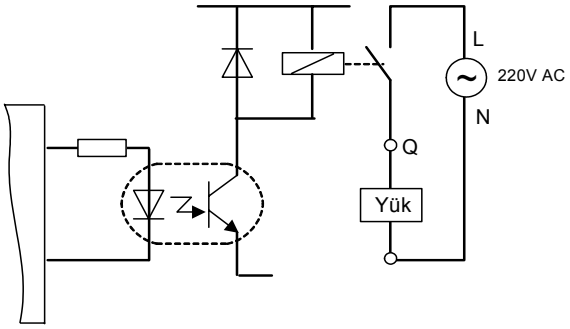
Şekil 1.3 115V AC giriş birimi



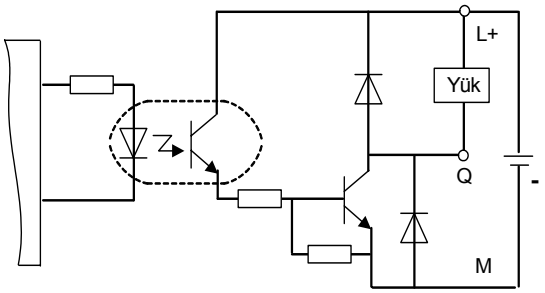
Şekil 1.4 220V AC giriş birimi

1.2.3 ÇIKIŞ BİRİMİ

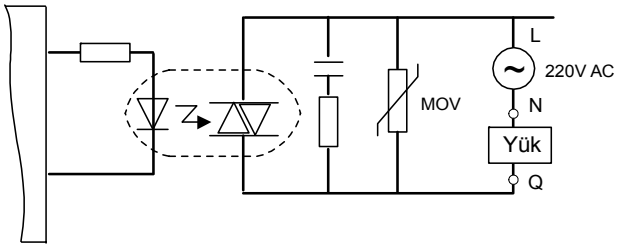
PLC işlemcisinde hesaplanan ve çıkış görüntü belleğine yazılan mantıksal işaretleri, kumanda edilen kontaktör veya röle gibi kumanda elemanlarını sürmeye uygun elektriksel işaretlere dönüştüren birimdir. Çıkış birimi röle, tiryak ya da transistörlü devrelerden oluşur. Çok sık devreye girmeyen ve elektriksel yalıtım gerektiren durumlarda röle çıkışlı (Şekil 1.5), yüksek hızlı açma-kapama gerektiren durumlarda, doğru akımda transistörlü (Şekil 1.6), alternatif akımda tiryak çıkışlı (Şekil 1.7) devreler kullanılır. PLC üzerindeki çıkış noktalarından çekilen akım değerleri (yüklenme akımı) kontak çıkışlı devreler için 1A ile 8A, triyak ve transistörlü devreler için 0.1A ile 2A arasında değişir.



Şekil 1.5 Röle çıkışlı birim



Şekil 1.6 Transistör çıkışlı birim



Şekil 1.7 Triyak çıkışlı birim

1.2.4 DİĞER BİRİMLER

Tüm PLC'lerde işlemcinin ve çevre birimlerinin çalışması için bir besleme güç kaynağı ve çevre birimleri ile iletişimi sağlayan en az bir iletişim arabirimi bulunur. PLC'yi programlamak için bu arabirimden yararlanılır. Bunun dışında, bazı PLC türlerinde programı yedeklemek ve başka bir PLC ye aktarmak için kalıcı bellek birimi, giriş-çıkış sayısını arttırmak için genişleme birimi, enerji kesilmeleri durumunda PLC'yi besleyen yedek güç kaynağı gibi birimler; çok küçük türleri dışında çoğu PLC'de yüksek hızlı sayma ve kesmeli çalışma gerektiren, analog giriş-çıkış olanakları sağlayan birimler de bulunur.

1.2.5 PROGRAMLAYICI BİRİMİ

Yazılan bir programın işletilmek üzere PLC program belleğine yüklenmesi bir programlayıcı birimi ile sağlanır. Programlayıcı birimi mikroişlemci tabanlı özel bir el aygıtı olabileceği gibi genel amaçlı kişisel bir bilgisayara yüklenmiş bir yazılım da olabilir. Bu birim; programın yazılması, PLC'ye aktarılması ve çalışma anında giriş/çıkış, sayıcı, zamanlayıcı ve veri belleğindeki çeşitli kaydedicilerin

durumlarının gözlenmesi ya da değiştirilmesi gibi olanakları da sağlar. Günümüzde genel amaçlı kişisel bilgisayarlara yüklenen özel yazılımların kullanımı yaygındır. Bu yazılımlar özellikle kumanda devreleri ile ilgili kişilerin kolayca kullanabilecekleri ve kolayca uyum sağlayabilecekleri özel programlarıdır.

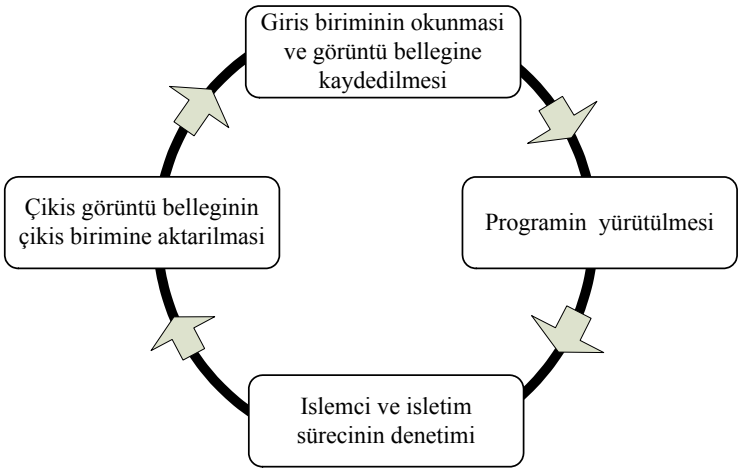
1.3 ÇALIŞMA BİÇİMİ

PLC'leri genel amaçlı bir mikroişlemcili sistemden ayıran en önemli özelliklerden biri de kullanıcı programının özel bir sistem programı ile denetlenmesidir. Bütün PLC sistemlerinde birbirine benzeyen sistem programları bulunur. Bu programlar üretim aşamasında kalıcı bir bellek alanına yüklenir. Genel olarak sistem programı

- Kullanıcı programını yürütür,
- Kesmeli çalışma ve iletişim olaylarını düzenler,
- Sistem çalışma durumlarını denetler.

1.3.1 KULLANICI PROGRAMININ YÜRÜTÜLMESİ

Bir PLC'de programın yürütülmesi, Şekil 1.8'de gösterildiği gibi sürekli döngü içinde belirli işlem evreleri yerine getirilerek yapılır. Kesmeli çalışma, analog biriminden veri okuma veya analog birime veri yazma, giriş biriminden anında okuma, çıkış birimine anında yazma veya iletişim gibi işlemlerin yapılmadığı durumlarda bu işlemler sürekli tekrarlanır.



Şekil 1.8 PLC işlem evreleri

PLC çalışma (RUN) durumuna getirildiğinde sırayla aşağıdaki işlemler gerçeklenir:

1. Giriş birimindeki işaretlerin durumu giriş görüntü belleğine kaydedilir. Bu değerler bir sonraki tarama çevrimine kadar değişmez.

2. Program belleğindeki komutlar adım adım sırayla işlenir. Programda kullanılan giriş değerleri giriş görüntü belleğine yazılan değerlerdir ve bir program çevrimi süresince değişmez. Hesaplanan çıkış değerleri çıkış görüntü belleğine kaydedilir.

3. İşlemcinin ve yapılan işlemlerin geçerliliği ve uygunluğu denetlenir; sorun yoksa çıkış görüntü belleğine kaydedilen değerler çıkış birimine aktarılır. Çıkış birimine aktarma işlemi tamamlandıktan sonra tekrar birinci adıma dönülür. Çıkış birimine aktarılan değerler bir sonraki çevrime kadar değişmez.

Bir PLC'de giriş ve çıkış işaret durumlarının her tarama çevriminin başında güncelleştirilmesi olağan çalışma biçimidir. Ancak, bazı özel komutlar kullanılarak girişlerin işaret durumları komutun işlendiği anda da okunabilir. Benzer şekilde çıkışa ilişkin bir değer hesaplandıktan sonra anında çıkış birimine gönderilebilir. Bu komutlar ivedi (anında) giriş-çıkış komutları (immediate I/O instructions) olarak adlandırılır. Bu komutlar kullanıldığında giriş görüntü belleğindeki değerler yerine komutun yürütüldüğü andaki giriş değerleri işlenir. Aynı şekilde çıkış adreslerine ilişkin hesaplanan bir değer, ivedi işlem komutunun yürütülmesi tamamlandığı anda çıkış birimine aktarılır.

PLC tarama çevriminden daha hızlı değişen işaretleri algılamak ve değerlendirmek için kullanılan kesmeli çalışma biçiminde de giriş işaretlerine doğrudan erişilir. Bu çalışma biçiminde hesaplanan değerlerin anında çıkış birimine aktarılması ise yine ivedi işlem komutları kullanılarak sağlanır. Kesmeli çalışmada kesme işareti geldiği anda olağan programın yürütülmesine ara verilir ve kesme olayı ile bağlantılı kesme altprogramı yürütülmeye başlanır. Kesme altprogramındaki komutların işlenmesi tamamlandığında tekrar ana programa dönülür. Bir kesme altprogramında çıkışa ilişkin hesaplanan bir değerin çıkış birimine aktarılması yine olağan program çevrimi tamamlandığında (çıkış görüntü belleğinin çıkış birimine aktarılması aşamasında) olur.

Simatic S7-200 PLC'lerinde, programın yürütülmesi evresinde, kısa süreli işaretlerin algılanmasını ve bir sonraki taramada bu işaretin değerlendirilmesini sağlama olanağı da vardır. Bu özellik MicroWIN'deki (System block>Pulse Catch bits) sistem ile ilgili ayarlar yapılırken belirlenir.

1.4 PROGRAMLAMA DİLLERİ

Bir kumanda ya da kontrol sisteminin istenen biçimde çalışması için bulunan sözel ya da matematiksel kuralların, PLC program belleğine aktarılmak üzere uygun bir dil kullanılarak düzenlenmesi

programlama olarak tanımlanır. Günümüzde kontaktör, röle ve yarıiletken mantıksal kapı elemanları ile gerçeklenen kumanda devrelerini tasarlayan kişilerin kolayca uyum sağlayabilecekleri ve kullanabilecekleri PLC programlama dilleri geliştirilmiştir. IEC 61131-3 standartlarına göre 5 türlü programlama dili tanımlanmıştır. Bunlar; komut, yapısal metin, merdiven mantığı, fonksiyon blok ve ardışıl fonksiyon gösterimi ile programlama biçimleridir.

Komut (STL:statement list, IL:instruction list) ile programlama, belirli bir komut kümesi kullanılarak gerçeklenir. Tüm PLC programlama yazılımlarında komut kümesi ile programlama olanağı genellikle bulunur. En düşük düzeyde bir programlama biçimi olduğundan komut ile işlemci etkileşimini bilmeyi gerektirir. Kullanımı diğer dillere göre daha karmaşıktır. Ancak, daha az program belleği kullanma olanağı sağlayabilir.

Yapısal metin dili (ST:structured text), Pascal diline benzeyen yüksek düzeyli bir programlama dilidir. Karmaşık ifadelerin daha kolay yazılmasına olanak sağlar. Ancak her PLC için geçerli bir programlama dili değildir. Daha çok orta ve büyük ölçekli bazı PLC türlerinde bulunur.

Merdiven mantığı (LAD:Ladder Diagram) ile programlama, açık kapalı kontak simgeleri ile mantıksal ilişkinin gösterildiği bir programlama dilidir. Elektriksel kumanda devrelerine benzediğinden en yaygın kullanılan programlama dilidir. Çok küçük ölçekli bazı PLC türlerinde geçerli değildir.

Fonksiyon blok diyagramı (FBD: Function block diagrams) ile programlama, standart mantıksal kapı simgeleri ile mantıksal devrenin oluşturulduğu grafiksel bir programlama biçimidir. Daha çok yarıiletken elektronik kapı elemanları ile kumanda devresi tasarımına yatkın kişilerin tercih ettiği bir dildir. Her PLC türü için geçerli değildir.

Ardışıl fonksiyon gösterimi (SFC:Sequential Function Charts) ile programlama, ardışıl özellikteki kumanda devrelerinin doğrudan gerçeklenmesine yönelik tasarlanmış bir dildir. Fransa'da yaygın olarak kullanılan "Grafcet" dilinden esinlenerek geliştirilmiştir. Her PLC türü için geçerli değildir.

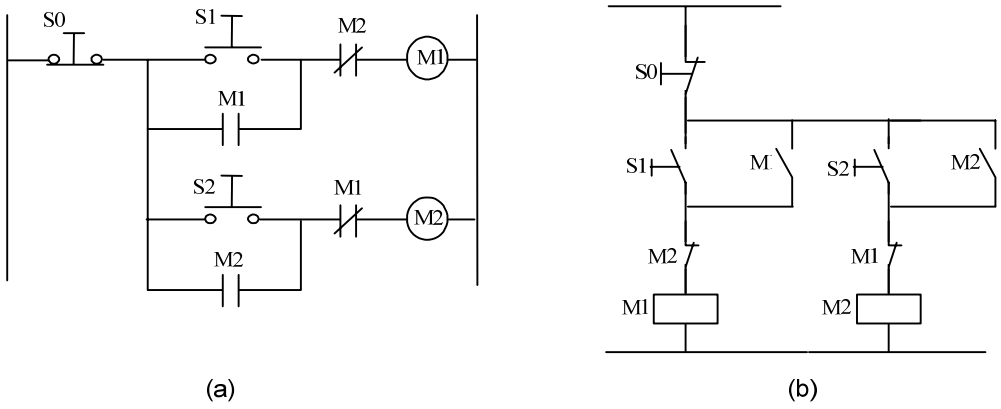
Bu kitapta ele alacağımız Siemens ürünü S7-200 sınıfı CPU 22x işlemcili PLC türlerinin programlanmasında kullanılan "MicroWIN" yazılımında STL, LAD ve FBD dilleri kullanılarak programlama yapılabilir. Siemens ürünü S7-300 ailesi ve S7-400 sınıfı PLC ailesinin programlanmasında kullanılan "Simatic Manager" yazılımında ise tüm programla dillerini kullanma olanağı vardır.

Tüm PLC türlerinde programla dilleri benzer özellikler taşır. LAD ve FBD dili aynı biçimde kullanılır ve aynı işlevi görür. STL dilinde bazı değişik komutlar ile karşılaşılabilir, ancak komutların yürütülüş biçimi, bazı PLC türleri dışında aynıdır. Siemens S7-200 sınıfı PLC türleri ile birçok üreticiye ait PLC türleri arasında büyük benzerlikler vardır.

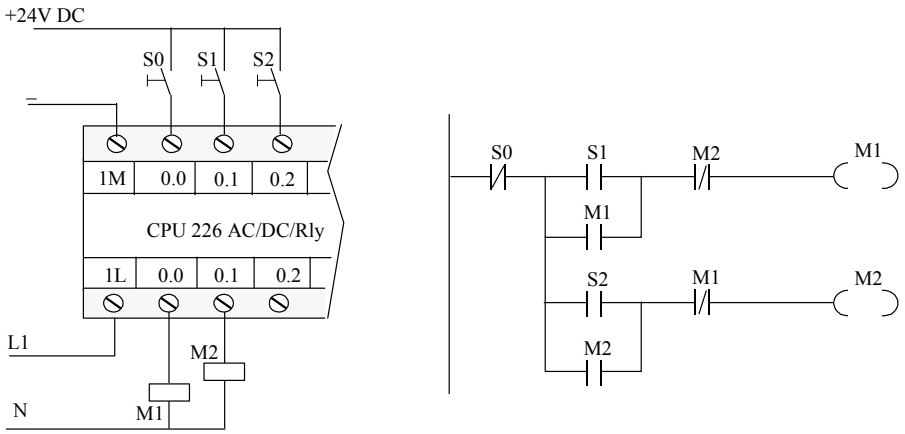
Siemens S7-300/400 sınıfı PLC türlerinde STL dilinde programlama ve komutlar yürütülüş biçimi farklıdır. Bu PLC'ye ilişkin bölümde bu komutların özellikleri ve yürütülmesi ayrıntılı olarak açıklanacaktır.

1.4.1 MERDİVEN MANTIĞI İLE PROGRAMLAMA

Merdiven mantığı ile programlama geleneksel kumanda devrelerinin standart çizimlerinden esinlenerek geliştirilmiş bir grafiksel programlama şeklidir. Geleneksel röleli kumanda devreleri, normalde açık (NA), normalde kapalı (NK) kontaklar, kontaktör veya yardımcı röle bobinlerinden oluşan devrelerdir. Bu devreler tasarlanırken belirli çizim kuralları ve simgeler kullanılır. Bu konuda çeşitli standartlar bulunur. En yaygın kullanılan standartlar Avrupa'da IEC (International Electrotechnical Commission) ve Kuzey Amerika'da ANSI (American National Standards Institute) standartlarıdır. Yurdumuzda daha yaygın bir kullanıma sahip IEC standartlarına göre çizilen kumanda devrelerinde, röle ve kontaktör bobinleri yatay, akım yolu çizgileri ise dikey çizilir. ANSI standartlarına göre gösterilen devrelerde ise röle ve kontaktör bobinleri dikey, akım yolu çizgileri ise yatay çizilir. Her iki çizim biçiminde kontak gösterimleri de değişiktir. Şekil 1.9'da ANSI (a) ve IEC (b) standartlarına göre çizilen iki kumanda devresi gösterilmiştir. Buradaki her iki devre aynı işlevi sağlar. Devrenin çalışması akım mantığına göre veya bu devrelere ilişkin mantık fonksiyonları yazılarak incelenebilir. Aynı devrenin PLC ile gerçeklenmesi durumunda, şekilde gösterilen giriş-çıkış bağlantılarına göre merdiven mantığı programı Şekil 1.10'daki gibi gösterilebilir. Buradaki tüm çizimlerin, simgelerdeki bazı değişiklikler dışında, benzerlik taşıdığı görülür. Bu benzerlik nedeniyle merdiven mantığı programı yaygın bir kullanım alanı bulur. Böylece herhangi bir kumanda işlevini kontaklı (anahtarlamalı) devre olarak tasarlayan kişi, bu devreyi PLC'ye kolayca uyarlayabilir.



Şekil 1.9 ANSI (a) ve IEC (b) standartlarına göre çizilmiş aynı işlevi sağlayan eşdeğer kumanda devreleri

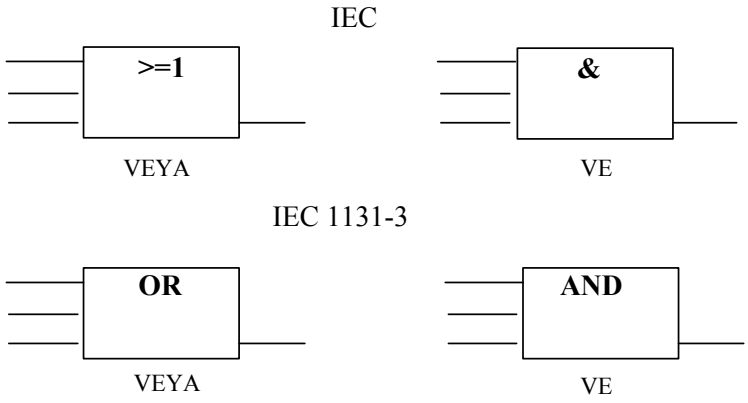


Şekil 1.10 Merdiven mantığı gösterimi

Bu devreler aynı işlevi sağlar: S1 düğmesine basıldığında M2 devrede değilse M1 kontaktörü, S2 düğmesine basıldığında M1 kontaktörü devrede değilse M2 kontaktörü devreye girer; S0 düğmesine basıldığında kontaktörler devreden çıkar.

1.4.2 FONKSİYON BLOK DİYAGRAMI İLE PROGRAMLAMA

Fonksiyon blok diyagramı (FBD) ile programlama, mantıksal "VE", "VEYA" kapıları art arda bağlanarak oluşturulur. Bunun için Şekil 1.11'de gösterilen IEC ve IEC 1131-3 standartlarına göre tanımlanan simgeler kullanılır.

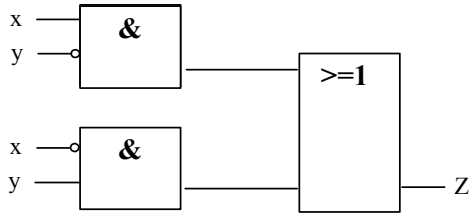


Şekil 1.11 Mantıksal VE, VEYA kapı simgeleri

Kumanda devrelerine ilişkin bir fonksiyon "VE" ,"VEYA" ve "DEĞİL" temel mantıksal işlemleri ile gerçeklenebilir. Tüm mantık fonksiyonları Boole cebri işlemleri ile çarpımların toplamı veya toplamların çarpımı biçiminde ifade edilebilir. Bu biçime getirilen mantık fonksiyonları kolayca programlanabilir. "DEĞİL" mantıksal işlemi kapı girişlerine konan küçük daire simgesi ile belirtilir. Örneğin,

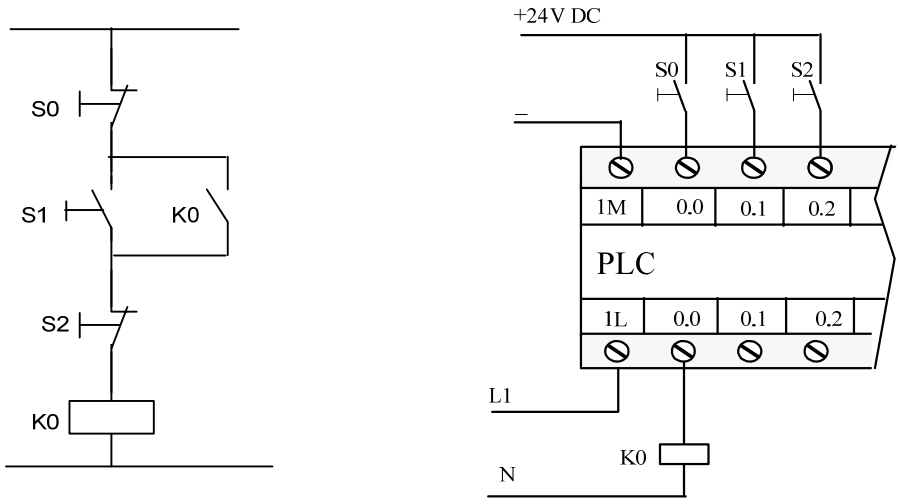
$$Z= x\ y' + x'\ y$$

biçiminde bir mantık fonksiyonuna ilişkin FBD programı Şekil 1.12'deki gibi verilebilir.

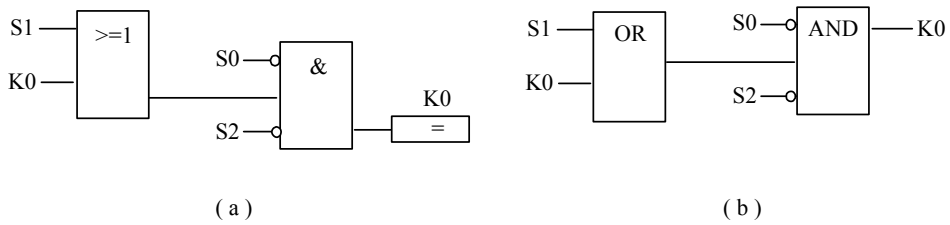


Şekil 1.12 Z= x y' + x' y mantık fonksiyonuna ilişkin FBD programı

Şekil 1.13'te gösterilen elektrikli kumanda devresine ilişkin FBD programı ise Şekil 1.14'teki gibi verilebilir.



Şekil 1.13 Kumanda devresi ve PLC bağlantı şeması



Şekil 1.14 FBD program örneği

1.4.3 KOMUT (STL, IL) İLE PROGRAMLAMA

Komut ile programlama, LAD ve FBD ile yapılan programlara göre daha karmaşıktır. Daha sonra SIEMENS S7-200 microPLC için ayrıntılı olarak incelenecek olan bu komutlar, diğer PLC modelleri için de benzer özellikler taşır. Çizelge 1.1'de çeşitli PLC modellerine ilişkin temel işlem komut örnekleri gösterilmiştir.

Çizelge 1.1

Komut	Hitachi	Omron	Mitsubishi	Texas Inst.	Simatic S7
LOAD	LD	LD	LD	STR	LD
AND	AND	AND	AND	AND	A
OR	OR	OR	OR	OR	O
NOT	NOT	NOT	I	NOT	NOT
LOAD NOT	LDI	LD NOT	LDI	STR NOT	LDN
AND NOT	ANI	AND NOT	ANI	AND NOT	AN
OR NOT	ORI	OR NOT	ORI	OR NOT	ON
AND BLOCK	ANB	AN LD	ANB	AND STR	ALD
OR BLOCK	ORB	OR LD	ORB	OR STR	OLD
OUT	OUT	OUT	OUT	OUT	=

Çizelge 1.1’de verilen komutlar ile her türlü mantık fonksiyonu programlanabilir. Aynı satırdaki komutlar aynı işlevi sağlar ve benzer biçimde programlanır.

Çizelgede ilk satırdaki komutlar (LOAD, LD, STR) mantıksal işlemi başlatma komutlarıdır. Bu komut ile işlenen adresin içeriği yığın yapısındaki bir belleğe yüklenir. Yığının üstündeki eski değer bir basamak aşağı itilir.

İkinci satırdaki komutlar (AND, A) VE mantıksal işlem komutlarıdır. Bu komutlar ile yığının üstündeki değer ile işlenen adresin içeriğine VE işlemi uygulanır, sonuç yığın belleğinin üstüne yazılır.

Üçüncü satırdaki komutlar (OR, O) mantıksal VEYA işlemi komutlarıdır. Bu komutlar ile yığının üstündeki değer ile işlenen adresin içeriğine VEYA işlemi uygulanır, sonuç yığının üstüne yazılır.

Dördüncü satırdaki komutlar (NOT, I) DEĞİL mantıksal işlem komutlarıdır. Bu komutlar ile yığının üstündeki verinin tümleyeni alınır ve sonuç yine yığının üstüne yazılır.

Beş, altı ve yedinci satırdaki komutlar (LOAD NOT, AND NOT, OR NOT, ...) ilk üç satırdaki komutlarla aynı işleve sahiptir. Ancak, mantıksal işlemlerde yığının üstündeki değerin tümleyeni kullanılır.

Sekiz ve dokuzuncu satırdaki komutlar (AND BLOCK, OR BLOCK, ...) yığının üstündeki değer ile bir basamak altındaki değere VE , VEYA mantıksal işlemlerini uygulayan ve sonucu yığının üstüne yazan komutlardır. İşlem sonunda yığın belleğinin diğer konumundaki veriler bir basamak yukarı kaydırılır.

Son satırdaki komut (OUT, =) yığının üstündeki değeri işlenen adrese yazar.

Komut ile programlamaya ilişkin bir örnek Çizelge 1.2'de gösterilmiştir. Bu örnekte, Texas Instruments PLC'sine ilişkin bir program parçası için her komutun yürütülmesinden sonra, yığında oluşan değerler belirlenmiş ve programa karşı düşen mantık fonksiyonu verilmiştir. Bu çizelgede program komutları 1. sütunda, yığının üstü 2. sütunda ve yığının 2. seviyesi 3. sütunda gösterilmiştir.

Çizelge 1.2

Program	Yığının tepesi	Yığının 2. seviyesi
STR X9	X9	-
OR C1	X9+C1	-
STR NOT X10	X10′	X9+C1
AND X11	X10′• X11	X9+C1
AND STR	(X10′•X11)•(X9+C1)	-
OUT C1	(X10′•X11)•(X9+C1)	-
STR X12	X12	(X10′•X11)•(X9+C1)
OR C2	X12+C2	(X10′•X11)•(X9+C1)
STR C3	C3	X12+C2
AND STR	C3•(X12+C2)	-
OUT C4	C3•(X12+C2)	-

Çizelge 1.2'deki komutların sırayla yürütülmesi ile

C1=(X10′•X11)•(X9+C1)

C4=C3•(X12+C2)

mantık fonksiyonları programlanmış olur.

Komut ile programlamaya ilişkin başka bir örnek Çizelge 1.3'te verilmiştir. Burada Simatic S7 200 PLC’si için yazılan bir program örneği gösterilmiş ve incelenmiştir.

Çizelge 1.3

Program	Yığın tepesi	Yığın 2. seviye	Yığın 3. seviye	Yığın 4. seviye
LD X0	X0	-	-	--
A X1	X0• X1	-	-	-
LD X2	X2	X0•X1	-	-
LD X3	X3	X2	X0•X1	-
LD X4	X4	X3	X2	X0•X1
OLD	X3+X4	X2	X0•X1	-
ALD	(X3 + X4)• X2	X0•X1	-	-
= Y1	(X3 + X4) • X2	X0 • X1	-	-
A X5	X5 • (X3 + X4)• X2	X0• X1	-	-
OLD	X5• (X3 + X4)•X2+X0• X1	-	-	-
= Y2	X5 • (X3 + X4)•X2+X0 •X1	-	-	-

Çizelge 1.3'teki program parçası ile

Y1=(X3 + X4) • X2 ve Y2= X5 • (X3 + X4)•X2+X0 •X1

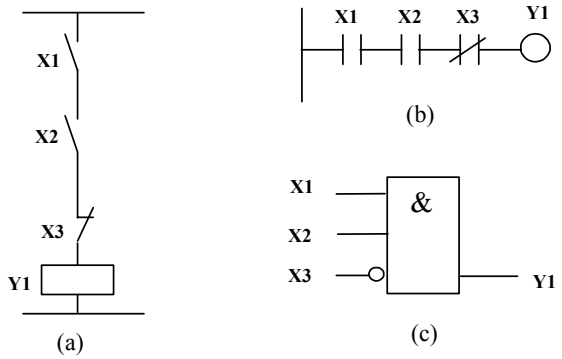
mantık fonksiyonları gerçeklenir.

Yukarıdaki örneklerden görüleceği üzere, çeşitli PLC'lere ilişkin işlem komutları ve bu komutların yürütülüş biçimleri birbirine benzer. Bu nedenle, herhangi bir PLC için yazılmış olan bir program başka bir PLC'ye kolayca uyarlanabilir.

1.5 PROGRAM ÖRNEKLERİ

1.5.1 SERİ BAĞLI ANAHTARLAMA DEVRESİ

Bir anahtarlama devresinde, seri bağlı kontaklar VE mantıksal işlemine karşı düşer. Şekil 1.15'te bu tür bir devre (a) ve bu devreye ilişkin LAD (b) ve FBD (c) program biçimleri verilmiştir.



Şekil 1.15 Seri bağlı anahtarlama devresine ilişkin LAD ve FBD programları

Şekil 1.15'te verilen seri bağlı anahtarlama devresine ilişkin mantık fonksiyonu

$$Y1 = X1 \bullet X2 \bullet X3'$$

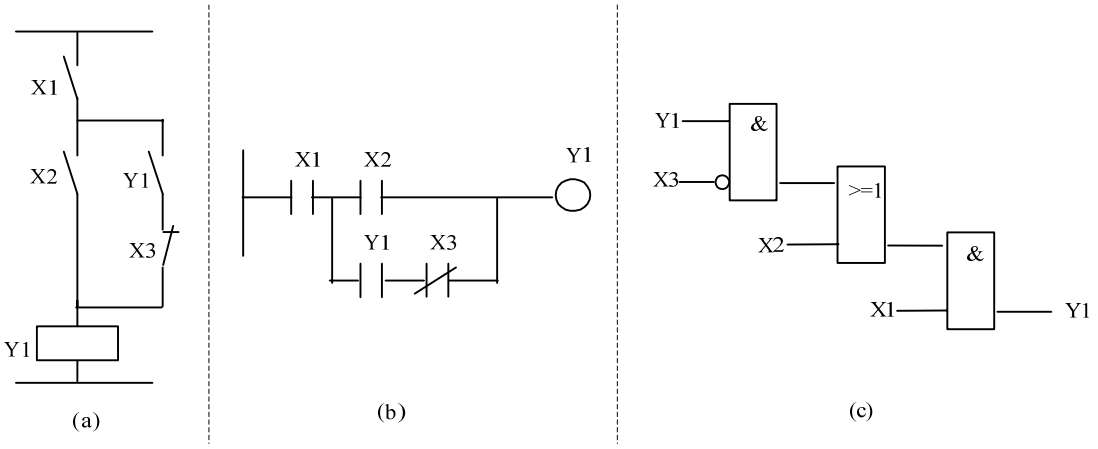
biçiminde ifade edilir. Bu fonksiyona ilişkin komut programı, çeşitli üretici firmalarına ilişkin PLC türleri için Çizelge 1.4'teki gibi yazılır.

Çizelge 1.4

Simatic S7-200	Hitachi	Omron	Mitsubishi	Texas Inst.
LD X1	LD X1	LD X1	LD X1	STR X1
A X2	AND X2	AND X2	AND X2	AND X2
AN X3	ANI X3	ANDNOT X3	ANI X3	ANDNOT X3
= Y1	OUT Y1	OUT Y1	OUT Y1	OUT Y1

1.5.2 SERİ-PARALEL BAĞLI ANAHTARLAMA DEVRESİ

Bir anahtarlamalı kumanda devresinde, seri ve paralel bağlı kontaklar VE ve VEYA mantıksal işlemleri ile gerçeklenir. Şekil 1.16'da bu tür bir kontaklı kumanda devresi (a) ve bu devreye ilişkin LAD (b) ve FBD (c) programları verilmiştir.



Şekil 1.16 Seri-paralel bağlı anahtarlama devresine ilişkin LAD ve FBD programları

Şekil 1.16'daki devrelere ilişkin mantık fonksiyonu

$$Y1= X1\bullet(X2+Y1\bullet X3')$$

biçiminde yazılır. Bu fonksiyon komut ile çeşitli üretici firmalarının PLC modelleri için Çizelge 1.5'teki gibi programlanır.

Çizelge 1.5

Simatic S7-200	Hitachi	Omron	Mitsubishi	Texas Inst.
LD X1	LD X1	LD X1	LD X1	STR X1
LD X2	LD X2	LD X2	LD X2	STR X2
LD Y1	LD Y1	LD Y1	LD Y1	STR Y1
AN X3	ANI X3	ANDNOT X3	ANI X3	ANDNOT X3
OLD	ORB	ORLD	ORB	ORSTR
ALD	ANB	ANLD	ANB	ANDSTR
= Y1	OUT Y1	OUT Y1	OUT Y1	OUT Y1

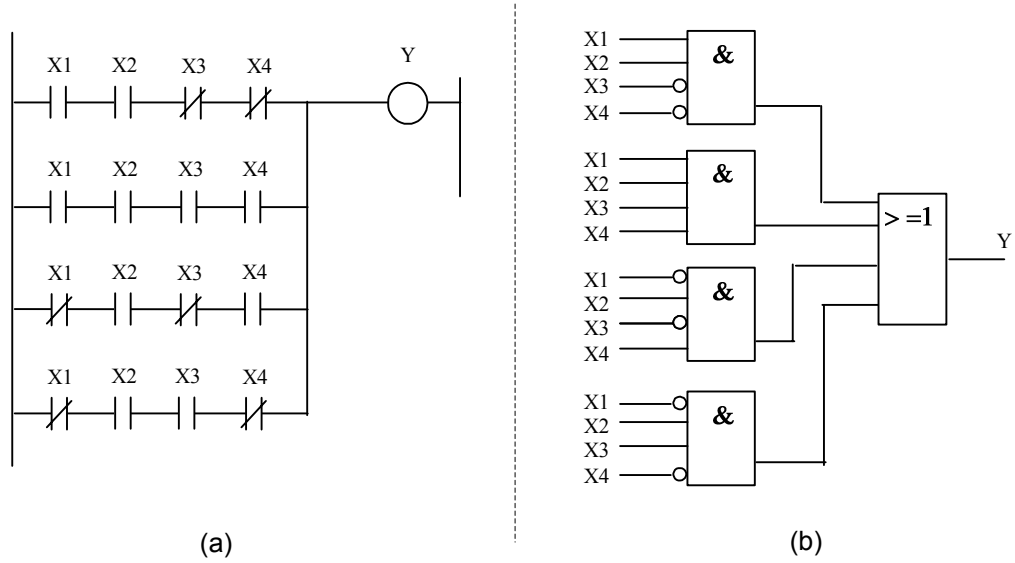
Herhangi bir anahtarlama devresine ilişkin mantık fonksiyonu, çarpımların toplamı veya toplamların çarpımı biçiminde düzenlenirse seri-paralel anahtarlama devreleri elde edilir ve bu biçime getirilen devreler kolayca programlanabilir. Örneğin,

$$Y=X1\bullet [X2\bullet(X3'\bullet X4'+X3\bullet X4)]+X1'\bullet [X2\bullet(X3'\bullet X4+X3\bullet X4')]$$

biçiminde bir mantık fonksiyonu verilmiş olsun. Bu fonksiyon mantıksal Boole cebri işlemleri ile

Y=X1•X2•X3′•X4′+X1•X2•X3•X4+X1′•X2•X3′•X4+X1′•X2•X3•X4′

biçimine getirilebilir. Bu biçimde verilen bir mantık fonksiyonuna ilişkin merdiven mantığı (LAD) ve fonksiyon blok diyagramı (FBD) Şekil 1.17'deki gibi gösterilebilir.



Şekil 1.17 Y=X1•X2•X3′•X4′+X1•X2•X3•X4+X1′•X2•X3′•X4+X1′•X2•X3•X4′ mantık fonksiyonuna ilişkin LAD (a) ve FBD (b) programları

Şekil 1.17'de verilen merdiven mantığı (LAD) ve fonksiyon blok diyagramı (FBD) programları, komut ile Çizelge 1.6'daki gibi yazılabilir.

Çizelge 1.6

Simatic S7-200	Hitachi	Omron	Mitsubishi	Texas Inst.
LD X1	LD X1	LD X1	LD X1	STR X1
A X2	AND X2	AND X2	AND X2	AND X2
AN X3	ANI X3	ANDNOT X3	ANI X3	ANDNOT X3
AN X4	ANI X4	ANDNOT X4	ANI X4	ANDNOT X4
LD X1	LD X1	LD X1	LD X1	STR X1
A X2	AND X2	AND X2	AND X2	AND X2
A X3	AND X3	AND X3	AND X3	AND X3
A X4	AND X4	AND X4	AND X4	AND X4
OLD	ORB	ORLD	ORB	ORSTR
LDN X1	LDI X1	LDNOT X1	LDI X1	STRNOT X1
A X2	AND X2	AND X2	AND X2	AND X2
AN X3	ANI X3	ANDNOT X3	ANI X3	ANDNOT X3
A X4	AND X4	AND X4	AND X4	AND X4
OLD	ORB	ORLD	ORB	ORSTR
LDN X1	LDI X1	LDNOT X1	LDI X1	STRNOT X1
A X2	AND X2	AND X2	AND X2	AND X2
A X3	AND X3	AND X3	AND X3	AND X3
AN X4	ANI X4	ANDNOT X4	ANI X4	ANDNOT X4
OLD	ORB	ORLD	ORB	ORSTR
= Y	OUT Y	OUT Y	OUT Y	OUT Y

Bir mantık fonksiyonu, toplamların çarpımı biçiminde ifade edilerek de programlanabilir. Örneğin,

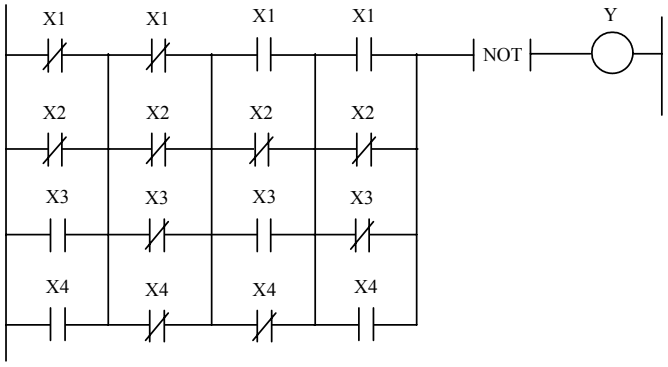
$$Y=X1\bullet X2\bullet X3'\bullet X4'+X1\bullet X2\bullet X3\bullet X4+X1'\bullet X2\bullet X3'\bullet X4+X1'\bullet X2\bullet X3\bullet X4'$$

çarpımların toplamı biçimindeki bir mantık fonksiyonu göz önüne alınsın. Bu fonksiyon, Boole Cebri işlemleri ile toplamların çarpımı biçimine getirilebilir. Yukarıdaki ifade De Morgan Teoremine göre

$$Y=[(X1\bullet X2\bullet X3'\bullet X4')'\bullet(X1\bullet X2\bullet X3\bullet X4)'\bullet(X1'\bullet X2\bullet X3'\bullet X4)'\bullet(X1'\bullet X2\bullet X3\bullet X4')']'$$

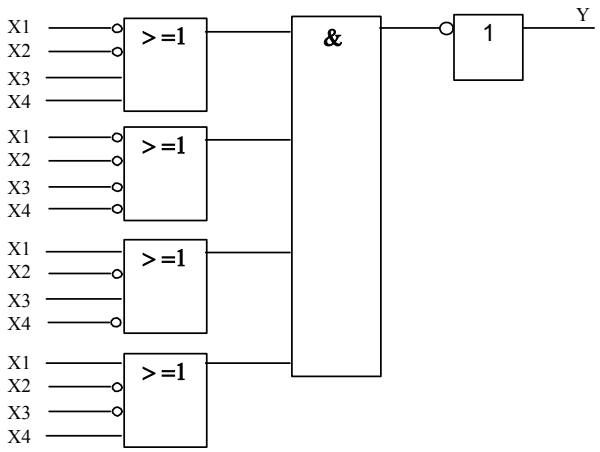
$$=[(X1'+X2'+X3+X4)\bullet(X1'+X2'+X3'+X4')\bullet(X1+X2'+X3+X4')\bullet(X1+X2'+X3'+X4)]'$$

toplamların çarpımı biçiminde yazılabilir. Bu fonksiyona ilişkin merdiven mantığı programı 1.18'de verilmiştir. Bu programda, Y çıkışı önündeki "NOT" simgesi yazılı kutu tümleme (DEĞİL) işlemini gösterir.



Şekil 1.18 Toplamların çarpımı biçimindeki mantıksal fonksiyonlara ilişkin merdiven mantığı programı

Bu mantık fonksiyonuna ilişkin fonksiyon blok diyagramı (FBD) Şekil 1.19'daki gibi verilebilir.



Şekil 1.19 Toplamların çarpımı biçimindeki mantıksal fonksiyonlara ilişkin FBD programı

Bu örneğe ilişkin

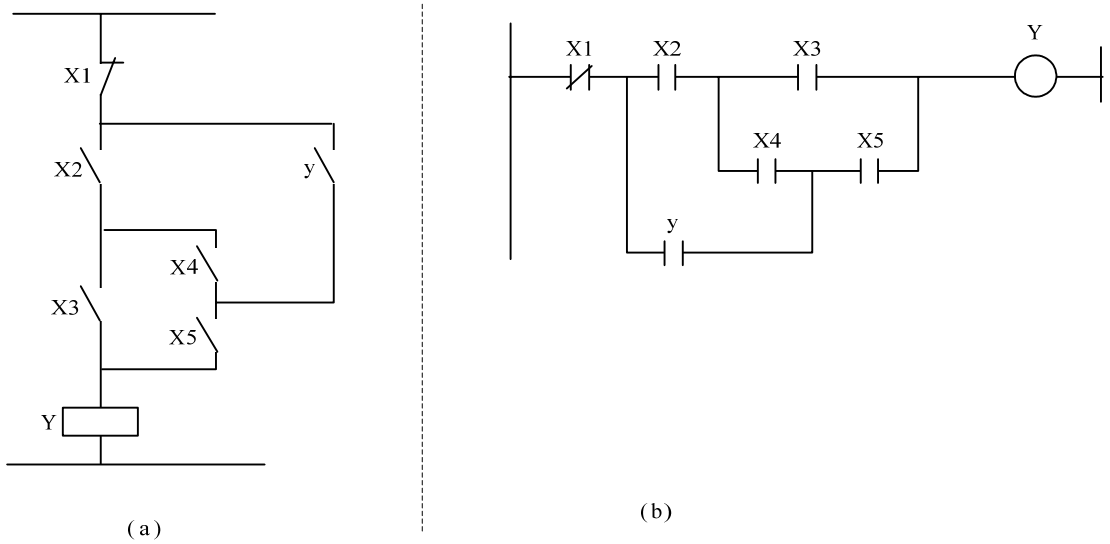
Y=[(X1′+X2′+X3+X4)•(X1′+X2′+X3′+X4′)•(X1+X2′+X3+X4′)•(X1+X2′+X3′+X4)]′

mantık fonksiyonu komut programı ile Çizelge 1.7'deki gibi yazılabilir.

Çizelge1.7

Simatic S7-200	Hitachi	Omron
LDN X1	LDI X1	LDNOT X1
ON X2	ORI X2	ORNOT X2
O X3	OR X3	OR X3
O X4	OR X4	OR X4
LDN X1	LDI X1	LDNOT X1
ON X2	ORI X2	ORNOT X2
ON X3	ORI X3	ORNOT X3
ON X4	ORI X4	ORNOT X4
ALD	ANB	LD X1
LD X1	LD X1	ORNOT X2
ON X2	ORI X2	OR X3
O X3	OR X3	ORNOT X4
ON X4	ORI X4	LD X1
ALD	LD X1	ORNOT X2
LD X1	ORI X2	ORNOT X3
ON X2	ORI X3	OR X4
ON X3	OR X4	ANLD
O X4	ANB	ANLD
ALD	ANB	ANLD
NOT	NOT	NOT
= Y	OUT Y	OUT Y

Genel olarak bir kontaklı (anahtarlamalı) kumanda devresi Boole Cebri özelliklerinden yararlanılarak her zaman kolay programlanabilir seri-paralel veya paralel-seri devreler biçimlere getirilebilir. PLC'de doğrudan programlamaya uygun olmayan kumanda devre yapıları bu şekilde düzenlenerek programlanır. Bu tür bir devreye örnek olarak Şekil 1.20'deki anahtarlama devresi gösterilebilir. Bu devreye karşı düşen merdiven mantığı programı (b), PLC'de programlamaya uygun olmayan bir yapıdadır.



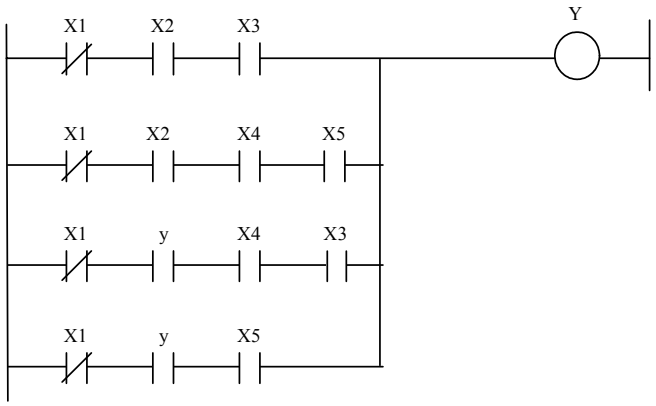
Şekil 1.20 Programlamaya uygun olmayan bir merdiven mantığı yapısı

Bu devreye ilişkin mantık fonksiyonu yazılır ve çarpımların toplamı biçimine getirilirse, PLC'de geçerli hem merdiven mantığı hem de komut programı kolayca yazılabilir.

Bu kumanda devresine ilişkin tüm akım yolları göz önüne alınarak mantık fonksiyonu

$$Y=X1'\bullet X2\bullet X3+X1'\bullet X2\bullet X4\bullet X5+X1'\bullet y\bullet X4\bullet X3+X1'\bullet y\bullet X5$$

biçiminde elde edilir. Bu fonksiyona ilişkin merdiven mantığı Şekil 1.21'de gösterilmiştir.



Şekil 1.21 Programlamaya uygun merdiven mantığı yapısı

Şekil 1.21'de gösterilen merdiven mantığı programı Simatic S7-200 ve Hitachi H-200 PLC'leri için Çizelge 1.8'de verildiği gibi programlanır.

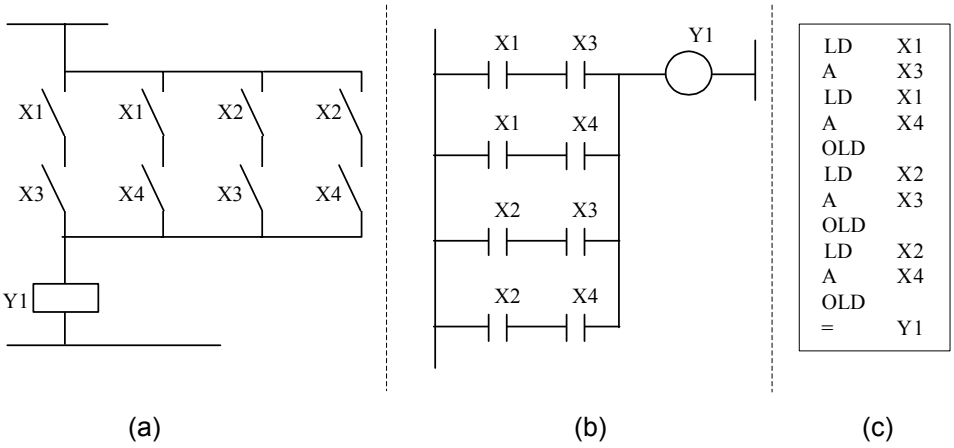
Çizelge 1.8

Simatic S7	Hitachi H-200
LDN X1	LDI X1
A X2	AND X2
A X3	AND X3
LDN X1	LDI X1
A X2	AND X2
A X4	AND X4
A X5	AND X5
OLD	ORB
LDN X1	LDI X1
A Y	AND Y
A X4	AND X4
A X3	AND X3
OLD	ORB
LDN X1	LDI X1
A Y	AND Y
A X5	AND X5
OLD	ORB
= Y	OUT Y

Mantık fonksiyonlarının, çarpımların toplamı veya toplamların çarpımı biçiminde düzenlenmesi, programlama işlemini kolaylaştırır. Ancak, bu şekilde yapılan programlama, genellikle gereğinden çok komut içerir. Aynı mantık fonksiyonunu daha az komut ile programlamak için, yine Boole cebri işlemlerinden yararlanılabilir. Örneğin,

$$Y1=X1\bullet X3 + X1\bullet X4 + X2\bullet X3 + X2\bullet X4$$

biçiminde verilmiş olan bir mantık fonksiyonu göz önüne alınsın. Bu fonksiyona ilişkin kontaklı anahtarlama devresi (a), merdiven mantığı (b) ve Simatic S7-200 PLC komut programı (c), Şekil 1.22'de gösterilmiştir.



Şekil 1.22 Çarpımların toplamı biçimindeki bir mantık fonksiyonuna ilişkin programlar

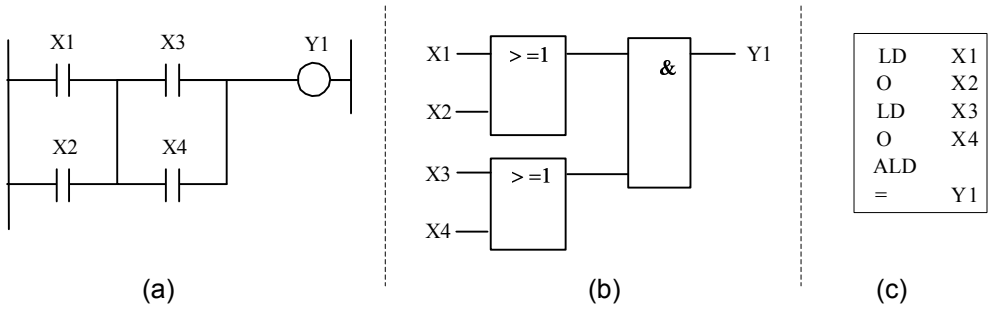
Bu mantık fonksiyonu, Boole cebri işlemleri ile

$$Y1=X1\bullet X3 + X1\bullet X4 + X2\bullet X3 + X2\bullet X4 =X1\bullet(X3 + X4)+X2\bullet(X3 + X4)$$

ve

$$Y1=(X1 +X2)\bullet(X3 + X4)$$

biçimine getirilebilir. Bu fonksiyona ilişkin merdiven mantığı (a), fonksiyon blok (b) ve S7-200 PLC komut programı (c), Şekil 1.23'deki gibi verilebilir.



Şekil 1.23 Daha az komutlu program örneği

Herhangi bir kumanda devresine ilişkin bir programın daha az komut ile yazılabilmesi, daha az program belleği gerektirmesi ve hız açısından önemlidir. Ancak, yazılan bir programın kolay izlenebilir ve anlaşılabilir olması da çoğu kez aranan bir özelliktir. Bazı durumlarda bu tür özelliklerin sağlanabilmesi için ara değişkenler kullanılması ve buna bağlı olarak programın genişletilmesi gibi durumlarla da karşılaşılabilir.

BÖLÜM 2
SIMATIC S7-200

2.1 GİRİŞ

Bu bölümde, Simatic S7-200 programlanabilir kontrolörünün genel özellikleri ve temel kumanda işlemlerine ilişkin komutları tanıtılacaktır. Burada verilen kontrol birimine ilişkin genel yapı, çalışma ilkesi, mantıksal ve zamanlama işlemleri ile ilgili verilen bilgiler, diğer PLC'ler için de geçerli veya uyarlanabilir bilgilerdir.

2.2 GENEL ÖZELLİKLER

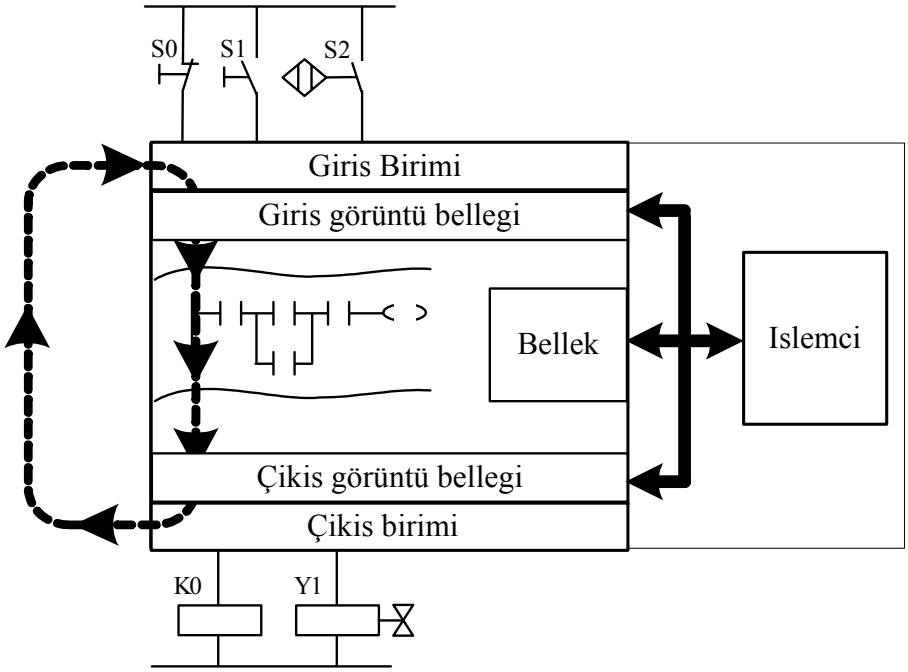
Simatic S7-200 CPU 22x işlemcili kontrolör, küçük ölçekli endüstriyel otomasyon uygulamalarının gerçeklenmesine yönelik tasarlanmış bir kontrol birimidir. Bu kontrol birimi ile giriş sayısı en çok 128 ve çıkış sayısı en çok 120 olan kumanda sistemleri gerçeklenebilir. Bu aileye ilişkin işlemcilerin genel özellikleri Çizelge 2.1'de, giriş-çıkış sayılarına ilişkin bilgiler EK 1'deki çizelgede verilmiştir. Bir otomasyon sistemi için uygun PLC seçiminde bu özellikler göz önüne alınır. Kumanda sistemlerinde PLC seçimi, giriş-çıkış sayısı, program ve veri belleği boyutu, sayıcı ve zamanlayıcı sayısı gibi özelliklere göre belirlenir. Bunun dışındaki özellikler uygulamaya göre önem kazanır. Örneğin, geri beslemeli kontrol uygulamaları için analog giriş-çıkış sayısı, kesmeli çalışma yeteneği, matematik işlem yeteneği ve hızı gibi özellikler aranır. Yüksek hızlı işaretlere bağlı kumanda sistemlerinin gerçeklenmesinde yüksek hızlı sayıcı, iletişim gerektiren uygulamalar için iletişim yeteneği, tarihe bağlı kumanda uygulamalarında gerçek zaman saati bulunması gereken özelliklerdir.

Çizelge 2.1

	CPU 221	CPU 222	CPU 224	CPU 226
Program belleği [kByte]	4	4	8	8
Veri belleği [kByte]	2	2	4	4
CPU üzerinde giriş /çıkış sayısı	6/4	8/6	14/10	24/16
Genişleme birimi sayısı	yok	2	7	7
G/C görüntü Belleği [bit]	256	256	256	256
İç Bellek (M bit) [bit]	256	256	256	256
Analog G/C Bellek alanı	-	16I/16Q	32I/32Q	32I/32Q
Sayıcı /Zamanlayıcı	256/256	256/256	256/256	256/256
Mantık. Kom. işleme hızı	0.37μs	0.37μs	0.37μs	0.37μs
Reel matematiksel işlem olanağı	var	var	var	var
Gerçek zaman saati	eklenebilir	eklenebilir	var	var
Hızlı sayıcı	4[20kHz]	4[20kHz]	6[20kHz]	6[20kHz]
Yüksek hızlı çıkış (PTO,PWM)	2[20kHz]	2[20kHz]	2[20kHz]	2[20kHz]
İletişim kesmesi	1tr / 2 rec	1tr / 2 rec	1tr / 2 rec	2tr / 4 rec
Donanım kesmesi	4	4	4	4
Zaman bağlı kesme	2(1:255ms)	2(1:255ms)	2(1:255ms)	2(1:255ms)
Şifre koruma	var	var	var	var

Program belleği gereksinimi; program uzunluğu, kullanılan komutlar ve programda işlenen veri boyutuna göre değişir. Örneğin, kumanda devrelerinin gerçeklenmesinde kullanılan mantıksal "VE", "VEYA" ve "DEĞİL" gibi işlem komutları için 2 bayt, zamanlama, sayıcı ve karşılaştırma komutları için 6-7 bayt, silme (SET) ve kurma (RESET) komutları için 8 bayt program belleği gerekir. Kontrol uygulamalarında kullanılan matematiksel işlem komutları 2 ile 7 bayt arasında program belleği gerektirir. Bazı uygulamalarda programın yürütülme hızı önem kazanır. Program yürütülme süresi kullanılan komutlara ve işlemin yapıldığı bellek alanına göre değişir. Örneğin mantıksal işlem komutlarının giriş görüntü belleğindeki bir veriyi işleme süresi 0.37μs, yerel bellek alanındaki bir veriyi işleme süresi 10.8μs, doğrudan giriş adreslerini (anında işlem) işleme süresi 27μs olarak verilir.

S7 200 sınıfı PLC, Bölüm 1'de değinildiği gibi tüm PLC'ler gibi Şekil 2.1'de gösterilen temel birimlerden oluşur.



Şekil 2.1 PLC temel birimleri

2.2.1 BELLEK ALANLARI VE ERİŞİM ADRESLERİ

S7-200 ailesi CPU 22x işlemcili PLC'lerde

Giriş görüntü belleği (I)

Çıkış görüntü belleği (Q)

Yardımcı bellek (M)

Değişken bellek (V)

Yerel (local) bellek (L)

Zamanlayıcı (T)

Sayıcı (C)

Yüksek hızlı sayıcı (HC)

Özel bellek (SM)

Analog giriş belleği (AI)

Analog çıkış belleği (AQ)

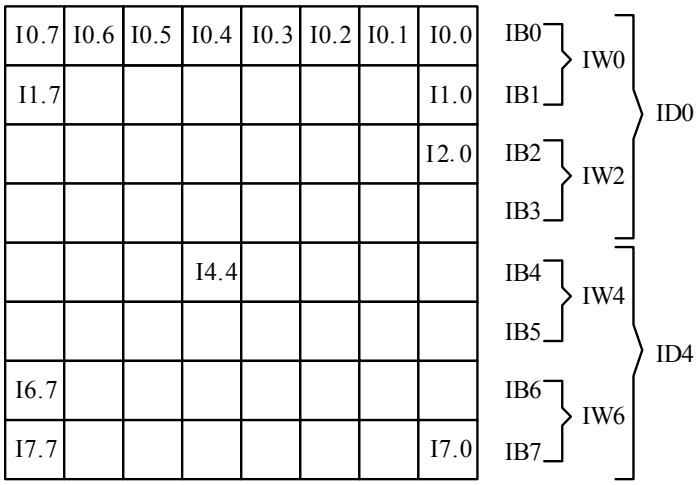
Ardışıl denetim belleği (S)

Akümülatör (AC)

gibi bellek alanları bulunur.

I, Q, M, V, L, SM ve S bellek alanlarına bit, 8 bit (bayt), 16 bit sözcük (word) ve 32 bit çift sözcük (double word) olarak erişilebilir. AI ve AQ 16 bit, T ve C bit ve 16 bit, HC 32 bit ve AC 8 bit, 16 bit ve 32 bit olarak erişilen alanlardır.

Bellek alanı adresleri alan simgesi ile başlar. 8 bit'lik alan adresleri, artan tamsayılar, bit adresleri düşük anlamlıdan (sağdan) yüksek anlamlıya (sola) doğru 0'dan 7'ye sıralanır. Şekil 2.2'de giriş görüntü belleğine ilişkin 1 bit, 8 bit, 16 bit ve 32 bitlik adresler gösterilmiştir.



Şekil 2.2 Giriş görüntü bellek alanının adreslenmesi

I, Q, M, V, L, SM ve S bellek alanları

Bit olarak erişimde Ix.bit {x: 0, 1, 2,...; bit: 0, 1, .. ,7}

8 bitlik erişimde IBx {x: 0, 1, 2,... }

16 bitlik erişimde IWx {x: 0, 1, 2,... }

32 bitlik erişimde IDx {x: 0, 1, 2,... }

biçiminde adreslenir. Burada x, alanın bayt başlangıç adresini gösterir. Buna göre

IW0={IB0, IB1},

IW1={IB1, IB2},

IW2={IB2, IB3}

IW3={IB3, IB4}

ID0={IB0, IB1, IB2, IB3} = {IW0, IW2}

ID1={IB1, IB2, IB3, IB4} = {IW1, IW3}

ID2={IB2, IB3, IB4, IB5} = {IW2, IW4}

alanlarını gösterir.

16 bit ve 1bit olarak erişilebilen zamanlayıcılara ilişkin adresler

Edited by Foxit Reader
Copyright(C) by Foxit Software Company,2005-2008
For Evaluation Only.

T0, T1, T2, T3, ...

sayıcılara ilişkin adresler

C0, C1, C2; C3, ...

olarak verilir.

16 bitlik analog giriş adresleri

AIW0, AIW2, AIW4, ...

Analog çıkış adresleri

AQW0, AQW2, AQW4, ...

biçiminde artan çift sayılarla gösterilir.

8 bit, 16 bit ve 32 bit olarak erişilen akümülatörlere

AC0, AC1, AC2 ve AC3

32 bitlik hızlı sayıcılara

HC0, HC1, HC2,...,HC5 adresleri ile erişilir.

2.2.2 ÖZEL BELLEK ALANI (SMX)

Özel bellek alanı, işlemcinin donanım ve çeşitli çalışma durumlarına ilişkin verilerin tutulduğu çeşitli boyutlardaki (SMx.x:1 bit, SMBx:8 bit, SMWx:16 bit ve SMD:32 bit) yazmaçlardan oluşur. Çalışma durumlarının (hızlı sayma, kesmeli çalışma, iletişim gibi) tanımlanması ve denetimine ilişkin işlemler, donanım bilgileri, bellek durumu, çeşitli işlemlerin durumu ile ilgili veriler bu yazmaçlarda tutulur. Örneğin,

SM0.0: İçeriği her zaman 1 olan özel bellek biti

SM0.1: İçeriği ilk tarama çevriminde 1 daha sonra 0 olan özel bellek biti

SM0.2: Kalıcı bellek alanında veri kaybı olduğunda bir tarama çevriminde 1 değerini alan özel bellek biti

SM0.3: PLC'ye enerji verildikten sonra ilk tarama çevriminde içeriği 1 olan özel bellek biti

SM0.4: İçeriği 30 saniye 1, 30 saniye 0 olan özel bellek biti

SM0.5: İçeriği 0.5 saniye 1, 0.5 saniye 0 olan özel bellek biti

SM0.6: Her tarama çevriminde içeriği değişen özel bellek biti

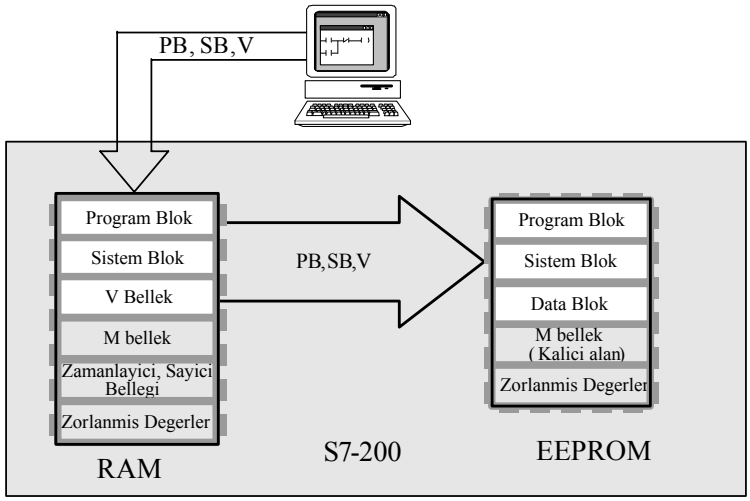
SM0.7: PLC çalışma durumu anahtarı "TERM" konumunda iken içeriği 0, "RUN" konumunda iken 1 olan özel bellek biti özelliklerini taşır.

Buna benzer birçok özel işleve sahip yazmaç (SMB0, SMB1, ... , SMB200) bulunur.

2.2.3 PROGRAM VE VERİLERİN PLC'YE AKTARILMASI VE PLC'DEN ALINMASI

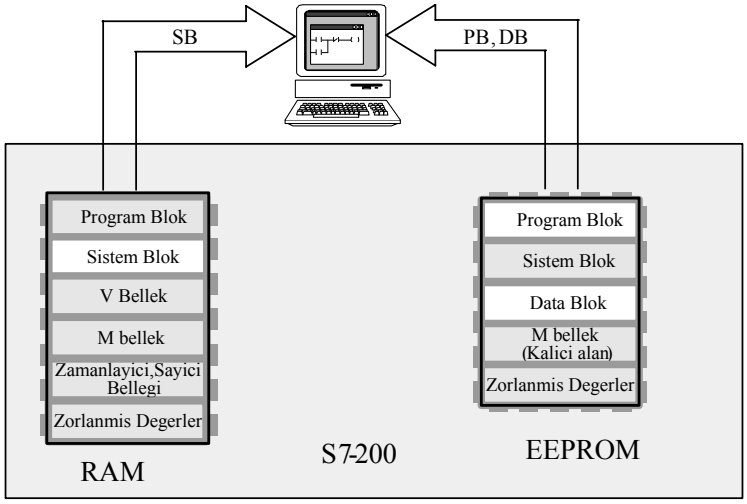
Program ve veriler kalıcı olmayan RAM ve kalıcı EEPROM saklama ortamlarında tutulur. İşlemcinin enerjisiz kalması durumunda RAM saklama ortamındaki veriler bir yüksek sığalı kondansatör yardımıyla belirli bir süre (CPU 221, CPU2 için 50 saat, CPU 224 için 100 saat ve CPU 226 için 190 saat) tutulur. EEPROM ortamına saklanmış olan veriler ise kalıcıdır.

Programın ve verilerin PLC'ye aktarılması (Download) durumunda veriler (PB: program blok, SB sistem blok, V:değişken bellek) hem RAM hem de EEPROM saklama ortamına yüklenir (Şekil 2.3). M yardımcı belleği, zamanlayıcı ve sayıcı içerikleri ve zorlanmış (force) değerler eski durumunu korur.



Şekil 2.3 Program ve verilerin PLC'ye yüklenmesi

Programın ve verilerin PLC'den PC'ye aktarılması (Upload) durumunda sistem verileri RAM, program ve kullanıcı verileri EEPROM'dan alınır (Şekil 2.4).



Şekil 2.4 Program ve verilerin PLC'den alınması

2.2.4 ENERJİ KESİLMESİ DURUMUNDA VERİLERİN KORUNMASI

Giriş (I) ve çıkış (Q) görüntü belleğinde tutulan veriler kalıcı değildir. Enerji kesilmelerinde veya PLC işlevsiz durumda (STOP durumunda) iken sıfırlanır.

V(değişken bellek), M (yardımcı bellek), T (kalıcı zamanlayıcı) ve C (sayıcı) bellek alanları kalıcı olarak tanımlanabilir. Eğer bu bellek alanları kalıcı olarak tanımlanırsa enerji kesilmesi durumunda içerikleri korunur. Kalıcı alanların tanımlanması MicroWIN yazılımı ile sistem blok ayarları altında yapılır. Bu alana erişmek için "view" > "component" > "System Block" > "Retentive ranges" yolu izlenir. Kullanıcı programı hem RAM hem de EEPROM ortamında saklandığından kalıcıdır.

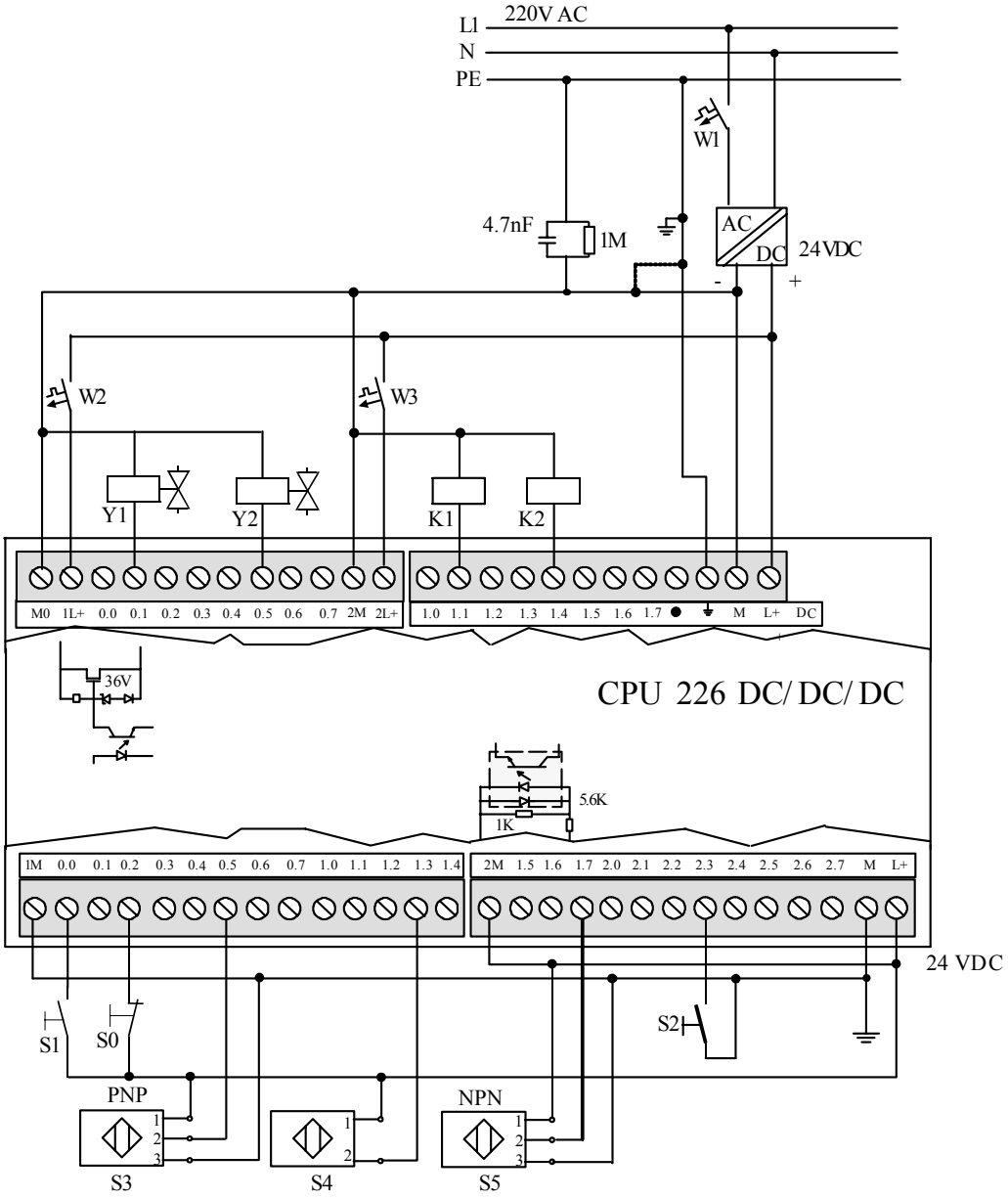
2.3 DIŞ DEVRE BAĞLANTILARI

PLC'li kumanda devrelerinde dış devre bağlantıları; güvenli işletim, süreklilik ve eleman ömürleri gibi etmenler göz önüne alınarak yapılır. Her üretici firma, çeşitli dış devre bağlantı şemaları önerir. Siemens S7-200 PLC ailesi işlemcilerine ilişkin bu tür dış devre bağlantı şemaları EK'te verilmiştir. Bir dış devre bağlantı şemasında güç kaynağı ve koruma topraklaması, giriş ve çıkış elemanlarının bağlantı biçimleri gösterilir. Siemens S7-200 PLC ailesi CPU 226 DC/DC/DC işlemcisine ilişkin bir dış devre bağlantı şeması Şekil 2.5'te gösterilmiştir. Koruma topraklama (PE) iletkeni ya doğrudan (kesikli çizgili hat) ya da 1MΩ direnç ve 4.7nF kondansatörden oluşan bir devre üzerinden işlemci M noktalarına ve güç kaynağı negatif ucuna bağlanır. Bu uygulamada çıkış birimine bağlı elemanların

uyarılması için gerekli enerji (1L+, 2L+ uçları) besleme kaynağından alınmıştır. Ancak, bu iş için ayrı bir güç kaynağı da kullanılabilir. W2 ve W3 anahtarlı sigortalar hem çıkış devresinin kısa devreye karşı korunması hem de PLC'den bağımsız olarak çıkış elemanları enerjisini kesmek için kullanılır. Giriş birimine bağlı elemanları uyarmak için gerekli enerji, doğrudan işlemcideki kaynaktan (L+, M uçları) alınabileceği gibi, bir dış kaynaktan da sağlanabilir.

Kumanda düğmeleri ve algılama elemanları özelliklerine göre çeşitli biçimlerde giriş birimine bağlanır. Kumanda devrelerinde en sık karşılaşılan

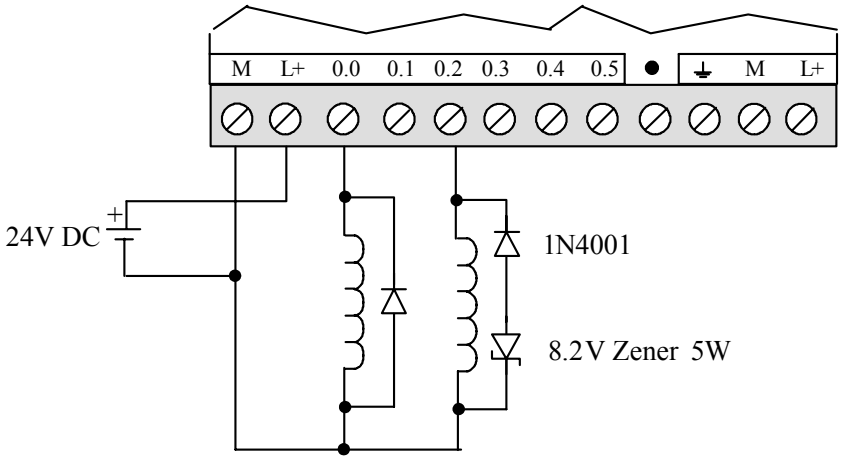
Durdurma düğmesi (S0),
Başlatma düğmesi (S1, S2),
Üç uçlu PNP çıkışlı yaklaşım ya da algılama elemanı (S3),
İki uçlu kontaklı (reed) yaklaşım elemanı (S4),
Üç uçlu NPN çıkışlı yaklaşım ya da algılama elemanı (S5)
bağlantı biçimleri Şekil 2.5'te gösterilmiştir.



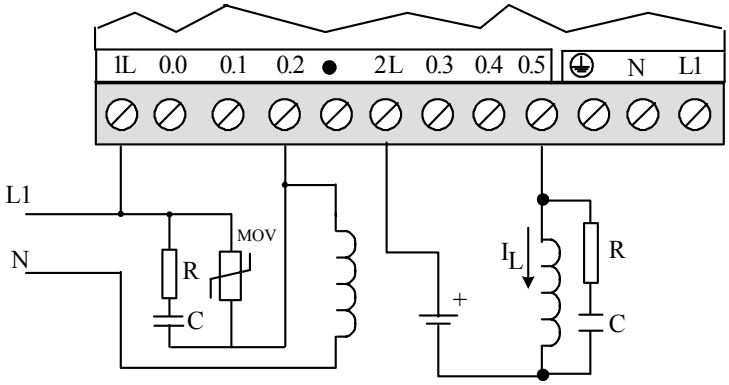
Şekil 2.5 SIMATIC S7-200 CPU 226 işlemcili bir PLC için dış devre bağlantıları

Çıkış birimine bağlanan elemanlar, anahtarlama nedeniyle oluşan aşırı gerilimlere karşı ek koruma devreleri ile korunur. Bu tür devreleri içeren elemanlar doğrudan çıkış birimine bağlanabilir. Şekil 2.5'teki elektropnömatik valf bobinlerinin çıkış birimine bağlantısı bu şekildedir.

Koruma devreleri içermeyen valf ve röle bobini gibi endüktif özellikteki elemanların; yarı iletken çıkışlı birime bağlanmasında Şekil 2.6'da verilen ek koruma devreleri, röle ya da tiryak çıkışlı birime bağlanmasında Şekil 2.7'de verilen ek koruma devreleri önerilir.



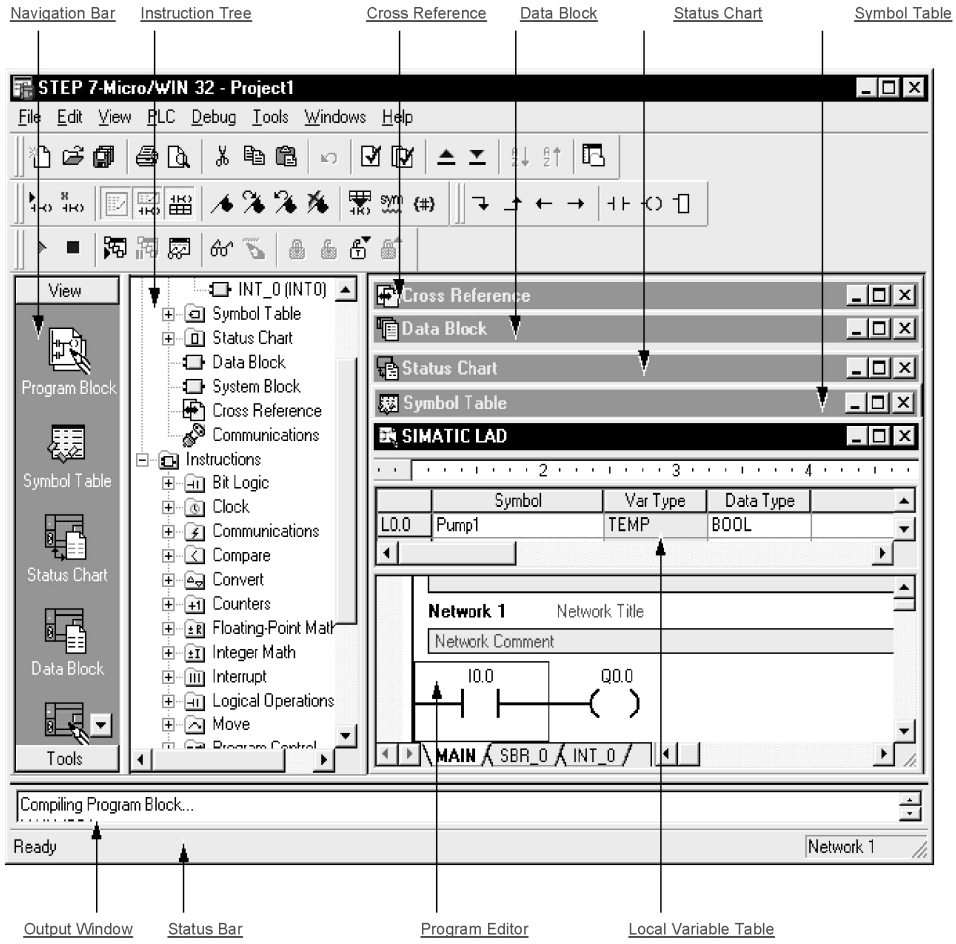
Şekil 2.6 Transistör ile sürülen endüktif yüklere ilişkin koruma devresi bağlantı şeması



Şekil 2.7 Röle ve triyak ile sürülen endüktif yüklere ilişkin koruma devresi bağlantı şeması

2.4 PROGRAMLAMA

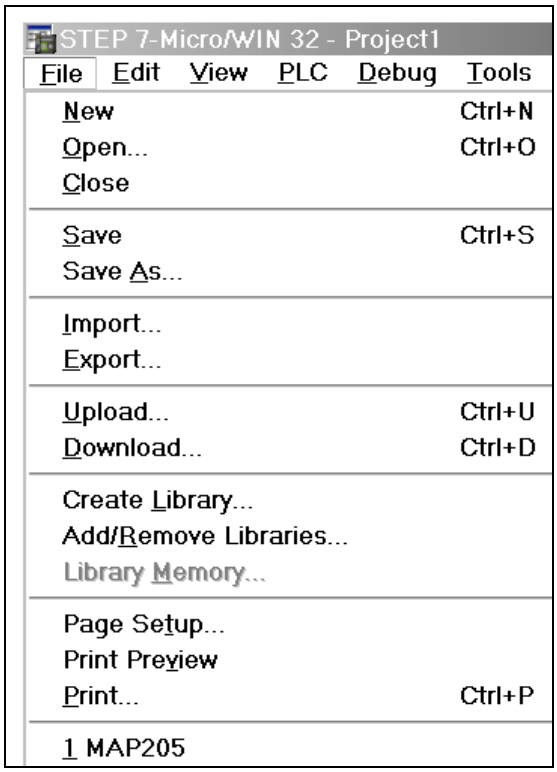
Simatic S7-200 PLC'lerini programlamak için kullanılan MicroWIN yazılımında komut ile programlama (statement list) "STL", merdiven mantığı ile programlama (ladder logic) "LAD" ve fonksiyon blok gösterimi ile programlama (function block diagram) "FBD" seçenekleri bulunur. Bu yazılım; programlama işlemi, çalışma durumlarını izleme, donanım bilgilerine erişim ve donanım ayarları gibi olanakları sağlar. Açılıştaki görünümü Şekil 2.8'de verilen bu yazılım "File", "Edit", "View", "PLC", "Debug", "Tools", "Windows" ve "Help" gibi menülerden oluşur.



Şekil 2.8 STEP 7 MicroWIN yazılımının genel görünümü

Yeni bir otomasyon projesinin hazırlanması; projeye isim verme, programlama dilinin seçimi, sembol tablosu oluşturma, PLC bilgisayar iletişim ayarlarının yapılması, donanım ayarlarının yapılması ve izleme çizelgesinin hazırlanması gibi adımlardan oluşur.

Yeni bir proje için "File">"New" seçeneği kullanılır ve bu projeye bir isim verilerek "File"> "Save As" kullanılarak saklanır. Bu işlemden sonraki tüm ayarlar bu proje dosyasında tutulur.



Şekil 2.9 "File" menüsü

"File" menüsü altında (Şekil 2.9) bulunan "Open", "Close" ve "Save" gibi seçenekler tüm Windows uygulamalarında geçerli olan işlevleri sağlarlar.

"Import" seçeneği MicroWin ortamı dışında yazılmış ASCII formatında bir kaynak dosyasının alınması için,

"Export" seçeneği MicroWin ortamında yazılmış programın ASCII formatında bir kaynak dosyası olarak saklanması için kullanılır.

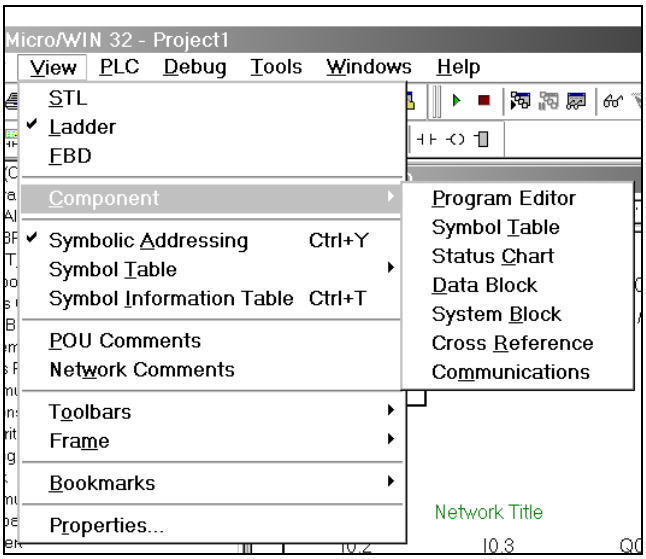
"Upload" seçeneği PLC belleğindeki programın alınması için,

"Download" seçeneği yazılan programın PLC belleğine aktarılması için kullanılır.

"Create Library" ve "Add/Remove Libraries" seçenekleri kütüphane oluşturma, ekleme ve kaldırma gibi işlevleri sağlarlar.

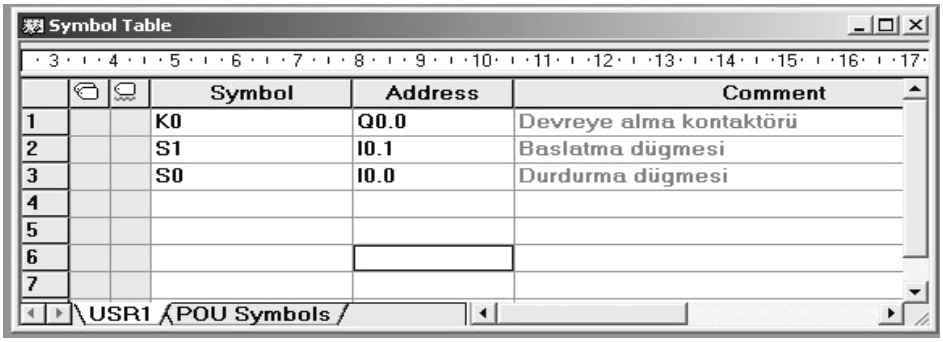
Programın hangi dilde yazılacağı, sembol tablosu, iletişim ayarları, izleme çizelgesi ve donanım ayarları "View" başlığı altındaki seçenekler kullanılarak tanımlanır.

Programlama dili "View" altındaki "STL", "Ladder" veya "FBD" seçilerek belirtilir (Şekil 2.10).



Şekil 2.10 "View" menüsü

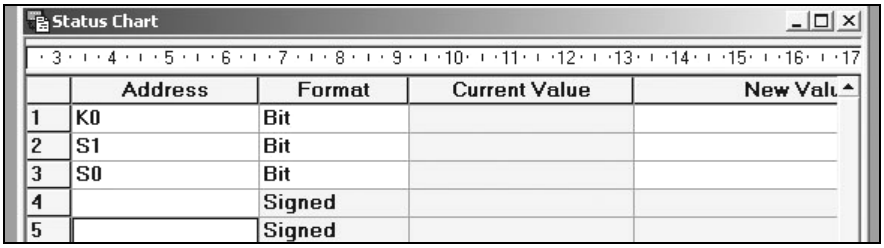
PLC giriş-çıkış ve bellek adresleri yerine bu adreslerle ilişkili simge adlarının kullanılması, programı daha kolay anlamak ve izlemek olanağını sağlar. Bunun için "View" > "Component" > "Symbol Table" seçilerek bir sembol tablosu alanı açılır ve bu alana adreslere karşı düşen simgeler yazılır. Program yazılırken PLC adresleri yerine bu simgeler kullanılır (Şekil 2.11).



Şekil 2.11 Sembol tablosu

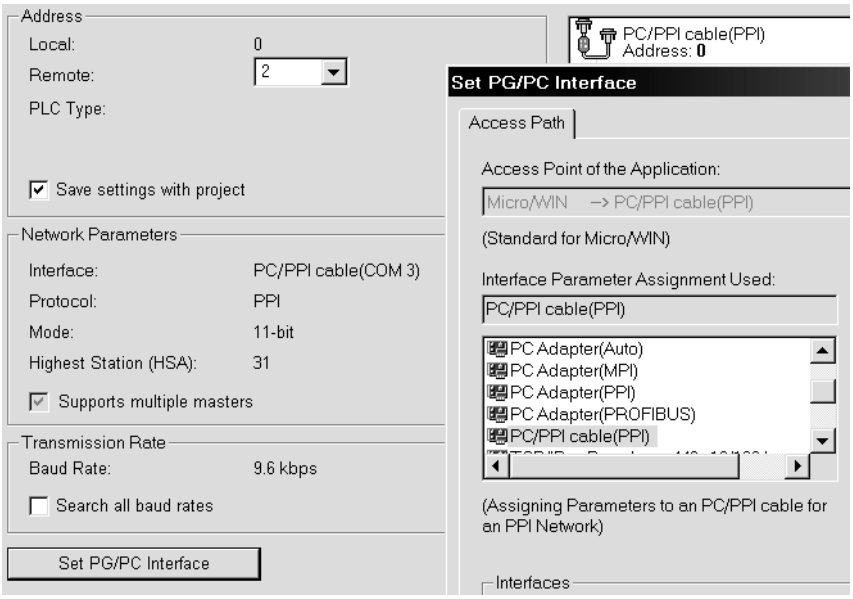
"Component" altındaki diğer seçenekler, durum çizelgesi (Status Chart), veri bloğu (Data Block), sistem ayarları (System Block) ve iletişim (Communications) ayarları için kullanılır.

Durum çizelgesi, PLC'de yürütülen bir programda değişkenlerin aldığı değerleri izlemek için kullanılır. Bunun için "Status chart" seçilerek bir durum çizelgesi alanı açılır ve bu alana izlenecek olan adresler veya simgeleri yazılır. PLC'deki programın işleyişini izlemek için "Debug" > "Program status" veya "Debug"> "Chart Status" seçilir (Şekil 2.12).



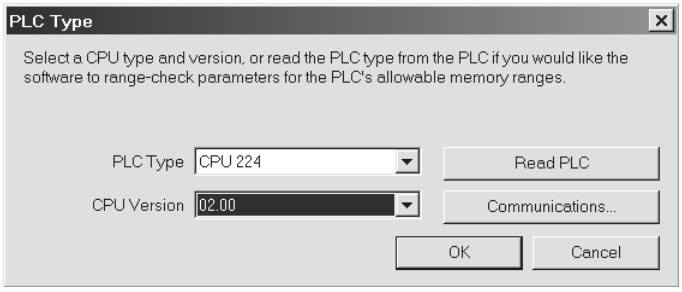
Şekil 2.12 Durum çizelgesi

Durum çizelgesinden değişken içeriklerinin izlenmesi için iletişim ayarlarının doğru yapılması gerekir. İletişim ayarları için "Communication" seçilir; iletişim protokolü, PLC adresi, iletişim iletkenin bağlı olduğu seri port, iletişim hızı gibi ayarlar yapılır (Şekil 2.13).



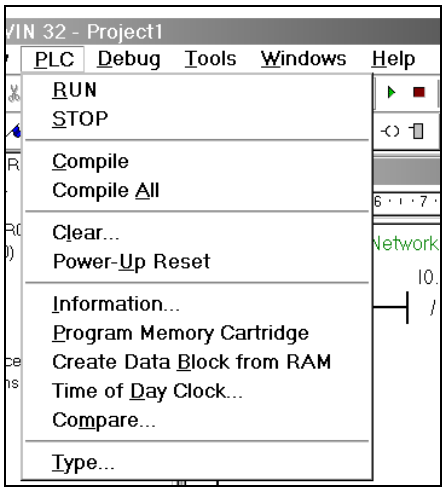
Şekil 2.13 İletişim ayarlarına ilişkin pencerenin görünümü

Programlanacak İşlemci türünün tanımlanması için "PLC" başlığı altındaki "Type" seçeneği kullanılır. İletişim ayarları buradan da yapılabilir. Şekil 2.14'teki pencereden PLC türü seçilir. Eğer iletişim ayarları doğru ise "Read PLC" seçeneği ile bilgisayara bağlı işlemci bilgileri okunabilir. Eğer iletişim ayarları doğru değilse "Communications" seçeneği ile ayarlara ilişkin pencereye erişilir ve gerekli ayarlar yapılır.



Şekil 2.14 İşlemci türünün seçimine ilişkin pencere

MicroWIN "PLC" menüsü (Şekil 2.15) altında



Şekil 2.15 PLC menüsü

PLC'yi çalıştırma (RUN), durdurma (STOP), yazılan programın derlenmesi (Compile), PLC'deki programın silinmesi (Clear), PLC" nin tekrar başlatılması (Power-Up R),

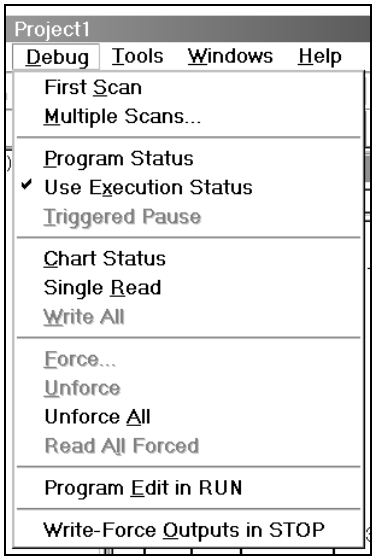
PLC ile bilgiler (Information), Programın dış bellek birimine yazılması (Program Memory Cartridge),

PLC veri belleğindeki değerlerin veri bloğuna yazılması (Create Data Block from RAM),

Gerçek zaman saatinin okunması (Time of Day Clock),

PLC belleğindeki program ile MicroWIN'deki programın karşılaştırılması (Compare) gibi işlemler yapılabilir.

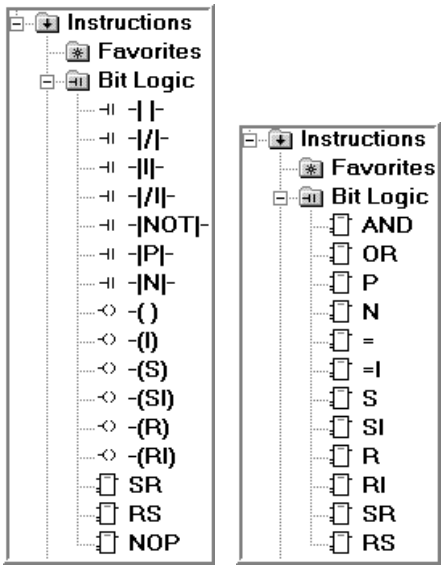
PLC'nin çalışma durumundaki bilgiler ve işlemler için "Debug" başlığı altındaki seçenekler kullanılır (Şekil 2.16). Buradaki komutlar, programın yürütülmesi sürecindeki işlemleri denetlemede kullanılır. Örneğin "First Scan" PLC'nin bir tarama işlemi yapıp durmasını, "Multiple Scans" belirtilen sayıda tarama yapıp durmasını sağlamak, "Program Status" programı izlemek, "Force" istenen giriş ve çıkışları belirli değerlere kurmak ve tutmak, "Unforce" "Force" işlemini sonlandırmak için kullanılır.



Şekil 2.16 "Debug" menüsü

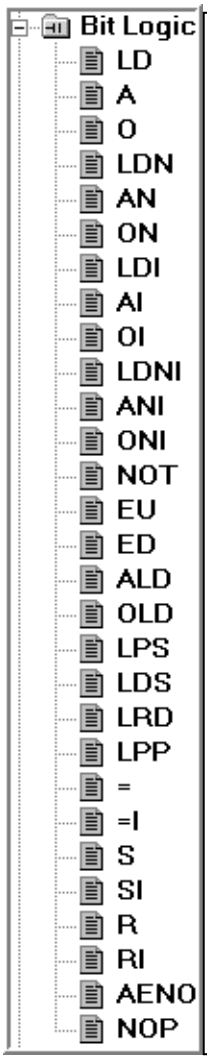
2.5 MANTIKSAL İŞLEMLER

Kumanda devrelerinin gerçeklenmesine ilişkin mantıksal işlem komutlarının STEP 7 Micro/WIN32 yazılımı komut ağacında (instruction tree) "LAD" ve "FBD" dilindeki görünümü Şekil 2.17'de verilmiştir.



Şekil 2.17 Merdiven mantığı (LAD) ve fonksiyon blok (FBD) komutları

Komut (STL) ile programlamada Şekil 2.17'de görülen komutlar ile aynı işlevi sağlayan daha çok komut kullanılır (Şekil 2.18).



LD: YIĞININ ÜSTÜNE YAZMA (İŞLEMİ BAŞLATMA)

A: VE İŞLEMİ

O: VEYA İŞLEMİ

LDN: TÜMLEYENİNİ YIĞINA YAZMA

AN: TÜMLEYENİNE VE İŞLEMİ

ON: TÜMLEYENİNE VEYA İŞLEMİ

LDI, AI, OI, LDNI, ANI, ONI: İVEDİ (ANINDA) İŞLEM

NOT: TÜMLEYENİ (DEĞİLİ)

EU: ÇIKAN KENAR

ED: İNEN KENAR

ALD, OLD, LPS, LDS, LRD, LPP: YIĞIN BELLEĞİ İŞLEM KOMUTLARI

= : ATAMA (AKTARMA)

S: KURMA

R: SİLME

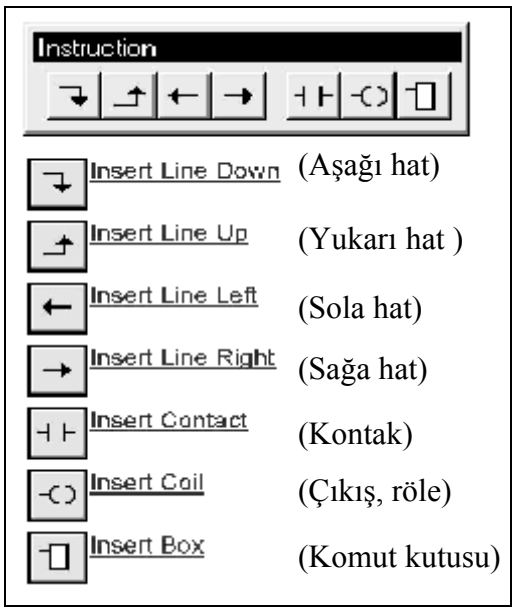
AENO: İŞLEM HATASIZ İŞARETİ

NOP: İŞLEM YOK

Şekil 2.18 Mantıksal işlem komutları

2.5.1 TEMEL MANTIKSAL İŞLEMLER

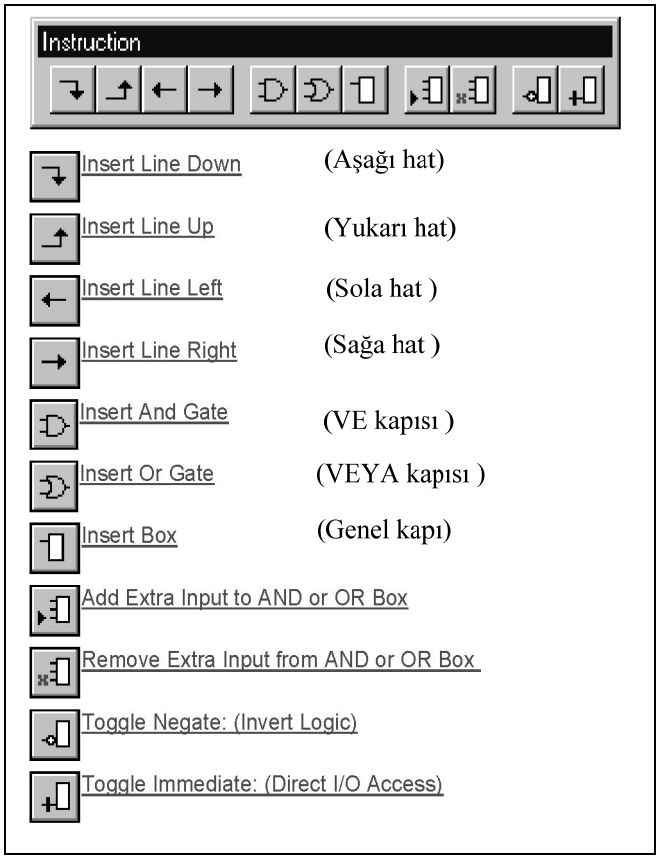
Merdiven mantığı ile programlamada kullanılan normalde açık (NA), normalde kapalı (NK) kontak ve hatlar MicroWIN yazılımında "Instruction Toolbar" altında bulunur (Şekil 2.19). Programlamada bu simgeler (ikonlar) kullanılır.



Şekil 2.19 Merdiven mantığı programına ilişkin hat simgeleri

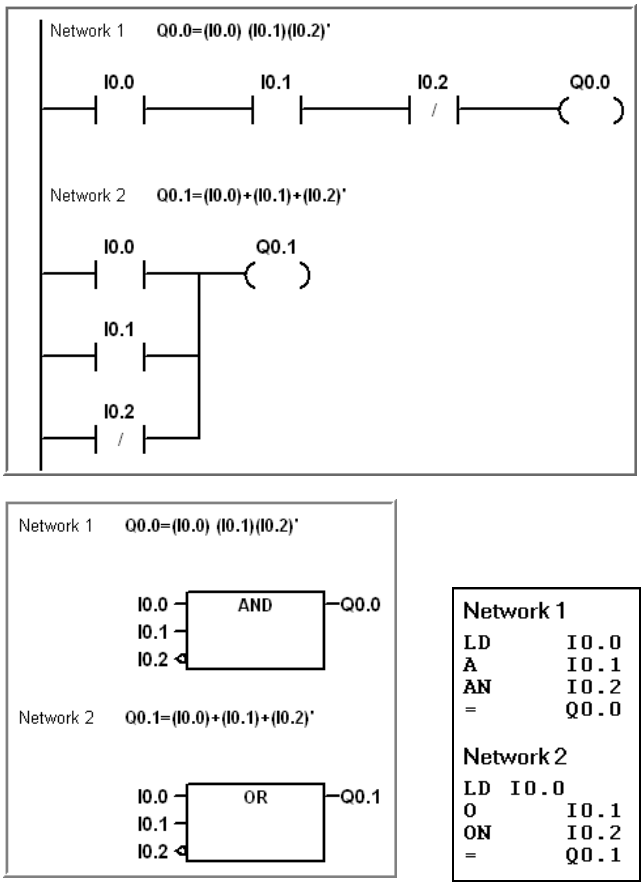
Fonksiyon blok diyagramı (FBD) ile programlamada ise "instruction toolbar" altında Şekil 2.20'de verilen simgeler yer alır. Bu simgeler ve hatlar kullanılarak program gerçeklenir.

Komut ile programlamada "STL" metin editörü kullanılır.



Şekil 2.20 FBD ile programlamada kullanılan simgeler

Temel mantıksal işlemlere ilişkin bir program örneği Şekil 2.21'de verilmiştir. Burada "VE", "VEYA" ve "DEĞİL" mantıksal işlemlerini içeren iki ayrı mantık fonksiyonunun her üç dildeki karşılıkları gösterilmiş ve geçerli adres alanları belirtilmiştir.

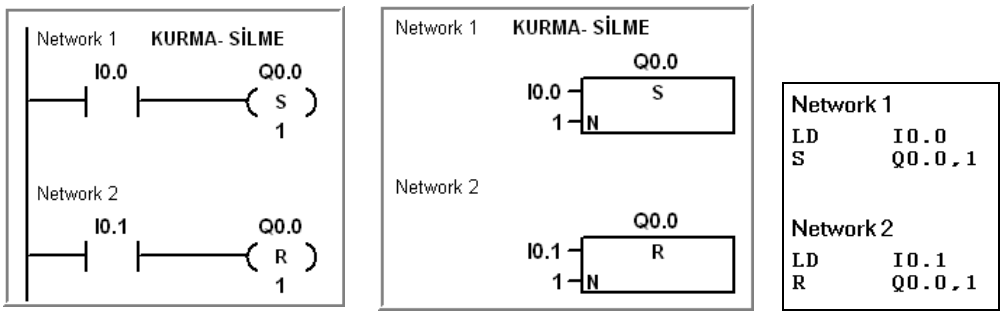


Geçerli adresler: I, Q, M, SM, T, C, S, L (bit)

Şekil 2.21 VE, VEYA ve DEĞİL işlemlerine ilişkin LAD, FBD ve STL programları

2.5.2 KURMA-SİLME (SET-RESET) İŞLEMLERİ

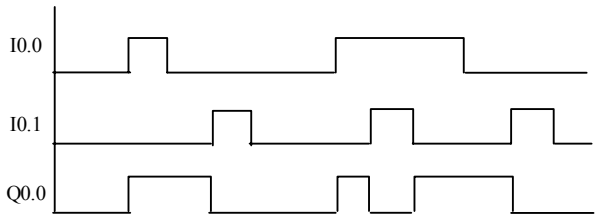
Kurma-silme komutları ile N (8 bit:1,2,3, .. , 255) adet ardışık adres içeriği 1'e kurulur veya silinir. Kurma işlemi için **S Qx.y,N** komutu, silme işlemi için **R Qx.y, N** komutu kullanılır. Bu komutlar ile Qx.y adresinden başlanarak N adet ardışık adres 1'e kurulur ve ya sıfırlanır. Örneğin, S Q0.0, 3 komutu ile Q0.0, Q0.1 ve Q0.2 adreslerinin içeriği 1'e kurulur; R Q0.0, 4 komutu ile Q0.0, Q0.1, Q0.2 ve Q0.3 adreslerinin içeriği silinir. Kurma-silme işlemlerine ilişkin bir program örneği Şekil 2.22'de, zamanlama diyagramı Şekil 2.23'te verilmiştir. Burada I0.0=1 için Q0.0 çıkışını 1'e kuran ve I0.1=1 için Q0.0 çıkışını sıfırlayan program parçasının her üç dildeki karşılıkları gösterilmiş ve geçerli adres alanları belirtilmiştir.



Geçerli adresler: : I, Q, M, SM, T, C, V, S, L [bit]

N: VB, IB, QB, MB, SMB, SB, LB, AC, sabit(1-255), *VD, *AC,*LD [byte]

Şekil 2.22 Kurma-silme işlemine ilişkin LAD, FBD ve STL programları

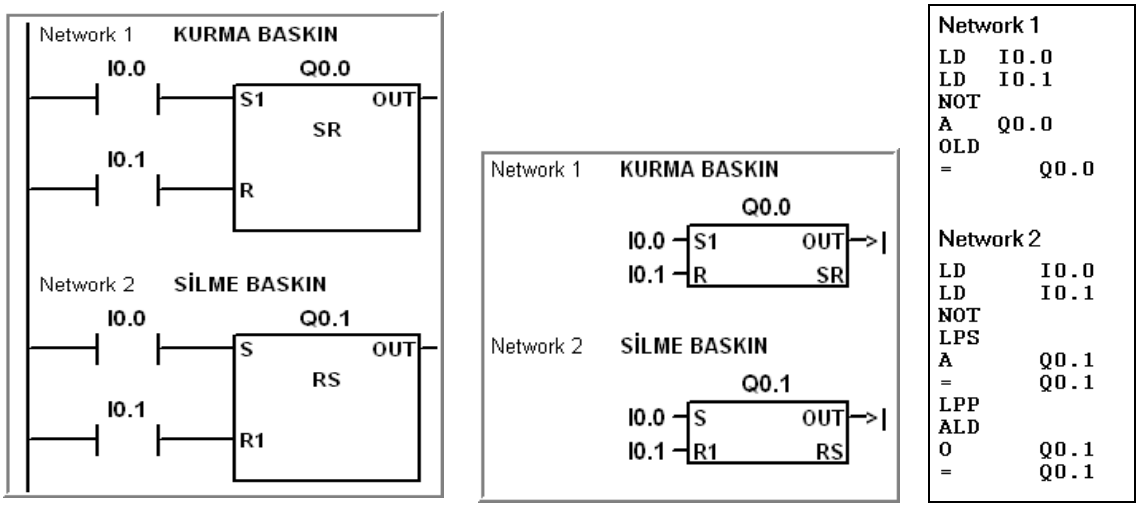


Şekil 2.23 Kurma-silme işlemine ilişkin zamanlama diyagramı

2.5.3 KURMA VEYA SİLME BASKIN İKİ KARARLI İŞLEMLER

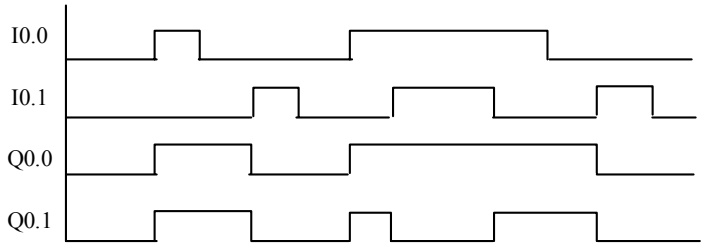
Kurma ve silme işlemlerine ilişkin komutların her ikisinin de etkin yapılması durumunda (S=1, R=1), son işlenen komut geçerli olur. Eğer komutların yazılış sırasından bağımsız olarak, SR=11 için çıkışa

istenen bir değerin atanması isteniyorsa bu komutlar kullanılabilir. Kurma baskın iki kararlı (SR) işlemde, her iki giriş de 1 yapıldığında çıkış 1, silme baskın iki karalı (RS) işlemde her iki giriş de 1 yapıldığında çıkış 0 olur. Bu işlemlere ilişkin bir program örneği Şekil 2.24'te, zamanlama diyagramı Şekil 2.25'te verilmiştir. Burada I0.0=1 ve I0.1=1 için Q0.0=1 olmasını sağlayan kurma baskın ve Q0.1=0 olmasını sağlayan silme baskın bir program parçasının her üç dildeki karşılıkları gösterilmiş ve geçerli adres alanları belirtilmiştir.



Geçerli giriş adresleri: : I, Q, M, SM, T, C, ,V, S, L [bit]
Geçerli çıkış adresleri: : I, Q, M, V, S[bit]

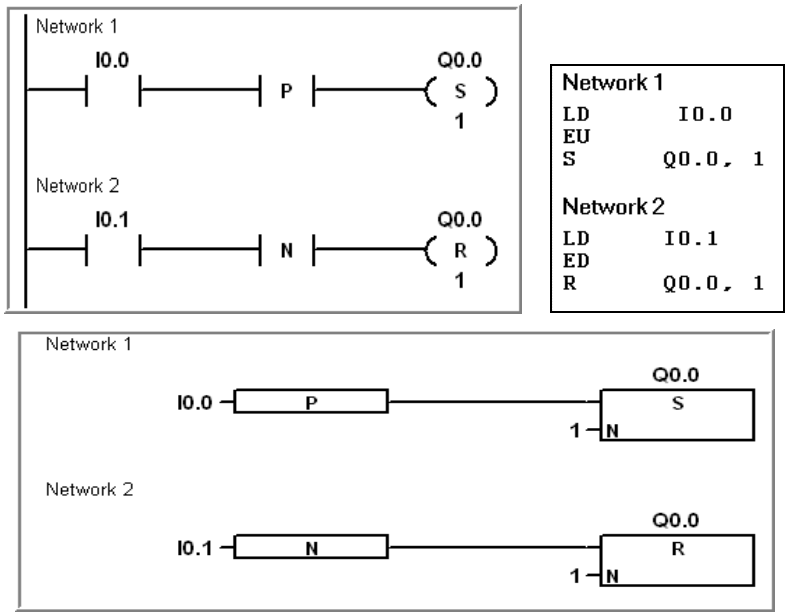
Şekil 2.24 Kurma veya silme baskın iki kararlı işlemlere ilişkin LAD, FBD ve STL programları



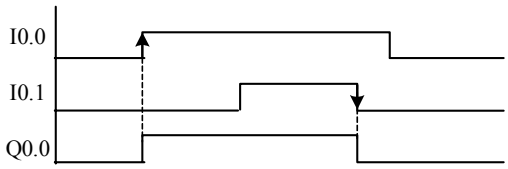
Şekil 2.25 Kurma veya silme baskın iki kararlı işlemlere ilişkin zamanlama diyagramı

2.5.4 ÇIKAN VE DÜŞEN KENARI ALGILAMA İŞLEMLERİ

Bir işaretin yalnız çıkan veya düşen kenarlarını algılamak için giriş işaretinin 0'dan 1'e değişmesi veya 1'den 0'a değişmesi durumunda 1 program tarama süresi mantıksal 1 işareti üreten komutlar kullanılır. Şekil 2.26'da çıkan kenar (P, EU) ve düşen kenar (N, ED) komutlarına ilişkin program parçaları, Şekil 2.27'de ilgili zamanlama diyagramı verilmiştir.



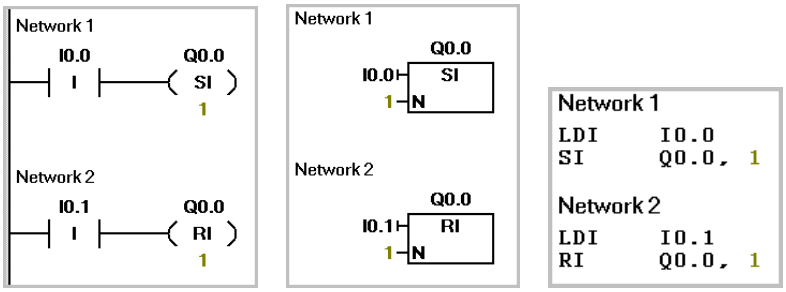
Şekil 2.26 Çıkan ve düşen kenarı algılama işlemlerine ilişkin LAD, STL ve FBD programları



Şekil 2.27 Çıkan ve düşen kenarı algılama işlemlerine ilişkin zamanlama diyagramı

2.5.5 DOĞRUDAN OKUMA VE YAZMA İŞLEMLERİ

PLC olağan çalışma durumunda, girişe bağlı elemanların işaret durumları okuma evresinde giriş görüntü belleğine kaydedilir ve komutlar bu değerlere göre işlem yapar. Çıkışa ilişkin hesaplanan bir değer çıkış görüntü belleğine kaydedilir ve yazma evresinde çıkış birimine gönderilir. Eğer komutun işlendiği andaki giriş değeri ile işlem yapılması ve hesaplanan bir çıkış değerinin anında çıkış birimine gönderilmesi isteniyorsa anında işlem (ivedi işlem) mantıksal komutları (LDI, AI, OI, LDNI, ANI, ONI) kullanılır. Şekil 2.28'de bu komutların kullanımına ilişkin LAD, FBD ve STL program örnekleri verilmiştir. Bu programa göre I0.0=1 ve I0.1=1 yapıldığında Q0.0 çıkış noktası aynı tarama çevrimi içinde önce 1 sonra 0 değerini alır. Eğer ivedi kurma (SI) yerine S Q0.0,1 ve ivedi silme (RI) yerine R Q0.0,1 komutları kullanılırsa (silme baskın) Q0.0 çıkış noktasının 1 değerini almadığı görülür.



Geçerli giriş adresleri: I (bit)
Geçerli çıkış adresleri: Q (bit)

Şekil 2.28 Doğrudan okuma ve yazma işlemine ilişkin komutlar

2.5.6 MANTIKSAL YIĞIN İŞLEMLERİ

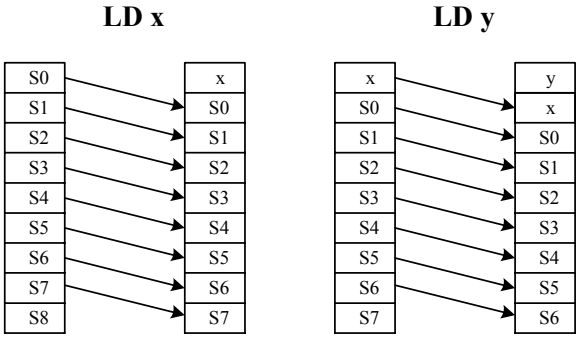
Mantıksal işlem komutları yığın yapısındaki bir bellek üzerinde yapılır. Boyutu 1 bit olan bu bellek 9 yığın katmanından oluşur. Komut ile etkin bir programlama için mantıksal yığın işlemlerinin bilinmesinde yarar vardır.

Mantıksal bir ifadeyi programlamaya başlamak için kullanılan LD, LDN, LDI ve LDNI komutlarının yürütülmesi sırasında, işlenen adresin içeriği yığının üstüne konur, yığındaki diğer alanlar bir basamak aşağı kaydırılır. Örneğin, x ve y simgesel adreslerinin

LD x

LD y

komutları ile art arda işletilmesi durumunda yığın belleğinin içeriği Şekil 2.29'deki gibi değişir. Burada S0, S1, .. S8 yığın katmanlarını göstermektedir.



Şekil 2.29 LD komutu ile yığın belleğinin değişimi

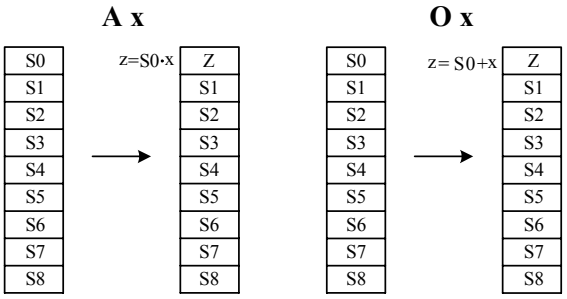
Mantıksal işlem komutları (A, O, AN, ON, AI, OI, ANI, ONI) yürütülürken, işlenen adresin içeriği ile yığın belleğinin üstündeki değer tanımlanan mantıksal işleme göre işlenir ve sonuç yine yığının üstüne konur. Yığın belleğinin diğer alanlarındaki verilerin konumu değişmez. Örneğin, x simgesel adresine ilişkin

A x

veya

O x

komutlarının yürütülmesi durumunda yığın belleğinin içeriği Şekil 2.30'daki gibi değişir.



Şekil 2.30 A ve O komutları işlendiğinde yığın belleği

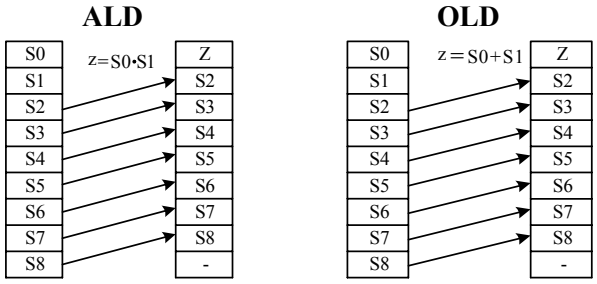
NOT, EU ve ED komutları yığın belleğinin üstündeki veriye uygulanır, işlem sonucu yine yığının üstüne konur. Yığının diğer alanlarında içerik ve konum değişikliği olmaz.

Atama komutu "=" yığının üstündeki verinin içeriğini belirtilen adrese aktarır.

Kurma ve silme komutları (S, R) yığının üstündeki verinin içeriği 1 değerinde iken işlevini yerine getirir.

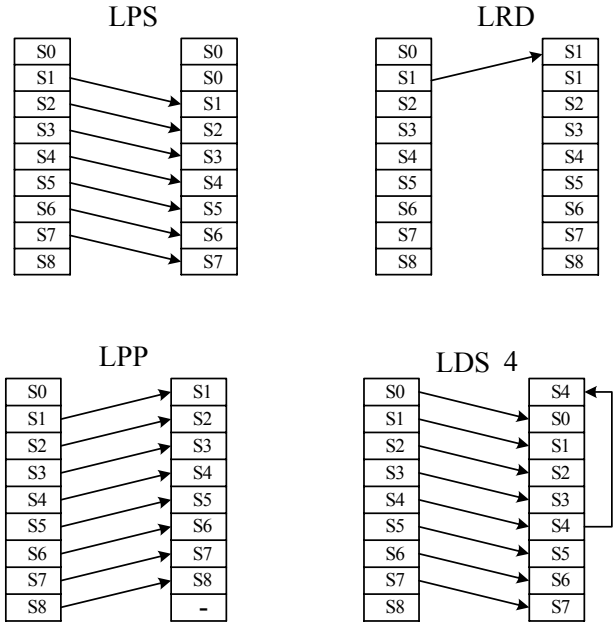
Yığın belleğinin katmanları arasında ALD, OLD, LPS, LPP, LRD ve LDS komutları kullanılarak çeşitli işlemler yapılabilir.

ALD ve OLD komutları yığının üstündeki (S0) ve birinci basamaktaki (S1) verilere uygulanır, işlem sonucu yığının üstüne yazılır, yığının diğer alanları bir basamak yukarı kaydırılır (Şekil 2.31).



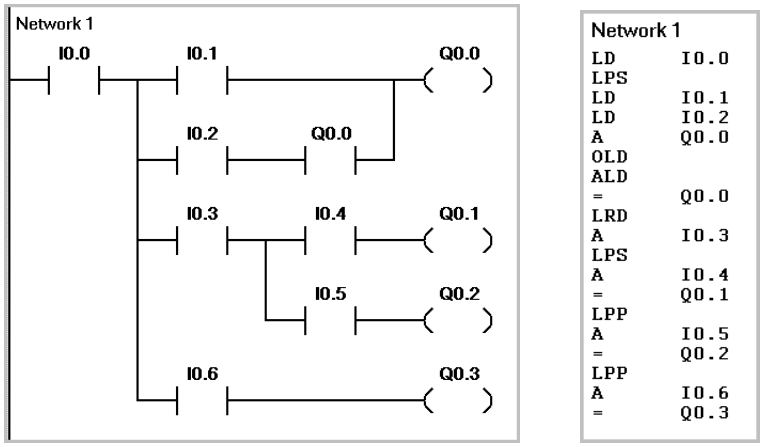
Şekil 2.31 ALD ve OLD komutları işlendiğinde yığın belleği

LPS, LRD, LPP ve LDS komutları yürütüldüğünde yığın belleği Şekil 2.32'de gösterildiği gibi değişir.

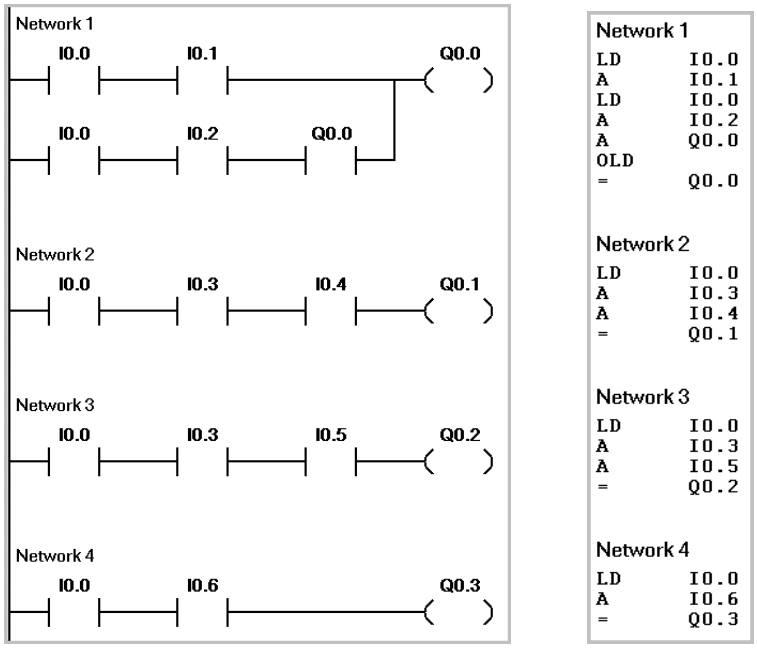


Şekil 2.32 LPS, LRD, LPP ve LDS komutları işlendiğinde yığın belleği

Yığın işlem komutları kullanılarak aynı işlevi sağlayan bir program parçası daha az komutla gerçeklenebilir. Örneğin, Şekil 2.33'te verilen merdiven mantığı programının gerçeklenmesi için sağda gösterilen komut dizisi yazılır. Burada verilen merdiven mantığı başka biçimde düzenlenerek yığın işlem komutları kullanılmadan da gerçeklenebilir. Ancak, bu tür bir gerçekleme genellikle daha çok komut gerektirir. Şekil 2.33'te verilen merdiven mantığı programına ilişkin eşdeğer devreler çeşitli şekillerde oluşturulabilir. Örneğin, Şekil 2.34'te bu tür bir devre gösterilmiştir. Bu programda yalnız OLD yığın işlem komutu kullanılmıştır. Genel olarak mantık fonksiyonları çarpımların toplamı veya toplamların çarpımı biçimine getirilerek, yığın işlem komutlarından ALD ve OLD kullanılarak gerçeklenebilir.



Şekil 2.33 Yığın belleği işlem komutları gerektiren bir program örneği



Şekil 2.34 ALD ve OLD ile gerçeklenen bir program örneği

2.6 ZAMANLAMA İŞLEMLERİ

Zamanlama işlemleri

- Gecikmeli kapatan zamanlayıcı (On-delay timer, TON)
- Kalıcı gecikmeli kapatan zamanlayıcı (retentive on-delay timer, TONR)
- Gecikmeli açan (Off-Delay Timer, TOF)

zamanlayıcı türleri ile gerçeklenir.

Bir zamanlayıcı kaydedicisi adresi aynı olan 16 bitlik veri (içerik) ve 1 bitlik durum bitinden oluşur. Zamanlayıcı komutu yürütüldüğünde zamanlayıcı içeriği belirli zaman aralıkları (1ms, 10ms veya 100ms) ile artar. Zamanlayıcı içeriği PT karşılaştırma değerine eşit veya büyük olduğunda zamanlayıcı durum işaret biti değişir. Karşılaştırma işlemi (zamanlayıcının güncelleştirilmesi)

1 ms artımlı zamanlayıcılarda PLC tarama çevriminden bağımsız olarak her 1 ms'de,

10 ms artımlı zamanlayıcılarda tarama çevriminin başında,

100 ms artımlı zamanlayıcılarda komut yürütüldüğünde gerçeklenir.

TON ve TOF türü zamanlayıcı adresleri ve artım değerleri

T32, T96 1ms

T33-T36, T97-T100 10ms

T37-T63, T101-T255 100ms

TONR türü zamanlayıcı adresleri ve artım değerleri

T0, T64 1ms

T1-T4, T65-T68 10ms

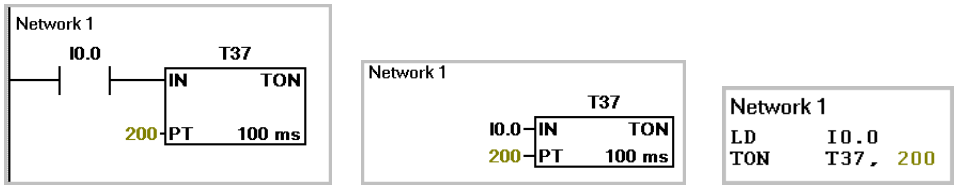
T5-T31, T69-T95 100ms

biçiminde değişir.

Karşılaştırma değeri (PT) sözcük (16 bit) boyutunda 1 ile 32767 tamsayı değerleri arasında değişir; PT değeri tamsayı sabit ya da 16 bitlik bir veri adresi (VW, T, C, IW, QW,...) ile tanımlanabilir.

2.6.1 GECİKMELİ KAPATAN ZAMANLAYICI (TON)

Gecikmeli kapatan zamanlayıcının LAD, FBD ve STL dilinde programlanması, geçerli veri ve zamanlayıcı adresleri Şekil 2.35'te, çalışma özelliklerine ilişkin zamanlama diyagramı Şekil 2.36'da verilmiştir.

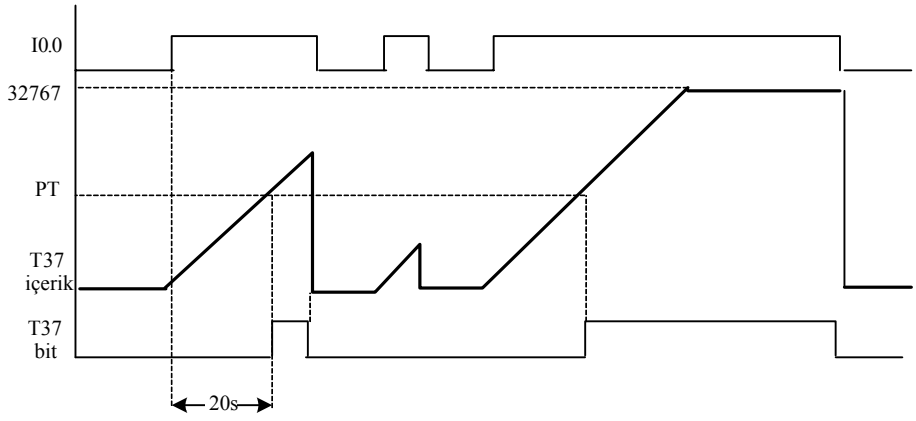


Geçerli veri adresleri:
IN: I, Q, M, SM, T, C, V, S, L [BOOL]
PT: VW, IW, QW, MW, SW, SMW, LW, AIW, T, C, AC, sabit, *VD, *LD, *AC [INT]

Geçerli zamanlayıcı adresleri:

T32, T96	1ms
T33-T36, T97-T100	10ms
T37-T63, T101-T255	100ms

Şekil 2.35 Gecikmeli kapatan zamanlayıcı komutları ve geçerli adresler



Şekil 2.36 Gecikmeli kapatan zamanlayıcıya ilişkin zamanlama diyagramı

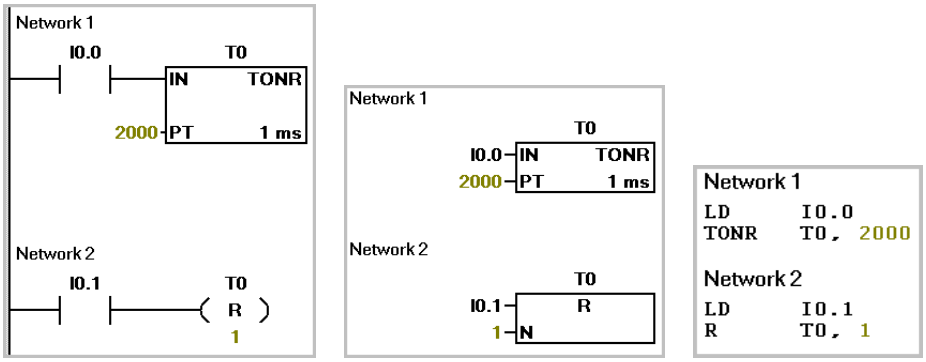
Zamanlayıcı etkin yapıldığında (IN=1) zamanlayıcı içeriği kullanılan zamanlayıcı adresine bağlı olarak belirli zaman değerlerinde (1ms, 10ms , 100ms) artar. Zamanlayıcı içeriği PT değerine eşit veya büyük olduğunda zamanlayıcı durum işareti 1 olur. Zamanlayıcı içeriği ve durum işareti giriş sıfırlandığında (IN=0) sıfırlanır. Karşılaştırma (PT: preset time) için geçerli en büyük değer 32767 olduğundan en uzun gecikme süreleri, 1ms artımlı zamanlayıcılar ile 32.767 saniye, 10ms artımlı zamanlayıcılar ile 327.67 saniye ve 100 ms artımlı zamanlayıcılar ile 3276.7 saniye olur.

Gecikmeli kapatan zamanlayıcı (TON), PLC işlevsiz duruma geçtiğinde (STOP veya enerji kesilmesi durumu) zamanlayıcı içeriği ve biti sıfırlanır. Bu tür zamanlayıcı içerikleri kalıcı (retentive) yapılamaz. Enerji kesilmesi durumunda zamanlayıcı içeriğinin korunması isteniyorsa kalıcı gecikmeli kapatan (TONR) zamanlayıcı kullanılır ve 2.2.4 alt bölümünde değinildiği gibi kalıcı olarak tanımlanır. TON zamanlayıcısının içeriği bir program parçası yazılarak da korunabilir. Ancak, bu işlem için içeriği kalıcı olarak tanımlanmış bir VWx değişken bellek alanı gerekir. Her taramada sayıcı içeriği bu alana yüklenir; enerji tekrar geldiğinde VWx değişken bellek alanındaki veri TON zamanlayıcısına aktarılır. Aktarma işlemi için daha sonra tanıtılacak olan MOVW komutu kullanılır. İşlemciye enerji verilmesi durumu alt başlık 2.2.2'de tanıtılan SM0.3 özel bellek biti değerlendirilerek belirlenir.

2.6.2 GECİKMELİ KAPATAN KALICI ZAMANLAYICI (TONR)

Gecikmeli kapatan kalıcı zamanlayıcıya ilişkin LAD, FBD ve STL programları, geçerli veri ve zamanlayıcı adresleri Şekil 2.37'de, çalışma özelliklerine ilişkin zaman diyagramı Şekil 2.38'de verilmiştir.

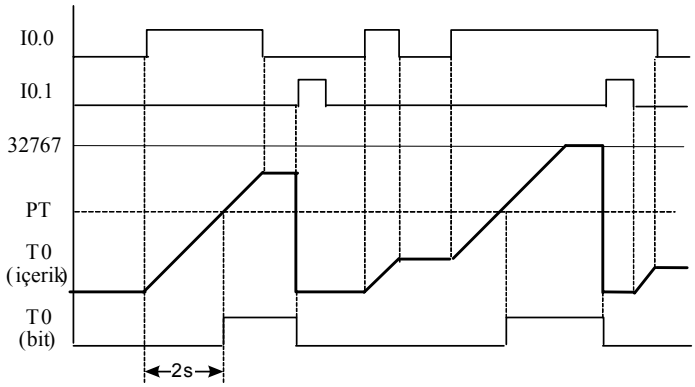
TONR zamanlayıcısı içeriği, zamanlayıcı komutunu etkin yapan koşul (I0.0=1) kaldırılsa da silinmez. Zamanlayıcı içeriğini sıfırlamak için silme komutu (R Tx,1) kullanılır. Eğer, TONR kalıcı olarak tanımlanmamışsa, enerji kesilmelerinde de zamanlayıcı içeriği sıfırlanır.



Geçerli veri adresleri:
IN: I, Q, M, SM, T, C, V, S, L [BOOL]
PT: VW, IW, QW, MW, SW, SMW, LW, AIW, T, C, AC, sabit, *VD, *LD, *AC [INT]

Geçerli zamanlayıcı adresleri:
T0, T64 1ms
T1-T4, T65-T68 10ms
T5-T31, T69-T95 100ms

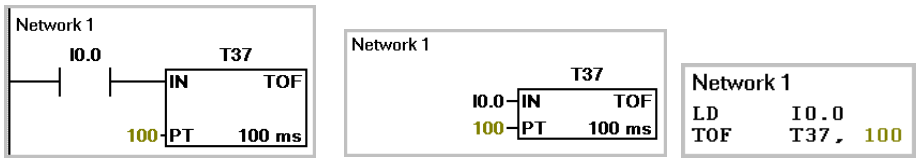
Şekil 2.37 Gecikmeli kapatan kalıcı zamanlayıcı komutları ve geçerli adresler



Şekil 2.38 Gecikmeli kapatan kalıcı zamanlayıcıya ilişkin zamanlama diyagramı

2.6.3 GECİKMELİ AÇAN ZAMANLAYICI (TOF)

Gecikmeli açan zamanlayıcı (Off-Delay) giriş işaretinin 0'dan 1'e değişimi ile uyarılan ve giriş işaretinin 1'den 0'a geçişi ile içeriği artan bir zamanlayıcı olup bu zamanlayıcıya ilişkin LAD, FBD ve STL programları Şekil 2.39'da çalışma özelliklerini tanımlayan zamanlama diyagramı Şekil 2.40'ta gösterilmiştir.



Geçerli veri adresleri:

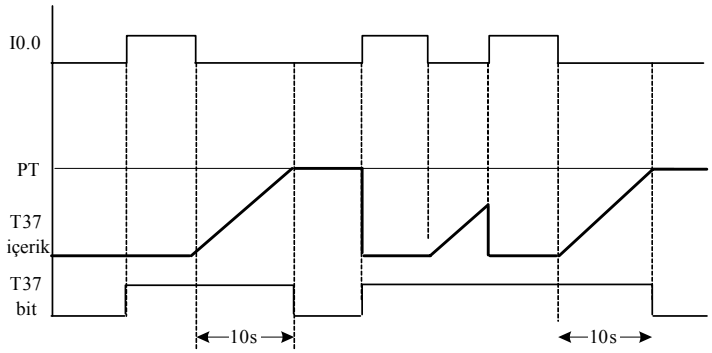
IN: I, Q, M, SM, T, C, V, S, L [BOOL]

PT: VW, IW, QW, MW, SW, SMW, LW, AIW, T, C, AC, sabit, *VD, *LD, *AC [INT]

Geçerli zamanlayıcı adresleri:

T32, T96	1ms
T33-T36, T97-T100	10ms
T37-T63, T101-T255	100ms

Şekil 2.39 Gecikmeli açan zamanlayıcı komutları ve geçerli adresler



Şekil 2.40 Gecikmeli açan zamanlayıcıya ilişkin zamanlama diyagramı

2.7 SAYMA İŞLEMLERİ

Saymaya bağlı kumanda işlemlerinin gerçeklenmesinde üç tür sayıcı kullanılır:

- İleri sayıcı (CTU)
- Geri sayıcı (CTD)
- İleri-geri sayıcı (CTUD)

Sayıcı kaydedicileri (C0, C1, ..., C255) aynı simge ile gösterilen 16 bit boyutunda sayıcı içeriği ve 1 bitlik sayıcı durumunu gösteren işaret alanından oluşur.

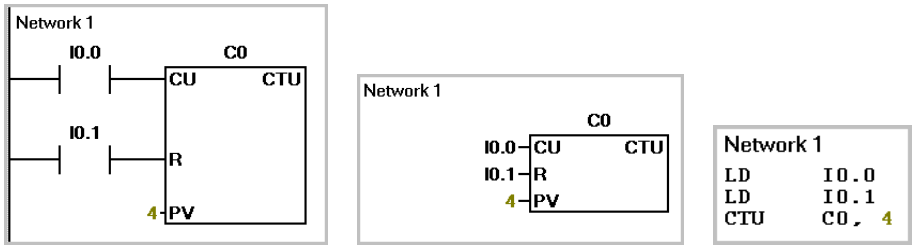
İleri sayıcı (CTU) ve ileri-geri sayıcıda (CTUD) sayıcı içeriği karşılaştırma değerine eşit veya büyük olduğunda sayıcı işaret biti 1 olur. Geri sayıcıda (CTD) sayıcı içeriğine 1 ile 32767 arasında bir değer yüklenir; sayıcı içeriği 0 olduğunda sayıcı işaret biti 1 olur.

Bir sayıcı kaydedicisi ileri ve geri sayıcı olarak kullanıldığında içeriği en az 0 en çok 32767; ileri-geri sayıcı olarak kullanıldığında içeriği en az –32768, en çok 32767 olur. İleri-geri sayıcıda sayıcı içeriği en küçük değerinde (–32768) iken bir geri sayma işareti geldiğinde sayıcı içeriği en büyük değeri (32767) alır. Sayıcı işareti en büyük değerinde iken bir ileri sayma girişi geldiğinde sayıcı içeriği en küçük değeri alır.

Sayıcılar kalıcı (retentive) olarak tanımlanabilir (alt başlık 2.2.4). Bu durumda enerji kesilmesi durumunda da sayıcı içerikleri korunur. Eğer enerji kesilmelerinde sayıcı içeriklerinin korunması istenmiyorsa kalıcı olma özellikleri kaldırılır veya PLC'ye enerji verilmesi durumunda bir tarama süresince etkin olan SM0.1 özel bellek biti kullanılarak kalıcı olarak tanımlanmış sayıcı içeriği sıfırlanır. Bunun için silme (R Cx, 1) komutu kullanılır.

2.7.1 İLERİ SAYICI

İleri sayıcıya -ilişkin LAD, FBD ve STL programları ve geçerli adresler Şekil 2.41'de, çalışma özelliklerini tanımlayan zamanlama diyagramı Şekil 2.42'de gösterilmiştir. İleri sayıcıda, CU girişindeki işaretin 0'dan 1'e değişmesi ile sayıcı içeriği 1 artar. Sayıcı içeriği PV değerine eşit ve büyük olduğunda sayıcı biti 1 olur. Sayıcı R girişindeki işaretin 1 olması durumunda sayıcı içeriği ve biti sıfırlanır. Sayıcı içeriği en az 0, en çok 32767 olabilir.



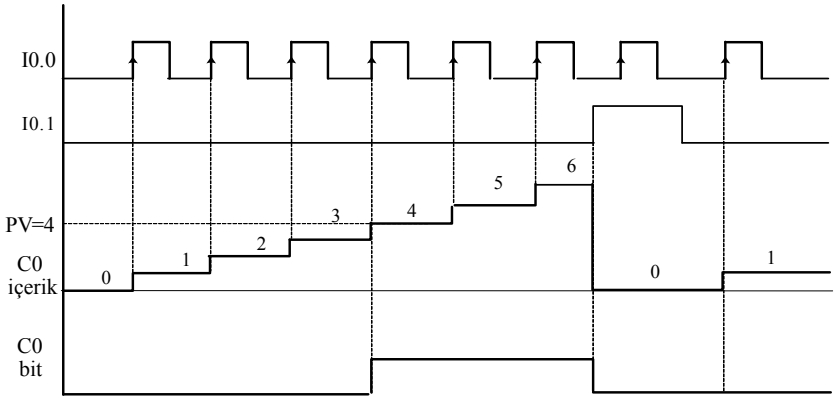
Geçerli veri adresleri:

CU: R: I, Q, M, SM, T, C, V, S, L [BOOL]

PV: VW, IW, QW, MW, SW, SMW, LW, AIW, T, C, AC, sabit, *VD, *LD, *AC [INT]

Geçerli sayıcı adresleri**:** C0-C255

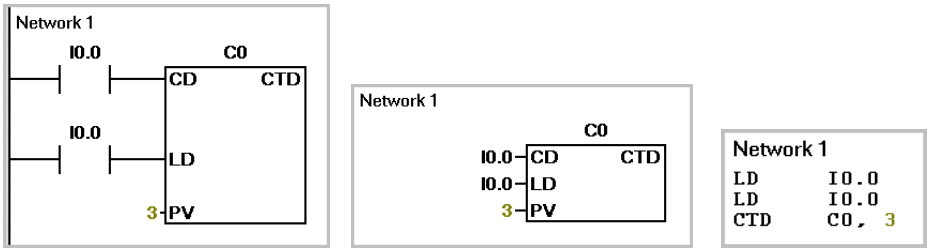
Şekil 2.41 İleri sayıcıya ilişkin komutlar ve geçerli adresler



Şekil 2.42 İleri sayıcının çalışmasına ilişkin zamanlama diyagramı

2.7.2 GERİ SAYICI

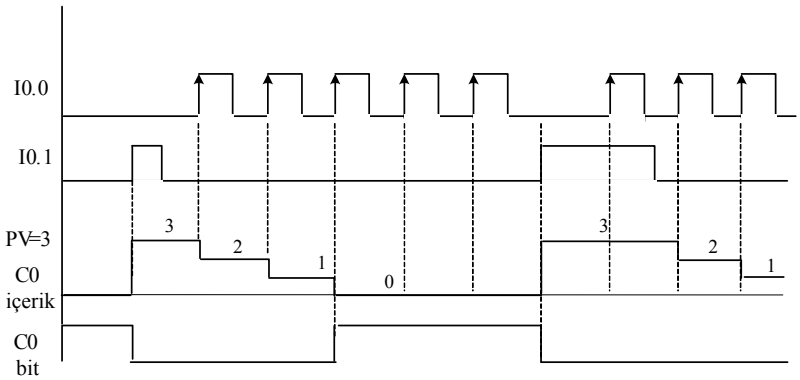
Geri sayıcıya -ilişkin LAD, FBD ve STL programları ve geçerli adresler Şekil 2.43'te, çalışma özelliklerini tanımlayan zamanlama diyagramı Şekil 2.44'de gösterilmiştir. Geri sayıcıda, PV değerine 1 ile 32767 arasında bir değer yüklenir; CD girişindeki işaretin 0'dan 1'e değişmesi ile sayıcı içeriği 1 azalır. Sayıcı içeriği 0 olduğunda sayıcı biti 1 olur. Sayıcı LD girişindeki işaretin 1 yapılması durumunda sayıcı içeriğine PV değeri yüklenir ve sayıcı biti sıfırlanır. Sayıcı, LD girişindeki işaret 0 yapıldığında işler duruma geçer. Sayıcı içeriği en az 0, en çok 32767 olabilir.



Geçerli veri adresleri:
CD: LD: I, Q, M, SM, T, C, V, S, L [BOOL]
PV: VW, IW, QW, MW, SW, SMW, LW, AIW, T, C, AC, sabit, *VD, *LD, *AC [INT]

Geçerli sayıcı adresleri: C0-C255

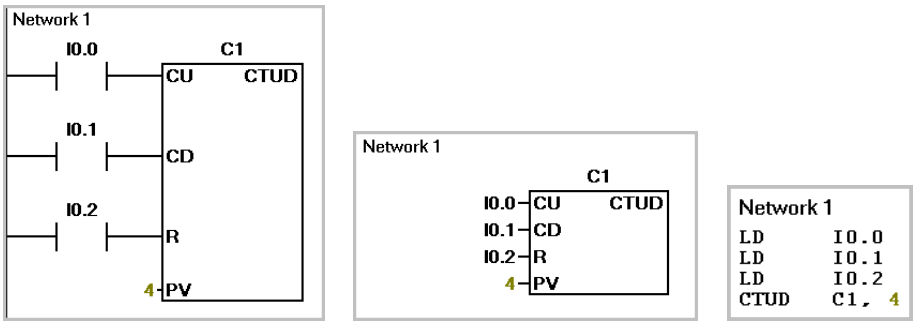
Şekil 2.43 Geri sayıcı komutları ve geçerli adresler



Şekil 2.44 Geri sayıcıya ilişkin zamanlama diyagramı

2.7.3 İLERİ-GERİ SAYICI

İleri-geri sayıcıya -ilişkin LAD, FBD ve STL programları ve geçerli adresler Şekil 2.45'te, çalışma özelliklerini tanımlayan zamanlama diyagramı Şekil 2.46'da gösterilmiştir. Bu sayıcı türünde CU girişindeki işaretin 0'dan 1' değişmesi ile sayıcı içeriği artar, CD girişindeki işaretin 0'dan 1'e değişmesi ile sayıcı içeriği azalır. R girişindeki işaretin 1 yapılması ile sayıcı içeriği ve biti sıfırlanır. İleri - geri sayıcı içeriği en az –32768, en çok 32767 olur. İleri-geri sayıcıda sayıcı içeriği en küçük değerinde (–32768) iken bir geri sayma işareti geldiğinde sayıcı içeriği en büyük değeri (32767), sayıcı içeriği en büyük değerinde iken (32767) bir ileri sayma işareti geldiğinde en küçük değeri alır.



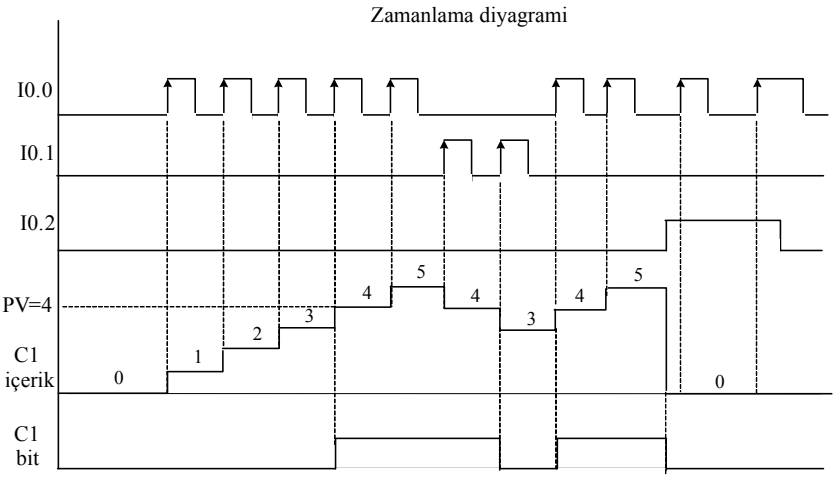
Geçerli veri adresleri:

CU, CD: R: I, Q, M, SM, T, C, V, S, L [BOOL]

PV: VW, IW, QW, MW, SW, SMW, LW, AIW, T, C, AC, sabit, *VD, *LD, *AC [INT]

Geçerli sayıcı adresleri**:** C0-C255

Şekil 2.45 İleri-geri sayıcı komutları ve geçerli adresler



Şekil 2.46 İleri-geri sayıcıya ilişkin zamanlama diyagramı

2.8 KARŞILAŞTIRMA İŞLEMLERİ

Kumanda devrelerinde karşılaştırma gerektiren çeşitli boyutlardaki veriler; büyüklük, küçüklük veya eşitlik ölçütlerine göre karşılaştırılarak değerlendirilebilir. Bu amaçla yapılan karşılaştırma işlemlerinde 8 bit, 16 bit ve 32 bit boyutundaki veriler için ayrı komutlar kullanılır.

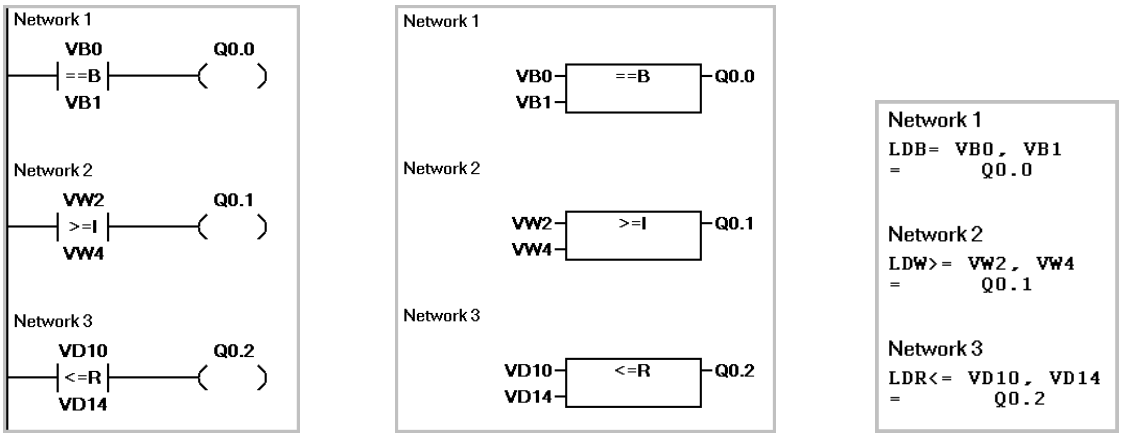
Karşılaştırma işlemlerinde boyutu 8, 16 veya 32 bit olan veriler

- Eşit (= =)
- Eşit-değil (<>)
- Büyük (>)
- Büyük-eşit (>=)
- Küçük (<)
- Küçük-eşit (<=)

ölçütlerine göre değerlendirilir; ölçüt sağlanıyorsa sonuç (yığın belleğinin üstü) mantıksal 1 sağlanmıyorsa 0 olur.

Karşılaştırma işlemleri 8 bitlik verilerde işaretsiz tamsayı, diğer verilerde işaretli tamsayı veya gerçek (reel) sayılar üzerinde yapılır.

Eşitlik ölçütüne göre çeşitli boyutlardaki verilerin karşılaştırılmasına ilişkin program ve geçerli adresler Şekil 2.47'de verilmiştir. Diğer karşılaştırma komutları, eşitlik ölçütü (= =) yerine eşit-değil (<>) , büyük (>), büyük-eşit (>=) , küçük (<) veya küçük-eşit (<=) ölçütleri yazılarak; LAD ve FBD dilinde ise ilgili karşılaştırma ölçütü simgesi taşıyan şekiller kullanılır.



Geçerli giriş adresleri:

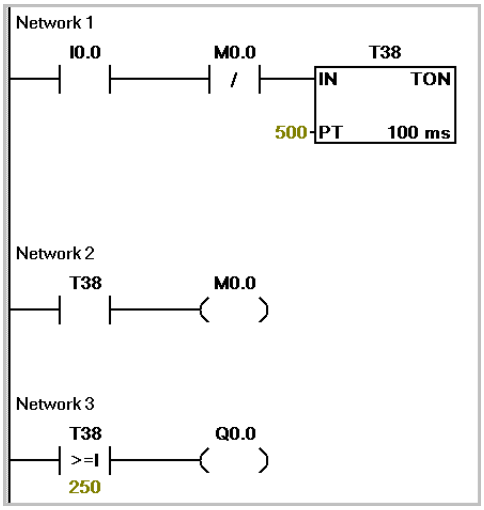
IB, QB, MB, SMB, VB, SB, LB, AC, sabit, *VD, *LD, *AC	(8 bit)
IW, QW, MW, SW, SMW, T, C, VW, LW, AIW, AC, sabit, *VD, *LD,*AC	(16 bit)
ID, QD, MD, SD, SMD, VD, LD, HC, AC, sabit, *VD, *LD, *AC	(32 bit)
ID, QD, MD, SD, SMD, VD, LD, AC, sabit, *VD, *LD, *AC	(reel)

Geçerli çıkış adresleri:

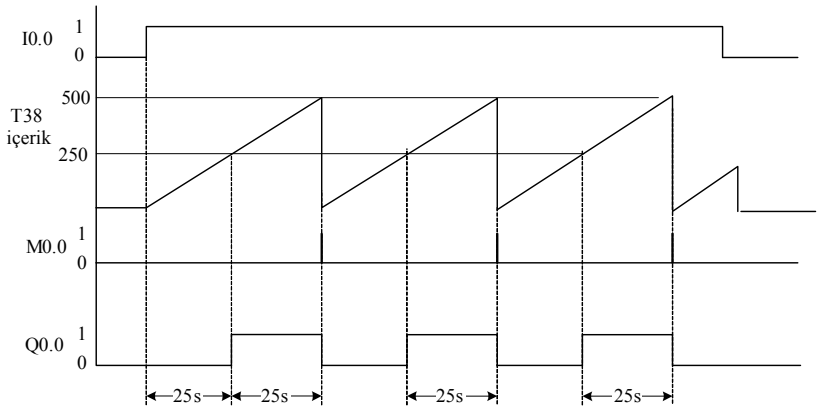
I, Q, M, SM, T, C, V, S, L (bit)

Şekil 2.47 Karşılaştırma işlemlerine ilişkin komutlar

Karşılaştırma komutları, zamanlayıcı ve sayıcılar ile birlikte kullanılarak çeşitli kumanda devreleri gerçeklenebilir. Şekil 2.48'de karşılaştırma komutunun bir zamanlayıcı ile birlikte kullanımına ilişkin bir program örneği, Şekil 2.49'da ilgili zamanlama diyagramı verilmiştir. Bu programa göre, I0.0=1 yapıldığında Q0.0 çıkışı 25s 0 ve 25s 1 değerinde kalarak sürekli değişir.



Şekil 2.48 Karşılaştırma komutu ile zamanlama işlemi



Şekil 2.49 Zamanlama diyagramı

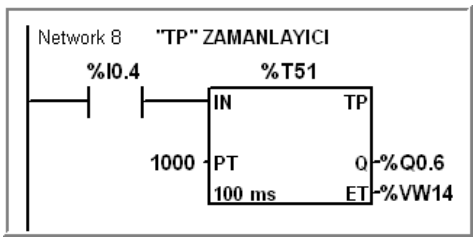
Edited by Foxit Reader
Copyright(C) by Foxit Software Company,2005-2008
For Evaluation Only.

2.9 IEC 1131-3 PROGRAMLAMA MODU

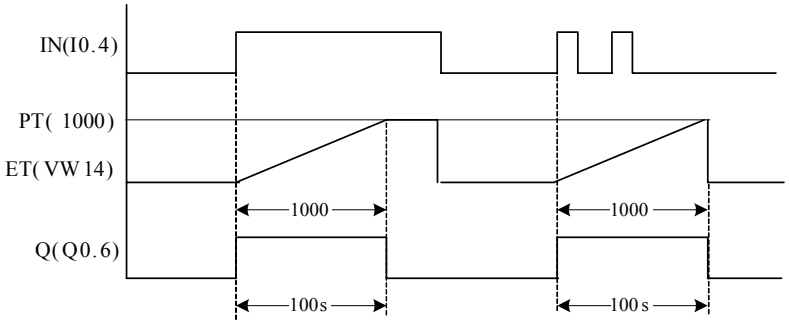
PLC'lerin programlanmasında ortak dil kullanımını amaçlayan bir girişimin sonucu ortaya çıkan IEC 1131-3, uluslararası bir programlama standardı olarak çoğu üretici kuruluş tarafından benimsenmiştir.

Simatic S7-200 CPU22x işlemcili PLC'ler IEC 1131-3 standart dil kurallarına göre de programlanabilir. Bu programlama modu seçildiğinde (<Tools>→<Option>→<General> IEC 1131-3) bazı mantıksal, zamanlama, sayma ve karşılaştırma komutlarının kullanımı ve işlevleri değişir. Örneğin mantıksal işlemlerde anında işlem (LDI, LDNI, AI, ANI, OI, ONI, SI, RI =I,) komutları yoktur. Zamanlayıcı türlerinden TONR programlanamaz, ancak TON ve TOF zamanlayıcısının yanı sıra darbe zamanlayıcısı (TP) programlanabilir. Sayıcılardan ileri-geri sayıcı farklı biçimde programlanır. Karşılaştırma komutlarının sayısı daha azdır. Ancak, karşılaştırma işleminde 8, 16 ve 32 bitlik tamsayı ve gerçel sayı biçimdeki veriler için aynı komutlar (eşit <EQ> , eşit değil <NE>, büyük-eşit<GE>, küçük-eşit<LE>, büyük <GT> ve küçük<LT> kullanılır. Temel kumanda işlemlerine ilişkin bu komutlar dışındaki bir çok komutun işlevi ve kullanımı değişiklik gösterir. SIMATIC programlama modunda geçerli olan gerçek zaman saati, kesmeli çalışma, iletişim, veri tablosu oluşturma gibi işlemlere ilişkin komutlar IEC programlama modunda kullanılamaz. MicroWIN 32 yazılımında hangi komutların geçerli olduğu komut ağacında görülebilir. Geçerli olmayan komutlar üzerinde eşkenar dörtgen biçiminde kırmızı bir simge bulunur.

IEC 1131-3 programlama modunda anahtarlama işlemleri aynı biçimde programlanır. Zamanlama işlemlerinde, TON ve TOF zamanlayıcıları aynı biçimde programlanır; TONR zamanlayıcısı programlanamaz. Ancak, bu programlama modunda, SIMATIC programlama modunda bulunmayan değişik özelliklerdeki bir zamanlayıcı vardır. Darbe zamanlayıcısı (TP) olarak tanımlanan bu zamanlayıcıya ilişkin program örneği Şekil 2.50'de ilgili zamanlama diyagramı Şekil 2.51'de gösterilmiştir.

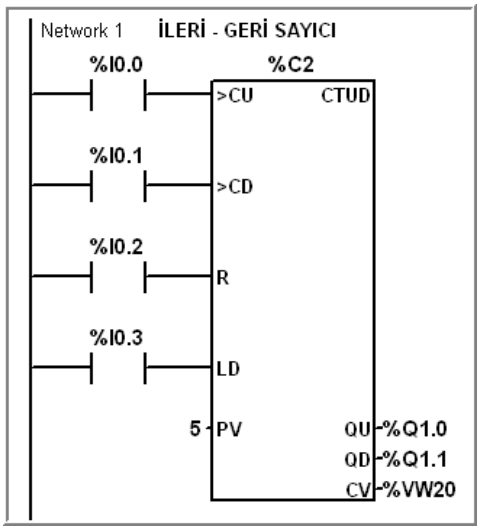


Şekil 2.50 IEC program modunda Darbe zamanlayıcı



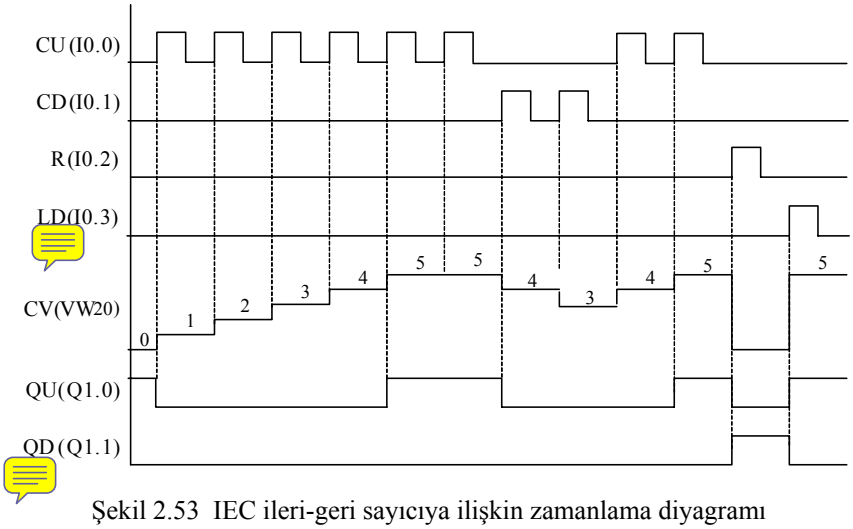
Şekil 2.51 Darbe zamanlayıcısına ilişkin zamanlama diyagramı

Sayma işlemlerine ilişkin ileri sayıcı (CTU) ve geri sayıcı (CTD) aynı şekilde programlanır. Ancak, ileri-geri sayıcının (CTUD) programlanması ve özellikleri değişiktir. Bu sayıcı hem SIMATIC CTUD sayıcısı hem de CTD sayıcılarının özelliklerini taşır. Şekil 2.52'de CTUD(IEC) ileri-geri sayıcısının program örneği ve Şekil 2.53'te ilgili zamanlama diyagramı verilmiştir. Zamanlama diyagramı incelendiğinde, bu sayıcının tüm sayıcı türlerinin (CTU, CTD ve CTUD) işlevlerini sağladığı görülür.



Şekil 2.52 IEC ileri-geri sayıcıya ilişkin LAD programı

Edited by Foxit Reader
Copyright(C) by Foxit Software Company,2005-2008
For Evaluation Only.



Şekil 2.53 IEC ileri-geri sayıcıya ilişkin zamanlama diyagramı

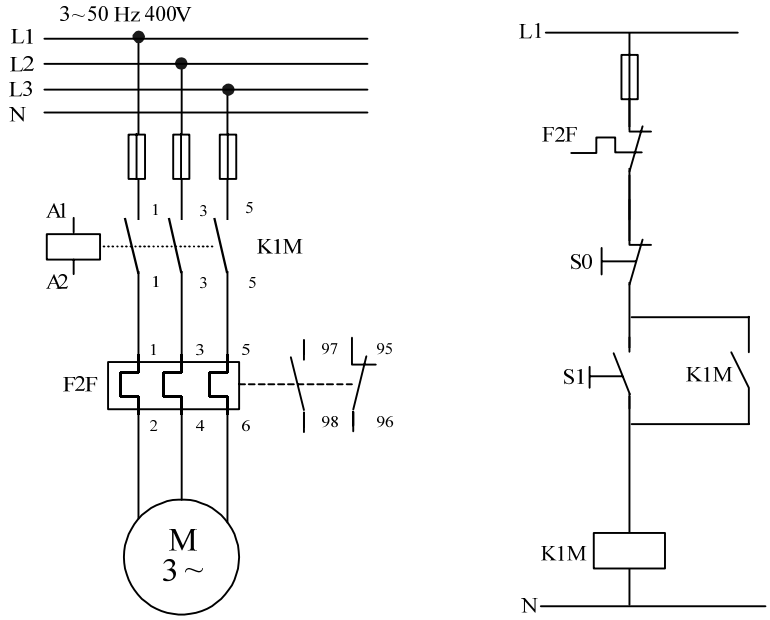
BÖLÜM 3
S7-200 CPU 22X
TEMEL KUMANDA UYGULAMALARI

3.1 GİRİŞ

Endüstriyel kumanda devrelerine ilişkin birçok temel uygulama şimdiye kadar tanıtılan komutlar kullanılarak gerçeklenebilir. Bu devrelerin PLC ile gerçeklenmesinde, kumanda devresi ile merdiven mantığı programı arasındaki benzerlikten yararlanılabilir. PLC ile gerçeklenen kumanda devresinin öncelikle elektrikli kumanda devresinin tüm işlevlerini yerine getirmesi istenir. Ancak, PLC kullanımının getirdiği yeni olanaklardan yararlanarak kumanda sistemi daha işlevsel biçimde gerçeklenebilir. Bu bölümde temel kumanda uygulamalarının PLC ile gerçeklenmesine ilişkin çeşitli örnekler verilecektir. Elektrikli kumanda devreleri ile yapılması olanaksız karmaşık kumanda devreleri, otomasyon için gerekli verilerin üretilmesi ve iletimi gibi uygulamalar PLC kullanılarak kolayca yapılabilir.

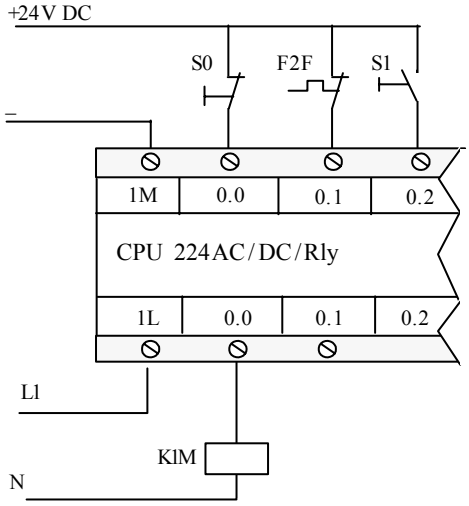
3.2 KISA DEVRE ASENKRON MOTORUN KUMANDASI

Bir kısa devre asenkron motorun devreye alınması ve devreden çıkartılmasına ilişkin güç ve kumanda devreleri Şekil 3.1'deki verilebilir. Bu devrede S0 durdurma düğmesi, F2F aşırı akım rölesi ve S1 başlatma düğmesidir. Bu sistemde, motorun devreye girmesi ve devrede kalması için K1M kontaktör bobinine sürekli enerji verilmelidir. Bunun sağlanması için L1 hattında enerji olmalı, kumanda iletkeni açık devre (kopuk) veya sigorta atmamış olmalı, F2F uyarılmamış (kontağı kapalı) olmalı, S0 düğmesine basılmamış (kontağı kapalı) olmalı ve S1 düğmesine bir kez basılmış olmalıdır. Bu devrede L1, F2F ve S0 elemanlarına ilişkin iletkenlerin, bir sorun nedeniyle, açık devre (kopuk) olması durumunda motor devreye girmez ya da devrede ise devreden çıkar.



Şekil 3.1 Kısa devre asenkron motora ilişkin güç ve kumanda devreleri

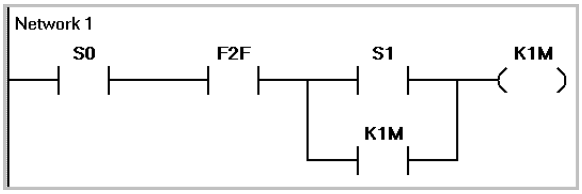
Bu kumanda devresinin PLC ile gerçeklenmesi durumunda en yalın PLC giriş çıkış bağlantıları Şekil 3.2'deki gibi yapılabilir. Burada PLC güç kaynağı ve çıkış elemanı koruma bağlantıları gösterilmemiştir.



Şekil 3.2 PLC giriş-çıkış bağlantıları

Şekil 3.2'deki bağlantı şemasına göre Şekil 3.1'deki kumanda devresi ile aynı işlevi sağlayan merdiven mantığı (LAD) ve komut (STL) programı Şekil 3.3'deki gibi verilebilir.

Symbol	Address
S0	I0.0
F2F	I0.1
S1	I0.2
K1M	Q0.0

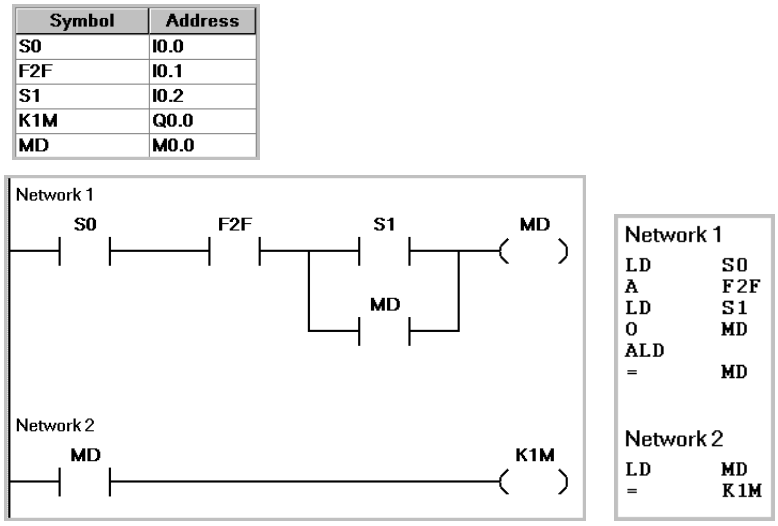



```
Network 1
LD      S0
A       F2F
LD      S1
O       K1M
ALD
=       K1M
```

Şekil 3.3 Sembol tablosu, merdiven mantığı ve komut programı

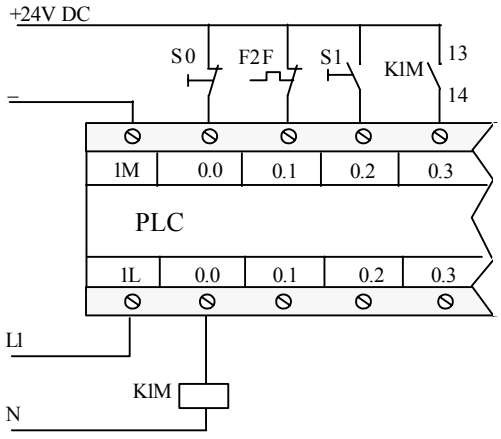
Şekil 3.2'deki bağlantı şeması ve Şekil 3.3'deki programa göre çalışan PLC'de Şekil 3.1'de verilen kumanda devresinin tüm özellikleri sağlanır. Şöyle ki, S0 ve F2F elemanları kapalı kontakları üzerinden PLC'ye bağlanmış olduğundan bu elemanlara ilişkin bağlantı iletkenlerinde herhangi bir sorun (açık devre) olduğunda programa göre Q0.0=0 olur ve buna bağlı olarak K1M kontaktörünün enerjisi kesilir. Eğer S0 ve F2F elemanları açık kontaklar üzerinden PLC'ye bağlanır, program buna göre düzenlenirse (bu durumda merdiven mantığı programında S0 ve F2F açık kontakları yerine kapalı kontakları kullanılır) bu koşulun sağlanmayacağı görülebilir.

Motor çalışırken enerjinin kesilmesi ve tekrar gelmesi durumunda, PLC çıkış görüntü belleği sıfırlandığından, Q0.0 çıkışı da 0 olur ve bu nedenle motor devreye girmez. Şekil 3.1'deki kumanda devresi için de benzer durum geçerlidir. Genel olarak motor kumanda sistemlerinde, enerji kesintisi sonrası motorun kendiliğinden devreye girmesi uygun değildir. Ancak, bazı uygulamalarda motorun devreye girmesi istenebilir. Şekil 3.1'de verilen kumanda devresi ile bu koşulun sağlanması ek devre ve elemanlar gerektirir. PLC ile gerçeklemede ise yalnız program üzerinde değişiklik yapmak yeterlidir. Bunun için motor durumunun kaydedildiği bir yardımcı bellek kullanılır ve bu bellek alanı alt başlık 2.3.4'te açıklandığı gibi kalıcı (retentive) olarak tanımlanır. Bu yeni durumda sembol tablosu, merdiven mantığı ve komut programı Şekil 3.4'teki gibi verilebilir. Burada "MD", motor durumunu gösteren M0.0 yardımcı bellek adresinin simgesidir.



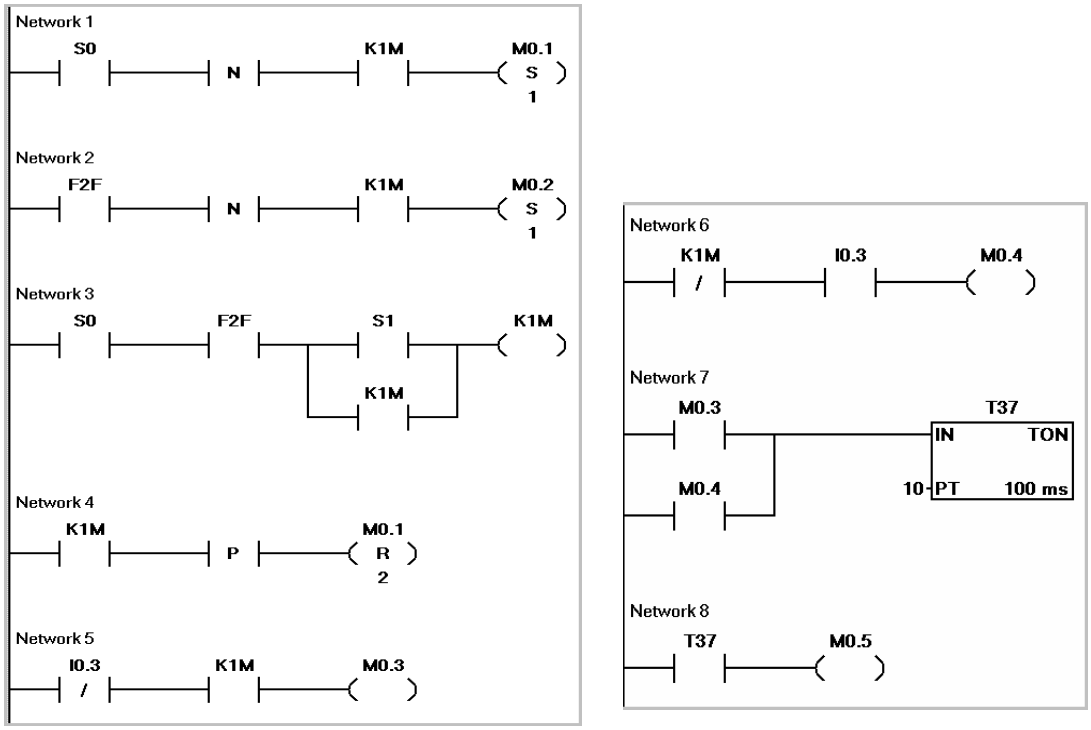
Şekil 3.4 Sembol tablosu, merdiven mantığı ve komut programı

PLC ile yapılan kumanda işleminde, elektrikli kumanda devresi ile gerçeklenmesi olanaksız birçok işlev yerine getirilebilir. Örneğin, motorun devreden çıkış nedeni (durdurma düğmesine basılarak veya aşırı akım nedeni ile gibi), kontaktöre enerji verildiği halde devreye girmeme durumu gibi bilgiler kaydedilebilir. Bu amaçla, PLC giriş-çıkış bağlantıları Şekil 3.5'de gösterilen devre için Şekil 3.6'daki merdiven mantığı programı kullanılabilir.



Şekil 3.5 PLC giriş-çıkış bağlantıları

Edited by Foxit Reader
Copyright(C) by Foxit Software Company,2005-2008
For Evaluation Only.



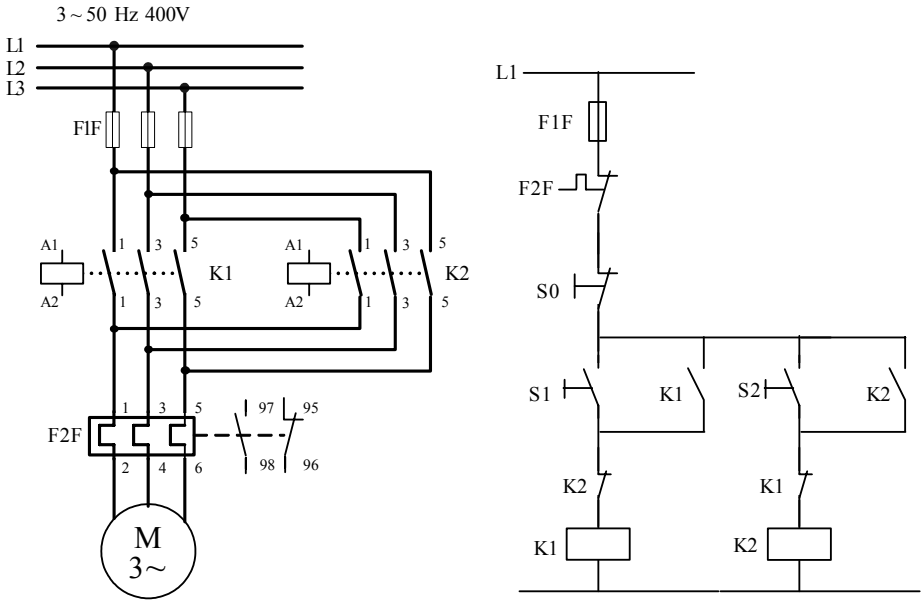
Şekil 3.6 Merdiven mantığı programı

Şekil 3.6'daki programa göre, motor çalışırken S0 durdurma düğmesine basılırsa K1M kontaktörü devreden çıkar, bu durum M0.1 biti 1'e kurularak kaydedilir (Network 1). Eğer motor çalışırken aşırı akım nedeni ile F2F rölesi kontağı açarsa motor durur ve bu durum M0.2 biti 1'e kurularak kaydedilir (Network 2). Motor devreye girdiğinde bu bilgiler silinir (Network 4). Ancak, bu işlem başka koşullara da bağlanabilir.

Kontaktör kontaklarının durumu K1M (13-14) yardımcı kontağı üzerinden alınan geri besleme ile PLC I0.3 girişine iletilir (Şekil 3.5). Bu giriş ve kontaktör bobininin bağlandığı Q0.0 çıkış noktasının değeri karşılaştırılarak bir sorun olup olmadığı belirlenebilir. Eğer I0.3=0 (kontaktör kontağı açık), Q0.0=1 (kontaktör bobinine enerji verildi) ise (I0.3′•K1M) kontaktör bobin devresinde bir sorun var demektir. Bu durumda M0.3=1 yapılır (Network 5). Eğer I0.3=1 (kontaktör kontağı kapalı), Q0.0=0 (kontaktör bobinine enerji verilmedi) ise kontaktör kontakları ile ilgili bir sorun var demektir. Bu durumda M0.4=1 yapılır (Network 6). Kontaktör gecikmesi nedeniyle (kontaktör bobinine gerilim uygulandıktan belirli bir süre sonra kontakları konum değiştirir) yanlış bilgi üretilmemesi için M0.3 veya M0.4, 1 değerini aldıktan belirli bir süre sonra M0.5=1 yapılır (Network 7, 8).

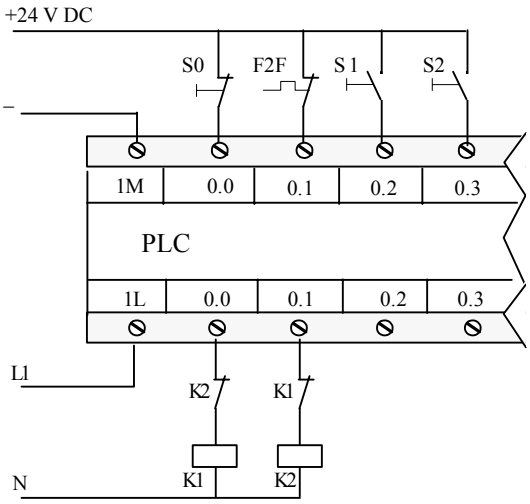
3.3 KISA DEVRE ASENKRON MOTORUN İKİ YÖNDE KUMANDASI

İki yönde çalışan bir kısa devre asenkron motorun devreye alınması ve devreden çıkartılmasına ilişkin güç ve kumanda devreleri Şekil 3.7'deki gibi verilebilir. Bu devrede, S0 düğmesine basıldığında veya motor aşırı yüklendiğinde devrede olan kontaktör devreden çıkar, S1 düğmesine basıldığında, eğer K2 devrede değilse K1 kontaktörü, S2 düğmesine basıldığında K1 kontaktörü devrede değilse K2 kontaktörü devreye girer.



Şekil 3.7 Kısa devre asenkron motorun iki yönde çalıştırılmasına ilişkin güç ve kumanda devreleri

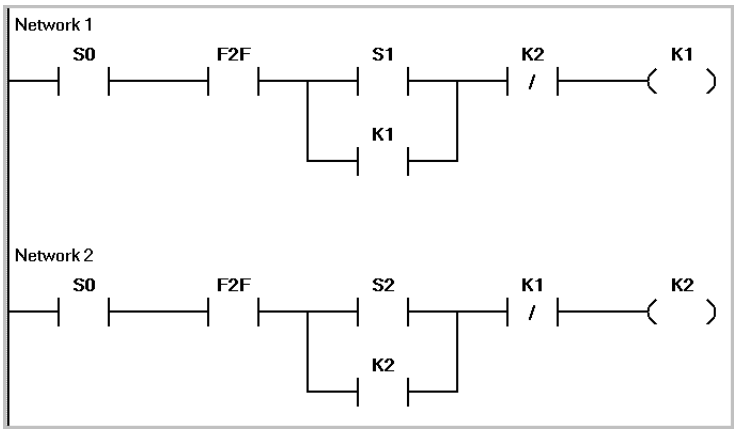
Bu kumanda devresinin PLC ile gerçeklenmesi durumunda en yalın PLC giriş çıkış bağlantıları Şekil 3.8'deki gibi verilebilir.



Şekil 3.8 PLC giriş-çıkış bağlantıları

Şekil 3.8'deki bağlantı şemasına göre Şekil 3.7'deki kumanda devresi ile aynı işlevi sağlayan merdiven mantığı ve komut programı Şekil 3.9 veya Şekil 3.10'daki gibi verilebilir.

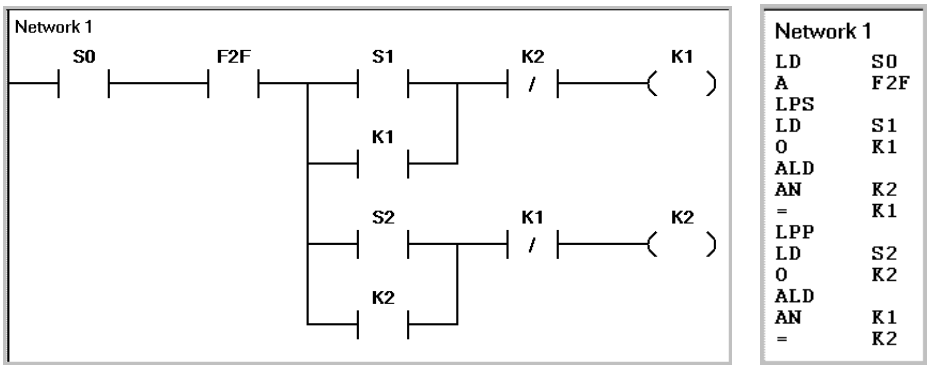
Symbol	Address
S0	I0.0
F2F	I0.1
S1	I0.2
S2	I0.3
K1	Q0.0
K2	Q0.1




```
Network 1
LD      S0
A       F2F
LD      S1
O       K1
ALD
AN      K2
=       K1

Network 2
LD      S0
A       F2F
LD      S2
O       K2
ALD
AN      K1
=       K2
```

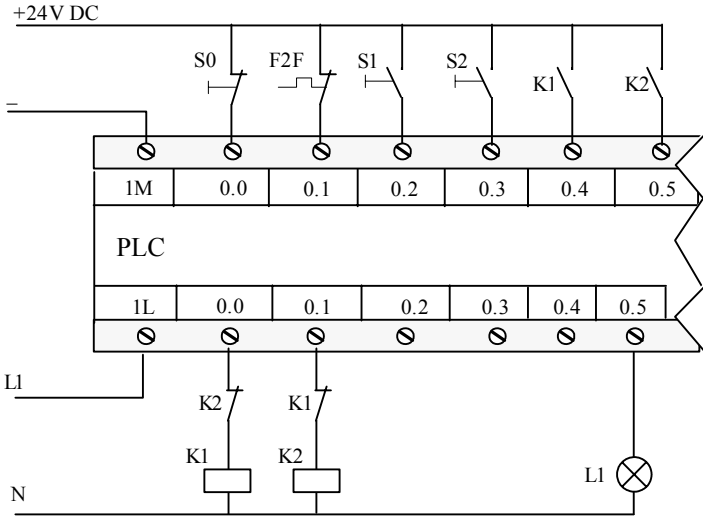
Şekil 3.9 İki yönde kumandaya ilişkin sembol tablosu, merdiven mantığı ve komut programı



Şekil 3.10 İki yönde kumandaya ilişkin merdiven mantığı ve komut programı

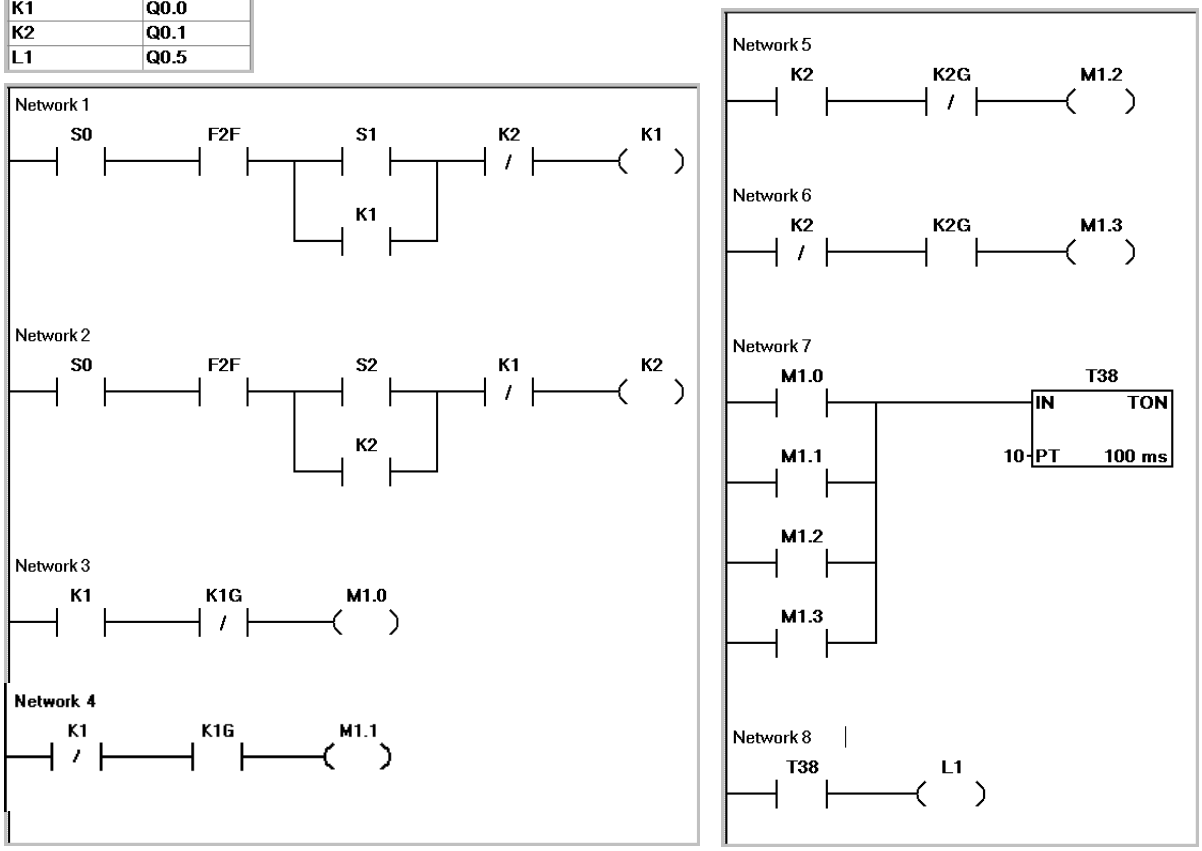
Bu şekilde gerçekleştirilen PLC'li kumanda devresi ile Şekil 3.7'de verilen kumanda devresi karşılaştırıldığında, her iki sistemin olağan çalışma koşullarında aynı işlevi sağladığı görülür. Ancak, bazı donanımsal sorunlar nedeniyle örneğin, K1 veya K2 kontaktörü kontaklarının yapışması (kaynaması) veya bağlantı iletkenlerinde bir kopukluk olması gibi durumlarda tam bir işlevsel eşdeğerlik sağlanamadığı görülür.

Kumanda sisteminde donanımsal sorunların da değerlendirilebilmesi için kontaktörlerin durumu ile bilginin alınabildiği bir bağlantı şeması düzenlenebilir. Şekil 3.11'de bu tür bir bağlantı şeması verilmiştir. Bu devrede PLC I0.4 ve I0.5 girişleri üzerinden K1 ve K2 kontaktörlerinin kontak durumu ile ilgili bilgi alınır. PLC Q0.0 çıkışı ile I0.4 girişi ve Q0.1 çıkışı ile I0.5 girişlerinin işaret durumları karşılaştırılarak olası bir donanım sorunu belirlenebilir. Örneğin, Q0.0=1 ve I0.4=0 ise ya K1 kontaktörü devreye girmemiştir ya da geribesleme iletkeninde (K1 NA kontağı üzerinden I0.4 girişine bağlanan iletken) bir kopukluk vardır. Örneğin, Q0.0=0 ve I0.4=1 ise K1 kontaktörü kontaklarının kaynamış olma olasılığı yüksektir. Buna benzer birçok hata ile ilgili bilgi üretilebilir. Şekil 3.12'de bu tür bir kumanda devresine ilişkin merdiven ve komut programı verilmiştir.



Şekil 3.11 PLC giriş-çıkış bağlantıları

Symbol	Address
S0	I0.0
F2F	I0.1
S1	I0.2
S2	I0.3
K1G	I0.4
K2G	I0.5
K1	Q0.0
K2	Q0.1
L1	Q0.5

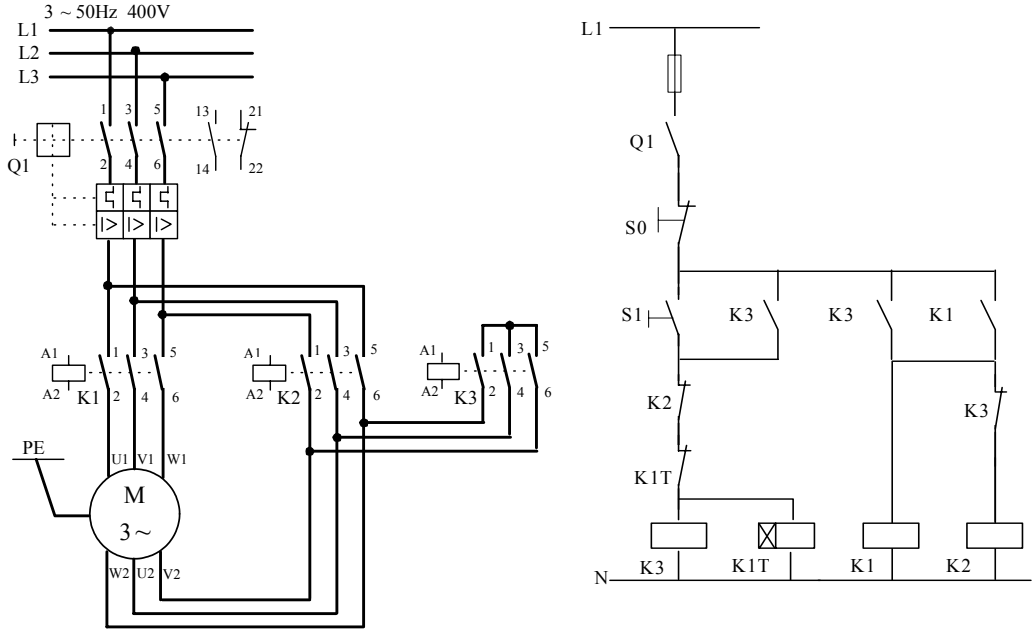


Şekil 3.12 İki yönde kumandaya ilişkin ayrıntılı merdiven mantığı programı

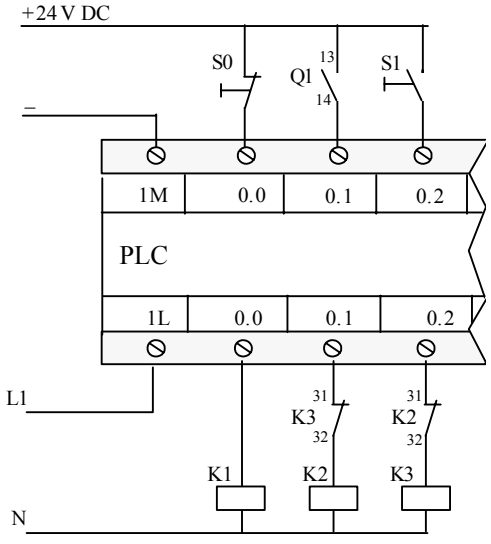
3.4 KISA DEVRE ASENKRON MOTORA YILDIZ-ÜÇGEN YOL VERME

Kısa devre asenkron motora yıldız-üçgen yol verme için Şekil 3.13'te gösterilen güç ve kumanda devreleri kullanılabilir. Burada verilen kumanda devresinin PLC ile gerçeklenmesi durumunda, en yalın PLC giriş çıkış bağlantıları Şekil 3.14, kumanda devresinden hareketle doğrudan elde edilen merdiven mantığı programı Şekil 3.15'teki gibi verilebilir. Hiçbir donanımsal sorunun olmadığı olağan çalışma koşullarında, Şekil 3.13'teki kumanda devresi ile Şekil 3.15'teki programa göre çalışan PLC'li kumanda devresi aynı işlevi sağlar. Ancak, bazı donanımsal sorunlar nedeniyle örneğin; K1, K2 veya K3 kontaktörlerine ilişkin kontakların yapışması (kaynaması) veya bağlantı iletkenlerinde bir kopukluk olması gibi durumlarda farklı sonuçlar elde edilebilir. Buna örnek olarak sistemin devreye alınması sürecinde yıldız bağlama kontaktörünün (K3), kontak yapışması nedeniyle devreden çıkmadığı durum gösterilebilir. Bu durumda Şekil 3.13'teki kumanda devresine göre çalışan sistemde K3 ve K1 kontaktörü sürekli devrede kalır ve motor yıldız bağlı çalışmadan üçgen bağlı çalışmaya geçemez. Üçgen bağlı çalışmaya geçilmemesi nedeniyle, motor (yük durumuna bağlı olarak) aşırı akım çeker. Aşırı akım ve kısa devre akımına duyarlı motor koruma açıcısı (eğer doğru seçilmiş ve ayarlanmışsa) motoru enerji kaynağından ayırır. Eğer motor koruma açıcısı tekrar devreye alınırsa, S1 başlatma düğmesine basılmadığı halde, K3 kontağı kapalı olduğundan motor tekrar yıldız bağlanarak devreye girer ve aynı olaylar yinelenir.

Şekil 3.15'teki merdiven mantığı programına göre çalışan kumanda devresinde ise devreye giriş sürecinde önce PLC Q0.2 (K3)=1 , Q0.0 (K1)=1 ve Q0.1 (K2)=0 olur. 10 saniye sonra PLC Q0.2 (K3)=0 ve Q0.0 (K1)=1 Q0.1 (K2)=1 olur. Ancak, Q0.2 çıkışına bağlı K3 yıldız kontaktörü kontakları yapışık olduğundan Q0.2=0 ve Q0.1=1 olsa da K2 üçgen kontaktörü devreye girmez ve yıldız bağlı çalışma durumu sürer. Bu durumda iken motor koruma açıcısının devreyi açtığını düşünelim. Q1 motor koruma açıcısının Q1(13-14) yardımcı kontağı nedeniyle PLC I0.1=0 ve merdiven mantığı programına göre PLC Q0.2 (K3)=0 ve Q0.0 (K1)=0 Q0.1 (K2)=0 olur. Eğer motor koruma açıcısı tekrar devreye alınırsa, motor kendiliğinden devreye girmez. Devreye girmesi için S1 başlatma düğmesine basılmalıdır.

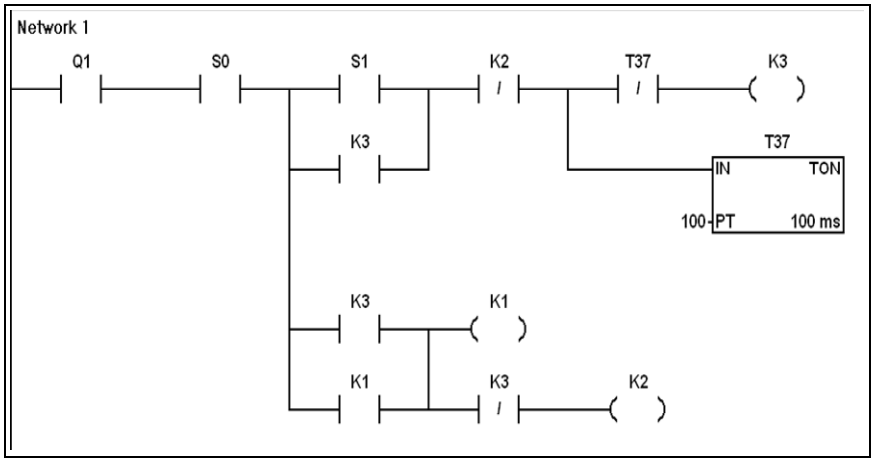


Şekil 3.13 Kısa devre asenkron motora yıldız-üçgen yol verme güç ve kumanda devreleri



Şekil 3.14 Yıldız-üçgen yol verme PLC giriş-çıkış bağlantıları

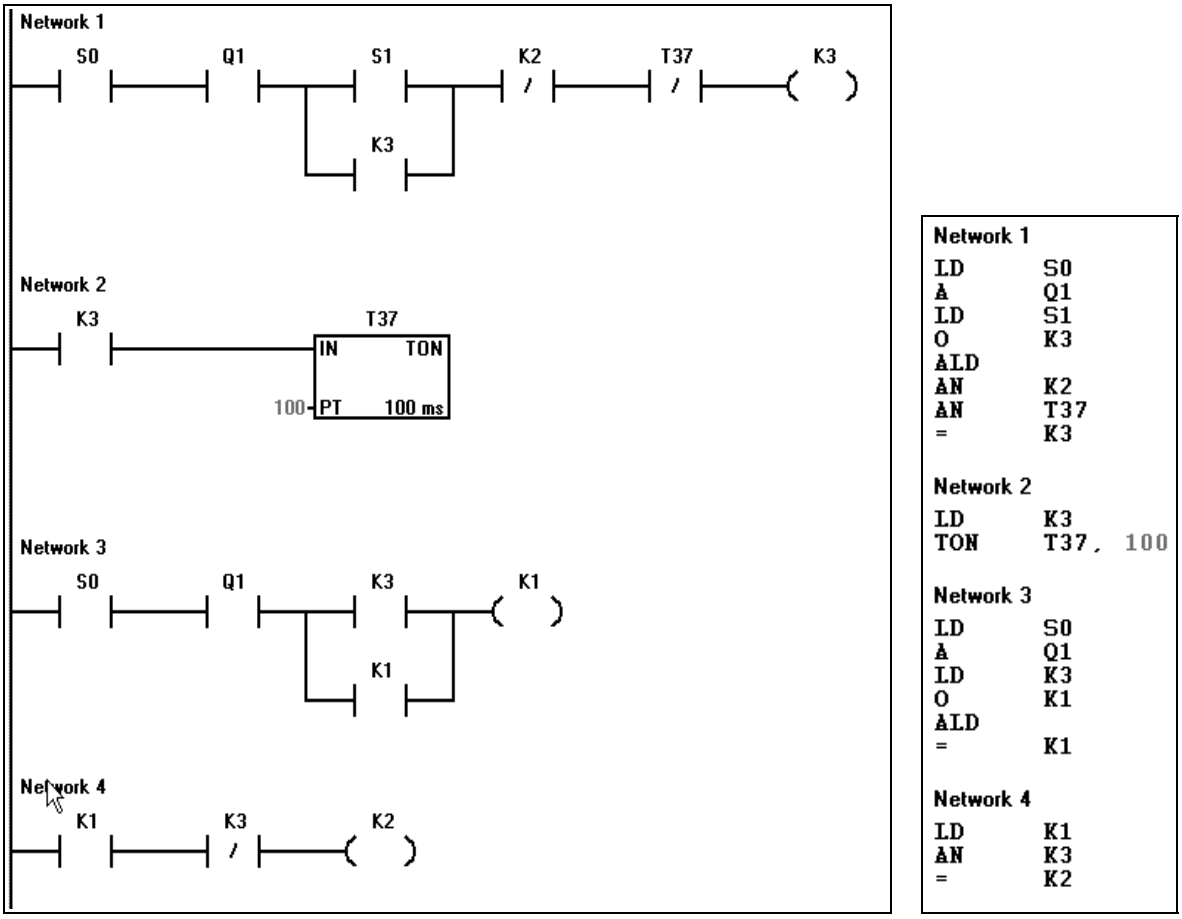
Symbol	Address
S0	I0.0
Q1	I0.1
S1	I0.2
K1	Q0.0
K2	Q0.1
K3	Q0.2




```
Network 1
LD      Q1
A       S0
LPS
LD      S1
O       K3
AN      K2
ALD
LPS
AN      T37
=       K3
LPP
TON     T37, 100
LPP
LD      K3
O       K1
ALD
=       K1
AN      K3
=       K2
```

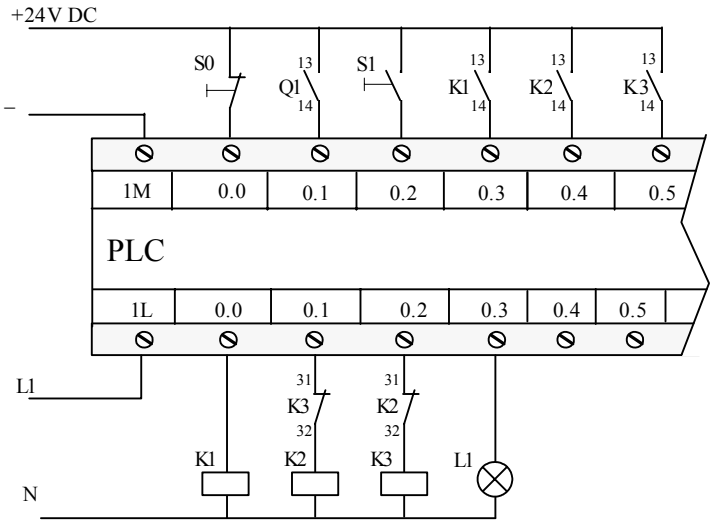
Şekil 3.15 Yıldız-üçgen yol verme kumandasına ilişkin merdiven mantığı ve komut programı I

Kumanda devresine bakılarak doğrudan elde edilen Şekil 3.15'teki kumanda devresi yerine STL dili daha basit ve izlenmesi daha kolay olan Şekil 3.16'daki merdiven mantığı programı kullanılabilir. Bu program, Şekil 3.13'teki kumanda devresinde akım yolları izlenerek elde edilen mantık fonksiyonlarından hareketle yazılabilir.



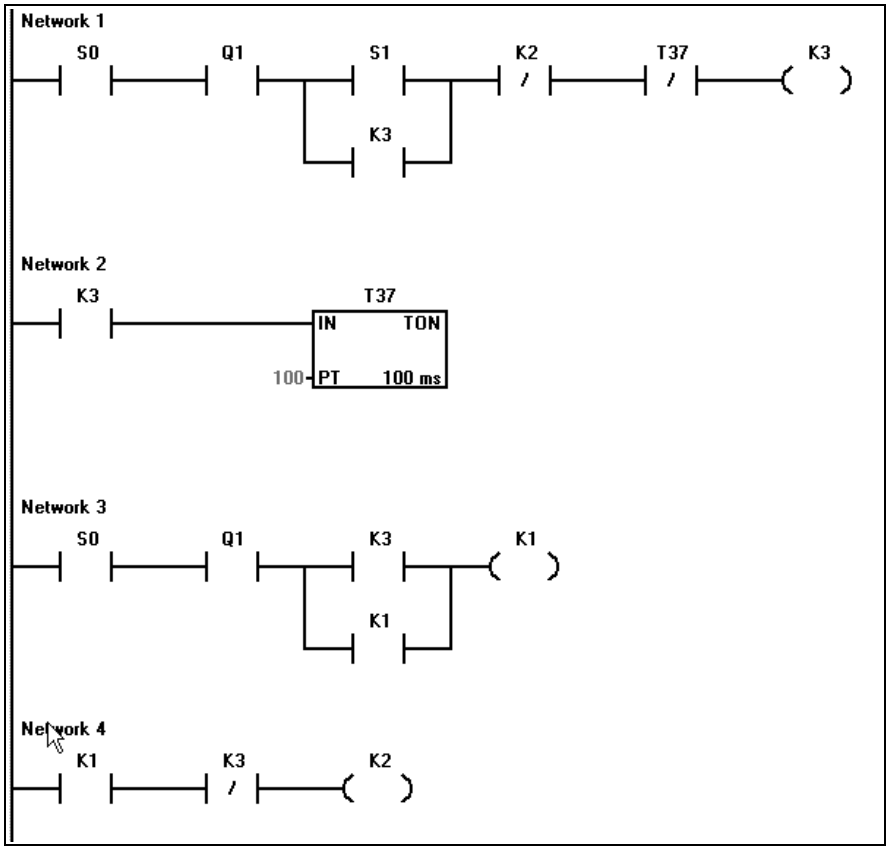
Şekil 3.16 Yıldız-üçgen yol verme kumandasına ilişkin merdiven mantığı ve komut programı II

Bu kumanda sisteminde, donanımsal sorunların değerlendirilebilmesi ve bu tür sorunlar ile karşılaşıldığında gerekli kumanda işaretlerinin üretilebilmesi için kontaktörlerin durumu ile ilgili bilginin alınabildiği bir bağlantı şeması düzenlenebilir. Şekil 3.17'de bu tür bir bağlantı şeması ve Şekil 3.18'de donanım hatalarının da değerlendirildiği bir merdiven mantığı programı verilmiştir.

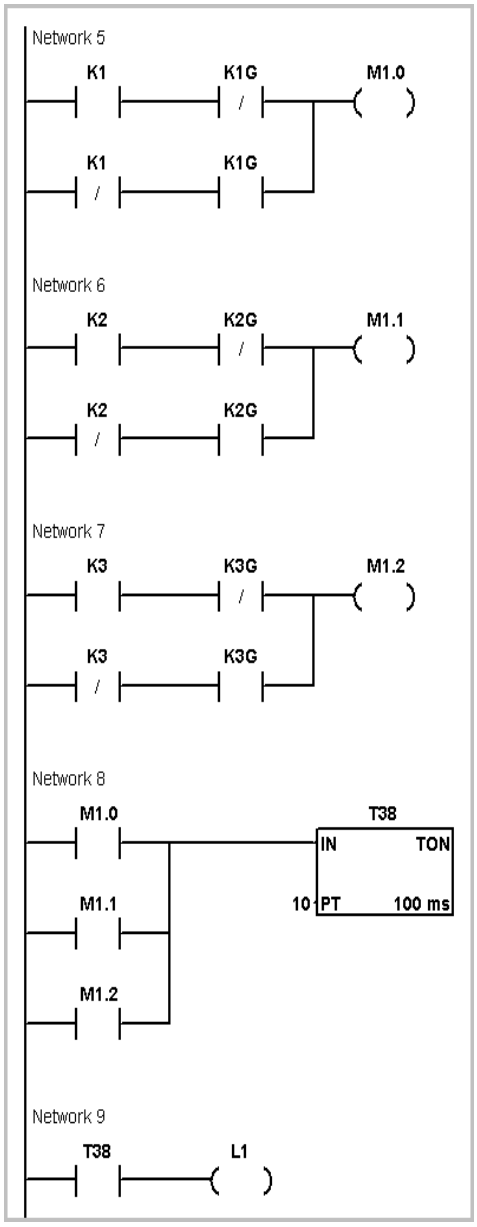


Şekil 3.17 Yıldız-üçgen yol verme geri beslemeli giriş-çıkış bağlantıları

Symbol	Address
S0	I0.0
Q1	I0.1
S1	I0.2
K1G	I0.3
K2G	I0.4
K3G	I0.5
K1	Q0.0
K2	Q0.1
K3	Q0.2
L1	Q0.3



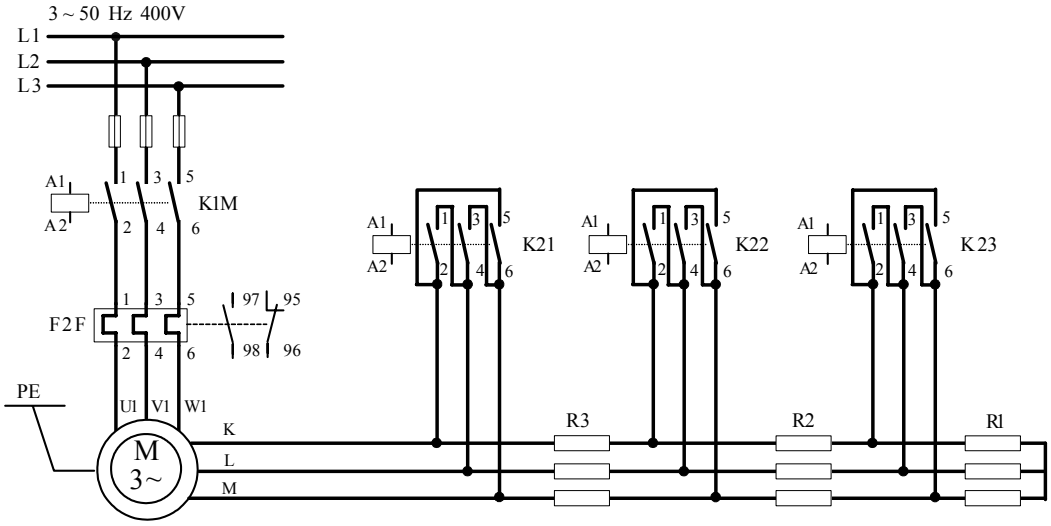
Şekil 3.18 Yıldız-üçgen yol vermeye ilişkin ayrıntılı merdiven mantığı programı (1/2)



Şekil 3.18 Yıldız-üçgen yol vermeye ilişkin ayrıntılı merdiven mantığı programı (2/2)

3.5 ROTORU SARGILI ASENKRON MOTORUN KUMANDASI

Rotoru sargılı (bilezikli) asenkron motorun devreye alınmasında rotor devresi etkin direnç değerini değiştirerek yol verme yaygın olarak kullanılan bir yöntemdir. Rotor etkin direnç değerinin değiştirilmesinde yarı iletken devre elemanları kullanıldığı gibi Şekil 3.19'da verilen basit bir devre de kullanılabilir. Bu devrede K21, K22 ve K23 kontaktörlerine kumanda edilerek rotor devresi direnci değiştirilir. Yol vermede, tüm dirençler devrede iken motor stator sargıları şebekeye bağlanır, motor yol almaya başladıktan sonra dirençler kademeli olarak kısa devre edilir. Böylece, en büyük yol alma momenti ile motorun hızlanması sağlanırken, yol alma akımı da sınırlandırılmış olur.



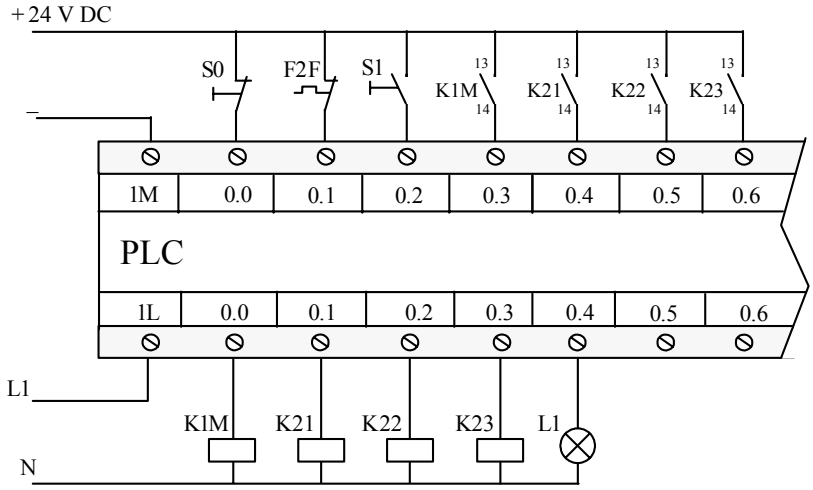
Şekil 3.19 Rotoru sargılı asenkron motora ilişkin güç devresi

Bu tür bir kumanda devresinin şu koşulları sağlaması beklenir:

1- Bir başlatma düğmesine basıldığında, eğer bütün dirençler devrede ise (K21, K22 ve K23 kontaktörleri devre dışı ise) K1M kontaktörü devreye girmeli ve K1M kontaktörü devreye girdikten belirli bir süre sonra sırayla K23 kontaktörü, daha sonra K22 ve K21 kontaktörleri devreye girmelidir.

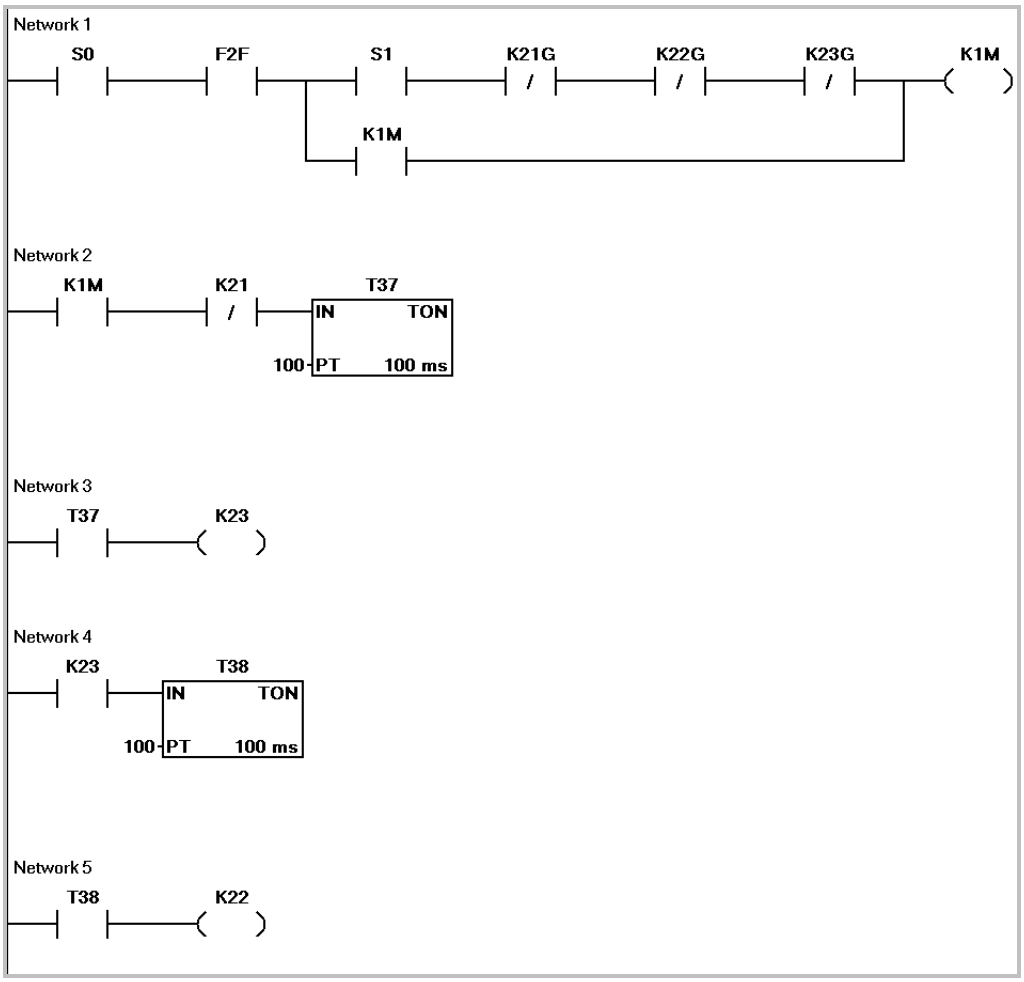
2- Herhangi bir anda durdurma düğmesine basıldığında veya motor aşırı yüklendiğinde bütün kontaktörler devreden ayrılmalıdır.

Bu sisteme ilişkin kumanda devresinin işlevi ve güvenli işletim koşulları göz önüne alınarak, aşağıdaki sembol tablosuna göre PLC giriş-çıkış bağlantıları Şekil 3.20, merdiven mantığı programı ise Şekil 3.21'deki gibi verilebilir.

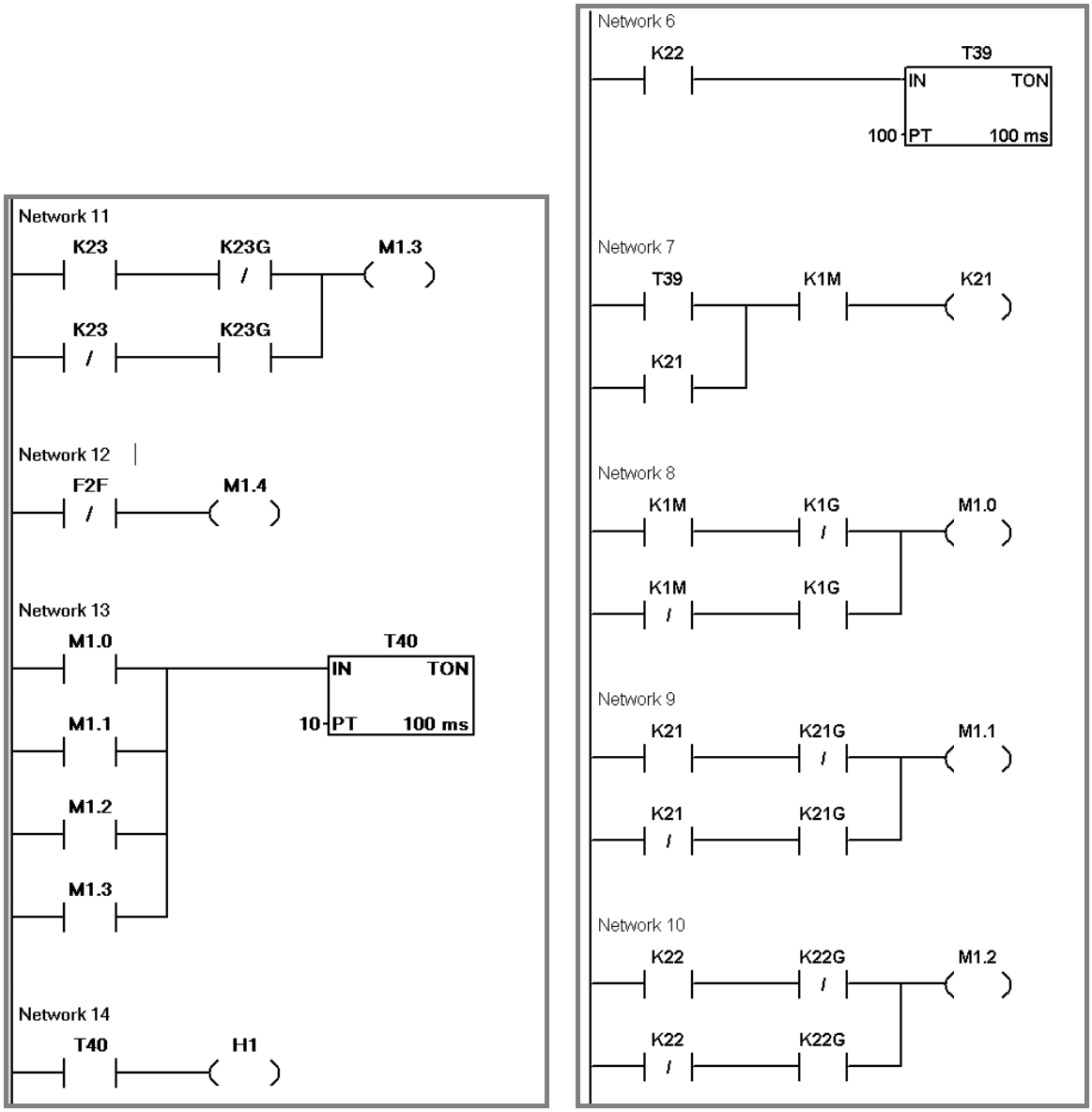


Şekil 3.20 PLC giriş-çıkış bağlantıları

Symbol	Address
S0	I0.0
F2F	I0.1
S1	I0.2
K1G	I0.3
K21G	I0.4
K22G	I0.5
K23G	I0.6
K1M	Q0.0
K21	Q0.1
K22	Q0.2
K23	Q0.3
H1	Q0.4



Şekil 3.21 Rotoru sargılı asenkron motorun kumandasına ilişkin merdiven mantığı programı (1/2)



Şekil 3.21 Rotoru sargılı asenkron motorun kumandasına ilişkin merdiven mantığı programı (2/2)

Şekil 3.21'de verilen merdiven mantığı programında (Network 1) ile (Network 7) arasındaki bölüm kumanda devresinin işlevini, diğer bölümler donanım hatalarının belirlenmesi ve bildirimini sağlar.

Kumanda devresinin çalışması şu şekilde özetlenebilir:

Eğer durdurma düğmesine basılmamış (S0=0, I0.0=1), motor aşırı yüklenmemiş (F2F=0, I0.1=1), K21, K22 ve K22 kontaktörleri devrede değilse (K21G=0, I0.4=0; K22G=0, I0.5=0; K23G=0, I0.6=0) başlatma düğmesine basıldığında (S1=1, I0.2=1) K1M kontaktörü devreye alınır (K1M=1, Q0.0=1). (Network 1).

K1M devreye alındıktan sonra, başlangıçta K21=0 olduğundan T37 zamanlayıcısı etkin olur, 10 saniye sonra T37 kontağı kapanır ve K23 kontaktörü devreye girer (Network 2, 3).

K23 kontaktörü devreye girdikten sonra T38 zamanlayıcısı etkin olur, T38 zamanlayıcı kontağı 10 saniye sonra kapanır ve K22 kontaktörü devreye girer (Network 4, 5).

K22 kontaktörü devreye girdikten sonra T39 zamanlayıcısı etkin olur, T39 zamanlayıcı kontağı 10 saniye sonra kapanır, K21 kontaktörü devreye girer ve K1M devreden ayrılıncaya kadar devrede kalır (Network 6, 7). K21 kontaktörünün devreye girmesiyle T37 devreden çıkar ve buna bağlı olarak T38, T39, K22 ve K23 devre dışı kalır (Network 2,3,4, 5, 6). Sürekli çalışmada yalnız K1M ve K21 devrededir.

Eğer PLC Q0.0=1, ancak K1G=0 (K1M kontaktörü devrede değil, kontakları açık) veya Q0.0=0, ancak K1G=1 (K1M kontaktörü devrede, kontakları kapalı) ise K1M kontaktörü ile ilgili bir donanım hatası vardır. Bu durumda M1.0=1 olur (Network 8). Benzer şekilde K21, K22 ve K23 kontaktörlerinin donanım hataları ile ilgili olarak sırayla M1.1, M1.2 ve M1.3 adreslerindeki yardımcı bellek bitlerinin içeriği 1 yapılır (Network 9,10,11).

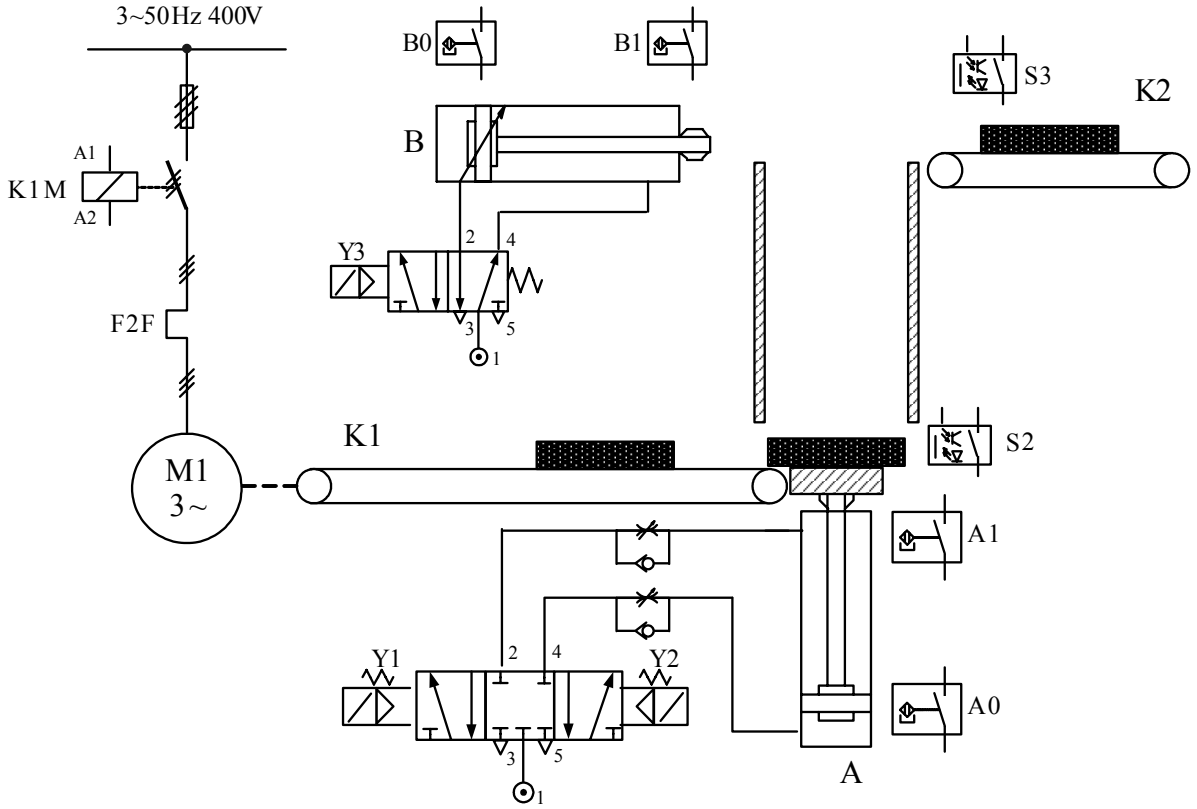
Eğer bu donanım hataları 1 saniyeden uzun sürerse L1 donanım hatası çıkışı etkin yapılır (Q0.4=1) (Network 13,14).

Motorun aşırı akım rölesinin durumu M1.4 adresine kaydedilir (Network 12). Eğer M1.4=1 ise motor aşırı yüklenmiştir.

Donanıma ve işletme durumuna ilişkin bu bilgiler (M1.0, M1.1, M1.2, M1.3 ve M1.4) kullanılarak bu kumanda devresine daha değişik özellikler kazandırabilecek bölümler eklenebilir.

3.6 BİR PNÖMATİK SİSTEMİN KUMANDASI

Pnömatik iş elemanları kullanılarak gerçeklenen bir aktarma sistemi Şekil 3.22'de gösterilmiştir. Burada K1 noktasından alınan parçanın K2 noktasına aktarılmasını sağlayan bir kumanda devresi, işletme güvenliği de göz önüne alınarak PLC ile gerçeklenecektir.

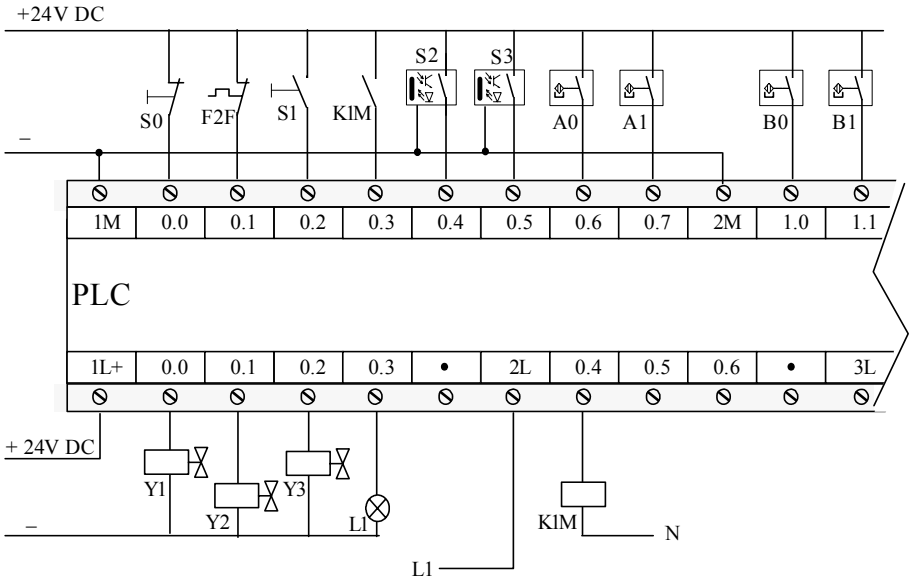


Şekil 3.22 Pnömatik iş elemanlı bir aktarma sistemi

Bu kumanda devresi; sistemi devreye alma ve devreden çıkarma, M1 motorunu devreye alma ve çıkarma, A pistonunu yukarı ve aşağı hareket ettirme, B pistonunu sağa hareket ettirme ile ilgili koşullar belirlenerek gerçeklenir.

Bu sisteme ilişkin kumanda devresinin işlevi ve güvenli işletim koşulları göz önüne alınarak aşağıdaki sembol tablosuna göre PLC giriş-çıkış bağlantıları Şekil 3.23'te, merdiven mantığı programı Şekil 3.24'teki gibi verilebilir.

Symbol	Address	Comment
S0	I0.0	Durdurma düğmesi
F2F	I0.1	Aşırı akım koruma rölesi
S1	I0.2	Başlatma düğmesi
K1G	I0.3	K1M devrede
S2	I0.4	A pistonu üzerinde parça var
S3	I0.5	K2 taşıyıcısı üzerinde parça var
A0	I0.6	A pistonu alt konumda
A1	I0.7	A pistonu üst konumda
B0	I1.0	B pistonu geri (sol) konumda
B1	I1.1	B pistonu ileri (sağ) konumda
Y1	Q0.0	A pistonu aşağı hareket valf bobini
Y2	Q0.1	A pistonu yukarı hareket valf bobini
Y3	Q0.2	B pistonu sağa hareket valf bobini
K1M	Q0.4	Motor kontaktörü
L1	Q0.3	Sorun var uyarı lambası
SD	M0.0	Sistem devrede
PVAR	M0.1	A pistonu üzerinde parça var



Şekil 3.23 PLC giriş-çıkış bağlantıları

Şekil 3.24'te verilen merdiven mantığı programına göre çalışan kumanda sisteminde; S1 düğmesine basıldığında sistem devreye girer ("SD"=M0.0=1), S0 durdurma düğmesine basıldığında sistem devreden çıkar ("SD"=M0.0=0) (Network 1).

Eğer sistem devrede (SD=1), motor aşırı akım uyarısı yok (F2F=0), A pistonu A0 konumunda (A0=1) ve üzerinde parça yok (PVAR=0) koşulları sağlanıyorsa K1M kontaktörü devreye girer (Network 2).

Eğer A pistonu 0 konumunda (A0=1) ve piston üzerinde parça var (S2=1) ise PVAR=1 yapılır. A pistonu A1 konumuna ve B pistonu B1 konumuna geldiğinde parçanın B pistonu ile itildiği varsayılarak PVAR=0 yapılır (Network 3).

Eğer sistem devrede (SD=1), B pistonu 0 konumunda (B0=1), A pistonu 1 konumunda değilse (A1=0) ve parça var (PVAR=1) ise A pistonunu yukarı hareket ettiren Y2 yol denetim valfi uyarılır (Network 4).

Eğer sistem devrede (SD=1), A pistonu üstünde parça yok (PVAR=0) ve A pistonu 0 konumunda değilse pistonunu aşağı hareket ettiren Y1 yol denetim valfi uyarılır (Network 5).

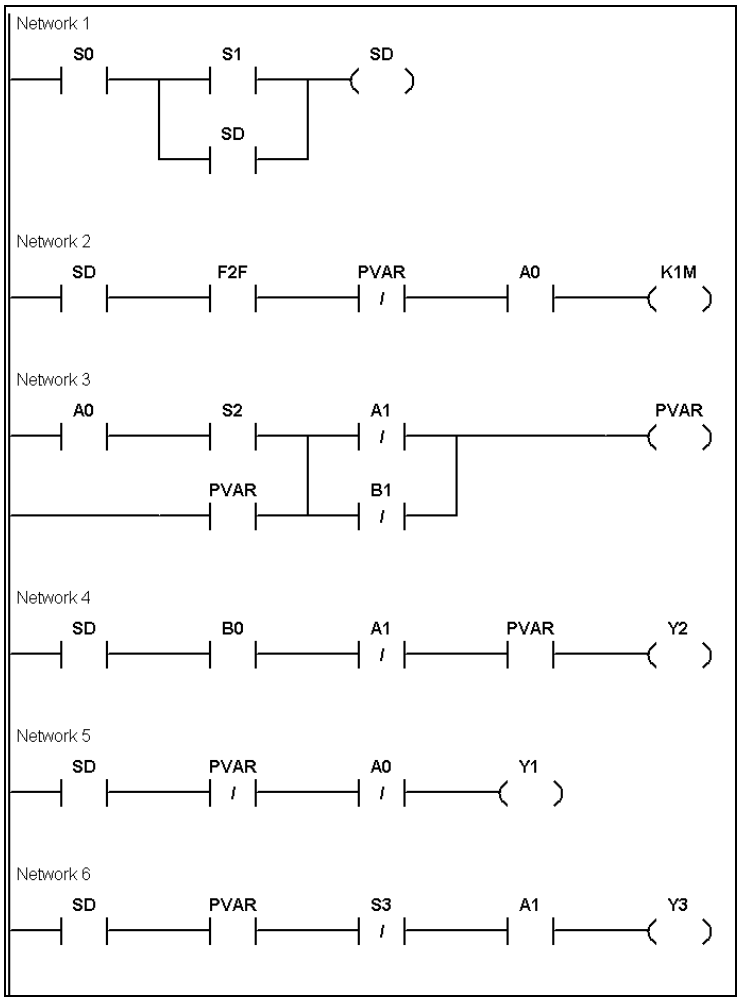
Eğer sistem devrede (SD=1), A pistonu üstünde parça var (PVAR=1), piston A1 konumunda (A1=1) ve K2 taşıyıcısı üzerinde parça yoksa (S3=0), B pistonunu sağa hareket ettiren Y3 yol denetim valfi uyarılır (Network 6).

Bu kumanda devresinde sistemin işletme durumu ile ilgili koşullar şu şekilde değerlendirilir:

Eğer K1M kontaktör bobini uyarılmış (Q0.0=1), ancak K1M yardımcı kontağı açık durumda (K1G=0) veya K1M kontaktörü bobini uyarılmamış (Q0.0=0), ancak K1M yardımcı kontağı kapalı durumda (K1G=1) ise K1M kontaktörü ile ilgili bir donanım hatası bildirim işareti (M1.0=1) üretilir (Network 7).

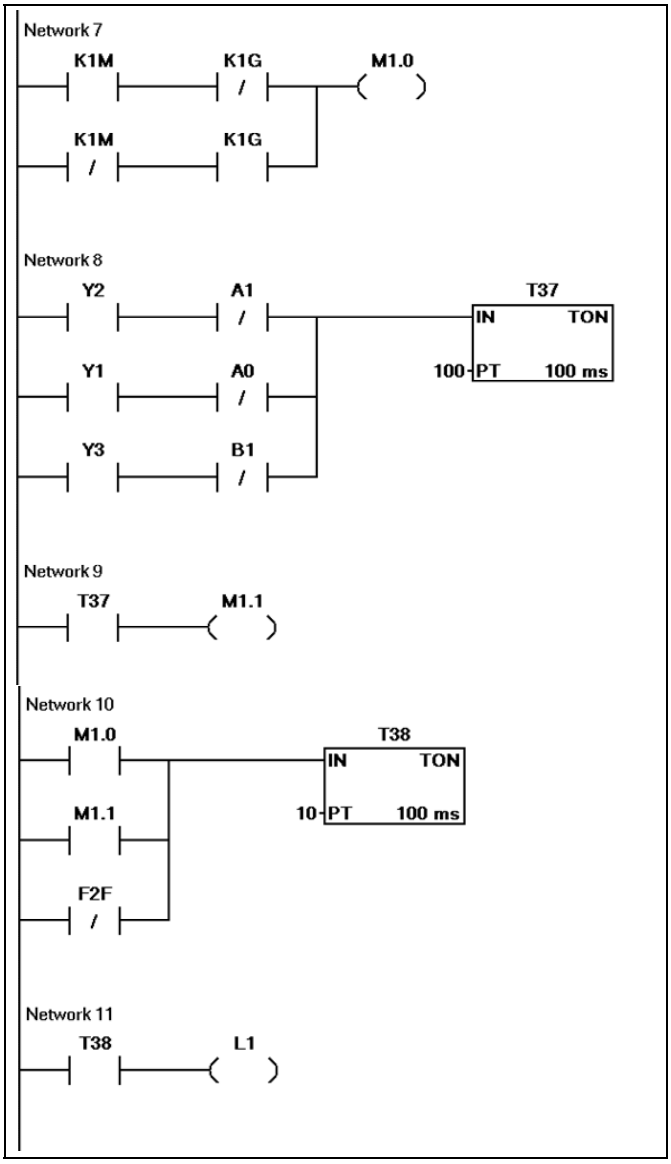
Eğer Y2, Y1 ve Y3 valfleri uyarıldığı halde ilgili pistonların belirli bir süre içinde varış noktalarına erişememeleri durumlarında piston hatası bildirim işareti (M1.1) üretilir (Network 8 ve Network 9).

Kontaktör, piston veya motor aşırı akım uyarılarının belirli bir süreyi aşması durumlarında L1 uyarı lambası işareti üretilir (Network 10 ve Network 11).



Symbol	Address
S0	I0.0
F2F	I0.1
S1	I0.2
K1G	I0.3
S2	I0.4
S3	I0.5
A0	I0.6
A1	I0.7
B0	I1.0
B1	I1.1
Y1	Q0.0
Y2	Q0.1
Y3	Q0.2
K1M	Q0.4
L1	Q0.3
SD	M0.0
PVAR	M0.1

Şekil 3.24 Pnömatik kumanda devresine ilişkin merdiven mantığı programı (1/2)



Şekil 3.24 Pnömatik kumanda devresine ilişkin merdiven mantığı programı (2/2)

3.7 ZAMANLAYICI VE SAYICI UYGULAMALARI

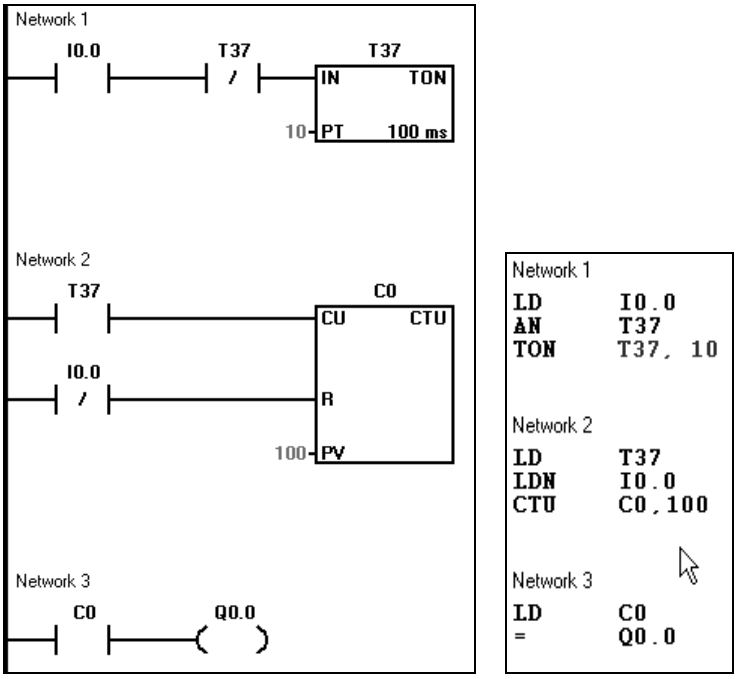
3.7.1 UZUN SÜRELİ GECİKME İŞARETİ ÜRETİLMESİ

Simatic S7-200 CPU 22x işlemcilerinde bir zamanlayıcının içeriği en çok 32767 değerini alır. Buna göre artım süresi (çözünürlüğü) 1ms olan zamanlayıcılar ile en uzun gecikme süresi 32767×1ms=32.767s, 10ms artımlı zamanlayıcılar ile en uzun gecikme süresi 32767×10ms =327670ms =327.67s ve 100ms artımlı zamanlayıcılar ile en uzun gecikme süresi 32767×100ms=3276700ms=3276.7s olarak elde edilebilir. Daha uzun süreli gecikme değerleri elde edebilmek için zamanlayıcı ve sayıcılar birlikte kullanılır. Bunun için zamanlayıcılar ile belirli süreli zamanlama işaret darbeleri elde edilir. Bu işaretler sayıcılar ile değerlendirilerek istenen uzunlukta gecikme süreleri elde edilebilir. Örneğin, artım süresi 100ms olan bir zamanlayıcı ve bir sayıcı ile 32767×32767×100ms uzunluğunda gecikme süresi elde edilebilir. Şekil 3.25'te bu amaçla kullanılabilecek bir program verilmiştir. Bu programa göre I0.0 girişi 1 yapıldığında 100ms artımlı T37 zamanlayıcısı her 1 saniyede bir işaret darbesi üretir. Bu işaretler C0 sayıcısı ile sayılır ve darbe sayısı 100 olduğunda Q0.0=1 olur. Bu biçimde yazılan bir programla zamanlayıcı karşılaştırma değeri PT=32767 ve sayıcı karşılaştırma değeri PV=32767 seçilerek gecikme süresi

32676×32767×100ms=1.07367×109ms=29824.347saat

=1242.6809gün=3.4046 yıl elde edilebilir.

Uzun süreli gecikme elde etmek için üretilen periyodik işaret darbelerinin sayılabilir olması için en az bir tarama çevrimi 1 değerinde kalması gerekir. Şekil 3.25'te verilen programda 100ms artımlı bir zamanlayıcı kullanılmıştır. Bu tür zamanlayıcılarda, zamanlayıcı içeriği ve biti komut (TON T37, 10) işlendiğinde güncellenir. Komut işlendiği anda eğer zamanlayıcı içeriği karşılaştırma değerine eşit veya büyükse T37 zamanlayıcı biti 1 olur ve bir sonraki taramaya kadar bu değerde kalır. Bir sonraki taramada zamanlayıcıyı etkin yapan koşul T37 bitine bağlı olarak sıfırlandığından zamanlayıcı içeriği ve biti sıfırlanır, zamanlayıcı tekrar sayma durumuna geçer. T37 biti 1 değerini aldıktan sonra bir tarama süresi 1 değerinde kaldığından C0 sayıcısı ile sayılabilir ve her 1 saniyede bir sayıcı içeriği 1 artar.

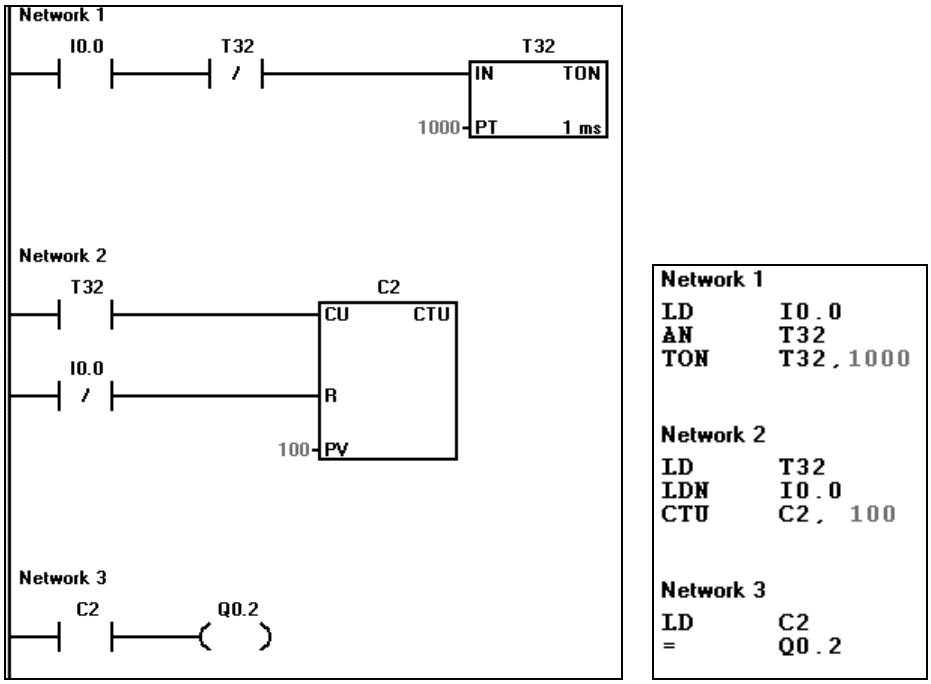


Şekil 3.25 100 ms artımlı zamanlayıcı ile uzun süreli gecikme işlevini sağlayan örnek program

Aynı işlevi sağlayan bir program aynı biçimde 1ms'lik artımlı (T32,T96) veya 10 ms artımlı (T33, T34, T35, ..) zamanlayıcılar ile gerçeklendiğinde aynı sonucun elde edilmediği görülür. Bunun nedeni bu tür zamanlayıcıların güncellenme koşullarının 100ms'lik zamanlayıcılardan farklı oluşudur. Daha önce Bölüm 2.7'de açıklandığı üzere, 1ms artımlı zamanlayıcılarda tarama çevriminden bağımsız olarak her 1ms'de ve 10ms artımlı zamanlayıcılarda ise her tarama çevriminin başında güncelleştirme yapılır (Alt Bölüm 2.7). Güncelleştirme anında, zamanlayıcı içeriği zamanlayıcı karşılaştırma (PT) değeri ile karşılaştırılır, eğer zamanlayıcı içeriği PT değerinden eşit veya büyük ise zamanlayıcı biti 1 yapılır.

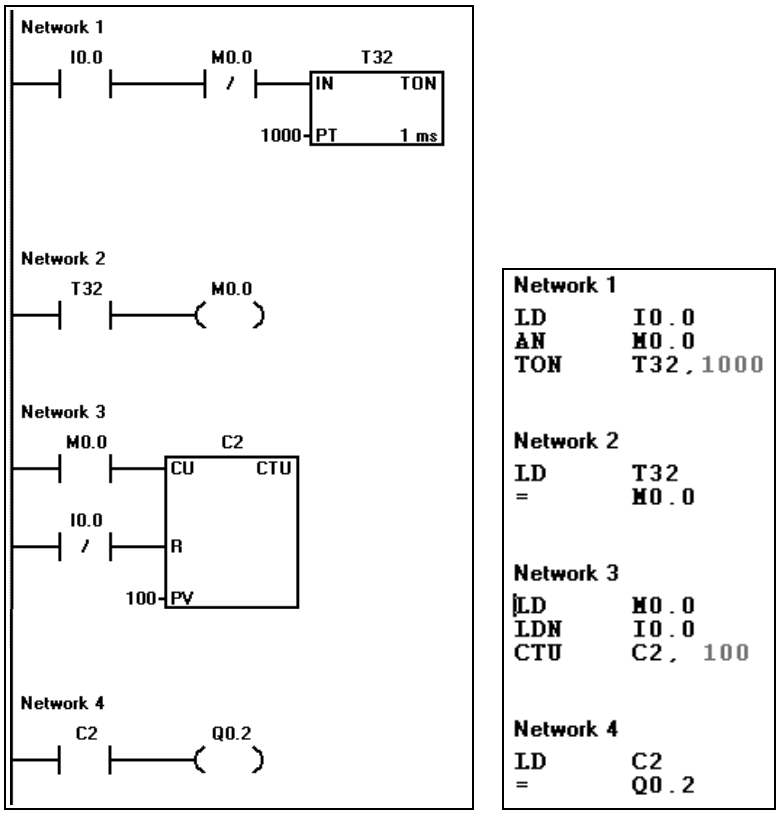
Uzun süreli 1ms çözünürlükte gecikme elde etmek için Şekil 3.26'da verilen program yazılmış olsun. Bu programın Şekil 3.25'te verilen program ile aynı davranışı göstermesi beklenir. Ancak 1ms artımlı T32 zamanlayıcısının güncelleştirme koşulundan kaynaklanan fark nedeniyle, Q0.2 çıkışının 100 saniyeden çok daha uzun süre sonra 1 değerini aldığı görülecektir. Bunun nedeni komutların işleniş zamanı ve sırası ile zamanlayıcı güncelleştirme anı göz önüne alınarak açıklanabilir. Bu programda yer alan T32 zamanlayıcısı her 1ms'de, program tarama çevriminden bağımsız olarak güncelleştirilir. Programın yürütülmesi sürecinde zamanlayıcı içeriği her 1ms'de bir 1000 ile karşılaştırılır ve zamanlayıcı içeriği 1000 değerine eşit veya büyükse T32=1 yapılır. Eğer zamanlayıcı komutu (TON

T32,100) işlenmeden önce T32=1 yapılmışsa, bu konut işlendiği anda T32 zamanlayıcısı içeriği ve biti sıfırlanacak ve sonraki komutların işlenmesi sürecinde T32=0 olarak değerlendirilecektir. Bu durumda C2 sayıcısının içeriği artmayacaktır. Eğer zamanlayıcı komutu işlendikten ve sayıcı komutu işlenmeden önce T32=1 olursa sayıcı içeriği artacaktır. Son olarak sayıcı komutu işlendikten sonra T32=1 olursa, bir sonraki taramada yine zamanlayıcı içeriği ve biti sıfırlanacağından sayıcı içeriği artmayacaktır. Şu halde bu programda sayıcı içeriğinin artması için gerekli koşul, güncelleştirme işleminin (Network 1)'deki (AN T32) komutu işlendikten sonra ve (Network 2)'deki (LD T32) işlenmeden önce yapılmasıdır. Bu aralığın diğer komutların işlenmesi ve giriş-görüntü belleği güncelleştirilmesi sürecinde geçen süreye göre çok daha kısa olması nedeniyle sayıcı içeriği gereğinden daha az artacaktır. Bu çözüm bir kumanda sistemi için kabul edilebilir değildir.



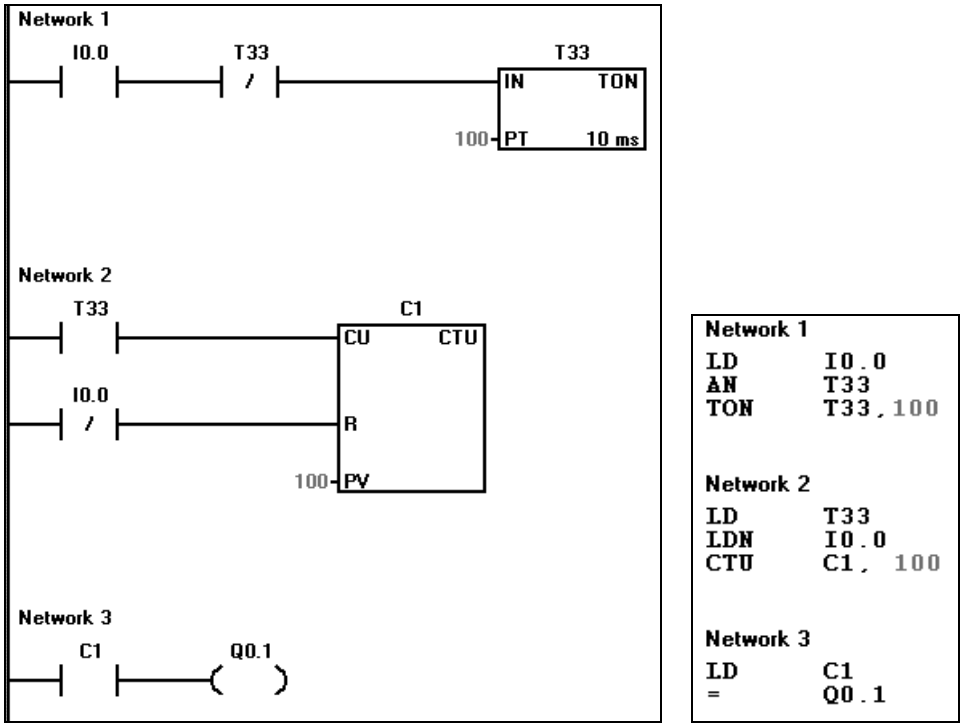
Şekil 3.26 1 ms artımlı zamanlayıcı ile uzun süreli gecikme işlevini sağlayan örnek program

Şekil 3.25'te verilen programla aynı işlevi sağlayan bir program Şekil 3.27'deki gibi yazılabilir. Burada T32 zamanlayıcı bitinin değeri bir M bellek alanında saklanarak 1 tarama çevrimi 1 değerinde tutulması sağlanır. Bu program incelendiğinde, T32 zamanlayıcı biti içeriğinin M0.0 bellek alanına atanması ile Şekil 3.26'da verilen programa ilişkin sorunların giderildiği görülür.



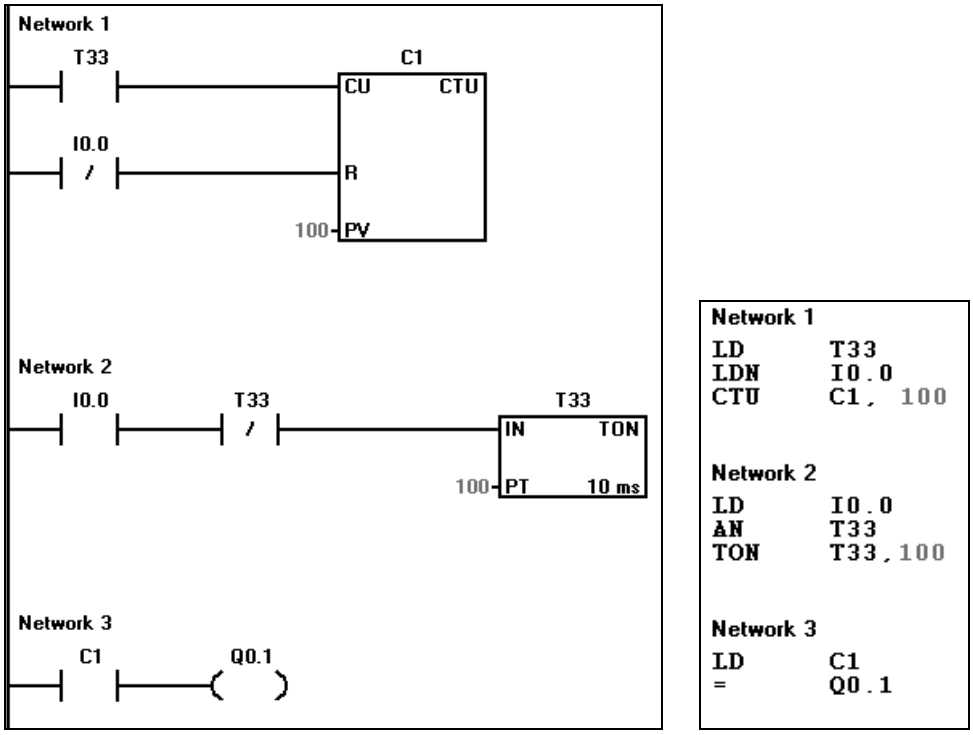
Şekil 3.27 1 ms artımlı zamanlayıcı ile uzun süreli gecikme işlevini sağlayan örnek program

Uzun süreli 10ms çözünürlükte gecikme elde etmek için Şekil 3.28'de verilen program yazılmış olsun. Bu programın işletilmesi durumunda sayıcı içeriğinin hiç artmadığı görülecektir. Bunun nedeni zamanlayıcının güncelleştirilmesi işleminin tarama çevriminin başında yapılmasıdır. Burada T33 zamanlayıcı biti tarama çevriminin başında 1 yapılmakta, sonraki süreçte zamanlayıcı komutu sayma komutundan önce işlendiği için sayma işlemi yapılmadan zamanlayıcı içeriği ve biti sıfırlanmaktadır.



Şekil 3.28 10 ms artımlı zamanlayıcı ile uzun süreli gecikme işlevini sağlayan örnek program

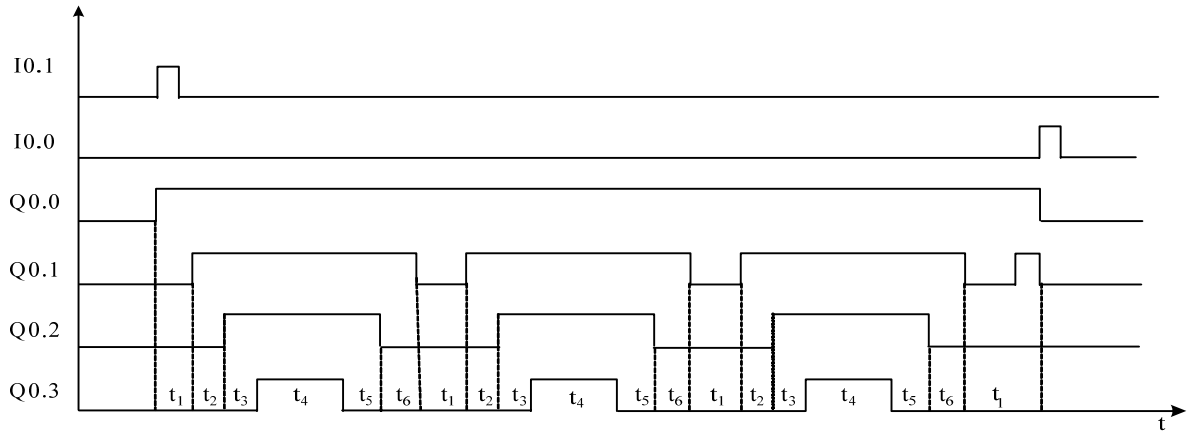
Bu programın istenen biçimde çalışması için, Şekil 3.27'deki gibi bir düzenleme yapılabilir. Diğer bir çözüm ise, sayma işleminin zamanlama işleminden önce yapılmasını sağlayan bir düzenleme yapmaktır. Şekil 3.29'da bu şekilde düzenlenmiş bir program verilmiştir. Bu biçimde yazılmış bir programla 10ms artımlı zamanlayıcılar için (tarama çevriminin başında güncelleştirilen zamanlayıcı türleri için) yardımcı bellek kullanımına gerek kalmaz.



Şekil 3.29 10 ms artımlı zamanlayıcı ile uzun süreli gecikme işlevini sağlayan örnek program

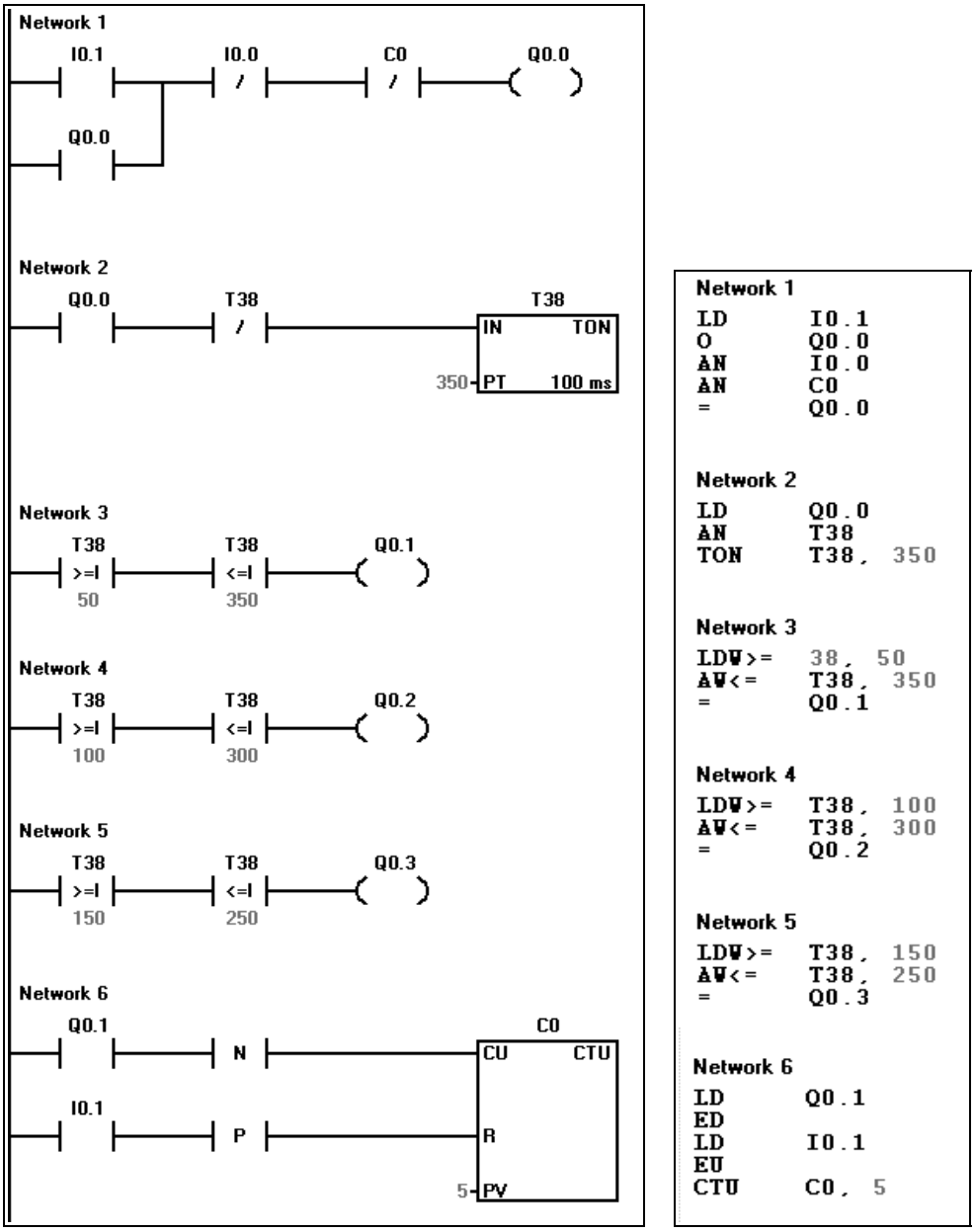
3.7.2 ZAMANLAYICI VE KARŞILAŞTIRMA İŞLEMLERİ İLE DEĞİŞİK ZAMANLAMA İŞARETLERİ ÜRETİLMESİ

Karşılaştırma komutları ve bir zamanlayıcı kullanılarak değişik süreli zamanlama işaretleri elde edilebilir. Örneğin, zamanlama diyagramı Şekil 3.30'da verilen bir kumanda sistemi için Şekil 3.31'deki program verilebilir. Burada t_1, t_2, t_3, t_5, t_6 süreleri 5s; t_4 süresi 10s alınmıştır.



Şekil 3.30 Değişik süreli işaretlere ilişkin zamanlama diyagramı

Şekil 3.31'deki programa göre, I0.1=1 yapıldığında, önce Q0.0=1, 5s sonra Q0.1=1, 10s sonra Q0.2=1 ve 15s sonra Q0.3=1 olmakta, 25s sonra önce Q0.3=0, 30s sonra Q0.2=0 ve 35s sonra Q0.1=0 olmaktadır. Bu zamanlama süreci 5 kez yinelendiğinde veya I0.0=1 yapıldığında sona ermektedir.



Şekil 3.31 Değişik süreli zamanlama işaretlerinin elde edilmesine ilişkin program

BÖLÜM 4
S7-200 CPU 22X
PROGRAM DENETİM İŞLEMLERİ

4.1 GİRİŞ

Program denetimi, program akışının veya komut yürütülüş sırasının değiştirilmesi veya düzenlenmesi olarak tanımlanabilir. Simatic S7-200 ailesi PLC'lerinde bu amaçla kullanılan çeşitli komutlar bulunur. Program denetiminde; atlama (JUMP), ardışıl denetim (SCR), program durdurma (STOP), döngü (FOR-NEXT) gözetim zamanlayıcısı denetimi (WDR), program sonlandırma (END), altprogram sonlandırma (RET) gibi komutlar kullanılır.

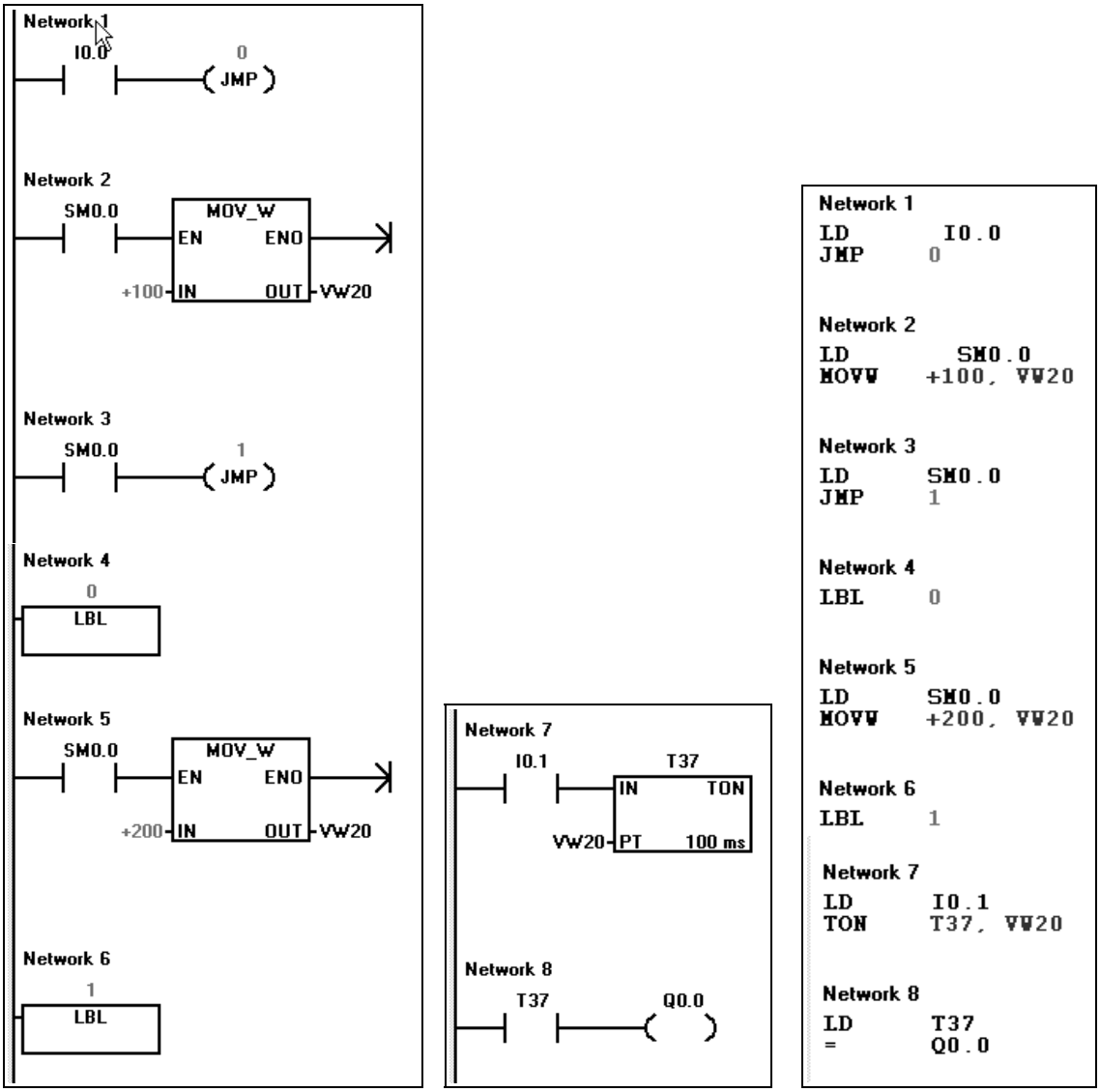
4.2 ATLAMA İŞLEMİ

Bir programda komutların yütütülüş sırası, JMP n atlama komutu ve LBL n etiket komutu çifti kullanılarak değiştirilebilir. JMP n komutu etkin olduğunda LBL n komutuna kadar yazılmış olan komutlar atlanır ve LBL n komutunu izleyen komut ile programın yürütülmesi sürdürülür.

JMP n atlama komutu ve varış noktasını belirten LBL n etiket komutu ana programda veya aynı altprogramlar içinde yazılır. Burada n bir tamsayı olup

CPU 212 işlemcisi olan PLC için n: 0 ~63,

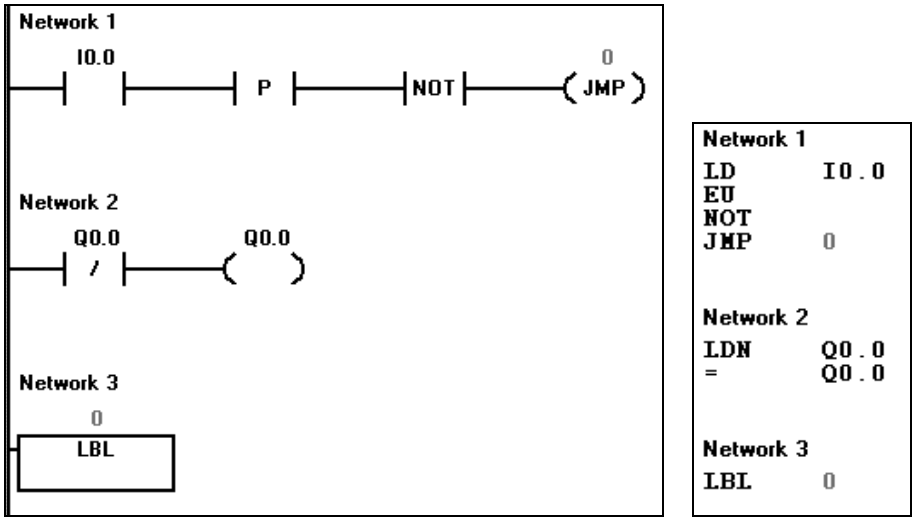
CPU 214, CPU 215 ,CPU 216 ve CPU 22x işlemcisi olan PLC'ler için n:0~255 değerleri geçerlidir. Şekil 4.1'de atlama komutlarının kullanımına ilişkin bir program örneği verilmiştir.



Şekil 4.1 Dallanma komutlarının kullanımına ilişkin bir program örneği

Bu programa göre, I0.0=0 yapıldığında (Network 2, 3, 6, 7 ve 8); I0.0 = 1 yapıldığında (Network 4, 5, 6, 7 ve 8) komutları işlenir. Böylece I0.0=0 yapıldığında T37 zamanlayıcısına PT=100 (10s) değeri, I0.0=1 yapıldığında T37 zamanlayıcısına PT=200 (20s) değeri yüklenmiş olur.

Şekil 4.2'de verilen örnek programda I0.0 girişinin her yükselen kenarında (Network 2)'deki komutlar yürütülür ve buna bağlı olarak Q0.0 işaretinin durumu değiştirilir.



Şekil 4.2 Çıkış durumunu değiştiren program örneği

Atlama işlemi, program içinde sonsuz döngüye neden olacak biçimde kullanılmamalıdır. Böyle bir durumda izin verilen maksimum tarama çevrimi süresi aşılacağından gözetim zamanlayıcısı (WDR) PLC'yi "STOP" durumuna götürür. Örneğin, geriye atlama işlemlerinde bu tür durumlarla karşılaşılabilir.

4.3 PROGRAM DENETİM KOMUTLARI

Program akışı <STOP>, <END>, <RET>, <WDR> <FOR-NEXT> gibi komutlar ile denetlenebilir.

STOP komutu: Etkin olduğunda, PLC'yi "RUN" durumundan "STOP" durumuna götürür.

END komutu: Etkin olduğunda, program tarama çevrimi sonlandırılır ve program baştan yürütülür.

RET komutu: Altprogramlarda kullanılır ve yürütüldüğünde altprogramdan çıkılır.

WDR komutu: Etkin olduğunda gözetim zamanlayıcısı içeriğini siler. Gözetim zamanlayıcısı (Watcdog Timer), program tarama çevrimi süresini denetleyen bir zamanlayıcıdır. Eğer bir tarama çevrimi 511ms'yi geçerse PLC'yi "STOP" durumuna götürür. Bu komut bir tarama çevrimindeki toplam işlem süresi 511ms'yi aşan programların yürütülmesinin gerekli olduğu durumlarda kullanılır.

FOR-NEXT komutu: Tarama çevrimi içinde döngü oluşturmak için kullanılır. Bu komut indeks (INDX) adresi, ilk değer (INIT) adresi ve son değer (FINAL) adresleri tanımlanarak, STL ile programlamada

FOR INDX, INIT, FINAL

.....

.....

NEXT

biçiminde yazılır. Komut yürütüldüğünde "INIT" değerinden başlanarak "FINAL" değerine kadar indeks 1 arttırılarak FOR-NEXT arasındaki komutlar yürütülür. Burada INDX, INIT, FINAL için

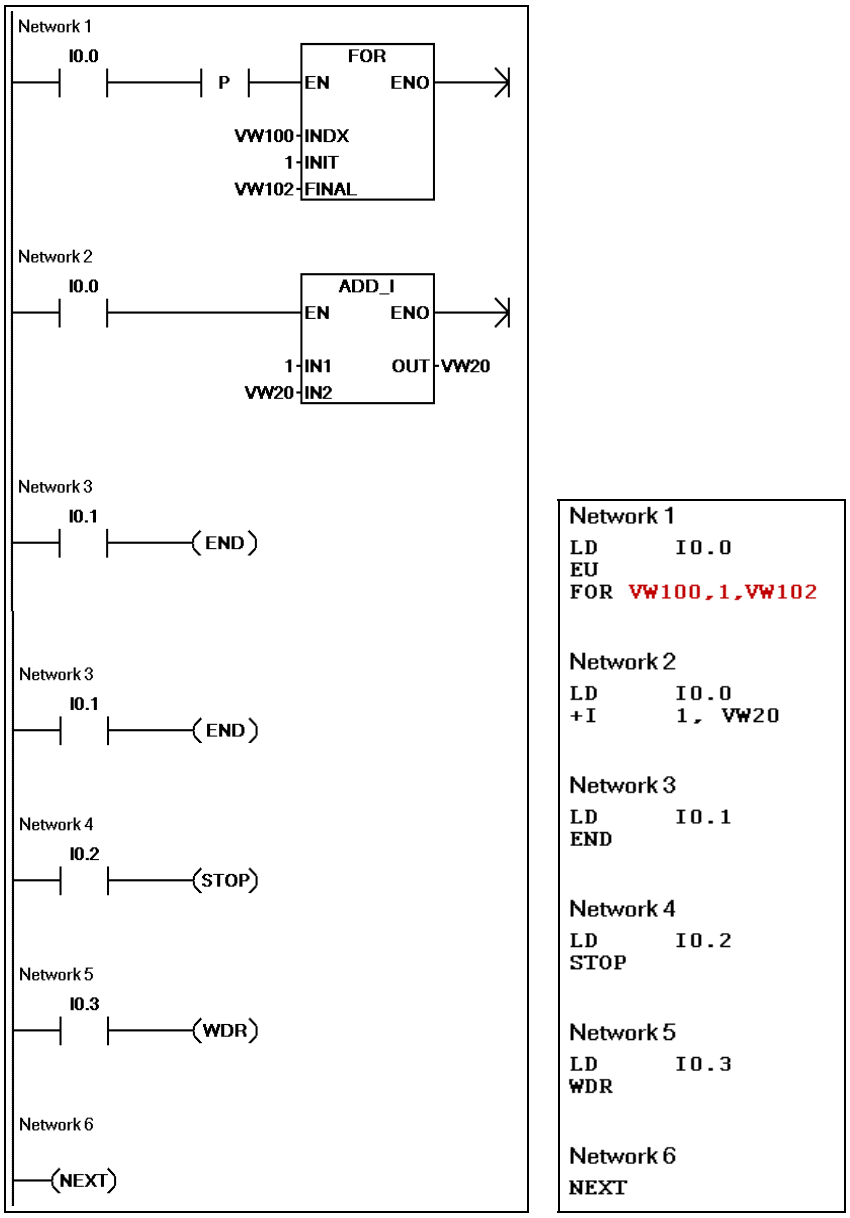
INDX:VW, IW, QW, MW, SW, SMW, LW, T, C, AC, *VD, *LD, *AC

INIT:VW, IW, QW, MW, SW, SMW, T, C, AC, LW, AIW, sabit, *VD, *LD, *AC

FINAL:VW, IW, QW, MW, SW, SMW, LW, T, C, AC, AIW, sabit, *VD, *LD, *AC

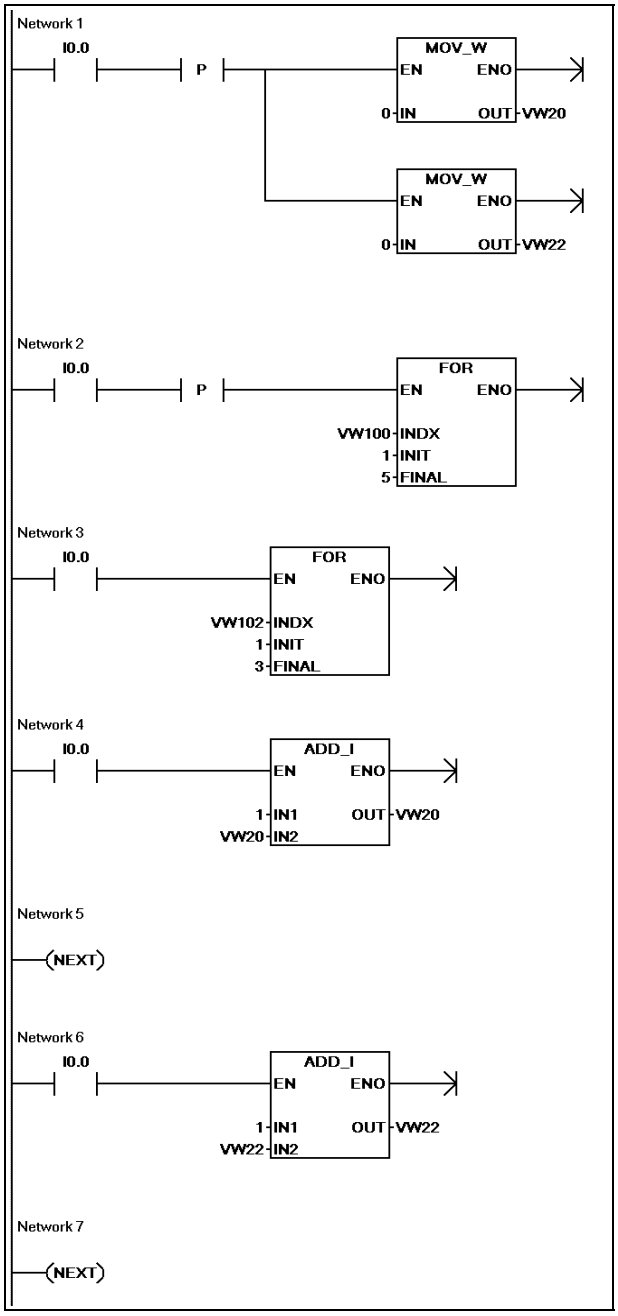
adresleri geçerlidir.

Şekil 4.3'te program denetim komutlarının kullanımına ilişkin bir program örneği verilmiştir. Bu programa göre I0.0=1 yapılırsa FOR ile NEXT arasındaki komutlar 10 kez işlenir. Bu durumda VW20 adresinin içeriği 10 artar. Çevrim sürecinde; I0.1=1yapılırsa END komutu etkin olur ve FOR-NEXT döngüsünden çıkılarak program başına dönülür. Eğer I0.2=1 yapılırsa PLC "STOP" durumuna geçer, I0.3=1 yapılırsa gözetim zamanlayıcısının içeriği silinir (Network 5).



Şekil 4.3 Denetim komutlarına ilişkin bir program örneği

Şekli 4.4'te iç içe FOR-NEXT komutlarının kullanımına ilişkin bir örnek program verilmiştir. Bu programa göre I0.0=1 yapıldığında (Network 4) 15 kez, (Network 6) 5 kez işlenir. Bu durumda VW20 adresinin içeriği 15 (her işlemde 1 arttırıldığından) , VW22 adresinin içeriği ise 5 olur.




```
Network 1
LD      I0.0
EU
MOVW    0, VW20
MOVW    0, VW22

Network 2
LD      I0.0
EU
FOR     VW100, 1, 5

Network 3
LD      I0.0
FOR     VW102, 1, 3

Network 4
LD      I0.0
+I      1, VW20

Network 5
NEXT

Network 6
LD      I0.0
+I      1, VW22

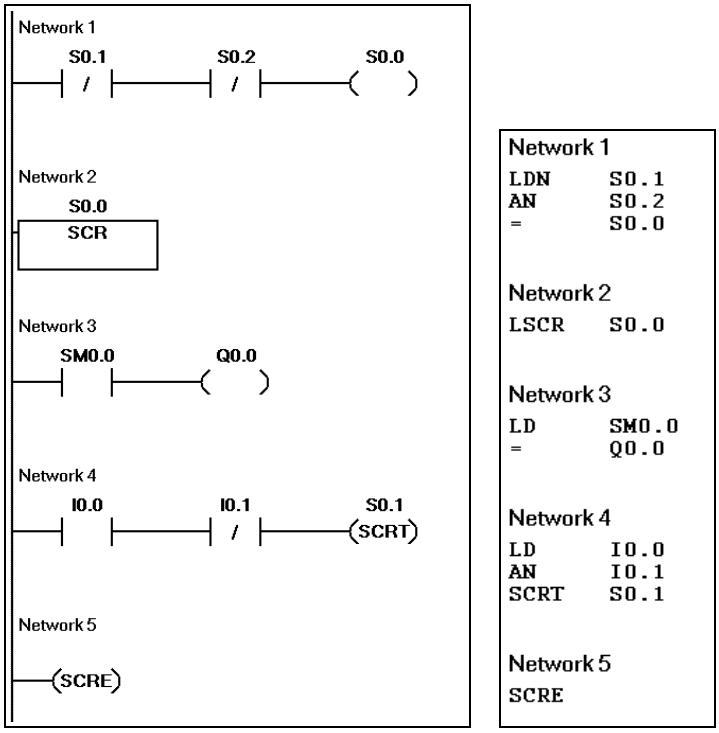
Network 7
NEXT
```

Şekil 4.4 İç içe FOR-NEXT döngülerine ilişkin program örneği

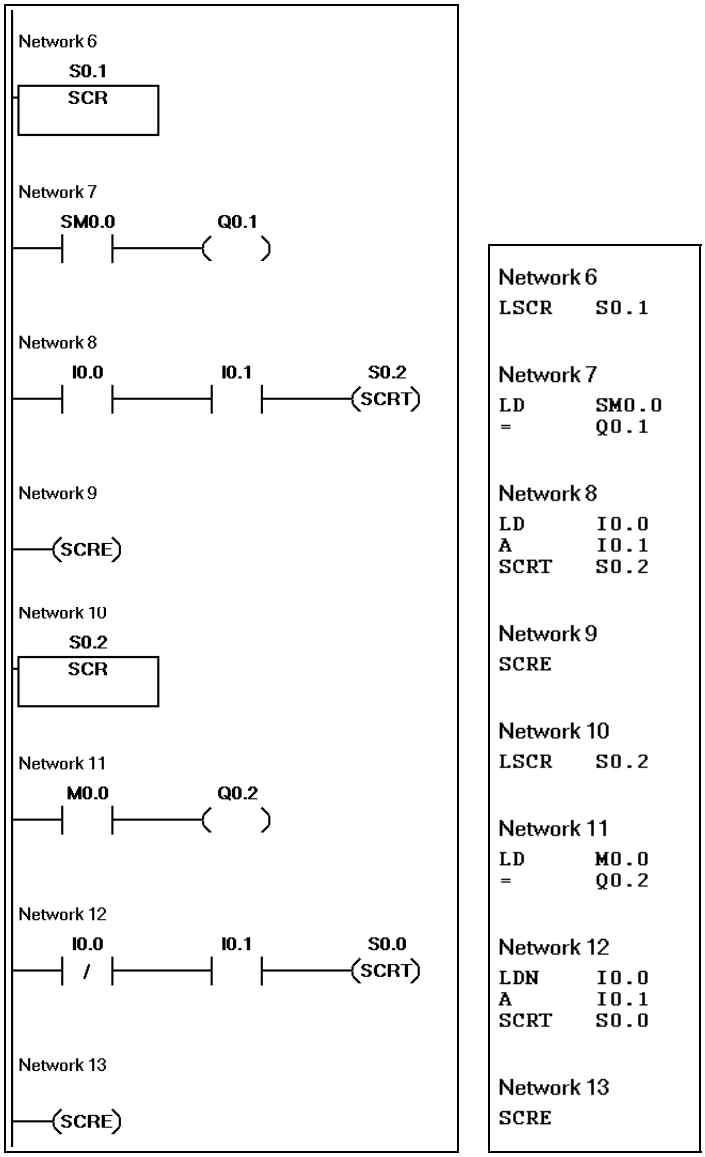
4.4 ARDIŞIL DENETİM KOMUTLARI

Ardışıl denetim komutları, ardışıl mantıksal devrelerinin doğrudan gerçeklenmesine olanak sağlayan komutlardır. Ardışıl devrelerde çıkışlar, girişlerin sırasına bağlı olarak değişir. Bu nedenle aynı giriş değerleri için farklı çıkış değerleri elde edilebilir.

Bir ardışıl devreye ilişkin durumu tanımlamak için durumun etkin olduğunu belirten (Sx.y: S0.0,.S0.1, .. , S31.7) bit adresi, etkin durumda işletilecek program bölgesinin başlangıç ve bitiş noktası, bir durumdan diğer duruma geçiş koşullarını belirten komutlar kullanılır. Bir duruma ilişkin bölgenin başlangıç noktası LSCR, son noktası SCRE, geçiş koşulu SCRT, koşullu durumdan çıkış CSCRE (STL) komutları ile sağlanır. Şekil 4.5'te ardışıl denetim komutlarının kullanımına ilişkin bir program verilmiştir.



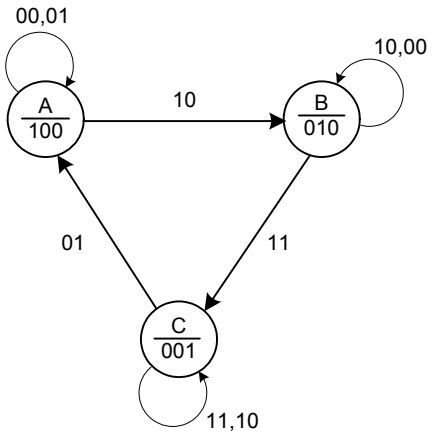
Şekil 4.5 Ardışıl denetim komutlarının kullanımına ilişkin bir program örneği (1/2)



Şekil 4.5 Ardışıl denetim komutlarının kullanımına ilişkin bir program örneği (2/2)

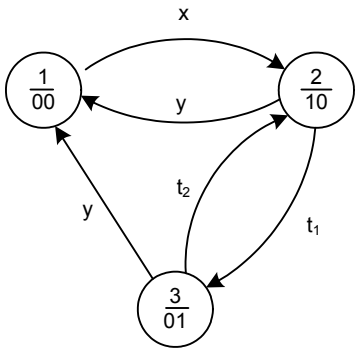
Şekil 4.5'te verilen programda 3 durum vardır. Birinci durum için S0.0, ikinci durum için S0.1 ve üçüncü durum için S0.2 adresleri seçilmiştir. Bu programa göre S0.0=1 yapıldığında (Network 2) ile (Network 5), S0.1=1 yapıldığında (Network 6) ile (Network 9), S0.2=1 yapıldığında (Network 10) ile

(Network 13) arasındaki komutlar işletilir. Her bölge (LSCR ile başlayan ve SCRE ile sonlanan alan) içinde yer alan SCRT komutu etkin yapıldığında, o bölgeye ilişkin Sx.y biti sıfırlanır ve SCRT komutu ile belirtilen Sx.y biti 1'e kurulur. Böylece bir bölgeden diğer bölgeye geçiş yapılmış olur. Bu programda (Network 1)'de yazılan komutla eğer devre S0.1 ve S0.2 durumunda değilse S0.0 durumuna kurulması sağlanır. S0.0=1 iken (Network 2)'den (Network 5)'e kadar yazılan komutlar yürütüldüğünden Q0.0=1 olur. Eğer bu durumda iken I0.0=1 ve I0.1=0 yapılırsa SCRT S0.1 komutu işletileceğinden S0.0=0 ve S0.1=1 olur. Bu durumda S0.1 bölgesi (Network 6- Network 9) etkin olur ve Q0.1=1 değerini alır. S0.0 bölgesi etkin durumdan çıktığından Q0.0=0 olur. S0.1 bölgesi etkin iken I0.0=1 ve I0.1=1 yapılırsa SCRT S0.2 komutu işletileceğinden S0.1=0 ve S0.2=1 olur. Bu durumda S0.1 bölgesi (Network 10- Network 13) etkin olur ve Q0.1=2 değerini alır. S0.1 bölgesi etkin durumdan çıktığından Q0.1=0 olur. S0.2 bölgesi etkin iken I0.0=0 ve I0.1=1 yapılırsa SCRT S0.0 komutu işletileceğinden S0.2=0 ve S0.0=1 olur. Bu durumda S0.0 bölgesi etkin olur ve başlangıç durumuna dönülür. Burada 2 giriş ve 3 çıkışlı bir ardışıl devre gerçeklenmiştir. Öyle ki başlangıçta I0.0=0 ve I0.1=0 iken Q0.0=1, başlangıç durumunda iken I0.0=1 ve I0.1=0 yapılırsa Q0.1=1, daha sonra I0.0=1 ve I0.1=1 yapıldığında Q0.2=1 değerlerini almaktadır. Bu devre eşzamansız bir ardışıl devre olarak göz önüne alındığında Şekil 4.6'da verilen durum geçiş grafı ile tanımlanabilir. Bu grafta daire içinde durum simgesi ve bu durumda iken çıkışların aldığı değerler (durum/çıkış), yönlü yay ve doğrular girişlerin aldığı değerleri gösterir. Bu grafa göre başlangıçta devre A durumunda ve çıkışlar Q0.0, Q0.1, Q0.2=1, 0, 0 değerindedir. Bu durumda iken I0.0, I0.1= 00 veya 01 olursa devre durumunu korumakta, I0.0, I0.1=10 olursa devre B durumuna geçmekte ve çıkışlar Q0.0, Q0.1, Q0.2=0, 1, 0 olmaktadır. Devre B durumunda iken girişler I0.0, I0.1= 1,1 olursa devre C durumuna geçmekte ve çıkışlar Q0.0,Q0.1,Q0.2=0,0,1 olmaktadır. C durumunda iken I0.0, I0.1= 0,1 yapılırsa devre A başlangıç durumuna dönmektedir.



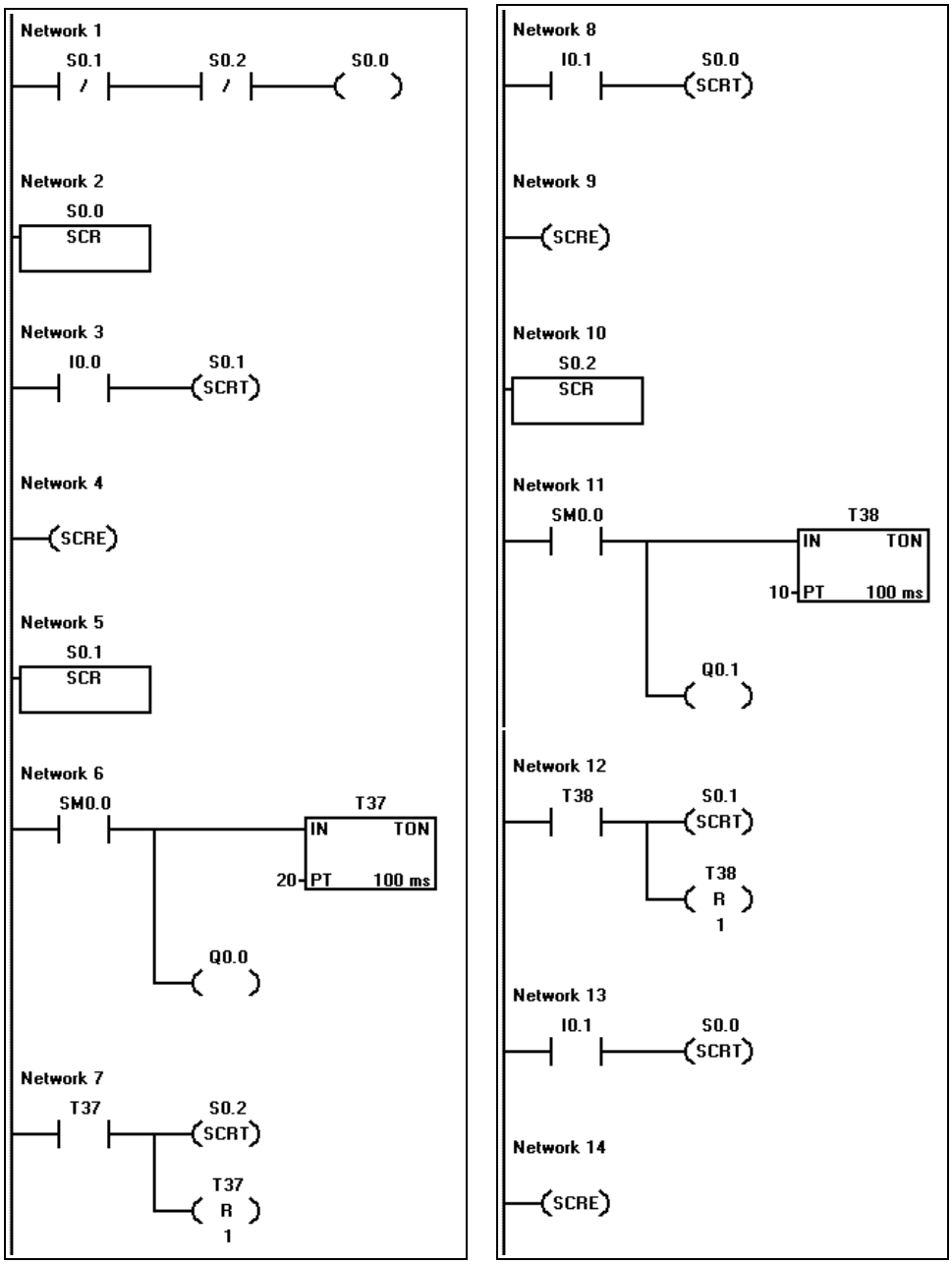
Şekil 4.6 Durum geçiş grafı

Durum geçiş grafı, Şekil 4.7'deki gibi düzenlenen bir kumanda devresine ilişkin program örneği Şekil 4.8'deki gibi verilebilir. Bu grafa göre, başlangıç durumunda (1) iken x düğmesine basıldığında yeni bir duruma (2) geçilmekte; bu durumda iken y düğmesine basılırsa başlangıç durumuna veya t_1 süre sonra (3) durumuna geçilmektedir. (3) durumunda iken y düğmesine basılırsa başlangıç durumuna veya t_2 süresi dolarsa tekrar (2) durumuna dönülmektedir. K1,K2 çıkış işaretleri (2) durumunda iken K1,K2=1,0; (3) durumunda iken K1,K2=0,1 olarak üretilmektedir.



Şekil 4.7 Durum geçiş grafı

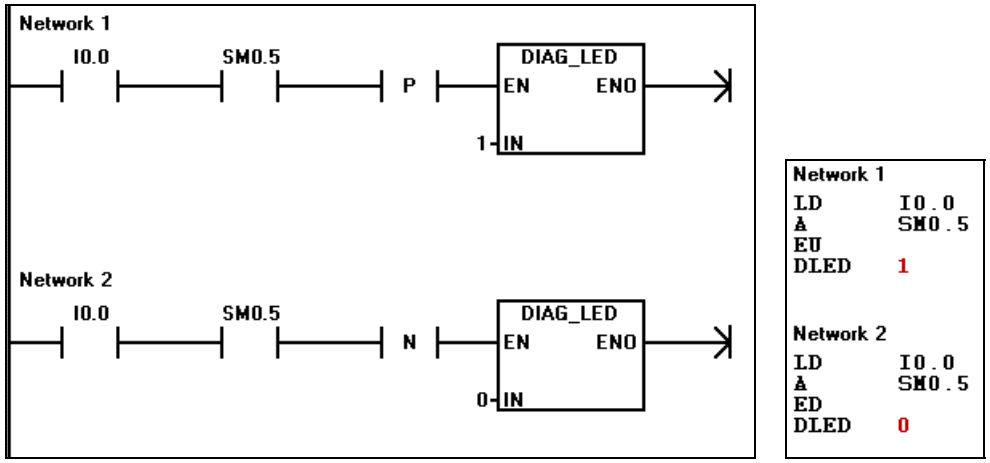
Durum geçiş grafındaki simgelere; x =I0.0, y=I0.1, K1=Q0.0, K2=Q0.1, t_1=3s (T37), t_2=2s (T38), (1) durumu=S0.0, (2) durumu=S0.1 ve (3) durumu=S0.2 atamaları yapılırsa merdiven mantığı programı Şekil 4.8'deki gibi olur.



Şekil 4.8 Ardışıl denetim komutlarına ilişkin bir program örneği

4.5 TANILAMA (DIAGNOSTIC) İŞARETİ ÜRETME

Yeni nesil CPU 22x işlemcili PLC üzerinde bulunan SF/DIAG lambası komutla denetlenebilir. Olağan durumlarda sistem hatası oluştuğunda yanan bu lamba, sistem hatası kapsamı içinde olmayan başka hatalar için de kullanılabilir. Bunun için DLED komutu kullanılır. Bu komutu izleyen 8 bitlik sayısal değer 0'dan farklı olduğunda lamba yanar, 0 ise lamba söner. Şekil 4.9'da bu komutun kullanımına ilişkin bir program örneği verilmiştir. Bu programa göre I0.0=1 yapıldığında, içeriği 0.5s 0 ve 0.5s 1 değerini alan SM0.5 özel bellek adresi nedeniyle SF/DIAG lambası periyodik olarak yanıp söner. SM0.5 adresinin içeriği 0' dan 1'e değiştiğinde DLED 1 komutu işletilir ve lamba yanar, 1'den 0'a değiştiğinde DLED 0 komutu işletilir ve lamba söner.



Şekil 4.9 Tanılama komutunun kullanımına ilişkin program örneği

4.6 ALTPROGRAM KULLANIMI

Altprogramlar, belirli bir işlevi yerine getirmek üzere yazılan program parçalarıdır. Altprogramlar büyük programların, parçalara bölünerek yazılması için de kullanılır. Böylece izleme, geliştirme ve değiştirme için daha elverişli bir program yapısı elde edilir.

Altprogramlar, altprogram için ayrılmış ayrı bir sayfaya yazılır (SBR_0, SBR_1, SBR_2…). Bir altprogram alanı açmak için, komut listesindeki "Program Block" > "SBR_0" , program editöründe Edit >Insert>Subroutine veya ana program sayfasında fare sağ tuşu ile Insert>Subroutine seçilir. Altprogramlar açıldıktan sonra, komut ağacındaki "call subroutines" seçeneği altında görünür. Buradan herhangi bir komut gibi alınarak ana programda veya altprogramlarda kullanılır. S7-200 CPU 212 işlemcili PLC'de 16 altprogram (n: 0 - 15); S7-200 CPU 214, CPU 215, CPU 214, CPU 221, CPU 222, CPU 224 işlemcili PLC'lerde 64 altprogram (n:0 - 63), CPU 224XP ve CPU 226XM işlemcili PLC'lerde 128 altprogram (n:0 - 127) yazılabilir. Bir altprogramdan diğer bir altprogramı çağıran yapılar kullanılabilir, ancak iç içe en çok 8 altprogram çağrılabilir.

Altprogramlar, bir programı parçalara ayırmak için parametresiz olarak, belirli bir işlevi yerine getirmek ve genel amaçlı kullanım için parametreli olarak kullanılır. Parametresiz kullanımda giriş ve çıkışlar için herhangi bir tanımlama yapılmaz. Parametreli kullanımda giriş (IN), çıkış (OUT) ve giriş-çıkış (IN-OUT) parametreleri altprogram yerel değişken tablosuna (Local Variable Table) yazılır. Parametre geçişleri bu tabloya göre yapılır. 64 bayt boyutunda olan yerel bellek alanının (L bellek alanı) 60 baytlık bölümü bu amaçla kullanılır. Parametre sayısı en çok 16 olabilir.

Yerel değişken tablosunda tanımlanan giriş (IN), giriş-çıkış (IN_OUT), çıkış (OUT) ve geçici (TEMP) değişkenleri için BOOL, BYTE, WORD, INT, DWORD, DINT, REAL, STRING veri türleri seçilebilir.

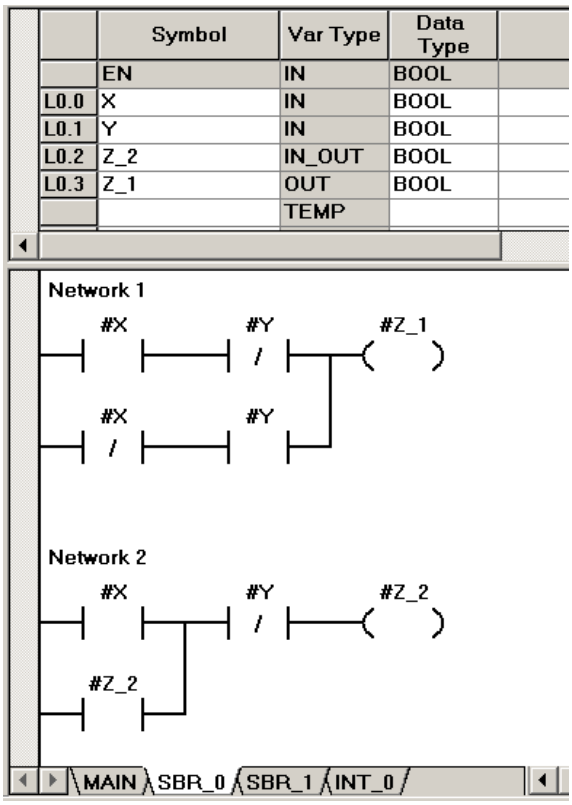
Bir altprograma ilişkin giriş, çıkış ve giriş-çıkış değişkenleri altprogram yürütüldüğünde yerel bellek alanına yazılır ve işlemler yerel bellek adresleri üzerinde yapılır. Altprogramdan çıkıldığında çıkış (OUT) ve giriş-çıkış (IN_OUT) değişkenlerine ilişkin verilerin değerleri çağırılan programa aktarılır.

Şekil 4.10'da bir mantıksal işleme ilişkin bir altprogram örneği verilmiştir. Burada X ve Y değişkenleri giriş, Z_2 değişkeni giriş-çıkış ve Z_1 değişkeni çıkış olarak tanımlanmıştır. Bu altprogram

$$Z_1=X\bullet Y' + X'\bullet Y \text{ ve } Z_2=(X+Z_2)\bullet Y'$$

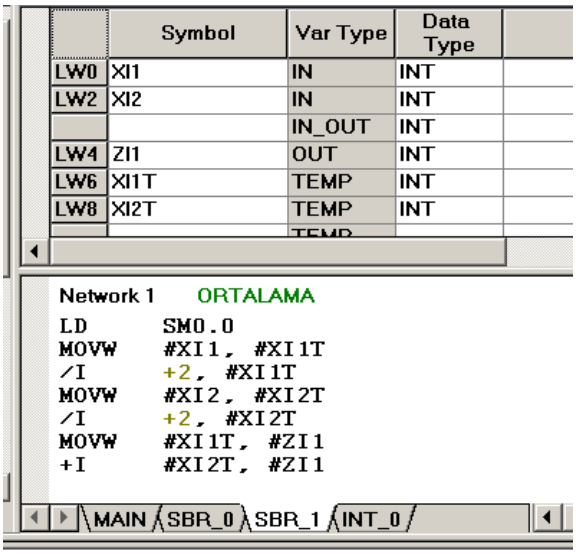
mantık fonksiyonlarını gerçekler. Burada Z_1 değişkeni X ve Y değişkenlerinin işlem anındaki değerlerine göre, Z_2 değişkeni ise X ve Y değişkeninin işlem anındaki ve Z_2 değişkeninin bir

tarama önceki değerine bağlı olarak hesaplanır. Z_2 değişkeninin hesaplanmasında, bu değişkene ilişkin önceki değerin de kullanılması gerektiğinden, bu değişken IN_OUT değişken türünde seçilir. Böylece Z_2'nin önceki değeri çağrılan programa aktarılır ve altprogramın bir sonraki çağrılmasında kullanılır.



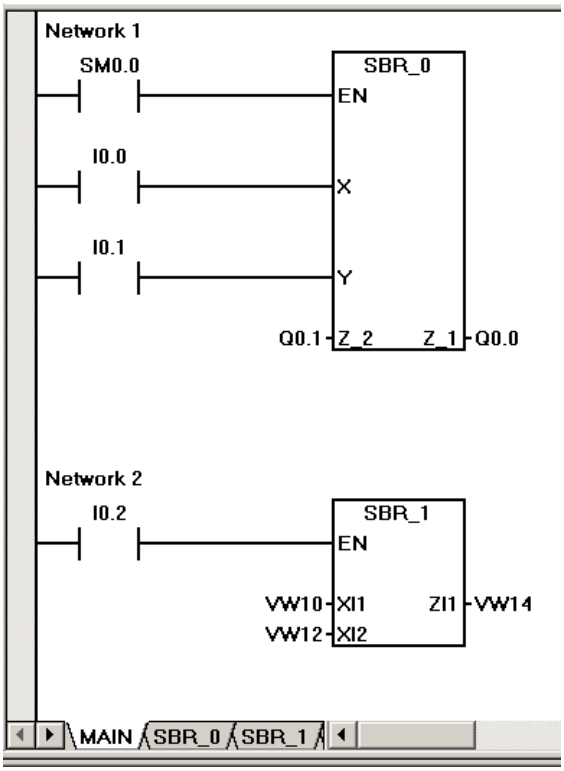
Şekil 4.10 Bir altprogram örneği

Altprogram kullanımına ilişkin diğer bir örnek Şekil 4.11'de verilmiştir. Burada iki işaretli tamsayının (INT) ortalama değeri alınmaktadır. ZI1=(XI1+XI2)/2 ifadesinin gerçeklendiği bu programda taşma olmaması için (değer aralığı dışına çıkılmaması), önce XI1 2'ye bölünür ve sonuç geçici değişken türünde (TEMP) bir adrese (burada XI1T (LW6)) yazılır. X12 değişkeni için de benzer işlem yapılır. TEMP türünde değişkenler, burada yapıldığı gibi, hesaplama sürecinde ara değerlerin saklanması amacıyla kullanılır.



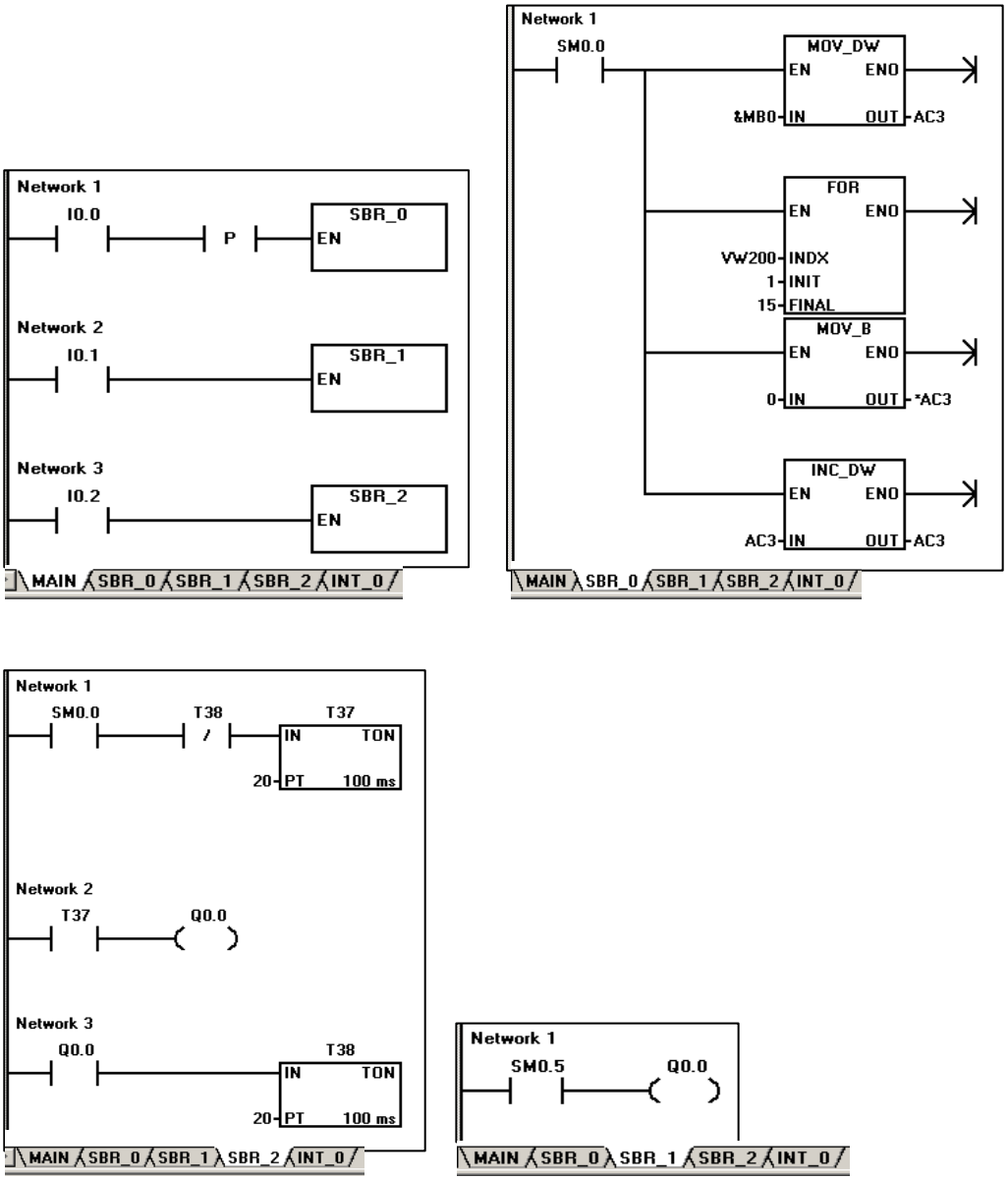
Şekil 4.11 Ortalama işlemi yapan altprogram

Şekil 4.10 ve Şekil 4.11'de verilen altprogramlar Şekil 4.12'deki gibi ana programdan çağrılarak işletilir. Bu program işletildiğinde I0.0 ve I0.1 girişleri SBR_0 altprogramına göre değerlendirilir ve sonuçlar Q0.0 ve Q0.1 çıkış adreslerine yazılır. I0.2=1 yapıldığında SBR_1 altprogramı işletilir ve bu durumda VW10 adresinin içeriği ile VW12 adresinin içeriği toplanır; sonuç VW14 adresine yazılır.



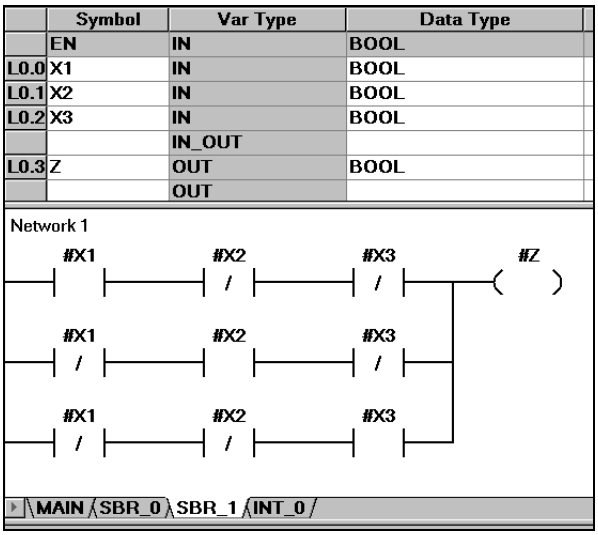
Şekil 4.12 Altprogramların ana programdan çağrılmasına ilişkin program örneği

Altprogramlar, parametresiz olarak, programların parçalara bölünerek yazılmasında da yaygın olarak kullanılır. Şekil 4.13'te bu amaçla yazılan bir program örneği verilmiştir. Burada SUB_0 alt programı I0.0 girişinin yükselen kenarında işletilir ve bu altprogramın işletilmesi ile MB0'dan MB14'e kadar olan bellek alanı sıfırlanır. SUB_1 alt programı I0.1=1 durumunda iken işletilir ve bu alt programın işletilmesi ile Q0.0 çıkışı 0.5 saniye 0, 0.5saniye 1 değerini alır. SUB_2 altprogramı I0.2=1 iken işletilir ve bu altprogramın işletilmesi ile Q0.0 çıkışı 2 saniye 1, 2 saniye 0 değerini alır.

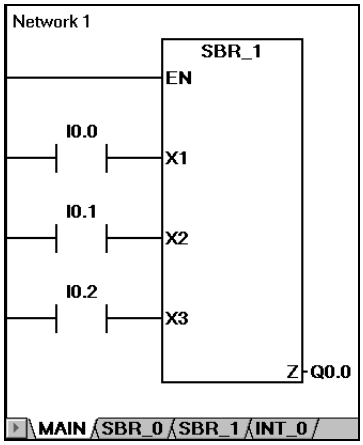


Şekil 4.13 Altprogramların parametresiz kullanımına ilişkin bir program örneği

Altprograma ilişkin başka bir örnek Şekil 4.14'te gösterilmiştir. Bu altprograma göre, 3 girişten yalnız biri etkin olduğunda çıkış etkin olur. Bu altprogram Şekil 4.15'de gösterildiği gibi ana programdan işletilir.



Şekil 4.14 Üç girişten yalnız biri etkin iken çıkışı etkin olan altprogram örneği



Şekil 4.15 Altprogramın ana programdan işletilmesi

BÖLÜM 5
SAYISAL İŞLEMLER

5.1 GİRİŞ

Sayısal kontrolör, sayısal süzgeç ya da sayısal işlem gerektiren kumanda sistemleri 16 bit veya 32 bitlik sayısal verileri işleyen komutlarla gerçeklenir. Sayısal işlemlerde veri aktarma (taşıma), toplama, çıkarma, çarpma, bölme ve çeşitli matematiksel fonksiyonların işlevlerini yerine getiren matematiksel işlem komutları kullanılır.

Sayısal işlemler, 16 ve 32 bitlik işaretli tamsayılar ve 32 bitlik gerçek (reel) sayılar üzerinde yapılır. 16 bitlik işaretli tamsayılar

-32768 (H8000) ile 32767 (H7FFF),

32 bitlik işaretli tamsayılar

-2,147,483,654 (H8000,0000) ile 2,147,483,653 (H7FFF,FFFF)

Reel sayılar

(-3.402823E+38) ile (+3.402823E+38)

değerleri arasında değişir.

Sayısal işlemlerde, sonucun geçerliliği ve durumu SMB1 özel durum baytına yazılır. Eğer, komut yürütüldüğünde

SM1.0=1 olursa sonuç sıfır

SM1.1=1 olursa taşma var, sonuç geçersiz

SM1.2=1 olursa sonuç negatif,

SM1.3=1 olursa sıfıra bölme var,

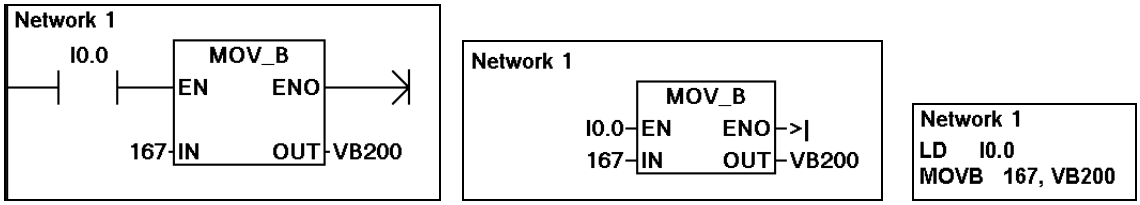
durumları belirlenir.

5.2 VERİ AKTARMA İŞLEMLERİ

Veri aktarma işlemleri 8 bit, 16 bit, 32 bit tamsayı ve reel veriler arasında değişik komutlar kullanılarak yapılır. Veri aktarma komutları yürütüldüğünde (etkin olduğunda) IN adresindeki veri OUT adresindeki veri alanına kopyalanır.

5.2.1 8, 16, 32 BİTLİK VERİ AKTARMA İŞLEMLERİ

Şekil 5.1'de 8 bitlik verilerin aktarılmasına ilişkin merdiven mantığı, FBD ve STL komutları ve geçerli veri adresleri verilmiştir. Bu komut yürütüldüğünde (I0.0=1), 167 işaretsiz tamsayısı VB200 adresine kopyalanır.

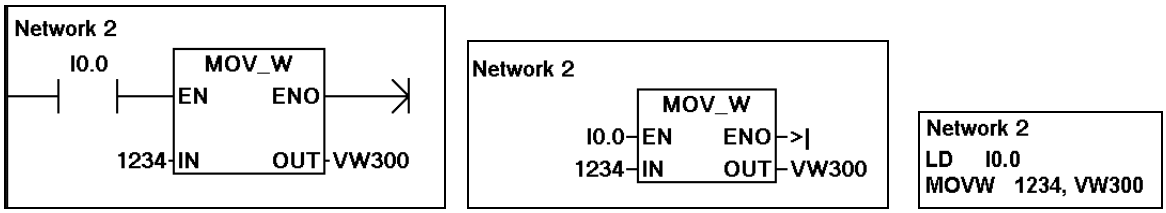


IN :VB, IB, QB, MB, SB, SMB, LB, AC, Sabit (0-255), *VD, *LD, *AC

OUT :VB, IB, QB, MB, SB, SMB, LB, AC, *VD, *LD, *AC

Şekil 5.1 8 bitlik verilerin aktarılması

Şekil 5.2'de 16 bitlik verilerin aktarılmasına ilişkin komutların her üç dildeki biçimi ve geçerli veri adresleri gösterilmiştir. Bu komut yürütüldüğünde (I0.0=1), 1234 işaretli tamsayısısı VW300 adresine kopyalanır.

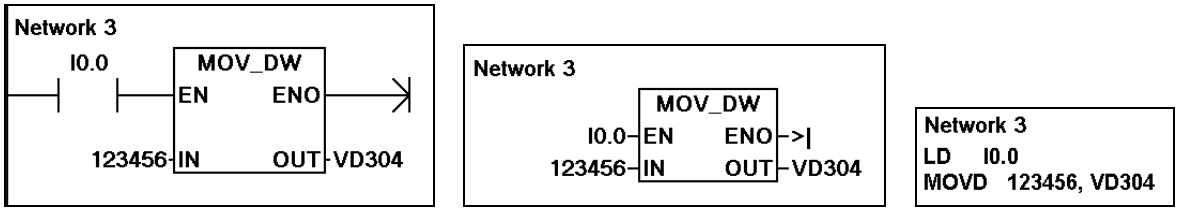


IN :VW, IW, QW, MW, SW, SMW, LW, T, C, AIW, Sabit, AC, *VD, *AC, *LD

OUT :VW, T, C, IW, QW, SW, MW, SMW, LW, AC, AQW, *VD, *AC, *LD

Şekil 5.2 16 bitlik verilere ilişkin aktarma işlemi

Şekil 5.3'te 32 bitlik verilerin aktarılmasına ilişkin komut ve geçerli veri adresleri gösterilmiştir. Bu komut yürütüldüğünde (I0.0=1), 123456 işaretli tamsayısısı VD304 adresine kopyalanır.

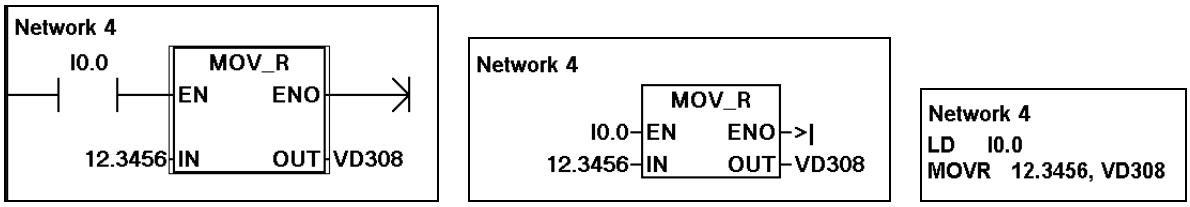


IN :VD, ID, QD, MD, SD, SMD, LD, HC, &VB, &IB, &QB, &MB, &SB, &T, &C, &SMB, &AIW, &AQW AC, Sabit, *VD, *LD, *AC

OUT :VD, ID, QD, MD, SD, SMD, LD, AC, *VD, *LD, *AC

Şekil 5.3 32 bitlik verilere ilişkin aktarma işlemi

Şekil 5.4'te reel biçimdeki verilerin aktarılmasına ilişkin komut ve geçerli veri adresleri gösterilmiştir. Bu komut yürütüldüğünde (I0.0=1), 12.3456 reel sayısı VD308 adresine kopyalanır.



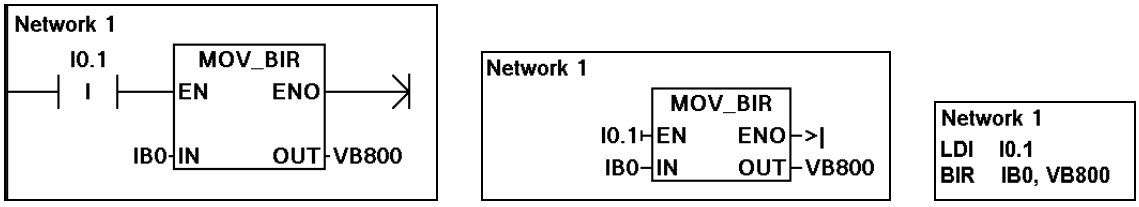
IN :VD, ID, QD, MD, SD, SMD, LD, AC, Sabit, *VD, *LD, *AC

OUT :VD, ID, QD, MD, SD, SMD, LD, AC, *VD, *LD, *AC

Şekil 5.4 Reel verilere ilişkin aktarma işlemi

5.2.2 ANINDA OKUMA İŞLEMİ

Program tarama çevriminin herhangi bir yerinde giriş biriminin doğrudan okunması ve 8 bitlik bir alana aktarılması için Şekil 5.5'teki komut ve adresler kullanılır. Bu komut yürütüldüğünde (I0.1=1) IB0 (I0.7, I0.6, …., I0.0) girişleri anında okunur ve VB800 adresine yazılır.



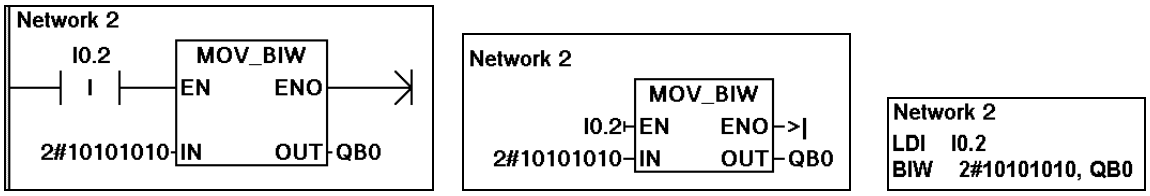
IN : IB, *VD, *LD, *AC

OUT : VB, IB, QB, MB, SB, SMB, LB, AC, *VD, *AC, *LD

Şekil 5.5 8 bitlik giriş biriminin anında okunması işlemi

5.2.3 ANINDA AKTARMA İŞLEMİ

Program tarama çevriminin herhangi bir yerinde 8 bitlik bir verinin doğrudan çıkış birimine aktarılması için Şekil 5.6'da verilen komutlar kullanılır. Bu programa göre I0.2=1 yapıldığı anda ikilik 2#10101010 değerleri doğrudan adresi QB0 (Q0.7, Q0.6, …., Q0.0) olan çıkış birimi ve çıkış görüntü belleğine yazılır.



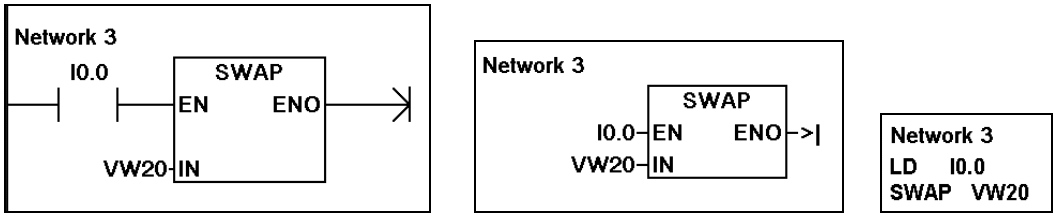
IN VB, IB, QB, MB, SB, SMB, LB, AC, Sabit, *VD, *AC, *LD

OUT QB, *VD, *LD, *AC

Şekil 5.6 8 bitlik veriyi anında çıkış birimine aktarma işlemi

5.2.4 YER DEĞİŞTİRME İŞLEMİ

Boyutu 16 bit olan bir verinin düşük anlamlı 8 biti ile yüksek anlamlı 8 biti arasında yer değiştirmek için Şekil 5.7'de verilen komut kullanılır. Bu programa göre I0.0=1 yapıldığında, 16 bitlik VW20 bellek alanının içerdiği 8 bitlik VB20 ve VB21 alanlarının içeriği yer değiştirir. Örneğin, komut işletilmeden önce VW20 adresinin içeriği onaltılık tabanda 16#FF00 ise, komut işletildiği anda VW20 adresinin içeriği 16#00FF olur.

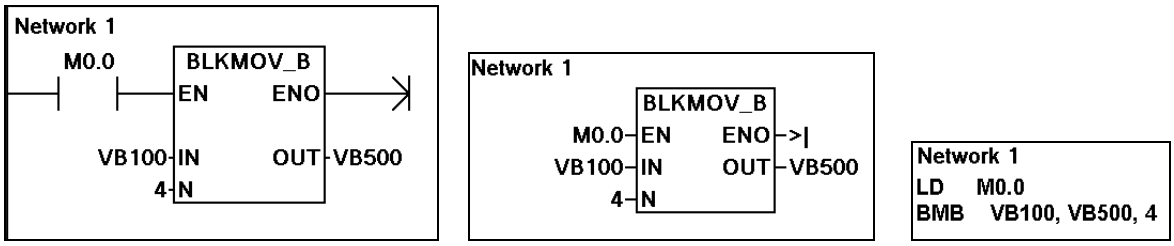


IN : VW, IW, QW, MW, SW, SMW, T, C, LW, AC, *VD, *AC, *LD

Şekil 5.7 16 bitlik veriler için yer değiştirme işlemi

5.2.5 VERİ BLOĞU AKTARMA İŞLEMİ

8 bitlik veri bloğunu aynı boyutta başka bir alana aktarmak için kullanılan komut ve geçerli adresler Şekil 5.8'de gösterilmiştir. Burada N bloktaki veri sayısı olup en çok 255 değerini alır. Bu komut ile IN adresinden başlanarak N adet ardışık veri OUT başlangıç adresini izleyen alana kopyalanır. Bu programa göre M0.0=1 olduğunda VB100, VB101, VB102, VB103 adreslerindeki veriler VB200, VB201, VB202, VB203 adreslerine yazılır.



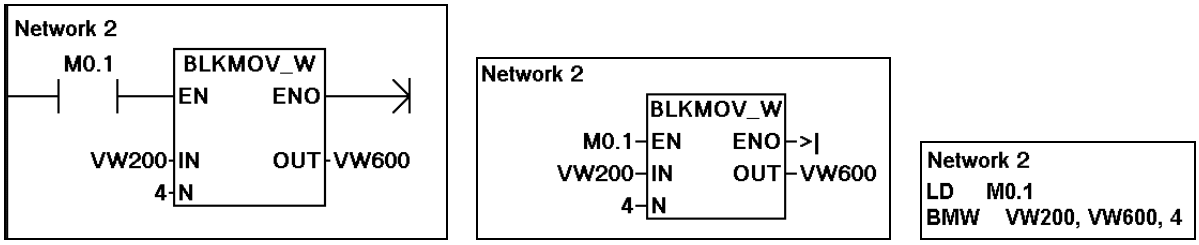
IN : VB, IB, QB, MB, SB, SMB, LB, *VD, *AC, *LD

N : VB, IB, QB, MB, SB, SMB, LB, AC, Sabit, *VD, *AC, *LD

OUT : VB, IB, QB, MB, SB, SMB, LB, *VD, *AC, *LD

Şekil 5.8 8 bitlik veri bloğu aktarma işlemi

16 bitlik veri bloğunu aynı boyutta başka bir alana aktarmak için Şekil 5.9'daki komut kullanılır. Bu komut yürütüldüğünde IN başlangıç adresini izleyen N (en çok 255) adet ardışık veri OUT başlangıç adresini izleyen alana kopyalanır. Bu programa göre M0.1 biti 1 yapıldığında VW200, VW202, VW204 ve VW206 adreslerindeki 4 adet 16 bitlik veri VW600, VW602, VW604 ve VW606 adreslerine yazılır.



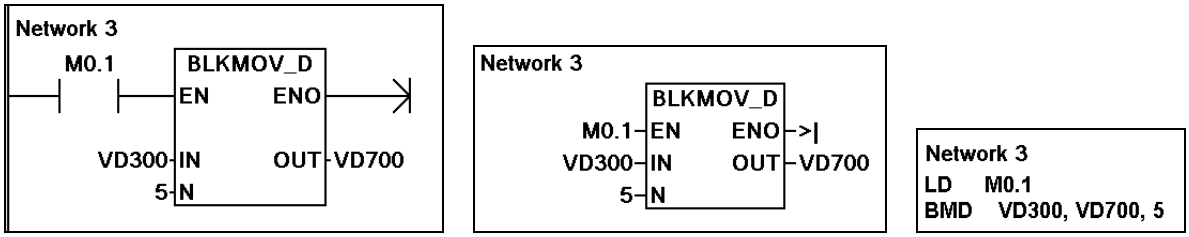
IN : VW, IW, QW, MW, SW, SMW, LW, T, C, AIW, *VD, *LD, *AC

N : VB, IB, QB, MB, SB, SMB, LB, AC, Sabit, *VD, *LD, *AC

OUT : VW, IW, QW, MW, SW, SMW, LW, T, C, AQW, *VD, *LD, *AC

Şekil 5.9 16 bitlik veri bloğu aktarma işlemi

32 bitlik veri bloğunu aynı boyuttaki başka bir alana aktarmak için Şekil 5.10'daki komut kullanılır. Bu komut ile IN başlangıç adresini izleyen N adet ardışık 32 bitlik veri, OUT başlangıç adresini izleyen alana kopyalanır. Şekil 5.10'daki programa göre M0.1=1 olduğunda VD300, VD304, VD308, VD312, VD316 adreslerindeki 5 adet 32 bitlik veri; VD700, VD704, VD708, VD712 ve VD716 adreslerine yazılır.



IN : VD, ID, QD, MD, SD, SMD, LD, *VD, *AC, *LD

OUT : VD, ID, QD, MD, SD, SMD, LD, *VD, *AC, *LD

N : VB, IB, QB, MB, SB, SMB, LB, AC, Sabit, *VD, *AC, *LD

Şekil 5.10 32 bitlik veri bloğu aktarma işlemi

5.3 İŞARETLİ TAMSAYILAR ÜZERİNDE İŞLEMLER

5.3.1 16 BİTLİK İŞARETLİ TAMSAYILAR ÜZERİNDE İŞLEMLER

İşaretli tamsayılar üzerinde toplama işlemine ilişkin komut Şekil 5.11'de gösterilmiştir. Bu komut yürütüldüğünde

OUT:=IN1 + IN2

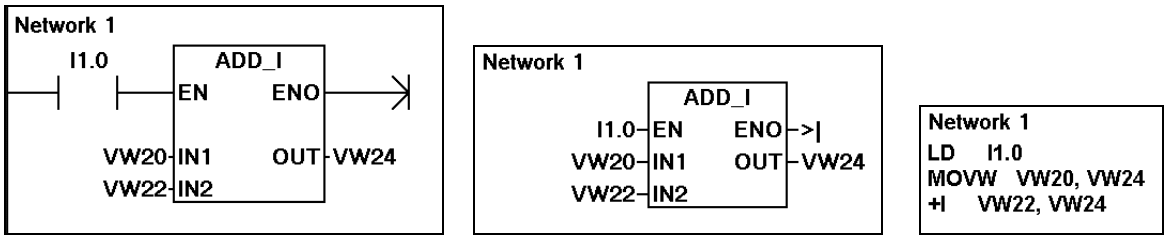
işlemi gerçeklenir. Şekil 5.11'deki programa göre I1.0=1 yapıldığında VW20 adresindeki işaretli tamsayı ile VW22 adresindeki işaretli tamsayı toplanır ve VW24 adresine yazılır. Eğer sonuç, işaretli tamsayı değer aralığı [-32768(H8000) ile +32767 (H7FFF)] dışında ise (taşma durumu) SM1.1=1 olur.

Örneğin, VW20=32767 (H7FFF) ve VW22=1 (H0001) olan iki sayı toplandığında elde edilen sonuç VW24=32768(H8000) ve SM1.1=1 (taşma) , SM1.2=1 (sonuç negatif) olur.

Eğer, VW20=2768 (H8000) ve VW22= (HFFFF) ise sonuç VW24 = 32767 (H7FFF) ve SM1.1=1 (taşma) ve SM1.2=0 (sonuç pozitif) olur.

Eğer, VW20=32768 (H8000) ve VW22=32768 (H8000) ise sonuç VW24= 0 (H0000) ve SM1.0=0 (sonuç sıfır) SM1.1=1 (taşma) ve SM1.2=0 (sonuç pozitif) olur.

Taşma durumunda, SM1.1=1 ve ENO=0 olur. ENO=0 sonucun geçersiz olduğu anlamına gelir.



IN1, IN2 : VW, IW, QW, MW, SW, SMW, T, C, AC, LW, AIW, Sabit, *VD, *LD, *AC

OUT : VW, IW, QW, MW, SW, SMW, T, C, LW, AC, *VD, *LD, *AC INT

Şekil 5.11 16 bit işaretli tamsayılara ilişkin toplama işlemi

Çıkarma işlemine ilişkin komut Şekil 5.12'de gösterildiği gibi yazılır. Bu komut yürütüldüğünde

OUT:=IN1-IN2

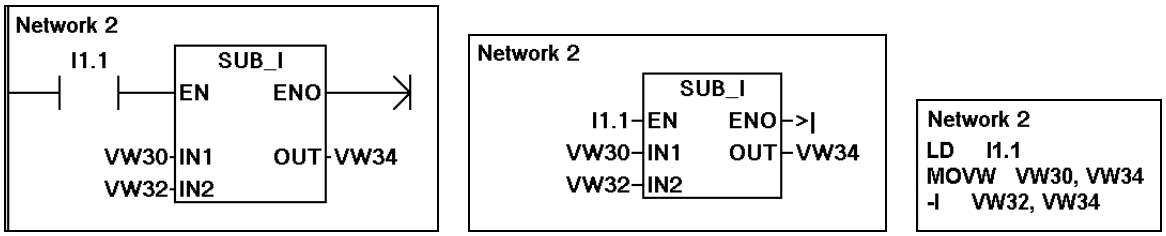
olur. Çıkarma işleminde STL komutu

-I IN1, OUT

biçiminde yazılır ve bu komut yürütüldüğünde

OUT:=OUT-IN1

işlemi gerçeklenir. İşlem sonucunda taşma olduğunda SM1.1=1 ve ENO=0 olur.



IN1, IN2 : VW, IW, QW, MW, SW, SMW, T, C, AC, LW, AIW, Sabit, *VD, *LD, *AC

OUT : VW, IW, QW, MW, SW, SMW, T, C, LW, AC, *VD, *LD, *AC INT

Şekil 5.12 16 bit işaretli tamsayılara ilişkin çıkarma işlemi

16 bit işaretli tamsayıların çarpımı için {MUL} ve {*I} gibi iki ayrı komut kullanılır: MUL çarpım komutu

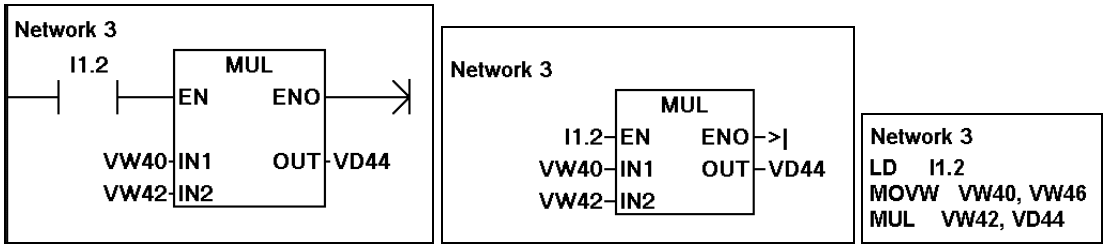
MUL IN1, OUT

biçiminde yazılır. Bu komut ile IN1 adresindeki 16 bitlik veri ile OUT adresindeki 32 bitlik verinin düşük anlamlı 16 bitlik bölümü çarpılır ve sonuç 32 bitlik OUT adresine yazılır. Örneğin,

MUL VW42, VD44

komutu yürütüldüğünde VW42 adresindeki işaretli tamsayı ile VW46 adresindeki işaretli tamsayı çarpılır ve sonuç VD44 adresine yazılır.

{MUL} çarpma komutunda işlenen IN1 ve OUT için geçerli adresler ve ilgili program Şekil 5.13'te gösterilmiştir. Çarpma işlemi sonucunda taşma olduğunda SM1.1=1 ve ENO=0 olur.



IN1, IN2 : VW, IW, QW, MW, SW, SMW, T, C, LW, AC, AIW, Sabit, *VD, *LD, *AC

OUT : VD, ID, QD, MD, SMD, SD, LD, AC, *VD, *LD, *AC

Şekil 5.13 16 bit işaretli tamsayılara ilişkin çarpma (MUL) işlemi

S7-200 CPU 22x işlemcili PLC'lerde çarpma işlemi için {*I} komutu da kullanılır ve

***I IN1, OUT**

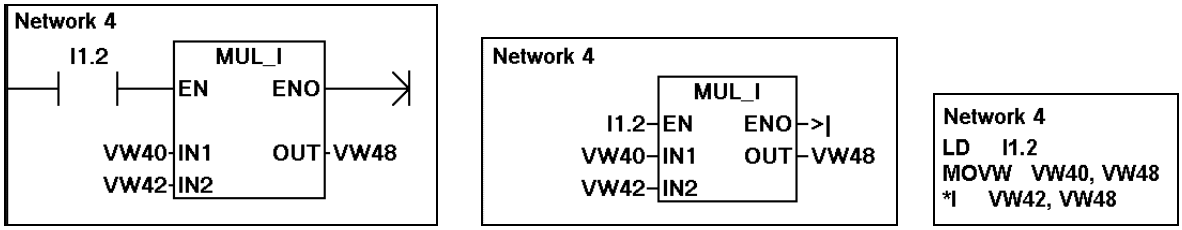
biçiminde yazılır. Komut yürütüldüğünde IN1 adresindeki 16 bitlik işaretli tamsayı ile OUT adresindeki 16 bitlik işaretli tamsayı çarpılır ve sonuç 16 bitlik OUT adresine işaretli tamsayı olarak yazılır.

OUT:=OUT*IN1

Şekil 5.14'teki programda merdiven mantığı ve FBD biçimindeki komut yürütüldüğünde

OUT:=IN1*IN2

işlemi gerçeklenir.



IN1, IN2 : VW, IW, QW, MW, SW, SMW, T, C, LW, AC, AIW, Sabit, *VD, *LD, *AC

OUT : VW, IW, QW, MW, SW, SMW, LW, T, C, AC, *VD, *LD, *AC

Şekil 5.14 16 bit işaretli tamsayılara ilişkin çarpma (*I) işlemi

CPU 22x işlemcili PLC'lerde 16 bit işaretli tamsayılara ilişkin bölme işlemi için {DIV} ve {/I} gibi iki ayrı komut kullanılır: {DIV} komutu

DIV IN1, OUT

biçiminde yazılır. Bu komut ile OUT adresindeki 32 bitlik verinin düşük anlamlı 16 bitlik bölümü, IN1 adresindeki 16 bitlik veriye bölünür ve bölüm, OUT adresinin 16 bitlik düşük anlamlı bölümüne, kalan, OUT adresinin 16 bitlik yüksek anlamlı bölümüne yazılır. Örneğin

DIV VW62, VD64

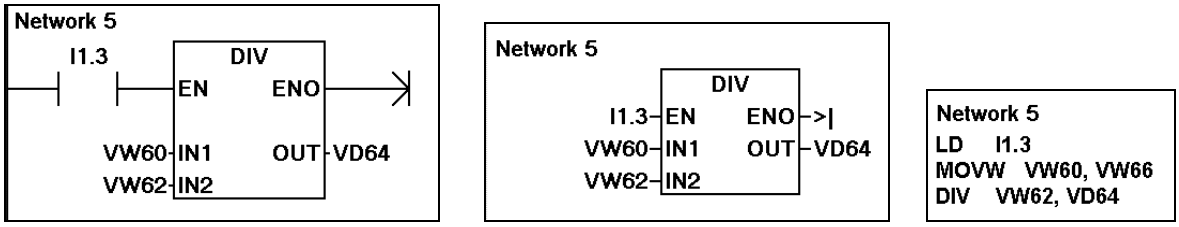
komutu yürütüldüğünde VW66 adresindeki işaretli tamsayı VW62 adresine bölünür; bölüm VW66 adresine, kalan VW64 adresine yazılır.

Şekil 5.15'teki programda merdiven mantığı ve FBD biçimindeki komut yürütüldüğünde

OUT:=IN1/IN2

işlemi gerçeklenir.

Bölme işlemi sonucunda taşma (SM1.1=1) veya sıfıra bölme (SM1.3=1) olursa ENO=0 olur



IN1, IN2 : VW, IW, QW, MW, SW, SMW, T, C, LW, AC, AIW, Sabit, *VD, *LD, *AC

OUT : VD, ID, QD, MD, SMD, SD, LD, AC, *VD, *LD, *AC

Şekil 5.15 16 bit işaretli tamsayılara ilişkin bölme { DIV } işlemi

16 bit işaretli tamsayılara ilişkin bölme işleminde {/I} komutu

/I IN1, OUT

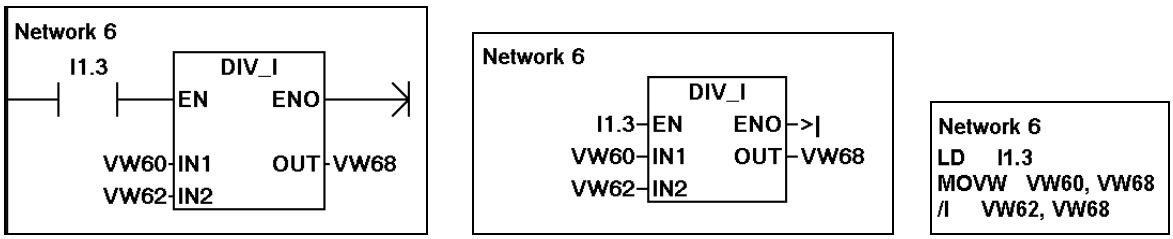
biçiminde yazılır. Bu komut ile OUT adresindeki 16 bitlik işaretli tamsayı, IN1 adresindeki 16 bitlik işaretli tamsayıya veriye bölünür ve bölüm 16 bitlik OUT adresine yazılır.

OUT:=OUT/IN1

Bölme işlemine ilişkin Şekil 5.16'da verilen programda komut yürütüldüğünde

OUT:=IN1/IN2

işlemi gerçeklenir.



IN1, IN2 : VW, IW, QW, MW, SW, SMW, T, C, LW, AC, AIW, Constant, *VD, *LD, *AC

OUT : VW, IW, QW, MW, SW, SMW, LW, T, C, AC, *VD, *LD, *AC

Şekil 5.16 16 bit işaretli tamsayılara ilişkin bölme (/I) işlemi

Bölme işlemi sonucunda taşma (SM1.1=1) veya sıfıra bölme (SM1.3=1) olursa ENO=0 olur.

5.3.2 32 BİT İŞARETLİ TAMSAYILAR ÜZERİNDE İŞLEMLER

32 bit işaretli tamsayıların toplamına ilişkin komut

+D IN1, OUT

biçiminde yazılır. Bu komut ile IN1 adresindeki 32 bitlik veri ile OUT adresindeki 32 bitlik veri toplanır, sonuç 32 bitlik OUT adresine yazılır.

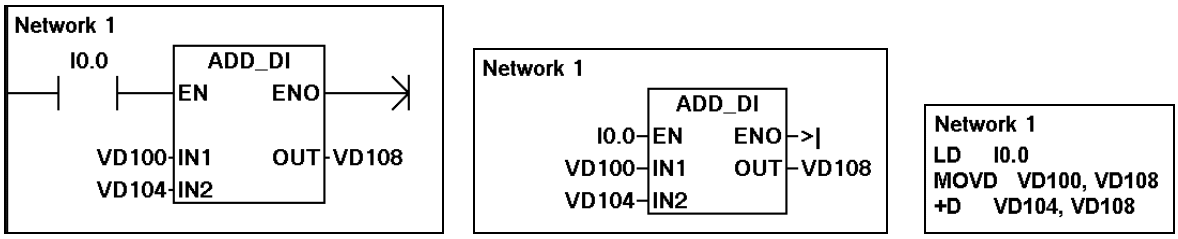
OUT:=OUT/IN1

32 bitlik işaretli tamsayıların toplamına ilişkin Şekil 5.17'de verilen programda komut yürütüldüğünde

OUT:=IN1/IN2

işlemi gerçeklenir.

Toplama sonucu taşma olursa (SM1.1=1) ENO=0 olur.



IN1, IN2 : VD, ID, QD, MD, SMD, SD, LD, AC, HC, Sabit *VD, *LD, *AC

OUT : VD, ID, QD, MD, SMD, SD, LD, AC, *VD, *LD, *AC

Şekil 5.17 32 bitlik işaretli tamsayılara ilişkin toplama işlemi

Çıkarma işlemi için

-D IN1, OUT

komutu kullanılır. Bu komut ile IN1 adresindeki veri OUT adresindeki veriden çıkarılır ve sonuç OUT adresine yazılır.

OUT:=OUT-IN1

Örneğin,

-D VD124, VD128

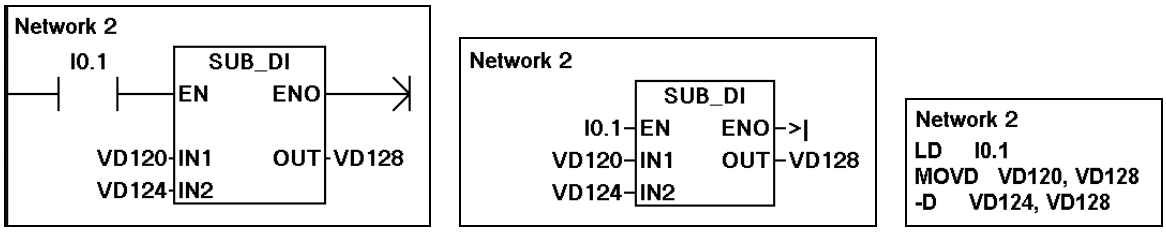
komutu yürütüldüğünde, VD124 adresindeki 32 bitlik işaretli tamsayı, VD128 adresindeki 32 bitlik işaretli tamsayıdan çıkartılır ve sonuç VD128 adresine yazılır.

Çıkarma işlemine ilişkin Şekil 5.18'deki programda komut yürütüldüğünde

OUT:=IN1-IN2

işlemi gerçeklenir.

Çıkarma sonucu taşma olursa (SM1.1=1) ENO=0 olur



IN1, IN2 : VD, ID, QD, MD, SMD, SD, LD, AC, HC, Sabit *VD, *LD, *AC

OUT : VD, ID, QD, MD, SMD, SD, LD, AC, *VD, *LD, *AC

Şekil 5.18 32 bitlik işaretli tamsayılara ilişkin çıkarma işlemi

32 bitlik işaretli tamsayılara ilişkin çarpma işlemi için

***D IN1, OUT**

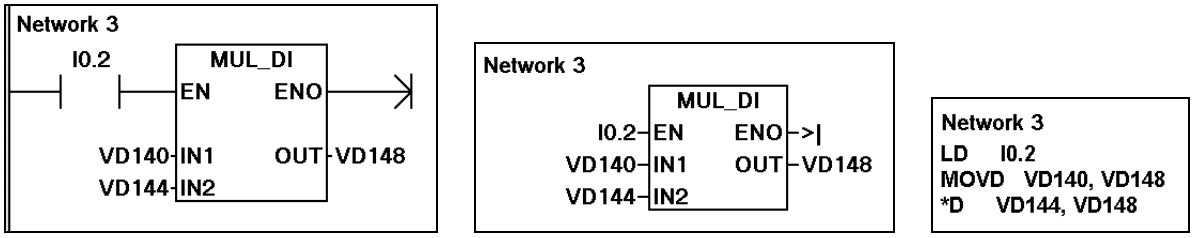
komutu kullanılır. Bu komutun yürütülmesi ile OUT adresindeki veri ile IN1 adresindeki veri çarpılır ve sonuç OUT adresine yazılır.

OUT:=OUT*IN1

Şekil 5.19'daki programda gösterilen merdiven mantığı veya FBD komutunun yürütülmesi durumunda

OUT:=IN1*IN2

işlemi yapılır.



IN1, IN2 : VD, ID, QD, MD, SMD, SD, LD, HC, AC, Sabit, *VD, *LD, *AC

OUT : VD, ID, QD, MD, SMD, SD, LD, AC, *VD, *LD, *AC

Şekil 5.19 32 bitlik işaretli tamsayılara ilişkin çarpma işlemi

32 bitlik işaretli tamsayılara ilişkin bölme işleminde

/D IN1, OUT

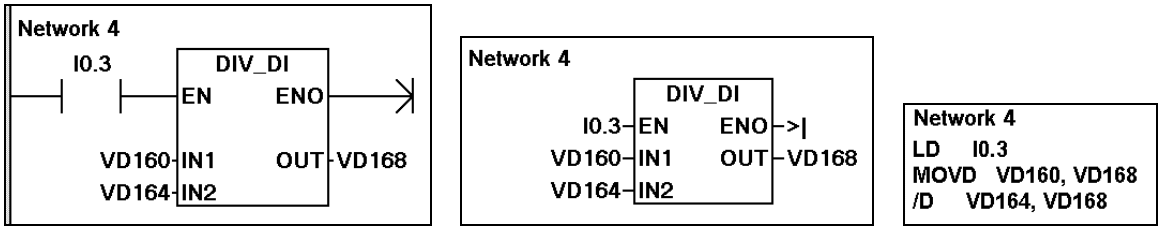
komutu kullanılır. Bu komutun yürütülmesi ile OUT adresindeki veri IN1 adresindeki veriye bölünür ve sonuç OUT adresine yazılır.

OUT:=OUT/IN1

Şekil 5.20'deki merdiven mantığı veya FBD programında komutun yürütülmesi ile

OUT:=IN1/IN2

işlemi yapılır.



IN1, IN2 : VD, ID, QD, MD, SMD, SD, LD, HC, AC, Sabit, *VD, *LD, *AC

OUT : VD, ID, QD, MD, SMD, SD, LD, AC, *VD, *LD, *AC

Şekil 5.20 32 bitlik işaretli tamsayılara ilişkin çarpma işlemi

Bölme işlemi sonucunda taşma (SM1.1=1) veya sıfıra bölme (SM1.3=1) olursa ENO=0 olur

5.4 REEL SAYILAR ÜZERİNDE İŞLEMLER

Reel sayılara ilişkin toplama işlemi için

+R IN1, OUT

komutu kullanılır. Bu komut yürütüldüğünde OUT adresindeki reel sayı ile IN1 adresindeki reel sayı toplanır ve sonuç OUT adresine yazılır.

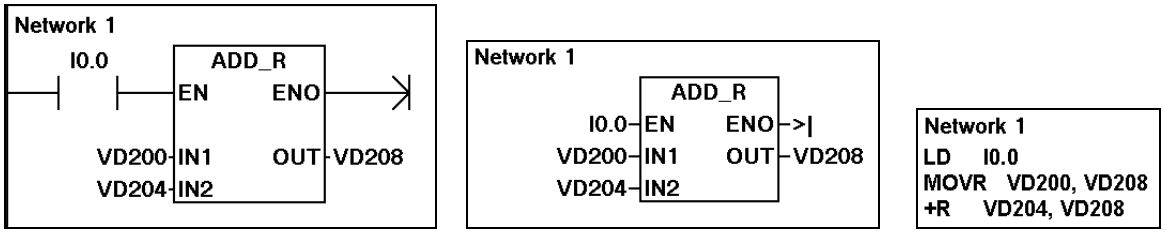
OUT:=OUT+IN1

Şekil 5.21'de gösterilen merdiven mantığı ve FBD programında komut yürütüldüğünde

OUT:=IN1+IN2

işlemi yapılır.

İşlem sonucunda taşma olduğunda (SM1.1=1), ENO=0 olur.



IN1, IN2 : VD, ID, QD, MD, SD, SMD, LD, AC, Sabit, *VD, *LD, *AC

OUT : VD, ID, QD, MD, SD, SMD, LD, AC, *VD, *LD, *AC

Şekil 5.21 Reel sayılara ilişkin toplama işlemi

Reel sayılara ilişkin çıkarma işlemi için

-R IN1, OUT

komutu kullanılır. Bu komut yürütüldüğünde IN1 adresindeki reel sayı OUT adresindeki reel sayıdan çıkarılır ve sonuç OUT adresine yazılır.

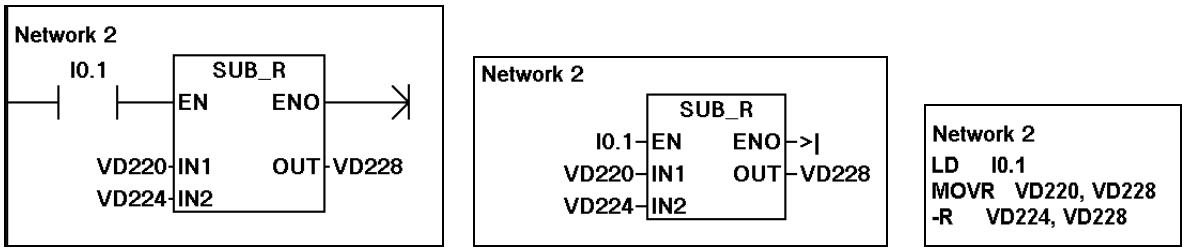
OUT:=OUT-IN1

Şekil 5.22de gösterilen merdiven mantığı ve FBD programında komut yürütüldüğünde

OUT:=IN1-IN2

işlemi yapılır.

İşlem sonucunda taşma olduğunda (SM1.1=1), ENO=0 olur.



IN1, IN2 : VD, ID, QD, MD, SD, SMD, LD, AC, Sabit, *VD, *LD, *AC

OUT : VD, ID, QD, MD, SD, SMD, LD, AC, *VD, *LD, *AC

Şekil 5.22 Reel sayılara ilişkin çıkarma işlemi

Reel sayılara ilişkin çarpma işlemi için

***R IN1, OUT**

komutu kullanılır. Bu komut yürütüldüğünde OUT adresindeki reel sayı ile IN1 adresindeki reel sayı çarpılır ve sonuç OUT adresine yazılır.

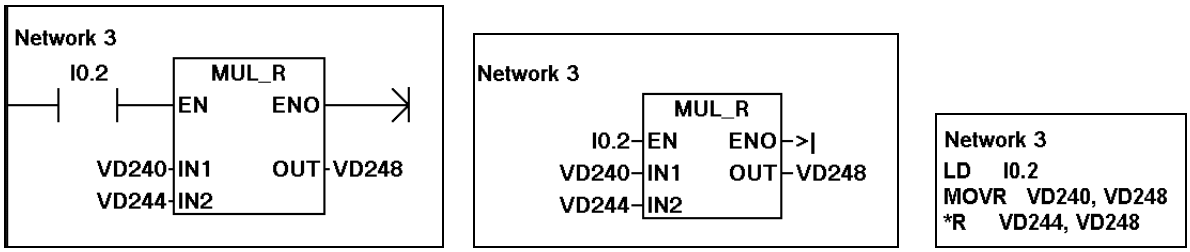
OUT:=OUT*IN1

Şekil 5.23'te gösterilen merdiven mantığı ve FBD programında çarpma komutu yürütüldüğünde

OUT:=IN1*IN2

işlemi yapılır.

İşlem sonucunda taşma olduğunda (SM1.1=1), ENO=0 olur.



IN1, IN2 : VD, ID, QD, MD, SD, SMD, LD, AC, Sabit, *VD, *LD, *AC

OUT : VD, ID, QD, MD, SD, SMD, LD, AC, *VD, *LD, *AC

Şekil 5.23 Reel sayılara ilişkin çarpma işlemi

Reel sayılara ilişkin bölme işlemi için

/R IN1, OUT

komutu kullanılır. Bu komut yürütüldüğünde IN1 adresindeki reel sayı OUT adresindeki reel sayıya bölünür ve sonuç OUT adresine yazılır.

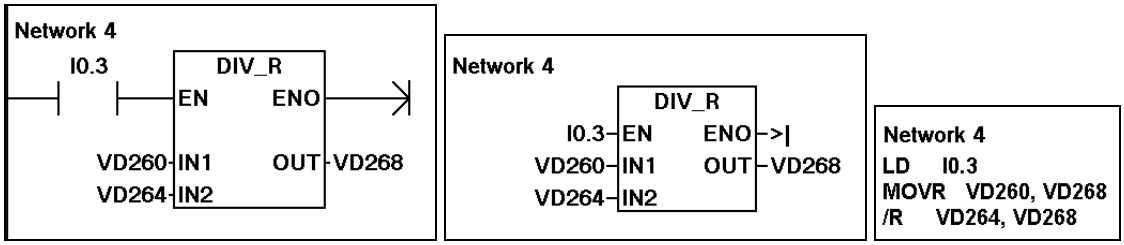
OUT:=OUT/IN1

Şekil 5.24'te gösterilen merdiven mantığı ve FBD programında bölme komutu yürütüldüğünde

OUT:=IN1/IN2

işlemi yapılır.

İşlem sonucunda taşma olduğunda (SM1.1=1), ENO=0 olur.



IN1, IN2 :VD, ID, QD, MD, SD, SMD, LD, AC, Sabit, *VD, *LD, *AC

OUT : VD, ID, QD, MD, SD, SMD, LD, AC, *VD, *LD, *AC

Şekil 5.24 Reel sayılara ilişkin bölme işlemi

5.5 REEL SAYILAR ÜZERİNDE DİĞER İŞLEMLER

5.5.1 KAREKÖK ALMA İŞLEMİ

Reel bir sayının karekökü

SQRT IN, OUT

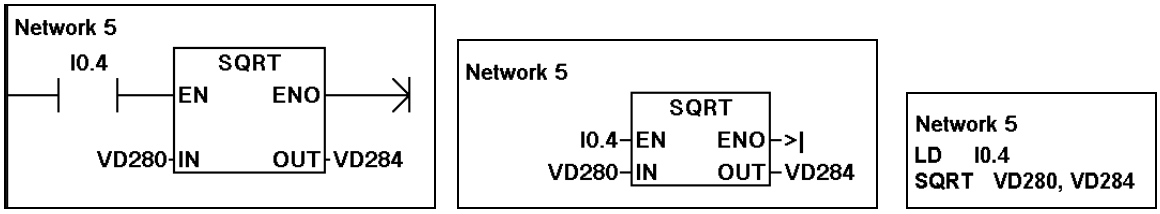
biçiminde yazılan komut ile bulunur. Komut yürütüldüğünde IN adresindeki reel sayının karekökü alınır ve OUT adresine yazılır.

Şekil 5.25'te gösterilen merdiven mantığı ve FBD programında bölme komutu yürütüldüğünde

$$OUT := \sqrt{IN}$$

işlemi yapılır.

İşlem sonucunda taşma olduğunda (SM1.1=1), ENO=0 olur.



IN : VD, ID, QD, MD, SMD, SD, LD, AC, Sabit, *VD, *LD, *AC

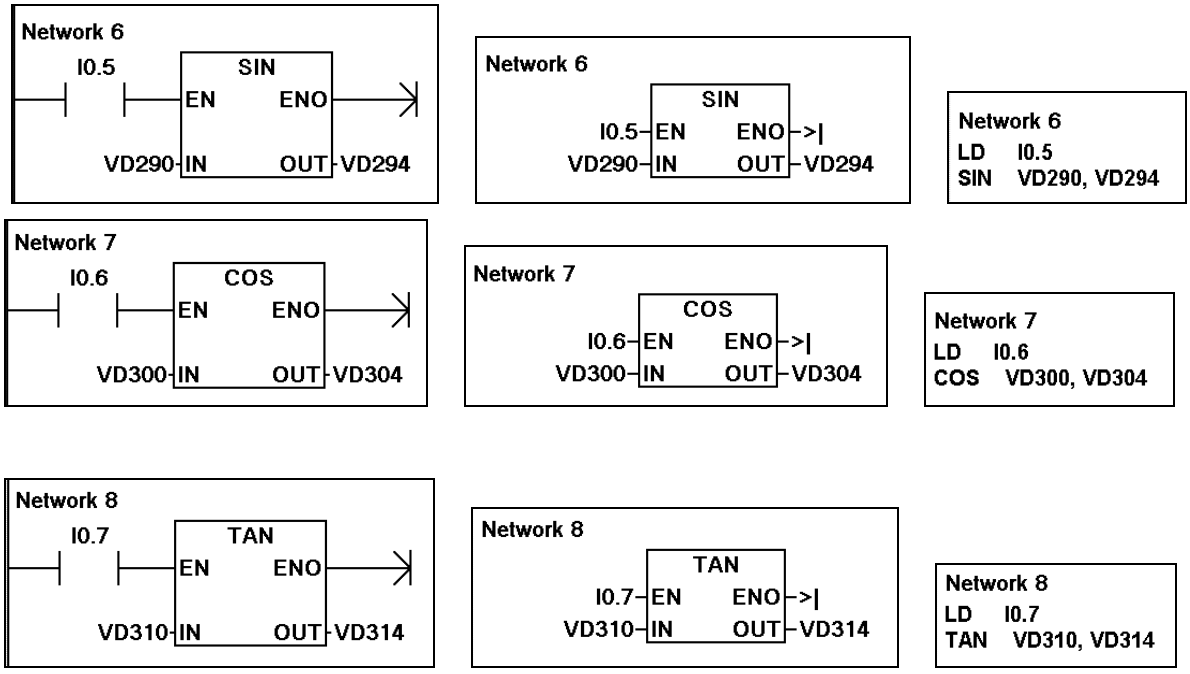
OUT : VD, ID, QD, MD, SMD, SD, LD, AC, *VD, *LD, *AC

Şekil 5.25 Reel sayılara ilişkin bölme işlemi

5.5.2 REEL SAYILAR ÜZERİNDE DİĞER İŞLEMLER

Simatic S7-200 CPU 22x işlemcili PLC'lerde **sin**, **cos** ve **tan** trigonometrik fonksiyonları, **ln** ve **exp** fonksiyonları ve PID kontrol işlevini sağlayan fonksiyonlar gerçeklenebilir.

Şekil 5.26'da trigonometrik fonksiyonlara ilişkin komutlar gösterilmiştir. Bu komutlar yürütüldüğünde IN adresindeki radyan boyutundaki reel sayının fonksiyonu OUT adresine yazılır. Diğer matematiksel işlemlerde olduğu gibi sonuç sıfır ise SM1.0=1, taşma varsa SM1.1=1 ve sonuç negatif ise SM1.2=1 olur. Taşma durumunda ENO=0 olur.



IN : VD, ID, QD, MD, SMD, SD, LD, AC, Sabit, *VD, *LD, *AC

OUT : VD, ID, QD, MD, SMD, SD, LD, AC, *VD, *LD, *AC

Şekil 5.26 Reel sayılar üzerinde trigonometrik işlemler

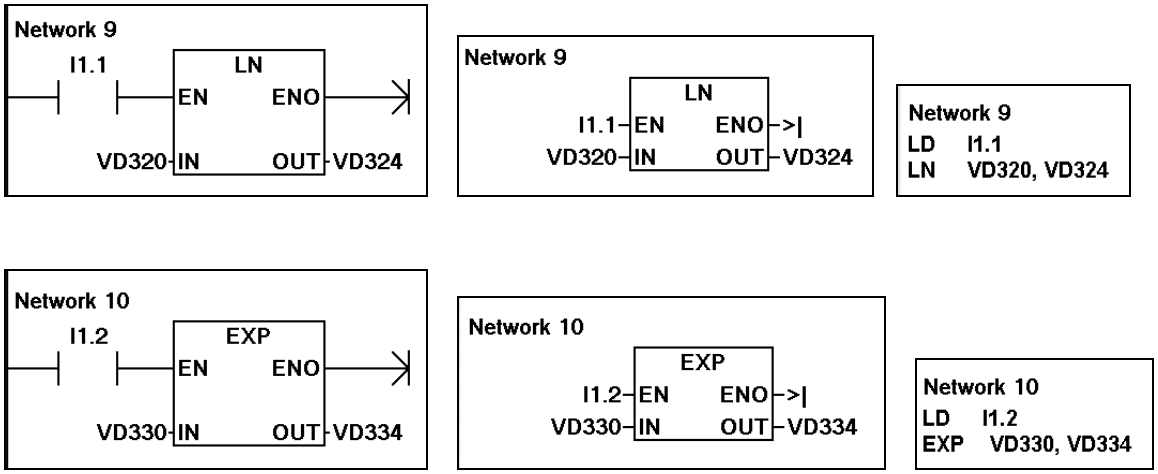
Bir reel sayının doğal logaritması

LN IN, OUT

üstel fonksiyonu

EXP IN, OUT

komutları ile hesaplanır. Merdiven mantığı ve FBD biçimi Şekil 5.27'de gösterilen bu komutlar yürütüldüğünde IN adresindeki veriye ilgili işlem uygulanır ve sonuç OUT adresine yazılır. Tüm matematiksel işlemlerde olduğu gibi sonuç sıfır ise SM1.0=1, taşma varsa SM1.1=1 ve sonuç negatif ise SM1.2=1 olur, taşma durumunda ENO=0 olur.



IN: **VD, ID, QD, MD, SMD, SD, LD, AC, sabit, *VD, *LD, *AC**

OUT: **VD, ID, QD, MD, SMD, SD, LD, AC, *VD, *LD, *AC**

Şekil 5.27 Reel sayılar üzerinde logaritma ve üstel fonksiyon alma işlemleri

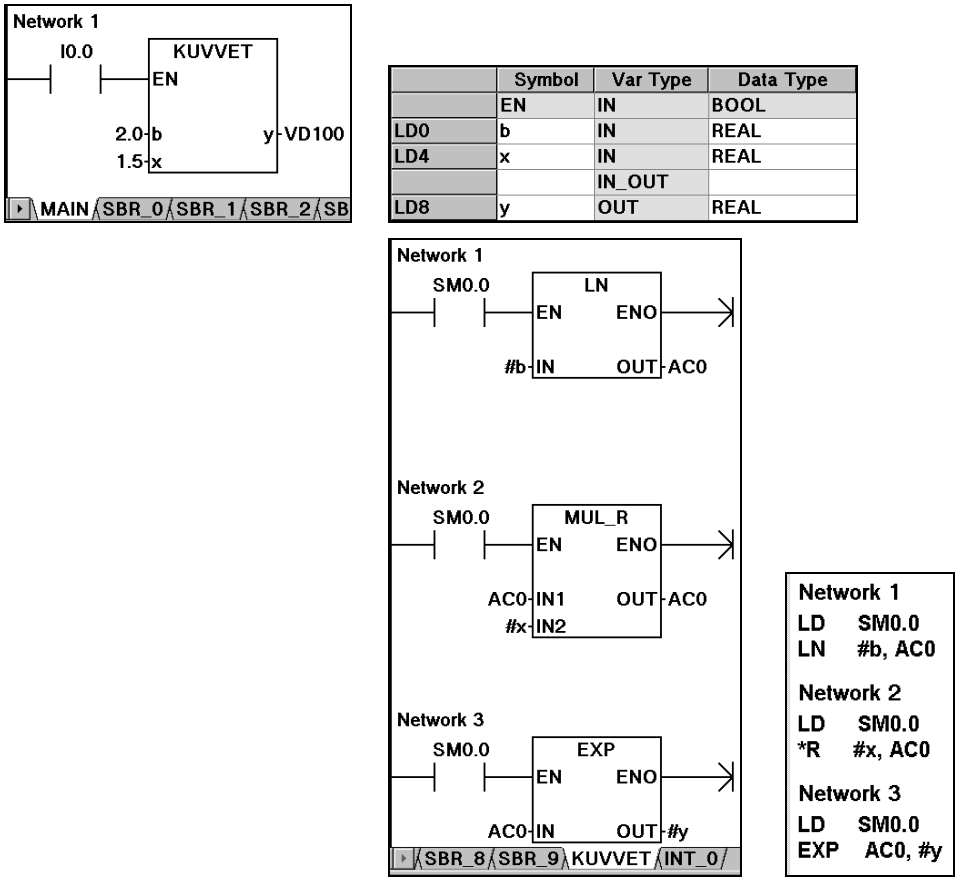
Birçok PLC'de olduğu gibi

$$y = b^x$$

biçiminde verilen bir reel sayının kuvvetini doğrudan hesaplayan komut yoktur. Ancak, exp ve ln komutları kullanılarak bu fonksiyon hesaplanabilir. Bunun için

$$y = \exp[x \ln b]$$

eşitliğinden yararlanılır. Bu ifade programlanırsa b sayısının x kuvveti hesaplanır. Şekil 5.28'de bu amaçla yazılan bir altprogram verilmiştir.



Şekil 5.28 Reel bir sayının kuvvetinin hesaplanmasına ilişkin altprogram

Şekil 5.28'de verilen altprogram anaprogramdan yürütüldüğünde

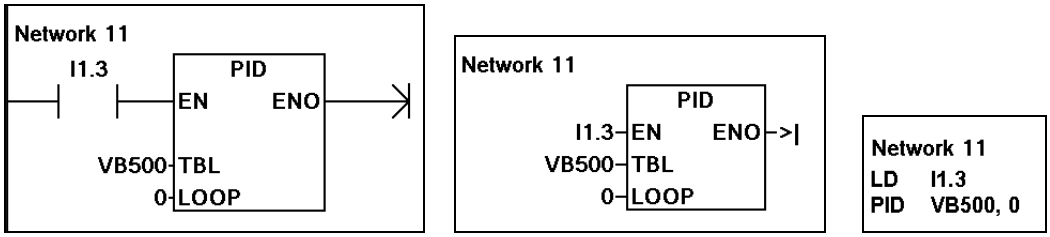
$$y = b^x = 2^{1.5}$$

işlemi yapılır ve VD100 adresine 2.828429 reel sayısı yazılır.

Geribeslemeli kontrol sistemlerinde kullanılan PID fonksiyonu Şekil 5.29'daki gibi programlanır.

Burada TBL, PID parametrelerinin tutulduğu 36 bayt boyutunda bir veri tablosudur, eğer Micro/WIN 4.0 sürümünde bulunan PID parametrelerinin otomatik ayarlama özelliği kullanılmak istenirse 80 bayt yer ayrılır. S7-200 CPU 22x işlemcili PLC'lerde 8 adet PID fonksiyonu kullanılabilir. Hangi PID fonksiyonunun kullanıldığı LOOP çevrim numarasına 0 ile 7 arasında bir tamsayı yazılarak belirtilir.

Bu komutun kullanımı kontrol sistemlerinin gerçeklenmesine ilişkin uygulamaların anlatıldığı Bölüm 8'de ayrıntılı olarak verilecektir.



TBL : VB

LOOP : Sabit (0 - 7)

Şekil 5.29 PID işlemine ilişkin komutun kullanımı

5.6 DÖNÜŞTÜRME İŞLEMLERİ

5.6.1 REEL SAYILARA DÖNÜŞTÜRME

Bir 8 bitlik işaretsiz tamsayı üzerinde reel işlem yapmak için, 8 bitlik tamsayı 16 bitlik işaretli tamsayıya, 16 bitlik işaretli tamsayı 32 bitlik işaretli tamsayıya ve 32 bitlik işaretli tamsayı reel sayıya dönüştürülür.

8 bitlik işaretsiz tamsayıyı 16 bitlik işaretli tamsayıya dönüştürmek için

BTI VB20, VW22

biçiminde yazılan komut kullanılır. Bu komut yürütüldüğünde VB20 adresindeki 8 bitlik işaretsiz tamsayı VW22 adresindeki 16 bitlik alana işaretli tamsayı biçiminde yazılır.

16 bitlik işaretli tamsayıyı 32 bitlik işaretli tamsayıya dönüştürmek için

ITD VW22, VD24

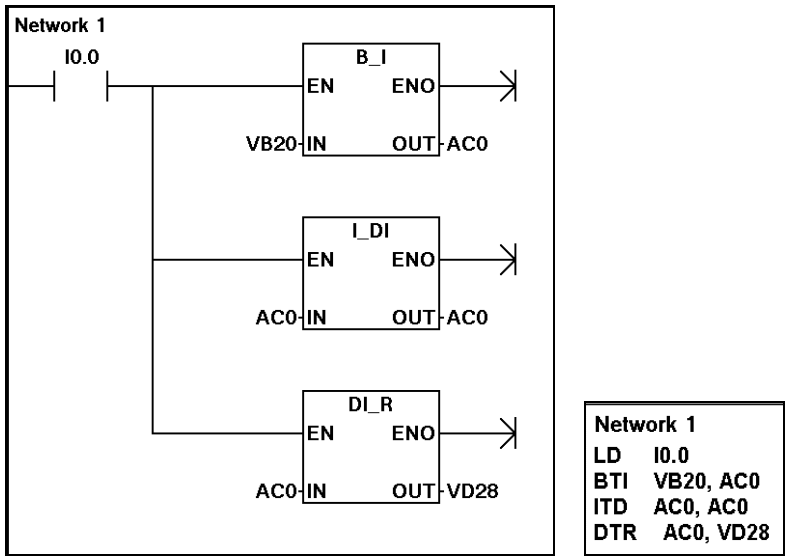
biçiminde yazılan komut kullanılır. Bu komut yürütüldüğünde VW22 adresindeki 16 bitlik işaretli tamsayı, VD24 adresine 32 bitlik işaretli tamsayı biçiminde yazılır.

32 bitlik işaretli tamsayıyı reel sayıya dönüştürmek için

DTR VD24, VD28

biçiminde yazılan komut kullanılır. Bu komut yürütüldüğünde VD24 adresindeki 32 bitlik işaretli tamsayı VD28 adresine reel sayı biçiminde yazılır.

Örneğin VB20 adresindeki 8 bitlik işaretsiz tamsayıyı VD28 adresine reel sayı biçiminde yazmak için Şekil 5.30'daki program kullanılabilir.



Şekil 5.30 8 bitlik işaretsiz tamsayıyı reel sayıya dönüştürme işlemi

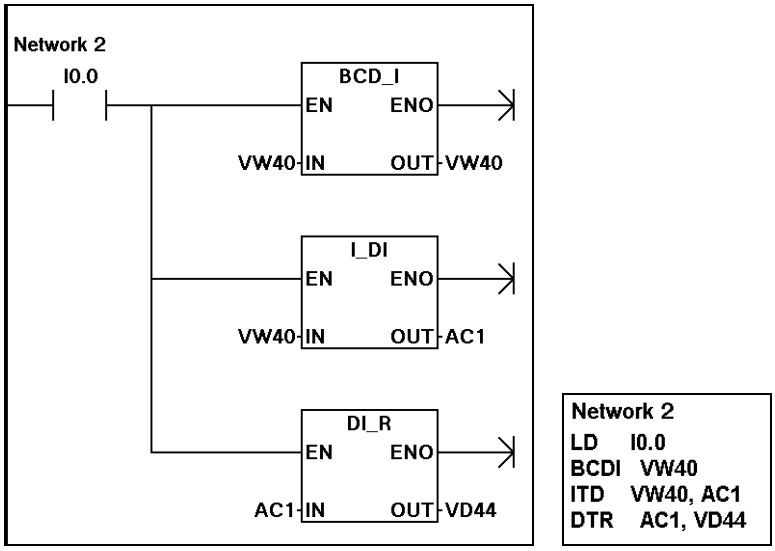
BCD biçiminde kodlanmış 0 ile 9999 arasındaki sayıları reel sayıya dönüştürmek için, BCD kodlanmış sayı 16 bitlik işaretli tamsayıya, 16 bitlik işaretli tamsayı 32 bitlik işaretli tamsayıya ve 32 bitlik işaretli tamsayı reel sayıya dönüştürülür.

BCD kodlanmış sayıyı işaretli tamsayıya dönüştürmek için

BCDI VW40

biçiminde yazılan komut kullanılır. Bu komutun işletilmesi ile VW40 adresindeki BCD kodlanmış sayı 16 bitlik işaretli tamsayıya dönüştürülür ve sonuç yine aynı adrese yazılır. İşaretli tamsayı bir önceki örnekteki gibi reel sayıya dönüştürülür.

Örneğin, VW40 adresindeki BCD kodlanmış sayıyı reel sayıya dönüştürmek için Şekil 5.31'deki program kullanılabilir.



Şekil 5.31 BCD kodlanmış tamsayıyı reel sayıya dönüştürme işlemi

5.6.2 İŞARETLİ 16 BİTLİK TAMSAYIYA DÖNÜŞTÜRME

Zamanlayıcı ve sayıcı içerikleri veya analog verilere ilişkin sayısal değerler 16 bitlik işaretli tamsayı biçimindedir. Matematiksel bir işlem sonucu elde edilen bir reel sayısal değerin zamanlayıcı, sayıcı veya analog çıkış birimine aktarılması için 16 bitlik işaretli tamsayıya dönüştürülmesi gerekir. Bunun için reel sayıyı işaretli tamsayıya dönüştürme işlemi yapılır. Reel sayıların işaretli tamsayıya dönüştürülmesinde kırpma (truncation) veya yuvarlama (round) komutları kullanılır. Kırpma komutu

TRUNC VD40, VD44

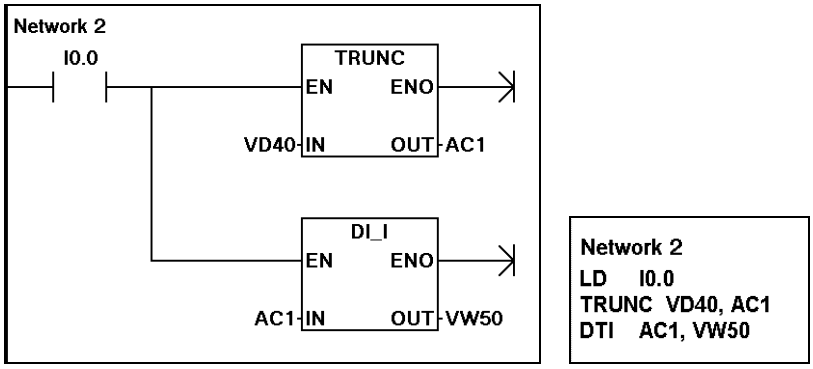
biçiminde yazılır. Bu komutun işletilmesi ile VD40 adresindeki reel sayının kesirli bölümü atılır ve kalan ondalık bölümü 32 bitlik işaretli tamsayı biçiminde VD44 adresine yazılır.

Yuvarlama komutu

ROUND VD40, VD44

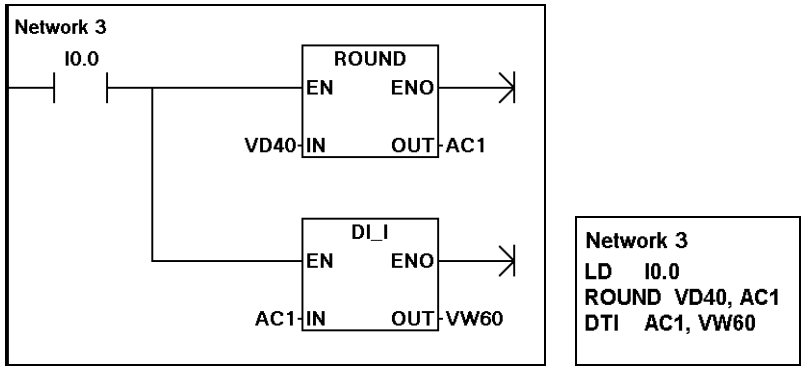
biçiminde yazılır. Bu komutun işletilmesi ile VD40 adresindeki reel sayının kesirli bölümü 5 ve daha büyükse ondalık bölümü 1 arttırılarak 32 bitlik işaretli tamsayı biçiminde, VD44 adresine yazılır. Eğer kesirli bölüm 5'ten küçükse kırpma komutu gibi kesirli bölüm atılır ve ondalık bölüm işaretli tamsayı olarak VD44 adresine yazılır.

Şekil 5.32'de TRUNC komutu, Şekil 5.33'te ROUND komutu kullanılarak bir reel sayıyı 16 bitlik işaretli tamsayıya dönüştüren program örnekleri verilmiştir.



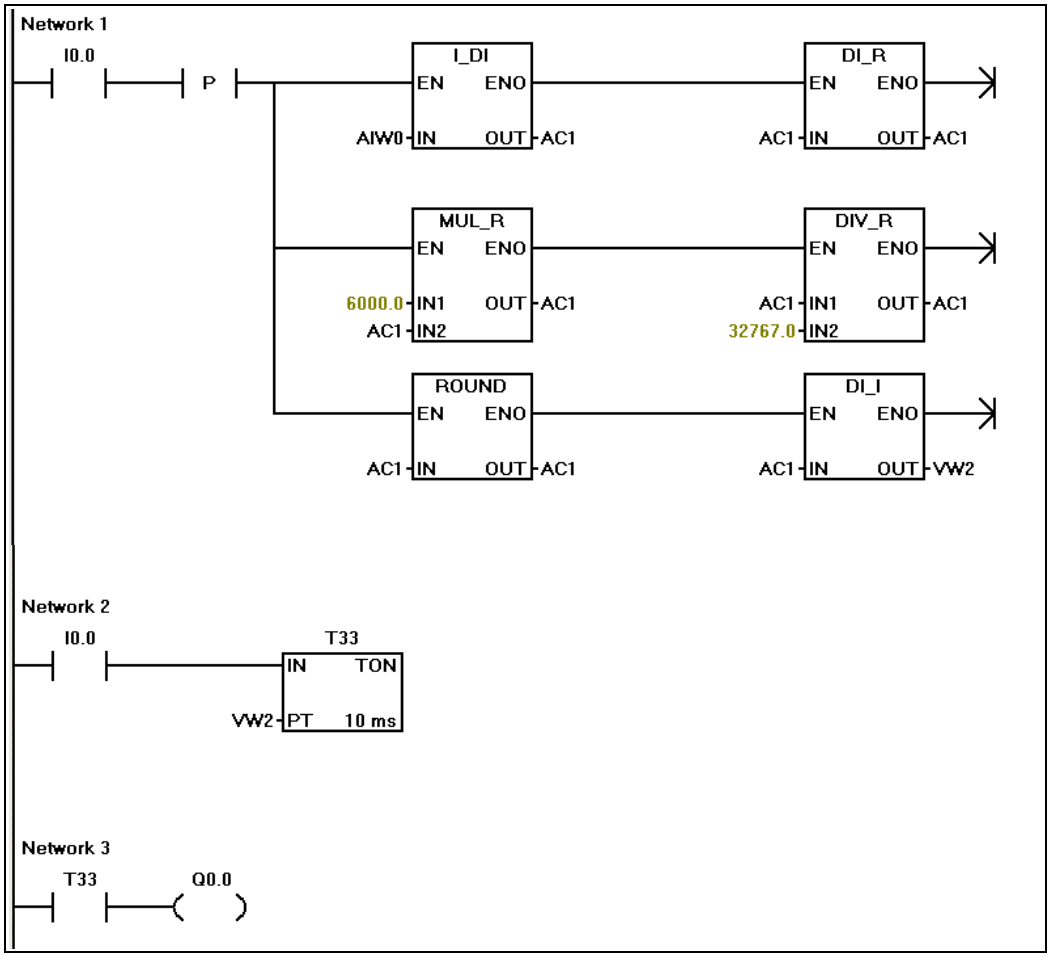
Şekil 5.32 Reel sayıyı işaretli tamsayıya dönüştürme işlemi

Şekil 5.30'da verilen program parçası ile VD40 adresindeki reel sayının kesirli bölümü atılır ve eğer kalan tamsayı bölümü -32768 ile +32767 arasında ise 16 bitlik işaretli tamsayı biçiminde VW50 adresine yazılır. Bir reel sayının 16 bitlik işaretli tamsayıya dönüştürülmesi için reel sayı ondalık bölümünün, işaretli tamsayı değer aralığı (-32768 : 32767) içinde olması gerekir.



Şekil 5.33 Reel sayıyı işaretli tamsayıya dönüştürme işlemi

Şekil 5.34'te PLC analog giriş birimine bağlı 0-10V arasında değişen bir gerilim kaynağı kullanılarak, gecikme süresi gerilim değeri ile değiştirilebilen bir zamanlayıcıya ilişkin programı örneği verilmiştir.



Şekil 5.34 Analog işaretle orantılı gecikme sağlayan program örneği

Bu uygulamada, işaretli tamsayı biçiminde olan PLC AIW0 analog adresinin içeriği önce reel sayıya dönüştürülür ve AC1'e yazılır. AC1'deki reel sayı 6000.0 reel sayısı ile çarpılır, sonra 32767.0 reel sayısına bölünür ve sonuç yine AC1'e yazılır. AC1'deki reel sayı yuvarlama komutu ile 32 bitlik işaretli tamsayıya, 32 bitlik işaretli tamsayı 16 bitlik işaretli tamsayıya dönüştürülerek VW2 adresine yazılır. Bu adres T33 zamanlayıcısının PT değeri olarak atanır. Böylece, AIW0 analog girişine uygulanan gerilimin 0-10V değeri ile doğrusal değişen 0-60s aralığında gecikme süreleri elde edilir. Bu programa göre, analog girişe uygulanan gerilim değeri ile orantılı bir gecikmeyle, I0.0=1 yapıldıktan belirli bir süre sonra, Q0.0=1 olur.

5.7 VERİ TABLOSU İŞLEMLERİ

Veri tablosu, 16 bitlik verilerin özel bir biçimde saklandığı bir veri alanıdır. Veri tablosu başlangıç adresine tablo boyutu veya derinliği, başlangıç adresini izleyen adrese veri sayısı yazılır. Diğer adreslerde ise saklanan veriler bulunur. Tablonun genel yapısı Şekil 5.35'te gösterilmiştir.

VWx	Başlangıç adresi
VW(x+2)	Veri sayısı (en çok 100)
VW(x+4)	İlk veri
VW(x+6)	İkinci veri

VW(x+n)	Son veri

Şekil 5.35 Veri tablosu genel yapısı

Belirli bir uzunlukta boş bir tablo oluşturmak için tablo başlangıç adresi alanına 1-100 arasında tamsayı bir değer yazılır. Başlangıç adresini izleyen 16 bitlik alana 0 yazılır. Örneğin,

MOVW 10,VW100

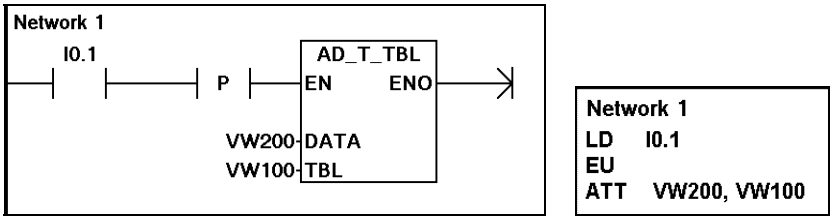
MOVW 0, VW102

komutları ile 10 veri saklanabilen boş bir veri tablosu tanımlanır.

Veri tablosuna veri eklemek için

ATT DATA, TBL

komutu kullanılır. Bu komutun yürütülmesi ile başlangıç adresi TBL olan veri tablosuna DATA adresinin içeriği eklenir ve (TBL+2) adresinin içeriği 1 arttırılır. Tablodaki veri sayısı başlangıç adresi ile belirtilen tablo boyutunu aştığında veya dolu bir tabloya yeni veri eklenmeye çalışıldığında SM1.4 özel bellek biti, komutun işlendiği tarama çevrimi içinde 1 değerini alır. Şekil 5.36'da bu komutun kullanımına ilişkin program ve geçerli veri adresleri verilmiştir. Bu programa göre I0.1 girişinin her çıkan kenarında başlangıç adresi VW100 olan tablonun VW104 adresine VW200 adresinin içeriği yazılır ve VW102 adresinin içeriği 1 arttırılır.



Şekil 5.36 Veri tablosuna veri eklemeye ilişkin program parçası

Tabloya eklenen son veriyi tablodan çıkartıp belirli bir adres yazmak için

LIFO TBL, DATA

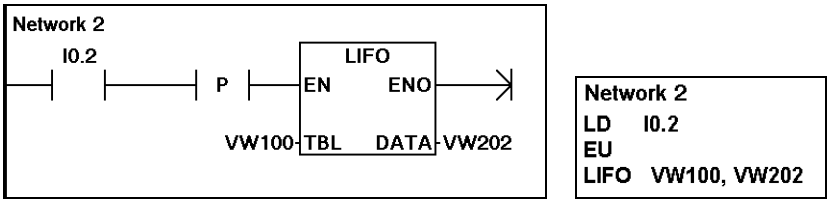
komutu kullanılır. Bu komutun yürütülmesi ile başlangıç adresi TBL olan tablodan son eklenen veri çıkartılır ve DATA alanına yazılır, (TBL+2) adresinin içeriği 1 azaltılır. Eğer tablo boş ise SM1.5 özel bellek biti, komutun işlendiği tarama çevrimi içinde 1 değerini alır. Örneğin, içeriği

VW100	10
VW102	5
VW104	1
VW106	2
VW108	3
VW110	4
VW112	5

biçimindeki veri tablosu için

LIFO VW100, VW200

komutu yürütüldüğünde, VW102 adresinin içeriği 4 olur ve VW112'nin içeriği VW200 adresine yazılır. Şekil 5.37'de bu komutun kullanımına ilişkin program verilmiştir. Bu programa göre I0.2 girişinin her çıkan kenarında başlangıç adresi VW100 olan tablodan son eklenen veri çıkartılır, VW202 adresine yazılır ve VW102'nin içeriği 1 azaltılır.



Şekil 5.37 Son eklenen verinin çıkartılmasına ilişkin program parçası

Tabloya eklenen ilk veriyi tablodan çıkarıp belirli bir adres yazmak için

FIFO TBL, DATA

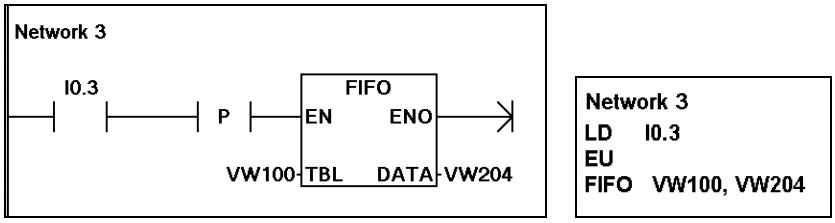
komutu kullanılır. Bu komutun işletilmesi ile başlangıç adresi TBL olan tablodan ilk eklenen veri çıkartılır ve DATA adresindeki alana yazılır, (TBL+2) adresinin içeriği 1 azaltılır. Eğer tablo boş ise SM1.5 özel bellek biti komutun işlendiği tarama çevrimi içinde 1 değerini alır. Örneğin,

VW100	10
VW102	5
VW104	1
VW106	2
VW108	3
VW110	4
VW112	5

biçimindeki bir veri tablosu için

FIFO VW100, VW200

komutu yürütüldüğünde VW102 adresinin içeriği 4 olur ve VW104 adresinin içeriği VW200 adresine yazılır. Şekil 5.38'de bu komutun kullanımına ilişkin program verilmiştir. Bu programa göre I0.2 girişinin her çıkan kenarında başlangıç adresi VW100 olan tablodan son eklenen veri çıkartılır, VW202 adresine yazılır ve VW102'nin içeriği 1 azaltılır.



Şekil 5.38 İlk eklenen verinin çıkartılmasına ilişkin program parçası

Bir tablodan eşitlik ölçütlerine göre bir verinin sırasını bulmak için

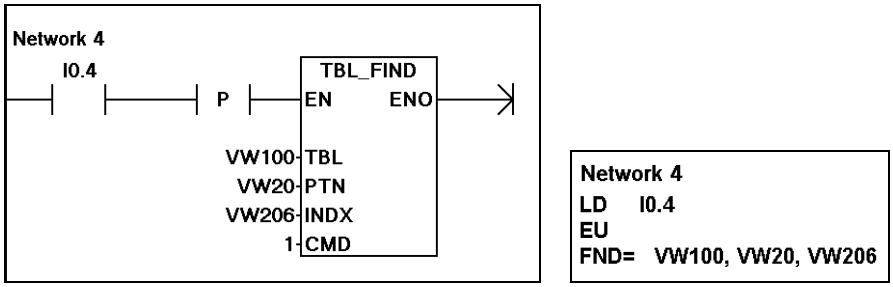
FND= TBL, PTN, INDX

FND<> TBL, PTN, INDX

FND< TBL, PTN, INDX

FND> TBL, PTN, INDX

komutları kullanılır. Bu komutlar yürütüldüğünde TBL tablosu taranır ve PTN ile =, <>, < veya > koşullarını sağlayan verinin sırası INDX alanına yazılır. Şekil 5.39'da bu komutun kullanımına ilişkin bir program gösterilmiştir.



Şekil 5.39 Veri bulma komutunun kullanımına ilişkin program parçası

Merdiven mantığı veya FBD ile programlamada, eğer arama eşitlik ölçütüne göre yapılacaksa CMD girişine 1, eşit-değil ölçütüne göre yapılacaksa 2, küçüktür ölçütüne göre yapılacaksa 3 ve büyüktür ölçütüne göre yapılacaksa 4 tamsayı değeri yazılır.

Bir arama sonucunda ölçütü sağlayan bir veri bulunamazsa INDX alanına tablonun en son sıra numarası yazılır. Eğer yeni bir arama yapılacaksa INDX değerine 0 değeri atanır. Arama işlemi INDX değerinden başlanarak yapılır. Örneğin,

VW100	10
VW102	5
VW104	1
VW106	2
VW108	3
VW110	4
VW112	5

biçimindeki bir tabloda değeri 3 olan bir verinin aranması için

FND= VW100, 3, VW206

komutu yazılır. Bu durumda, eğer başlangıçta VW206 (INDX) <3 ise VW206 adresine 3 değeri, eğer VW206 (INDX) >3 ise VW206 adresine 10 değeri yazılır.

Veri tablosu komutlarının işlev ve özelliklerini incelemek için Şekil 5.40'taki program verilebilir. Bu programa göre I0.0 girişinin çıkan kenarında başlangıç adresi VW100 olan boş bir tablo tanımlanır (Network 1).

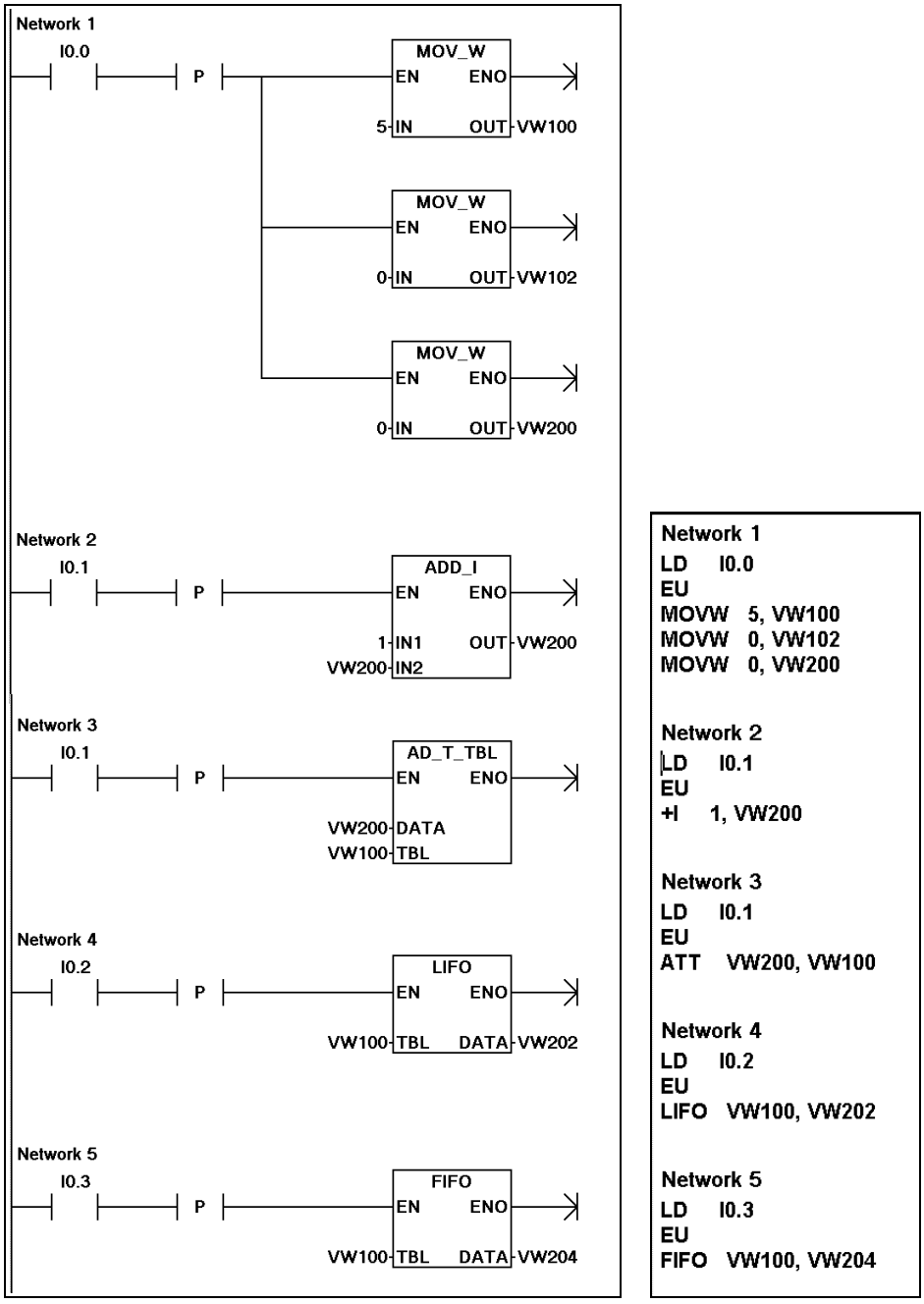
I0.1 girişinin çıkan kenarında VW200 adresini içeriği 1 arttırılır (Network 2) ve bu değer tabloya eklenir (Network 3). Bu durumda VW102 adresinin içeriği 1 olur.

I0.2 girişinin çıkan kenarında tabloya eklenen son veri tablodan çıkartılır ve VW202 adresine yazılır (Network 4). Bu durumda VW102 adresinin içeriği tekrar 0 olur.

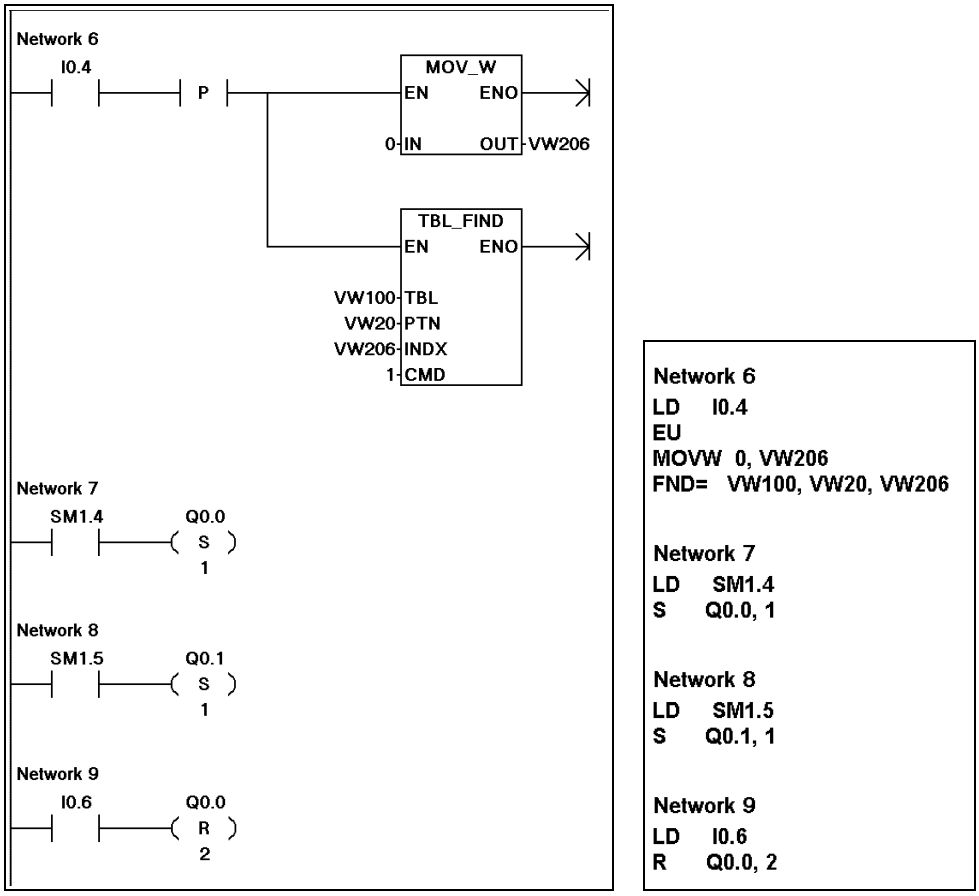
I0.3 girişinin çıkan kenarında tabloya ilk eklenen veri çıkartılır ve VW204 adresine yazılır (Network 5). Eğer tablo boşsa (VW102=0) SM1.5 biti bir tarama çevrimi 1 olur ve Q0.1 çıkışı 1'e kurulur (Network 8).

I0.4 çıkışının çıkan kenarında bu tabloda içeriği VW20 ile aynı olan bir veri varsa sıra numarası VW206'ya yazılır, yoksa VW206 adresine 5 yazılır.

Bu programda I0.0 girişi 1 yapıldıktan sonra art arda 6 kez I0.1 girişi 1 yapılırsa, tablonun dolduğunu belirten SM1.4 biti ve buna bağlı olarak Q0.0=1 olur. Eğer I0.2 veya I0.3 girişleri art arda 6 kez 1 yapılırsa tablonun boş olduğunu belirten SM1.5 biti ve buna bağlı olarak Q0.1=1 olur.



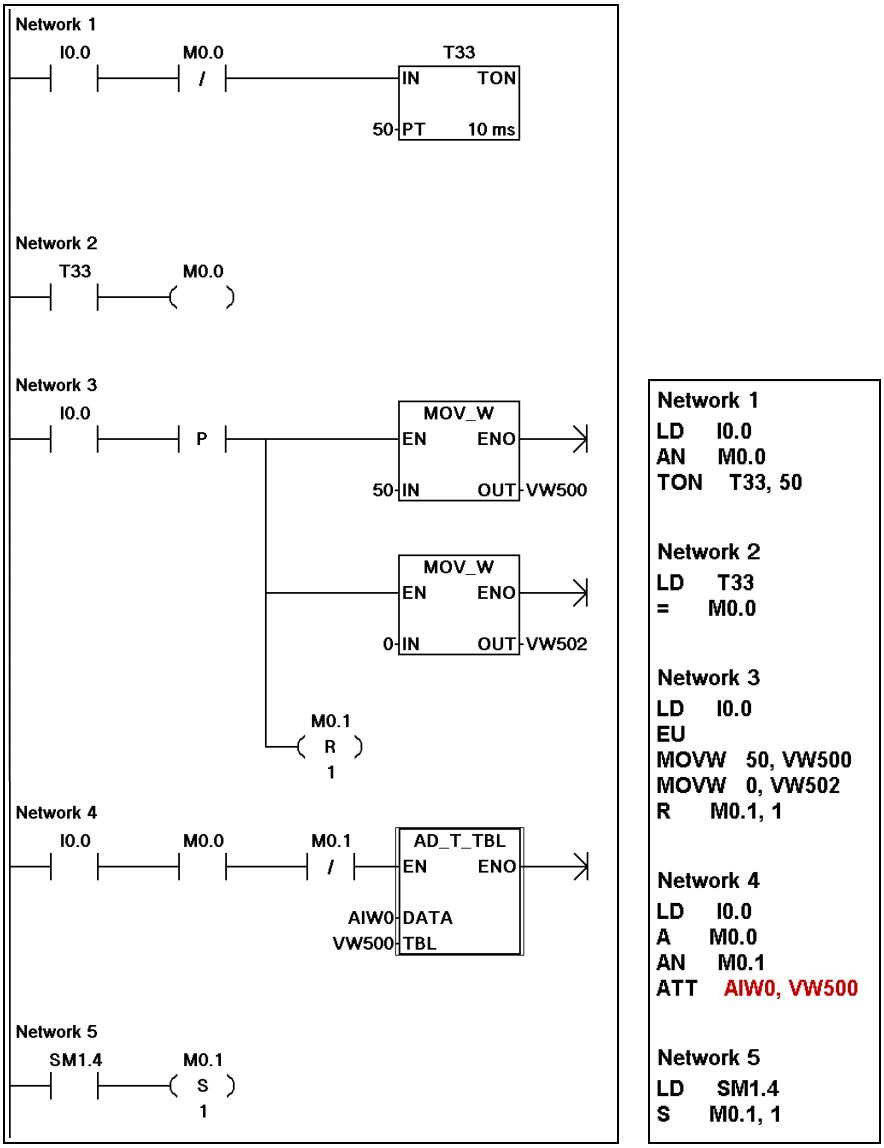
Şekil 5.40 Veri tablosu komutlarının incelenmesine ilişkin program örneği (1/2)



Şekil 5.40 Veri tablosu komutlarının incelenmesine ilişkin program örneği (2/2)

Veri tablosu kullanımına ilişkin bir program örneği Şekil 5.41'de verilmiştir. Bu program, PLC analog giriş birimine ilişkin AIW0 adresindeki 16 bitlik sayısal verilerin 500ms aralıklarla ve 25 saniye süreyle, VW504 adresinden başlanarak kaydedilmesi amacıyla kullanılabilir.

Bu programda I0.0=1 yapıldığında periyodu 500ms olan işaret darbeleri (M0.0) üretilir (Network 1 ve Network 2). I0.0 girişinin çıkan kenarında başlangıç adresi VW500 olan boş bir tablo oluşturulur ve tablo doldu durumunu gösteren M0.1 biti sıfırlanır (Network 3). Eğer I0.0=1 (başla), M0.0=1 (500ms süreli darbe işareti var) ve tablo dolu değil (M0.1=0) ise AIW0 adresinin içeriği tabloya eklenir. Tablo dolduğunda M0.1 adresinin içeriği 1'e kurulur. Böylece bir sonraki I0.0 girişinin 0'dan 1'e değiştirilmesine kadar yeni veri kaydedilmez.



Şekil 5.41 Veri kaydedilmesine ilişkin program örneği

5.8 DOLAYLI ADRESLEME

Dolaylı adresleme, bellek alanlarına gerçek adresi yerine bir işaretçi kullanılarak erişme yöntemidir. Bu yöntemde bellek adresleri işaretçi olarak tanımlanan 32 bitlik bir adreste tutulur. İşaretçi olarak 32 bitlik V bellek, L bellek ve akümülatör (AC1, AC2, AC3) kullanılabilir. Dolaylı adresleme ile I, Q, V, M, S, alanlarına erişilebilir; ancak 1 bitlik alanlara ve AI, AQ, HC, SM, L alanlarına erişilemez.

Bir bellek alanına ilişkin işaretçiyi tanımlamak için adresin önüne & işareti konur. Örneğin,

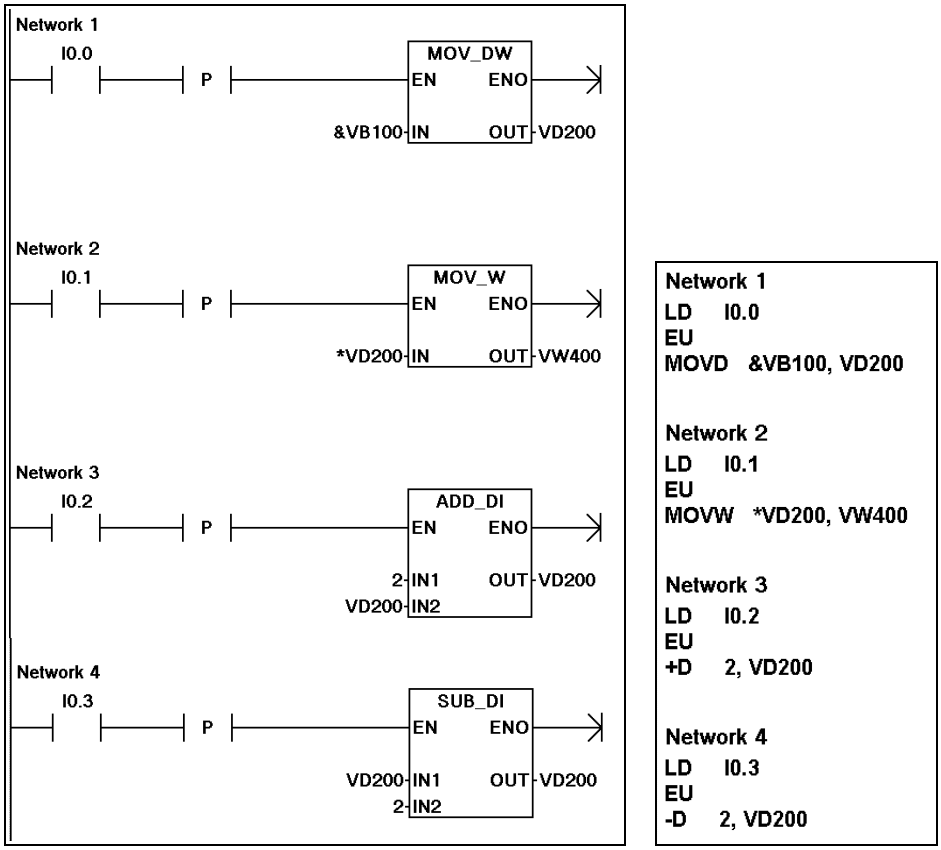
MOVD &VB100, VD200

komutu ile VB100 bellek alanının işaretçisi olarak VD200 adresi tanımlanır. VD200 işaretçisinin içeriği değiştirilerek VB100 adresini izleyen alanlara erişilir. İşaretçi, önüne * işareti konularak belirtilir. Örneğin,

MOVW *VD200, VW500

komutu ile VD200 işaretçisinin gösterdiği adresin içeriği VW500 adresine yazılır.

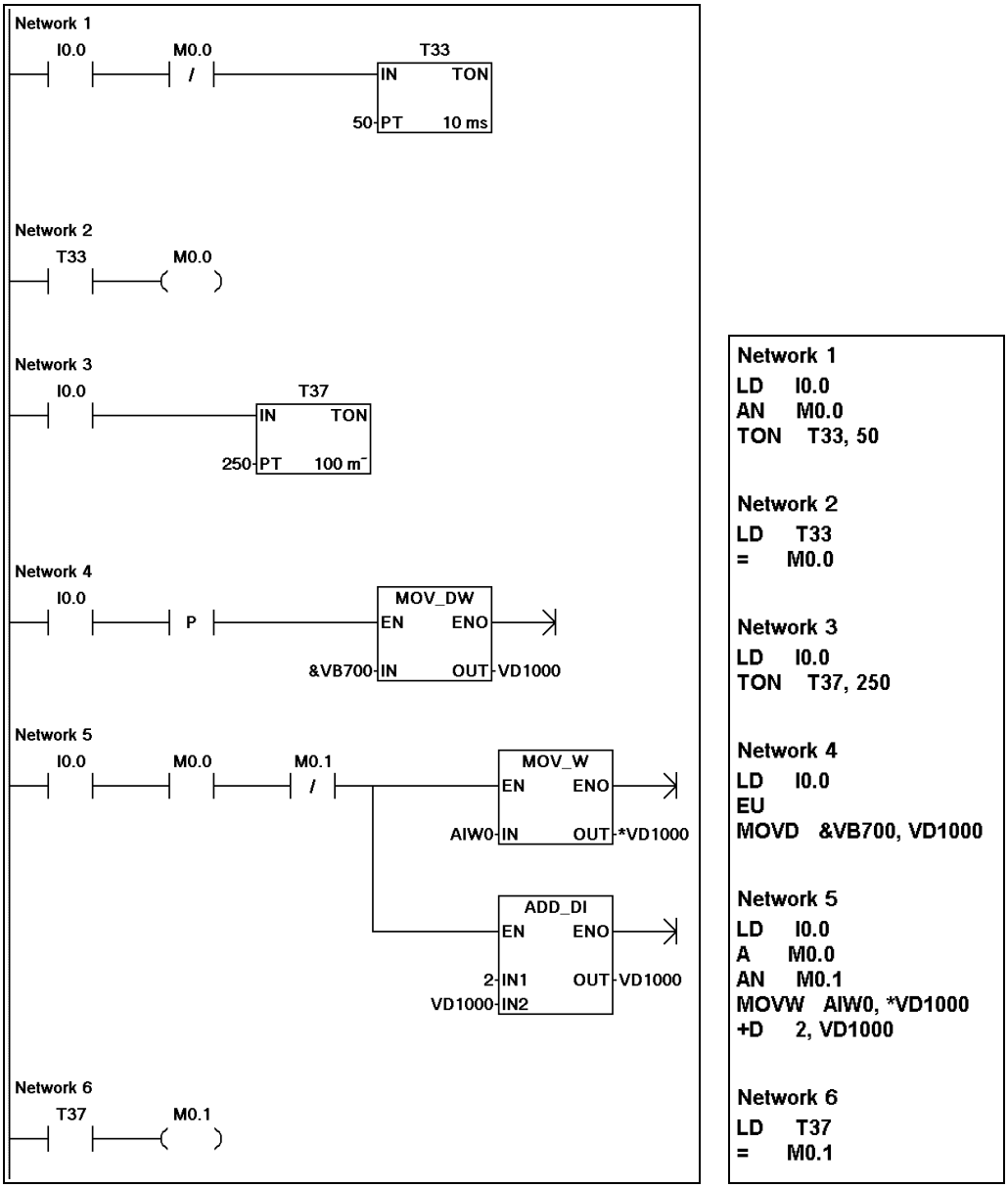
Dolaylı adreslemenin kullanımına ilişkin bir program parçası Şekil 5.42'de verilmiştir. Bu programa göre, I0.0=1 yapıldığında VB100 başlangıç adresine VD200 adresi işaretçi olarak atanır. I0.1=1 yapılırsa VW100 adresinin içeriği VW400 adresine yazılır. Eğer I0.2=1 yapıldıktan sonra I0.1=1 yapılırsa VW102 adresinin içeriği, I0.3=1 yapıldıktan sonra I0.1=1 yapılırsa VW98 adresinin içeriği VW400 adresine yazılır.



Şekil 5.42 Dolaylı adresleme program örneği

Bir önceki örnekte ele alınan AIW0 adresindeki 16 bitlik sayısal verilerin 500ms aralıklarla ve 25 saniye süreyle belirli bir değişken bellek alanına (V) kaydedilmesi uygulaması, dolaylı adresleme yöntemiyle Şekil 5.43'te verilen bir programla gerçeklenebilir.

Bu programda, (Network 1) ve (Network 2) I0.0=1 yapıldığında T33 zamanlayıcısı ile her 500ms'de, bir tarama çevrimi 1 değerini alan bir işaret üretilmesi (M0.0); (Network 3) sürenin başlatılması; (Network 4) VB700 başlangıç adresinin işaretçisi olarak VD1000 adresi seçilmesi işlevini sağlar. I0.0=1, M0.0=1 (500ms doldu) ve M0.1=0 (25s dolmadı) ise AIW0 analog adresinin içeriği VW700 adresinden başlanarak kaydedilir (Network 5).



Şekil 5.43 Dolaylı adresleme yöntemiyle veri toplamaya ilişkin program örneği

5.9 VERİLER ÜZERİNDE MANTIKSAL İŞLEMLER

Boyutu 8, 16 ve 32 bitlik veriler üzerinde birler tümleyeni, VE (AND), VEYA(OR) ve ÖZELVEYA (EXOR) mantıksal işlemleri yapılabilir.

8 bitlik verilere ilişkin mantıksal işlem komutları

INVB OUT

ANDB IN,OUT

ORB IN,OUT

XORB IN,OUT

16 bitlik verilere ilişkin mantıksal işlem komutları

INVW OUT

ANDW IN,OUT

ORW IN,OUT

XORW IN,OUT

32 bitlik verilere ilişkin mantıksal işlem komutları

INVD OUT

ANDD IN,OUT

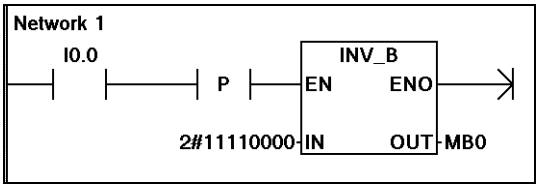
ORD IN,OUT

XORD IN,OUT

Birler tümleyenine ilişkin komutlar yürütüldüğünde OUT adresindeki verinin birler tümleyeni alınır ve sonuç yine OUT adresine yazılır.

Diğer mantıksal işlemlere ilişkin komutlar yürütüldüğünde IN adresindeki veri ile OUT adresindeki veriye ilgili mantıksal işlem uygulanır ve sonuç OUT adresine yazılır.

Şekil 5.44'te 8 bitlik bir sayısal verinin birler tümleyenine ilişkin bir program örneği verilmiştir. Bu programa göre I0.0=1 yapıldığında 2#11110000 ikilik sayının birler tümleyeni alınır ve 2#00001111 değeri MB0 adresine yazılır.




```
Network 1
LD    I0.0
EU
MOVB  2#11110000, MB0
INVB  MB0
```

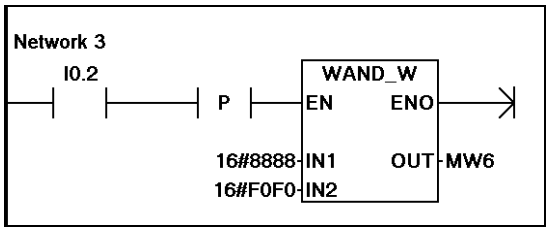
IN : VB, IB, QB, MB, SB, SMB, LB, AC, Sabit, *VD, *AC, *LD

OUT : VB, IB, QB, MB, SB, SMB, LB, AC, *VD, *AC, *LD

Şekil 5.44 8 bitlik verinin birler tümleyenine ilişkin program örneği

Şekil 5.45'te 16 bitlik sayısal verilere VE mantıksal işlemi uygulanmasına ilişkin bir program verilmiştir. Bu programa göre I0.2=1 yapıldığında

onaltılık 8888 (2#1000_1000_1000_1000) sayısı ile F0F0 (1111_0000_1111_0000) sayısına VE işlemi uygulanır ve onaltılık 8080 (2#1000_0000_1000_0000) değeri MW6 adresine yazılır.




```
Network 3
LD    I0.2
EU
MOVW  16#8888, MW6
ANDW  16#F0F0, MW6
```

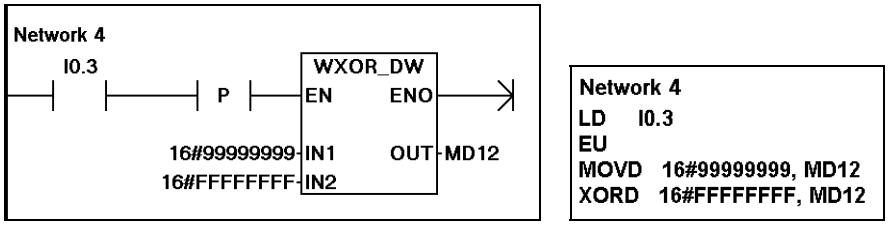
IN1, IN2 : VW, IW, QW, MW, SW, SMW, T, C, AC, LW, AIW, Sabit, *VD, *AC, *LD

OUT : VW, IW, QW, MW, SW, SMW, T, C, LW, AC, *VD, *AC, *LD

Şekil 5.45 16 bitlik sayısal veriler üzerinde VE işlemine ilişkin program örneği

Şekil 5.46'da 32 bitlik sayısal verilere ÖZELVEYA (EXOR) mantıksal işlemi uygulanmasına ilişkin bir program verilmiştir. Bu programa göre I0.3=1 yapıldığında

onaltılık 99999999 (1001_1001_1001_1001) sayısı ile FFFF (1111_1111_1111_1111) sayısına EXOR işlemi uygulanır ve 66666666 (0110_0110_0110_0110) sayısal değeri MD12 adresine yazılır.



IN1, IN2 : VD, ID, QD, MD, SMD, AC, LD, HC, Sabit, *VD, *AC, SD, *LD

OUT : VD, ID, QD, MD, SMD, LD, AC, *VD, *AC, SD, *LD

Şekil 5.46 32 bitlik sayısal veriler üzerinde ÖZELVEYA işlemine ilişkin program örneği

5.10 VERİLER ÜZERİNDE KAYDIRMA VE DÖNDÜRME İŞLEMLERİ

Boyutu 8, 16 ve 32 bitlik veriler üzerinde sağa veya sola kaydırma ve döndürme işlemleri yapılabilir.

8, 16 ve 32 bitlik veriler üzerinde sola kaydırma işlemi için

SLB OUT, N

SLW OUT, N

SLD OUT, N

komutları,

sağa kaydırma işlemi için

SRB OUT, N

SRW OUT, N

SRD OUT, N

komutları kullanılır. Bu komutlar yürütüldüğünde OUT adresindeki bitler N sayısı kadar sola veya sağa kaydırılır ve sonuç OUT adresine yazılır.

8, 16 ve 32 bitlik veriler üzerinde sola döndürme işlemi için

RLB OUT, IN

RLW OUT, IN

RLD OUT, IN

sağa döndürme işlemi için

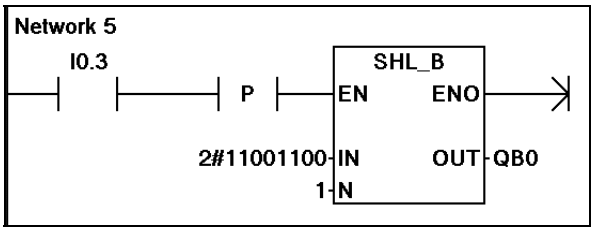
RRB OUT, IN

RRW OUT, IN

RRD OUT, IN

komutları kullanılır. Bu komutlar yürütüldüğünde OUT adresindeki bitler N sayısı kadar sola veya sağa döndürülür ve sonuç OUT adresine yazılır.

Şekil 5.47'de 8 bitlik veriler üzerinde sola ve Şekil 5.48'de sağa kaydırma işlemlerine ilişkin program örnekleri verilmiştir.




```
Network 5
LD    I0.3
EU
MOVB  2#11001100, QB0
SLB   QB0, 1
```

IN (LAD, FBD) : VB, IB, QB, MB, SB, SMB, LB, AC, Sabit, *VD, *LD, *AC

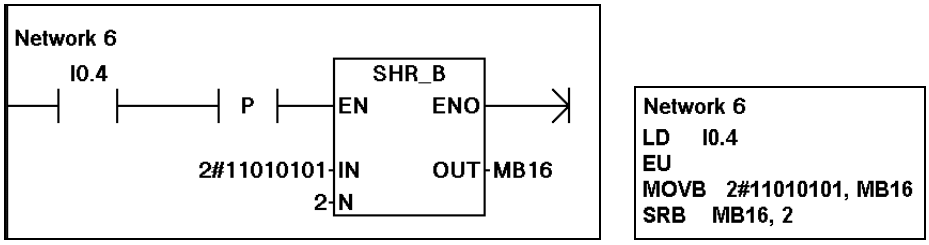
N : VB, IB, QB, MB, SB, SMB, LB, AC, Sabit, *VD, *LD, *AC

OUT : VB, IB, QB, MB, SB, SMB, LB, AC, *VD, *LD, *AC

Şekil 5.47 8 bitlik veriler üzerinde sola kaydırma işlemi

Şekil 5.47'de verilen programa göre I0.3=1 yapıldığında 2#11001100 ikilik sayısı 1 basamak sola kaydırılır ve elde edilen 2#10011000 değeri QB0 [Q0.7, Q0.6, Q0.5, Q0.4, Q0.3, Q0.2, Q0.1, Q0.0] adresine yazılır. Bu durumda Q0.7=1, Q0.4=1 ve Q0.3=1 olur.

Şekil 5.48'de verilen programa göre I0.4=1 yapıldığında 2#11010101 ikilik sayısı 2 basamak sağa kaydırılır ve elde edilen 2#0011_0101 değeri MB16 adresine yazılır.



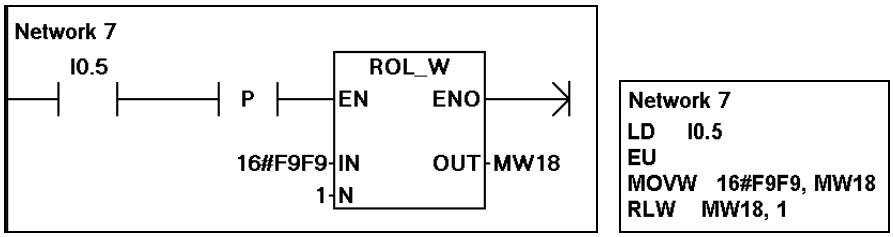
IN (LAD, FBD) : VB, IB, QB, MB, SB, SMB, LB, AC, Sabit, *VD, *LD, *AC

N : VB, IB, QB, MB, SB, SMB, LB, AC, Sabit, *VD, *LD, *AC

OUT : VB, IB, QB, MB, SB, SMB, LB, AC, *VD, *LD, *AC

Şekil 5.48 8 bitlik veriler üzerinde sağa kaydırma işlemi

Şekil 5.49'da 16 bitlik veriler üzerinde sola ve Şekil 5.50'de sağa döndürme işlemlerine ilişkin program örnekleri verilmiştir.



IN (LAD, FBD) : VW, T, C, IW, QW, MW, SW, SMW, LW, AC, AIW, Sabit, *VD, *LD, *AC

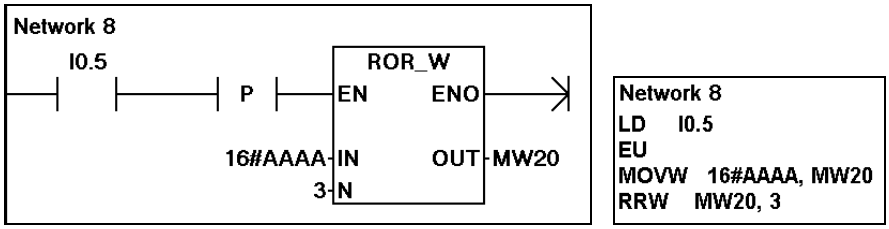
N : VB, IB, QB, MB, SB, SMB, LB, AC, Sabit, *VD, *LD, *AC

OUT : VW, T, C, IW, QW, MW, SW, SMW, LW, AC, *VD, *LD, *AC

Şekil 5.49 16 bitlik veriler üzerinde sola döndürme işlemi

Şekil 5.49'da verilen programa göre I0.5=1 yapıldığında onaltılık F9F9 (2#1111_1001_1111_1001) sayısı 1 basamak sola döndürülür ve elde edilen 2#1111_0011_1111_0011 değeri MW18 adresine yazılır.

Şekil 5.50'de verilen programa göre I0.5=1 yapıldığında onaltılık AAAA (2#1010_1010_1010_1010) sayısı 1 basamak sağa döndürülür ve elde edilen (2#0101_0101_0101_0101) değeri MW20 adresine yazılır.



IN (LAD, FBD) : VW, T, C, IW, QW, MW, SW, SMW, LW, AC, AIW, Sabit, *VD, *LD, *AC

N : VB, IB, QB, MB, SB, SMB, LB, AC, Sabit, *VD, *LD, *AC

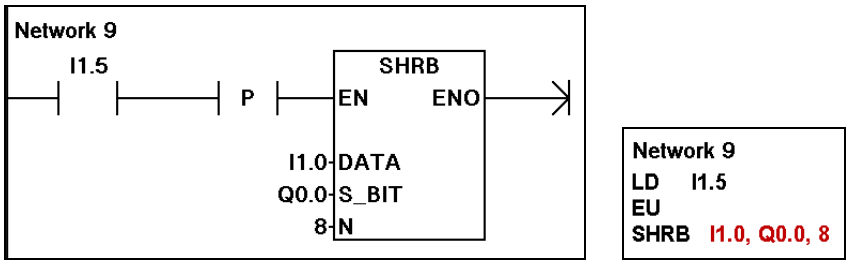
OUT : VW, T, C, IW, QW, MW, SW, SMW, LW, AC, *VD, *LD, *AC

Şekil 5.50 16 bitlik veriler üzerinde sağa döndürme işlemi

Kaydırma işlemleri ilişkin diğer bir komut bit kaydırma işlevini sağlayan ve

SHRB DATA, S_BIT, N

biçiminde yazılan komuttur. Burada DATA kaydırılan bit adresi, S_BIT kaydedilecek alanın en düşük anlamlı bit adresi, N kaydedilecek alanın boyutudur. Eğer N pozitif olursa sola, negatif olursa sağa kaydırma işlemi yapılır. Bu komuta ilişkin bir program örneği Şekil 5.51'de verilmiştir.



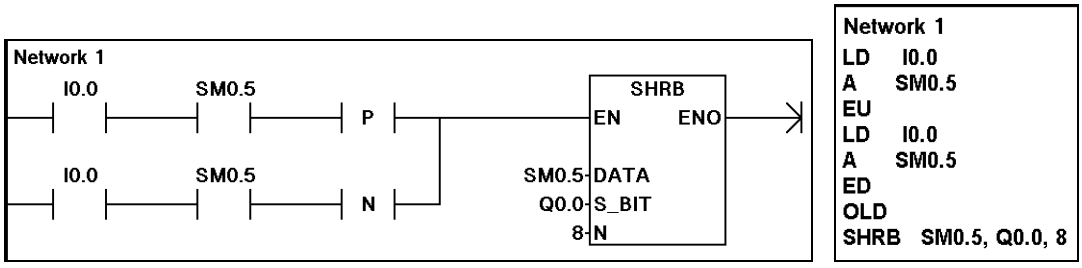
DATA, S_BIT : I, Q, M, SM, T, C, V, S, L

N : VB, IB, QB, MB, SB, SMB, LB, AC, Sabit, *VD, *LD, *AC

Şekil 5.51 Bit kaydırma işlemine ilişkin örnek program

Bu programa göre, en düşük anlamlı bit adresi S_BIT:=Q0.0, kaydedici alanı [Q0.7, Q0.6, Q0.5, Q0.4, Q0.3, Q0.2, Q0.1, Q0.0] olarak tanımlanır. I1.5=1 yapıldığında I1.0 adresinin içeriği Q0.0 adresine yazılır, önceki değerler bir basamak sola kaydırılır. Eğer N=-8 yazılır ve aynı işlem yapılırsa bu kez I1.0 adresinin içeriği kaydedici Q0.7 adresine yazılır ve önceki değerler bir basamak sağa kaydırılır.

Bu komutun kullanımına ilişkin diğer bir örnek Şekil 5.52'de gösterilmiştir.



Şekil 5.52 Bit kaydırma işlemine ilişkin bir örnek program

Bu programa göre I0.0=1 yapıldığında 1 saniye periyotlu SM0.5 özel bellek bitinin çıkan kenarında 1 değeri, düşen kenarında 0 değeri Q0.0 adresine yazılır ve diğer adresler kaydırılarak kaydedilir.

BÖLÜM 6
GERÇEK ZAMAN İŞLEMLERİ

6.1 GİRİŞ

Bir kumanda sisteminde yıl, ay, gün, saat ve saniye gibi tarihe bağlı kumanda işaretlerinin üretilmesi gerçek zaman işlemlerine ilişkin komutlar ile gerçeklenir. Gerçek zaman işlemleri, zaman bilgisini okuma, yazma ve karşılaştırma gibi işlevleri sağlayan komutlar ile yapılır.

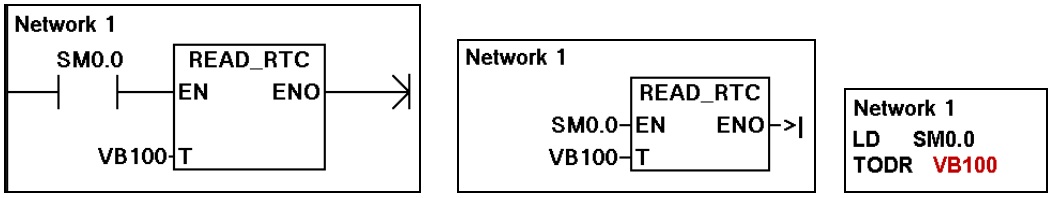
6.2 ZAMAN BİLGİSİNİN OKUNMASI

Gerçek zaman bilgilerini okumak için TODR veya TODRX komutları kullanılır. CPU 21x ve CPU 22x işlemcilerinde geçerli olan TODR komutu ile okunan zaman bilgisi 8 baytlık bir veri tablosuna yazılır. Yalnız yeni nesil (CPU 22x) işlemcilerde geçerli olan TODRX komutu ile okunan zaman bilgisi için ise 19 baytlık veri alanı kullanılır.

Zaman bilgisini belirli bir bellek alanına yazmak için

TODR T

biçiminde yazılan komut kullanılır. Bu komutun işletilmesi ile gerçek zaman bilgileri T ile belirtilen adrese ve bu adresi izleyen 7 baytlık alana yazılır. Örneğin, Şekil 6.1'de verilen program parçası ile gerçek zaman bilgileri VB100, VB101, ... , VB107 adreslerine yazılır.



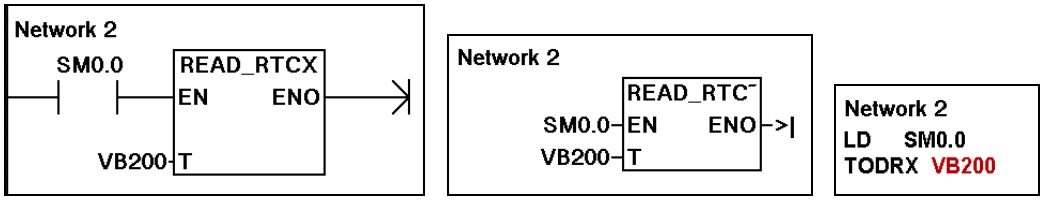
T : VB, IB, QB, MB, SMB, SB, LB, *VD, *AC, *LD

ŞEKİL 6.1 Gerçek zaman değerlerinin okunmasına ilişkin program örneği

Tarihe ilişkin bilgiler BCD formatında aşağıda belirtilen sırada bellek alanına yazılır.

Bellek adresi	veri	Açıklama
VBxx	Yıl (0-99)	VB100
VB(xx+1)	Ay (1-12)	VB101
VB(xx+2)	Gün (1-31)	VB102
VB(xx+3)	Saat (0-23)	VB103
VB(xx+4)	Dakika (0-59)	VB104
VB(xx+5)	Saniye (0-59)	VB105
VB(xx+6)	00	VB106
VB(xx+7)	Haftanın günü (1-7)	VB107, 1=Pazar

Yeni nesil (CPU 22x) işlemcilerde geçerli olan TODRX komutu ile gerçek zaman bilgisi 19 baytlık veri alanına yazılır. Bu komutun farkı yaz-kış saati uygulamasına ilişkin bilgileri de içermesidir. Veri tablosunun ilk 8 baytı TODR komutu ile kullanılan tablo ile aynı, diğer 11 bayt yaz ve kış saati uygulamasına ilişkin verileri içerir. Örneğin, Şekil 6.2'de verilen program parçası ile gerçek zaman bilgisine ilişkin veriler VB200, VB201, ...,VB218 adreslerine aşağıdaki tabloda belirtildiği gibi yazılır.

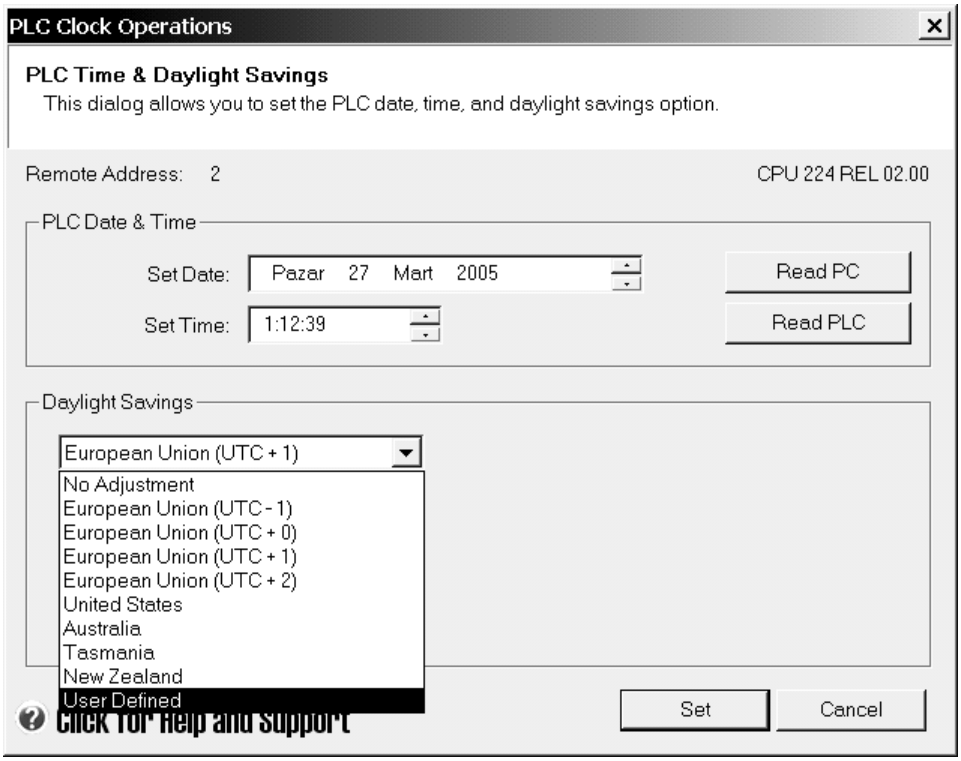


T : VB, IB, QB, MB, SMB, SB, LB, *VD, *AC, *LD

Şekil 6.2 Genişletilmiş gerçek zaman değerlerinin okunmasına ilişkin program örneği

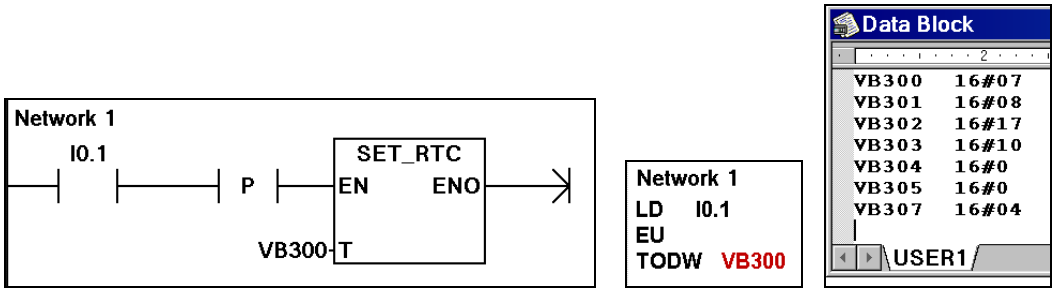
Bellek adresi	veri	Açıklama
VBxx	Yıl (0-99)	VB200
VB(xx+1)	Ay (1-12)	VB201
VB(xx+2)	Gün (1-31)	VB202
VB(xx+3)	Saat (0-23)	VB203
VB(xx+4)	Dakika (0-59)	VB204
VB(xx+5)	Saniye (0-59)	VB205
VB(xx+6)	00	VB206
VB(xx+7)	Haftanın günü (1-7)	VB207, 1=Pazar
VB(xx+8)	Düzeltme modu	FF: Kullanıcı tanımlı
VB(xx+9)	Düzelteme saati	16#01: 1 saat
VB(xx+10)	Düzeltme dakikası	16#00: 0 dakika
VB(xx+11)	Yaz saati başlama ayı	16#03: Mart
VB(xx+12)	Yaz saati başlama günü	16#25 : ayın 25. günü
VB(xx+13)	Yaz saati başlama saati	16#01: saat 1
VB(xx+14)	Yaz saati başlama dakikası	16#00: dakika 00
VB(xx+15)	Kış saati başlama ayı	16#10: Ekim
VB(xx+16)	Kış saati başlama günü	16#28 : ayın 28. günü
VB(xx+17)	Kış saati başlama saati	16#02: saat 02
VB(xx+18)	Kış saati başlama dakikası	16#00: dakika 00

Gerçek zamana bağlı kumanda işlemlerinin doğru olarak yapılabilmesi için PLC gerçek zaman saatinin doğru ayarlanması gerekir. Bunun için programlama ana penceresinden "PLC > Time of Day Clock" sırası izlenerek Şekil 6.3'te gösterilen pencere açılır ve gerekli ayarlar yapılır.



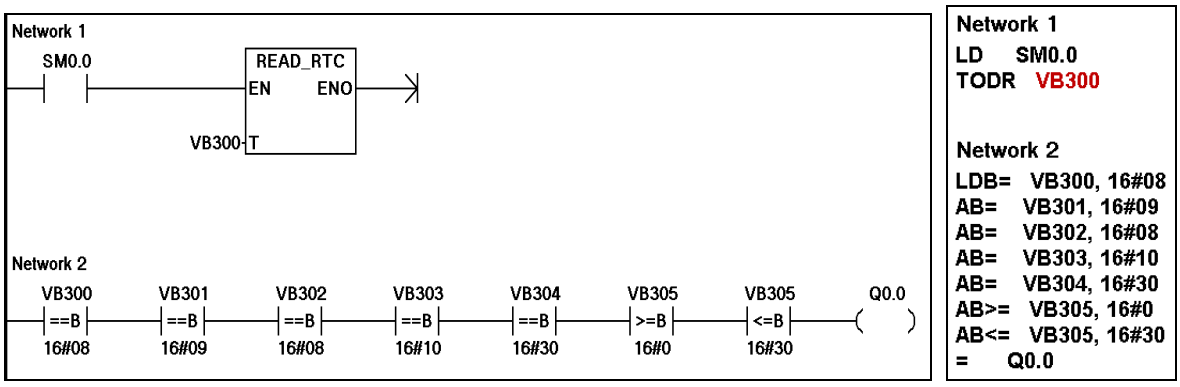
Şekil 6.3 Gerçek zaman saatine ilişkin ayarların yapıldığı pencere

Gerçek zaman saatine ilişkin ayarlar PLC'ye yazılan bir programla da değiştirilebilir. Bunun için zaman bilgilerini içeren bir veri tablosu oluşturulur. Yaz-kış saati seçeneği olmayan bir tablo için 8 bayt, yaz-kış saati bilgilerini içeren bir tablo için 19 baytlık alan ayrılır ve istenen değerler ilgili bellek alanlarına yazılır. Ayar yapılmak istendiğinde bu tabloların başlangıç adresleri ile TODW ve TODWX komutları kullanılarak ayar bilgileri PLC'ye yazılır. Örneğin, PLC gerçek zaman saatini 17/08/2007 tarihinde saat 10'a ayarlamak için Şekil 6.4'te verilen program parçası yazılabilir. Bu programa göre ayarların güncelleştirilmesi için "Data Block" alanına veri tablosu yazılabilir ve I0.1 girişi 0'dan 1'e değiştirilir.



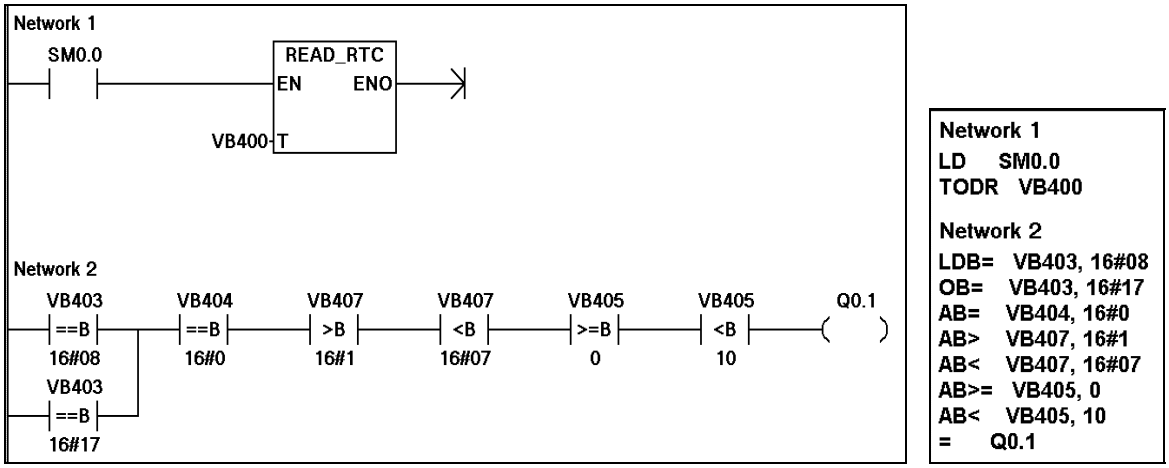
Şekil 6.4 Gerçek zaman saatinin ayarlayan program örneği

Gerçek zamana bağlı kumanda işaretlerinin üretilmesi için veri tablosundan 8 bitlik veriler okunur ve 8 bitlik karşılaştırma komutları kullanılır. Şekil 6.5'te tarihe bağlı kumanda işlemlerine ilişkin bir program örneği verilmiştir. Bu programa göre 08/09/2008 tarihinde saat 10.30'da 30 saniye süreli bir kumanda işareti (Q0.0=1) üretilecektir.



Şekil 6.5 Tarihe bağlı kumanda işlemlerine ilişkin bir program örneği

Şekil 6.6'da tarihe bağlı kumanda işlemlerine ilişkin başka bir program örneği verilmiştir. Bu örnekte Cumartesi ve Pazar dışında her gün saat 8'de ve 17'de 10 saniye süreli bir kumanda işareti (Q0.1=1) üretilmektedir.

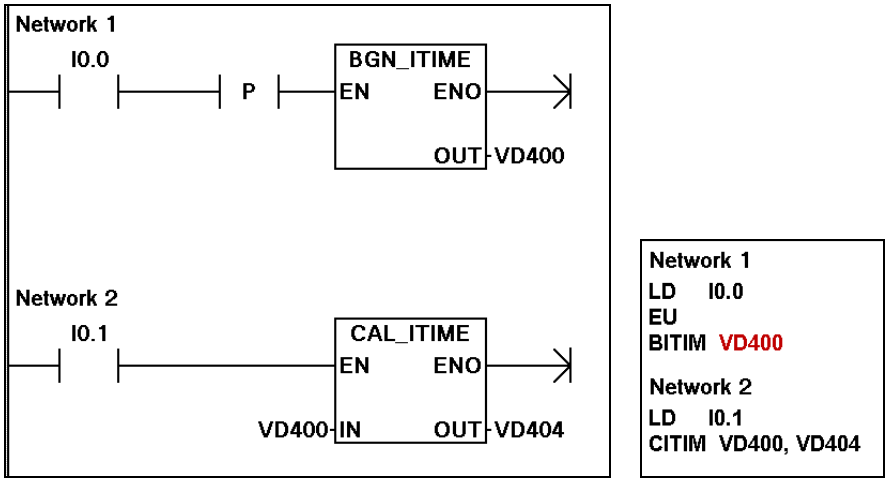


Şekil 6.6 Tarihe bağlı kumanda işlemlerine ilişkin bir program örneği

6.3 UZUN SÜRELİ 1MS ARTIMLI ZAMAN BİLGİSİ

S7-200 PLC ailesinde ms çözünürlükte zamana bağlı kumanda işaretlerinin üretilmesinde 1ms artımlı (T32, T96 gibi) zamanlayıcılar kullanılabilir. Ancak bu zamanlayıcılar ile elde edilebilecek en uzun süre 32767ms=32.767s değerindedir. Bölüm 3.7'de değinildiği gibi daha uzun süreler elde etmek için zamanlayıcı ve sayıcılar birlikte kullanılabilir. Yeni nesil CPU 22x işlemcili PLC'lerde bu amaçla kullanılabilecek bir komut çifti bulunur. Bu komut çifti ile 1ms artımlı uzun süreler elde edilebilir. Zaman bilgisi 32 bitlik bellek alanına yazıldığından 2^{32} = 4294967296ms=1193.046 saat =49.7 gün gecikme süresi elde edilebilir.

Bu komutların kullanımına ilişkin bir program parçası Şekil 6.7'de verilmiştir. Bu programa göre I0.0 girişinin yükselen kenarı ile BITIM komutu etkin olur ve süre başlar. I0.1 girişi 1 yapıldığında CITIM komutu etkin olur ve geçen süre ms olarak VD404 bellek alanına yazılır. Sürenin başlaması için BITIM komutunun etkin durumdan çıkartılması gerekir. Bunun için yalnız bir tarama çevrimi BITIM komutunun etkin olmasını sağlayan çıkan kenar komutu (EU) kullanılır.



Şekil 6.7 Yüksek çözünürlüklü uzun süreli gecikme işareti elde edilmesini sağlayan program örneği

BÖLÜM 7
KESMELİ ÇALIŞMA

7.1 GİRİŞ

Kesmeli çalışma, kontrol biriminin veya işlemcinin olağan program tarama çevrimi sürecinde yaptığı işlemleri geçici olarak durdurmak ve öncelikli veya süreli başka görevlere yönlendirmek olarak tanımlanabilir. İşlemcinin olağan tarama çevriminden bağımsız işlem yapmayı gerektiren görevlerin yerine getirilebilmesi bu şekilde sağlanır. S7-200 CPU 22x işlemcilerinde olay, süre, iletişim veya hızlı sayıcı durumlarına bağlı kesmeli çalışma türleri vardır.

7.2 KESMELİ ÇALIŞMAYA İLİŞKİN GENEL KONULAR

Kesmeli çalışmanın gerçeklenebilmesi için izin veya yetki, kesme kaynağı (olay, süre, iletişim veya hızlı sayıcı) kesmeli çalışmada işletilecek altprogram ve kesme türü ile ilgili koşullar tanımlanır ve güncelleştirilir. Bu amaçla çeşitli komutlar ve özel denetim yazmaçları kullanılır. Kesmeli çalışma koşulları sağlandığı anda, yürütülen komut işlenir ve kesme ile ilişkili altprogram yürütülmeye başlanır, altprogramdaki tüm komutlar işlendikten sonra tekrar ana programa, işlenen son komutu izleyen komuta dönülür. Kesmeli çalışmanın gerçeklenebilmesi için kesmeli çalışma durumuna izin verilmesi gerekir. Bunun için

ENI

Kesmeli çalışma sürecine son vermek için

DISI

komutu kullanılır. Bu komut kesme olaylarına bağlı altprogramların yürütülmesini önler. Ancak, yürürlükte olan kesme istekleri kesme kayıt kuyruğunda sıralanır. Kesmeli çalışmaya tekrar izin verildiğinde kuyruktaki kesme istekleri tekrar öncelik sırasına göre işleme konur.

Kesme ile ilgili altprogramdan belirli koşullara bağlı olarak çıkmak için

CRETI

koşullu dönüş komutu kullanılır.

Herhangi bir kesme olayı ile işletilecek olan altprogramı belirtmek için

ATCH INT, EVENT

komutu kullanılır. Burada INT işletilecek kesme altprogramı, EVENT ise kesme olay numarasıdır. Her kesme olayı değişik bir sayı değeri ile tanımlanır. Örneğin, zamana bağlı kesme olayları için 10 ve 11; olaya bağlı kesmeler için 0, 1, 2, 3, 4, 5, 6, 7; iletişim kesmeleri için 8 ve 9; yüksek hız sayıcısı kesmeleri için 12, 13, 14, 15, 16, 17, 18 değerleri kullanılır.

Kesme olayı ile altprogramı ayırmak için

DTCH EVENT

komutu kullanılır. Bu komutla, kesme olayı ile kesme altprogramının işletilmesine son verilir.

Kesme kuyruğundaki belirli bir olaya ilişkin kayıtları silmek için

CEVNT EVENT

komutu kullanılır.

Simatic S7-200 ailesi CPU 22x işlemcili PLC'lerde işlemci türüne bağlı olarak, CPU 221ve CPU 222 işlemcili PLC'lerde

8 adet olaya bağlı,

2 adet zamana bağlı,

2 adet zamanlama kesmesi (T32, T96)

2 adet seri iletişim,

7 adet yüksek hız sayıcısı ve

2 adet darbe çıkışı kesme işlemi gerçeklenebilir.

CPU 224 ve CPU 226 işlemcili PLC'lerde, ayrıca değişik özelliklerde hızlı sayıcı ve iletişime ilişkin kesmeli çalışma durumları vardır.

Kesme altprogramlarında, DISI, ENI, HDEF ve END gibi komutlar geçerli değildir.

Bir kesme olayına ilişkin altprogram yürütülmeye başlandıktan sonra, bu altprograma ilişkin tüm komutlar işlenmeden ana programa dönülmez. Bir kesme altprogramı yürütülürken oluşan yeni kesme istekleri kesme kayıt kuyruğuna kaydedilir. Her kesme altprogramı tamamlandıktan sonra, kesme kayıt kuyruğundaki kesme istekleri öncelik sırasına göre işleme alınır. Kesme olayları yüksek, orta ve düşük öncelikli olarak sınıflandırılır. Her sınıf içindeki olayların ayrıca bir öncelik sırası vardır.

S7-200 CPU 22x işlemcilerinde

İletişim kesmeleri yüksek öncelikli,

Olaya bağlı kesme istekleri orta öncelikli,

Zamana bağlı kesme istekleri düşük öncelikli olarak sınıflandırılır.

İşlemciye iletilen kesme isteklerinden işleme alınmayanlar, kesme kayıt kuyruğunda tutulur. Ancak,

İletişim kesmeleri için en çok 4 veya 8 (CPU 226) kayıt,

Olaya bağlı kesmeler için 16 kayıt

Zamana bağlı kesmeler için 8 kayıt yapılabilir.

Kesme kayıt kuyruğu dolduktan sonra yeni kesme istekleri kayıt edilmez. Eğer,

İletişim kaynaklı kesmelere ilişkin kayıt kuyruğu dolu ise SM4.0,

Olay kaynaklı kesmelere ilişkin kuyruk dolu ise SM4.1,

Zaman kaynaklı kesmelere ilişkin kuyruk dolu ise SM4.2 durum bitleri 1'e kurulur. Kesme kayıt kuyruğu boşaldığında işlemci bu bitleri tekrar sıfırlar.

Uygulamalarda, kesme altprogramlarının işletildiği anda sonuçların aktarılması için, genellikle, anında (ivedi) işlem komutları kullanılır. Böylece işlem sonucunun olanaklı en kısa sürede tamamlanması sağlanır.

7.3 OLAYA BAĞLI KESMELİ ÇALIŞMA

Olaya bağlı kesmeli çalışma; kesme kaynağı olay olan bir kesmeli çalışma türüdür. Olaylar PLC'nin belirli giriş noktalarına gelen işaretlerin yükselen ya da düşen kenarları ile algılanır. Olaya bağlı 8 adet kesme kaynağı vardır. Bunlar:

0 sayılı kesme olayı PLC I0.0 giriş noktasına uygulanan işaretin yükselen kenarı

1 sayılı kesme olayı PLC I0.0 giriş noktasına uygulanan işaretin düşen kenarı

2 sayılı kesme olayı PLC I0.1 giriş noktasına uygulanan işaretin yükselen kenarı

3 sayılı kesme olayı PLC I0.1 giriş noktasına uygulanan işaretin düşen kenarı

4 sayılı kesme olayı PLC I0.2 giriş noktasına uygulanan işaretin yükselen kenarı

5 sayılı kesme olayı PLC I0.2 giriş noktasına uygulanan işaretin düşen kenarı

6 sayılı kesme olayı PLC I0.3 giriş noktasına uygulanan işaretin yükselen kenarı

7 sayılı kesme olayı PLC I0.3 giriş noktasına uygulanan işaretin düşen kenarı

olarak tanımlanır.

Bir kesme olayı ile hangi kesme altprogramının işletileceği

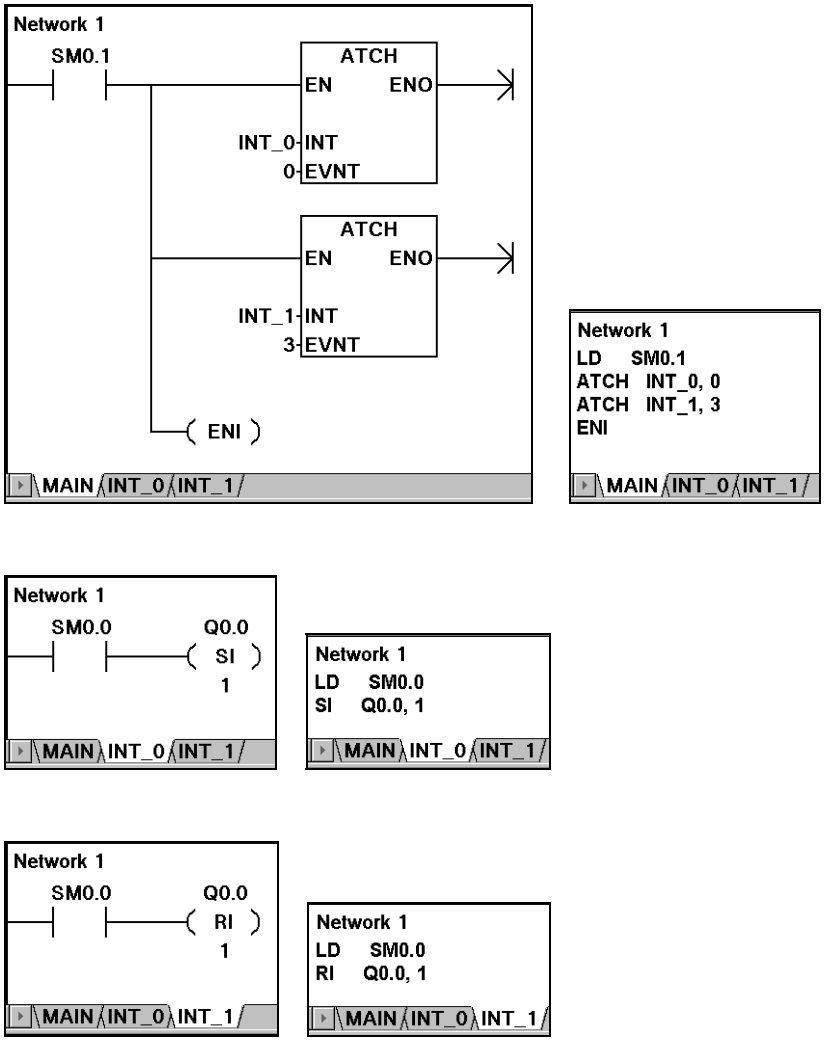
ATCH INT, EVENT

komutu ile belirtilir.

Örnek 7.1 PLC I0.0 girişine bağlı normalde açık düğmeye basıldığında en kısa sürede Q0.0=1 olmasını sağlayan, I0.1 girişine bağlı normalde kapalı düğmeye basıldığında en kısa sürede Q0.0=0 olmasını sağlayan programı yazınız.

Bu koşulları sağlayan bir program örneği Şekil 7.1'de gösterilmiştir. Burada, program çerçevesinin alt bölümündeki açık zemin üzerindeki koyu yazılar ilgili programı göstermektedir. Bu programa göre "MAIN" ana programına yazılan komutlar ile ilk tarama çevriminde (SM0.1=1); 0 numaralı kesme olayı (I0.0 girişine uygulanan işaretin yükselen kenarı) INT_0 kesme altprogramı ile, 3 numaralı kesme olayı (I0.1 girişine uygulanan işaretin düşen kenarı) INT_1 kesme altprogramı ile ilişkilendirilir ve kesmeli çalışmaya izin verilir.

INT_0 kesme altprogramında Q0.0 çıkışını 1'e kuran, INT_1 kesme altprogramında Q0.0 çıkışını sıfırlayan ivedi işlem komutları kullanılarak, sonucun komut işlendiği anda çıkışa iletilmesi sağlanır.



Şekil 7.1 Olaya bağlı kesmeli çalışmaya ilişkin program örneği

7.4 ZAMANA BAĞLI KESMELİ ÇALIŞMA

Zamana bağlı kesmeli çalışma, tarama süresinden bağımsız olarak, belirli sürelerde gerçekleşen bir kesmeli çalışma türüdür. Bu çalışma türü kesme süresi ve ilişkili altprogram belirtilerek tanımlanır. CPU 22x işlemcilerinde kesme olay numarası 10 ve 11 olan zamana bağlı iki ayrı kesmeli çalışma gerçeklenebilir. Kesme işaretlerinin üretilme süresi 10 olay sayılı kesmeli çalışma için SMB34 ve 11 olay sayılı kesmeli çalışma için SMB35 özel bellek yazmaçlarına kaydedilir. Bu yazmaçlara 1 ile 255

arasında yazılan tamsayı değerler [ms] olarak kesme sürelerini belirtir. Süre değerinin yazmaçlara kaydedilmesi MOVB komutu ile yapılır.

Örneğin,

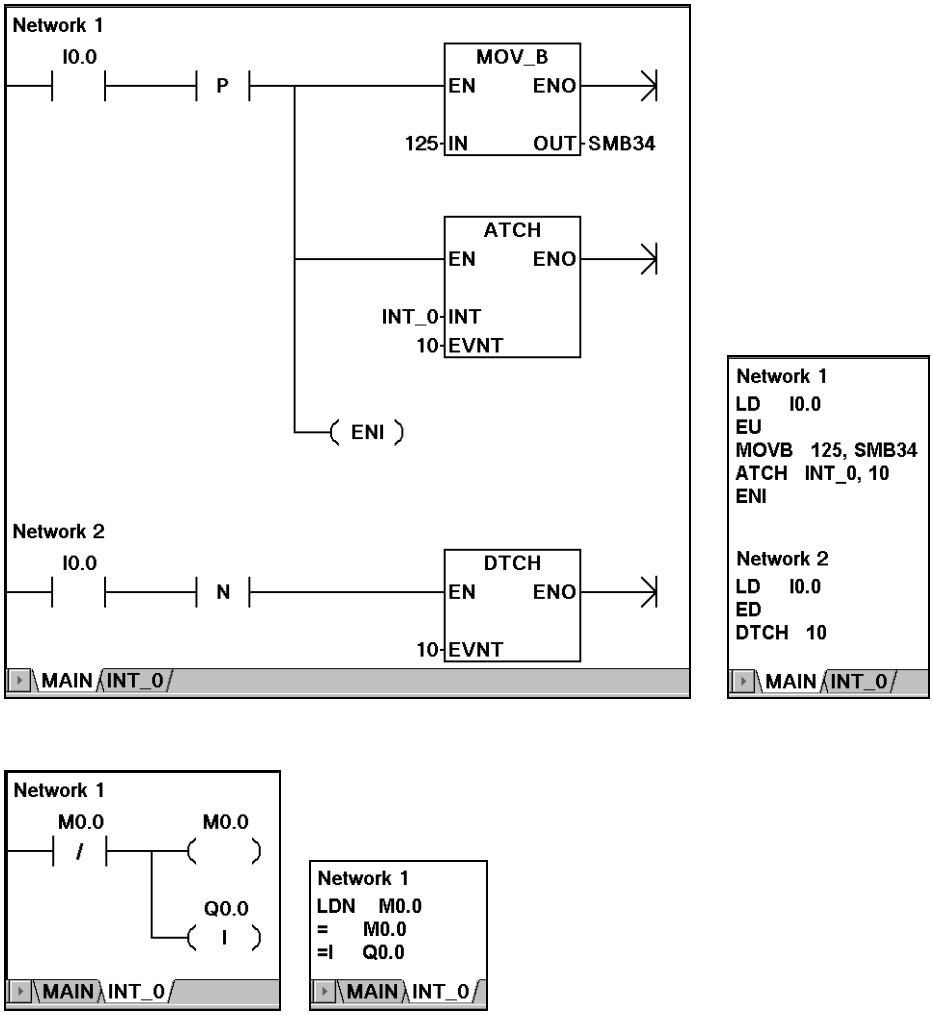
MOVB 10, SMB34

komutu ile SMB34 özel yazmacına 10 değeri yazılır ve süreli kesme işleminin her 10 ms'de bir yapılacağı belirtilmiş olur.

Zamana bağlı kesmeli çalışma, genellikle, matematiksel ifadelerin sayısal olarak gerçek zamanlı gerçeklenmesinde kullanılır. Kontrol uygulamalarında, belirli bir örnekleme zamanı için verilen sayısal kontrolör veya filtrelere ilişkin ifadeler, her örnekleme anında işletilen altprogramlara yazılarak gerçeklenir.

Örnek 7.2 PLC I0.0 girişi 1 yapıldığında her 125ms'de bir Q0.0 çıkışının durumunu değiştiren ve I0.0=0 yapıldığında bu işleme son veren bir program yazınız.

Bu özellikleri sağlayan bir program örneği Şekil 7.2'de gösterilmiştir. Bu programa göre "MAIN" anaprogramına yazılan komutlar ile I0.0 girişi 0'dan 1'e değiştirildiğinde 10 numaralı kesme olayına ilişkin süre değeri SMB34 özel bellek alanına yazılır, INT_0 kesme altprogramı ile 10 numaralı kesme olayı ilişkilendirilir ve kesmeli çalışmaya izin verilir. I0.0 giriş1'den 0'a değiştirdiğinde kesme olayı ile ilgili altprogramın ilişkisine son verilir (Network 2).



Şekil 7.2 Zamana bağlı kesmeli çalışma program örneği

7.5. ZAMANLAYICI KAYNAKLI KESMELİ ÇALIŞMA

CPU 22x işlemcili PLC'lerde 1ms zaman artımlı T32 ve T96 zamanlayıcıları ile kesmeli çalışma yapılabilir. T32 zamanlayıcısı ile 21 olay sayılı kesme, T96 zamanlayıcısı ile 22 olay sayılı kesme işareti üretilir. Kesme işareti zamanlayıcı içeriği PT ayar değerine erişildiğinde oluşur. Zamanlayıcı ile işletilen kesme altprogramı T32 zamanlayıcısı için

ATCH INT_X, 21

T96 zamanlayıcısı için

ATCH INT_X, 22

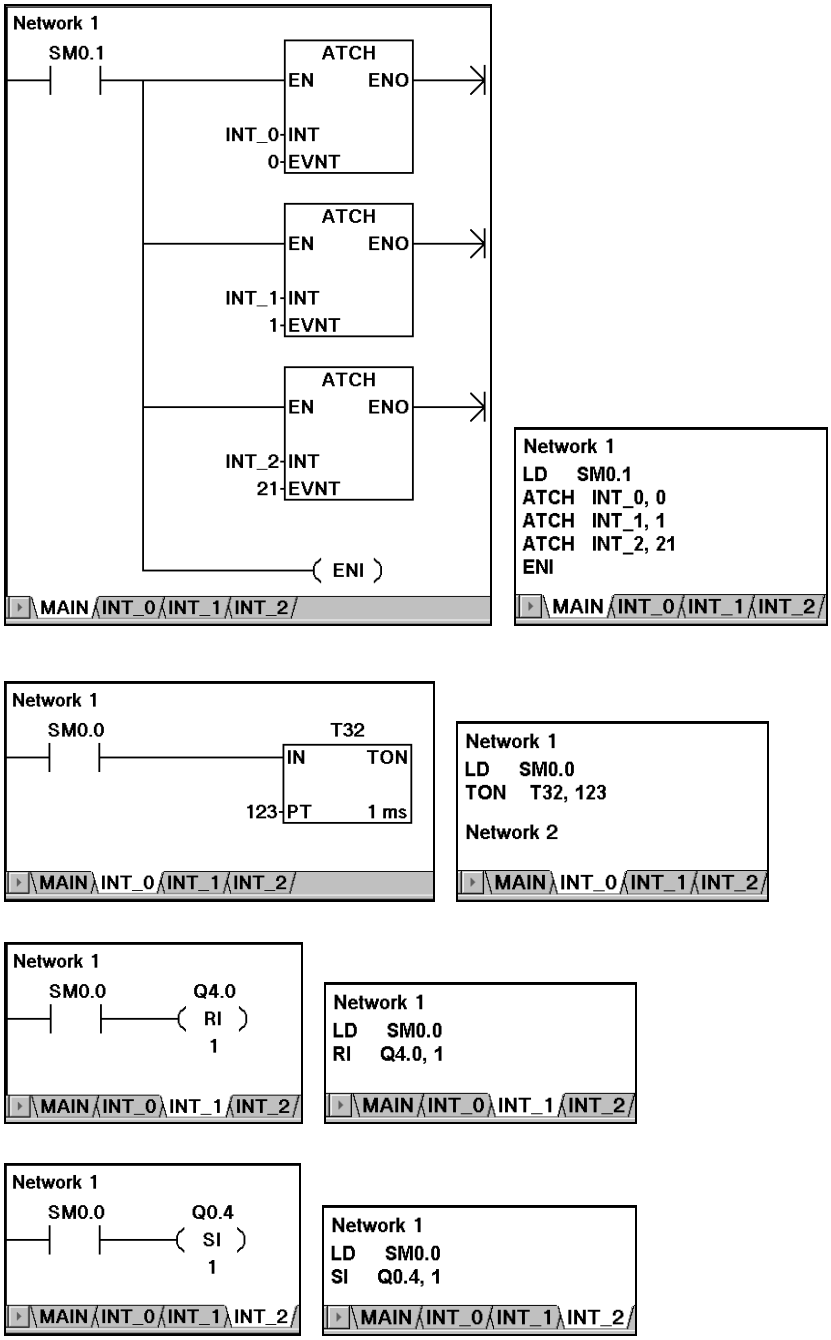
komutları ile belirtilir. Bu kesme türü ile süreye çok duyarlı kumanda olayları gerçeklenebilir. Herhangi bir olaydan belirli bir süre sonra bir kumanda işaretinin üretilmesi için olaya bağlı kesmeli çalışma ile zamanlayıcı kaynaklı kesmeli çalışma birlikte kullanılır. Bu şekilde daha duyarlı gecikmeler elde edilmesi sağlanır.

Örnek 7.3 PLC I0.0 girişi 1 yapıldığı andan 163ms sonra PLC Q0.4 çıkışını 1'e kuran, I0.1 girişi 1 yapıldığı anda Q0.4 çıkışını sıfırlayan bir uygulamaya ilişkin programı yazınız.

Bu özelikleri sağlayan bir program örneği Şekil 7.3'te gösterilmiştir.

Bu programda I0.0 girişine ilişkin 0 olay sayılı kesme INT_0, T32 zamanlayıcısına ilişkin 21 olay sayılı kesme INT_2, I0.1 girişine ilişkin 1 olay sayılı kesme INT_1 altprogramları ile ilişkilendirilir ve kesmeli çalışmaya izin verilir.

Bu programa göre I0.0=1 yapıldığında INT_0 altprogramı işletilir ve T32 zamanlayıcına ilişkin süre başlar, 123ms sonra süre dolar ve 21 olay sayılı kesme işareti oluşur, 21 sayılı olayla ilgili INT_2 kesme altprogramı işletilir ve Q0.4 çıkışı anında 1'e kurulur. I0.0=0 yapıldığında 1 sayılı kesme olayına ilişkin INT_1 altprogramı yürütüldüğünden anında Q0.4=0 yapılır.



ŞEKİL 7.3 Zamanlayıcı kaynaklı kesmeli çalışmaya ilişkin program örneği

7.6 HIZLI SAYICI KAYNAKLI KESMELİ ÇALIŞMA

Hızlı sayıcılar, PLC ' nin olağan program tarama çevrimi süresine göre çok daha kısa sürelerde değişen hızlı işaretlerin değerlendirilmesi ve bu işaretlere ilişkin kumanda işaretlerinin gerçeklenmesinde kullanılır.

Simatic S7-200 CPU 221 ve CPU 222 işlemcili PLC'lerde dört hızlı sayıcı (HSC0, HSC3, HSC4 ve HSC5), CPU 224 ve CPU 226 işlemcili PLC'lerde altı adet hızlı sayıcı (HSC0, HSC1, HSC2, HSC3, HSC4 ve HSC5) bulunur. Hızlı sayıcılar ile sıklığı 30 kHz'e kadar olan işaretler sayılabilir. Olağan program tarama çevrimi içinde hızlı sayıcıların içeriği 32 bitlik HC0, HC1, HC2, HC3, HC4 ve HC5 yazmaçlarından okunabilir ve değerlendirilebilir. Ancak, hızlı sayıcı içeriklerinin bu şekilde değerlendirilmesi, istenen duyarlıkta sonuçların elde edilmesi için yeterli değildir. İşlem sonucu program tarama çevrim süresi ve komutun işlendiği ana bağlı olarak değişir. Bu sakıncanın önlenebilmesi için hızlı sayıcı kaynaklı kesmeli çalışma olanağından yararlanılır.

7.6.1 HIZLI SAYICININ PROGRAMLANMASI

Bir hızlı sayıcı,

- Sayıcı ve çalışma biçiminin tanımlanması,
- Sayıcı ayarlarının denetim yazmacına yazılması,
- Sayıcı başlangıç değerinin ilgili yazmaca yazılması,
- Sayıcı karşılaştırma değerinin ilgili yazmaca yazılması,
- Kesmeli çalışma ve ilgili kesme altprogramlarının tanımlanması,
- Tüm ayar değerlerinin sayıcıya yüklenmesi ve sayıcının işler duruma getirilmesi

adımlarından oluşur.

Sayıcı ve çalışma biçiminin tanımlanması:

Sayıcı ve çalışma biçiminin tanımlanması

HDEF HSC, MODE

komutu ile yapılır.

Burada HSC: 0, 1, 2, 3, 4, 5 tamsayı değerlerini alır ve kullanılan sayıcıyı belirtir. HSC0 hızlı sayıcısı için 0, HSC1 hızlı sayıcısı için 1, HSC2 hızlı sayıcısı için 2, ..., HSC5 hızlı sayıcısı için 5 sayıları kullanılır.

MODE çalışma biçimini belirtir. Bu tanımlama için 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11 tamsayı değerleri kullanılır. Her tamsayı ayrı bir çalışma biçimini belirtir.

0 sayılı çalışma biçimi: Sayıcının, bir sayma girişi ile ileri-geri sayıcı olarak kullanıldığı bir çalışma biçimini tanımlar. Sayma yönü, ilgili denetim yazmacındaki yön denetim biti ile yapılır.

1 sayılı çalışma biçimi: Sayıcının, bir sayma ve bir silme girişi ile ileri-geri sayıcı olarak kullanıldığı bir çalışma biçimidir. Sayma yönü, ilgili denetim yazmacındaki yön denetim biti ile yapılır.

2 sayılı çalışma biçimi: Sayma, silme ve başlatma girişleri olan ileri-geri sayıcı türünü tanımlar. Sayma yönü, ilgili denetim yazmacındaki yön denetim biti ile yapılır.

3 sayılı çalışma biçimi: Sayma ve yön girişleri olan bir ileri-geri sayıcı türünü tanımlar. Sayma yönü, yön girişi ile denetlenir.

4 sayılı çalışma biçimi: Sayma, yön ve silme girişleri olan bir ileri-geri sayıcı türünü tanımlar. Sayma yönü, yön girişi ile denetlenir.

5 sayılı çalışma biçimi: Sayma, yön, silme ve başlatma girişleri olan bir ileri-geri sayıcı türünü tanımlar. Sayma yönü, yön girişi ile denetlenir.

6 sayılı çalışma biçimi: İleri sayma ve geri sayma girişleri olan bir ileri-geri sayıcı türünü tanımlar.

7 sayılı çalışma biçimi: İleri sayma, geri sayma ve silme girişleri olan bir ileri-geri sayıcı türünü tanımlar.

8 sayılı çalışma biçimi: İleri sayma, geri sayma, silme ve başlatma girişleri olan bir ileri-geri sayıcı türünü tanımlar.

9 sayılı çalışma biçimi: Enkoder A ve B işaretlerini sayan ve işaretlerin sırasına bağlı olarak ileri ve geri sayan sayıcı türünü tanımlar.

10 sayılı çalışma biçimi: Enkoder A ve B işaretleri ve bir de silme girişi olan bir sayıcı türünü tanımlar.

11 sayılı çalışma biçimi: Enkoder A ve B işaretleri, silme ve başlatma girişleri olan bir sayıcı türünü tanımlar.

Bir sayıcıda başlatma girişi etkin durumda iken silme girişi etkin yapıldığında sayıcı içeriği sıfırlanır. Başlatma girişi etkin yapıldığında saymaya izin verilir. Başlatma girişi etkin durumdan çıkartıldığında sayma işlevi sona erer ancak sayıcı içeriği korunur. Silme girişi, başlatma girişi etkin değilse etkisizdir. Ancak, silme girişi etkin iken başlatma girişi etkin yapılırsa sayıcı içeriği sıfırlanır. Başlatma ve silme girişlerinin etkin olma durumları düşük veya yüksek işaret düzeylerinde olabilir. Bu

girişlerin etkin olma durumları, ilgili denetim yazmacının (Çizelge 7.1) özel bellek bitleri 0 veya 1 yapılarak belirlenir.

Çizelge 7.1

HSC0	HSC1	HSC2	HSC4	AÇIKLAMA
SM37.0	SM47.0	SM57.0	SM147.0	Silme girişinin etkin düzeyi: yüksek=0, düşük=1
-	SM47.1	SM57.1	-	Başlatma girişinin etkin düzeyi: yüksek=0, düşük=1
SM37.2	SM47.2	SM57.2	SM147.2	Enkoder sayıcı hız seçimi 4x =0, 1x =1

Yukarıda açıklanan çalışma biçimleri tüm hızlı sayıcılar için geçerli değildir. Hızlı sayıcılar için geçerli olan çalışma biçimleri ve ilgili giriş adresleri Çizelge 7.2'de verilmiştir.

Çizelge 7.2

Hızlı sayıcı	Sayma girişi	Yön girişi	Silme girişi	Başlatma girişi	Çalışma biçimi
HSCO	I0.0	I0.1	I0.2		0,1,3,4,6,7,9,10
HSC1	I0.6	I0.7	I1.0	I1.1	Tümü
HSC2	I1.2	I1.3	I1.4	I1.5	Tümü
HSC3	I0.1				0
HSC4	I0.3	I0.4	I0.5		0,1,3,4,6,7,9,10
HSC5	I0.4				0

Sayıcı ayarlarının denetim yazmacına yazılması:

Sayıcı ayarları, sayıcıya ilişkin denetim yazmaçlarına Çizelge 7.3'te anlamı verilen değerler yazılarak yapılır. Bunun için MOVB komutu kullanılır. Örneğin

MOVB 2#11111000, SMB137

komutu ile HSC3 hızlı sayıcısına tüm ayar değerlerinin yazılması ve güncelleştirilmesi izni verilir ve sayıcının etkin duruma getirilmesi sağlanır.

Çizelge 7.3

HSC0	HSC1	HSC2	HSC3	HSC4	HSC5	Açıklama
SM37.3	SM47.3	SM57.3	SM137.3	SM147.3	SM157.3	Sayma yönü: ileri=1, geri=0
SM37.4	SM47.4	SM57.4	SM137.4	SM147.4	SM157.4	Sayma yönünü güncelleştir = 1
SM37.5	SM47.5	SM57.5	SM137.5	SM147.5	SM157.5	Karşılaştırma değerini güncelleştir =1
SM37.6	SM47.6	SM57.6	SM137.6	SM147.6	SM157.6	Sayıcı içeriğini güncelleştir =1
SM37.7	SM47.7	SM57.7	SM137.7	SM147.7	SM157.7	Hızlı sayıcı çalışma durumu: Etkin=1

Sayıcı başlangıç ve karşılaştırma değerlerinin ilgili yazmaca yazılması:

Sayıcının çalışmaya başladığı andaki içeriği ve karşılaştırma değerinin belirtilmesi için hızlı sayıcı ile ilgili Çizelge 7.4'te verilen 32 bitlik yazmaçlar kullanılır. Bu işlem MOVD komutu ile yapılır. Örneğin,

MOVD 0, SMD48

MOVD 123, SMD52

komutları ile HSC1 hızlı sayıcısının başlangıç değeri 0, karşılaştırma değeri 123 olarak tanımlanmış olur.

Çizelge 7.4

HSC0	HSC1	HSC2	HSC3	HSC4	HSC5	Açıklama
SMD38	SMD48	SMD58	SMD138	SMD148	SMD158	Başlangıç değeri
SMD42	SMD52	SMD62	SMD142	SMD152	SMD162	Karşılaştırma değeri

Kesmeli çalışma ve ilgili kesme altprogramlarının belirtilmesi :

CPU 221 ve CPU 222 işlemcilerinde hızlı sayıcı kaynaklı 8 kesme olayı, CPU 224 ve CPU 226 işlemcilerinde ise 14 kesme olayı gerçeklenebilir. Kesme olayları; sayıcı içeriği karşılaştırma değerine eşit olduğunda, sayma yönü değiştiğinde ve sayıcı silme girişi etkin yapıldığında oluşur. Kesme olayı ile ilgili altprogramı belirtmek için ATCH komutu kullanılır.

Örneğin,

ATCH INT_0, 12

komutu ile HSC0 hızlı sayıcısının içeriği karşılaştırma değerine eşit olduğunda INT_0 kesme altprogramının işletileceği,

ATCH INT_1, 14

komutu ile HSC1 hızlı sayıcısının sayma yönü değiştiğinde INT_1 kesme altprogramının işletileceği tanımlanmış olur. Tüm hızlı sayıcılara ilişkin kesme olayları Çizelge 7.5'te verilmiştir.

Çizelge 7.5

Hızlı sayıcı	Olay Numarası	Olay
HSC0	12	Sayıcı içeriği karşılaştırma değerine erişti.
HSC0	27	Sayma yönü değişti.
HSC0	28	Sayıcı silme girişi etkin yapıldı.
HSC1	13	Sayıcı içeriği karşılaştırma değerine erişti.
HSC1	14	Sayma yönü değişti.
HSC1	15	Sayıcı silme girişi etkin yapıldı
HSC2	16	Sayıcı içeriği karşılaştırma değerine erişti.
HSC2	17	Sayma yönü değişti.
HSC2	18	Sayıcı silme girişi etkin yapıldı.
HSC4	29	Sayıcı içeriği karşılaştırma değerine erişti.
HSC4	30	Sayma yönü değişti.
HSC4	31	Sayıcı silme girişi etkin yapıldı.
HSC3	32	Sayıcı içeriği karşılaştırma değerine erişti.
HSC5	33	Sayıcı içeriği karşılaştırma değerine erişti.

Tüm ayar değerlerinin sayıcıya yüklenmesi ve sayıcının işler duruma getirilmesi:

Hızlı sayıcının denetim yazmaçlarına yazılan değere göre çalışması ve işler duruma getirilmesi için

HSC N

komutu kullanılır. Burada N sayıcı numarası olup HSC0 için 0, HSC1 için 1, HSC2 için 2 , ..., HSC5 için 5 tamsayısı değerleri geçerlidir.

Bir sayıcı işler duruma alındıktan sonra sayıcının durumu özel bir yazmaca kaydedilir. Bu yazmaçta sayma yönü, sayıcı içeriği ile karşılaştırma değeri arasındaki büyüklük ilişkisi bilgileri tutulur. Çizelge 7.6'da tüm hızlı sayıcılara ilişkin durum yazmacı bitleri ve anlamları verilmiştir.

Çizelge 7.6

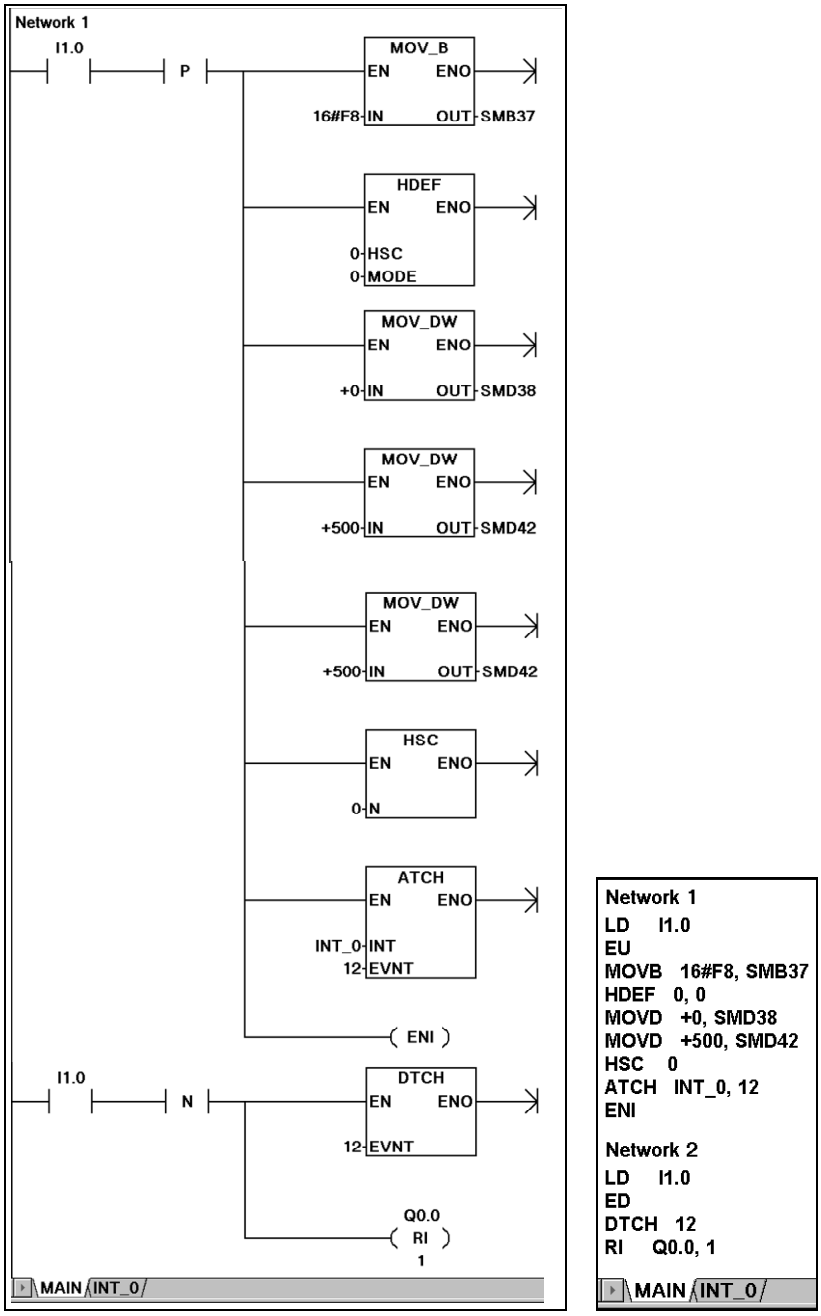
HSC0	HSC1	HSC2	HSC3	HSC4	HSC5	Açıklama
SM36.5	SM46.5	SM56.5	SM136.5	SM146.5	SM156.5	Sayma yönü ileri=1, geri=0
SM36.6	SM46.6	SM56.6	SM136.6	SM146.6	SM156.6	Sayıcı içeriği karşılaştırma değerine eşit=1
SM36.7	SM46.7	SM56.7	SM136.7	SM146.7	SM156.7	Sayıcı içeriği karşılaştırma değerinden büyük=1

Örnek 7.4 I1.0=1 yapıldığında, bir kaynaktan alınan yüksek hızlı işaretleri sayan ve her 500 darbede Q0.0 çıkışının durumunu değiştiren, I1.0=0 yapıldığında sayıcı içeriği ve Q0.0 çıkışını sıfırlayan bir program yazınız.

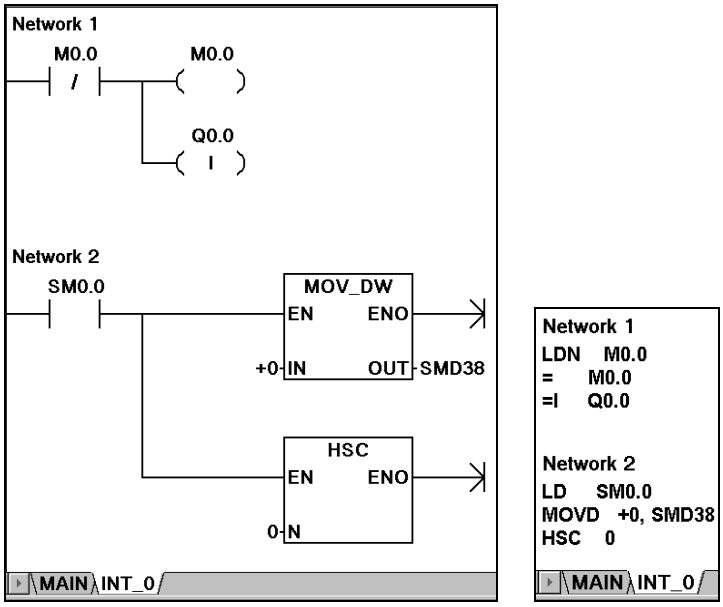
Bu uygulama, yalnız sayma girişi işlevi olan 0 sayılı çalışma biçimi ile gerçeklenebilir. Çizelge 7.2 incelendiğinde, tüm sayıcıların bu çalışma özelliğine sahip olduğu ve bu amaçla kullanılabileceği görülür.

Bu uygulamanın HSC0 hızlı sayıcısı ile gerçeklenmesi durumunda, yüksek hızlı işaret girişi için I0.0 girişi kullanılır ve Şekil 7.4'teki program yazılabilir.

Eğer aynı özellikleri sağlayan bir program HSC3 hızlı sayıcısı ile gerçeklenirse Şekil 7.5'te gösterilen program elde edilir.



Şekil 7.4 HSC0 hızlı sayıcısı ile 0 sayılı çalışma biçimine ilişkin program örneği (1/2)



Şekil 7.4 HSC0 hızlı sayıcısı ile 0 sayılı çalışma biçimine ilişkin program örneği (2/2)

Şekil 7.4'teki ana programdaki (Network 1)

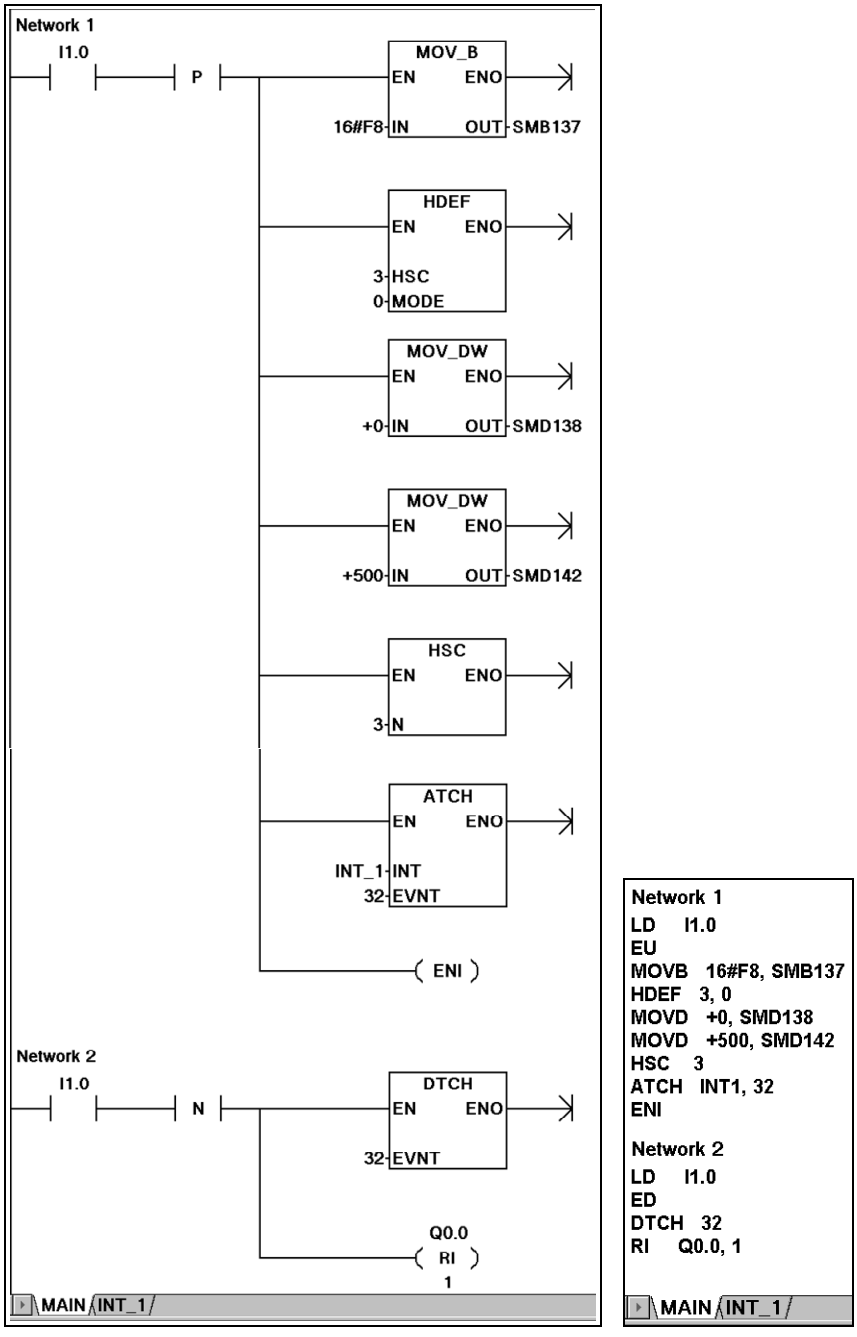
MOVB 16#F8, SMB37

komutu ile HSC0'a ilişkin SMB37 denetim yazmacına (16#F8=2#11111000: SM37.7=1 hızlı sayıcı etkin, SM37.6=1 sayıcı içeriğini güncelleştir, SM37.5=1 karşılaştırma değerini güncelleştir, SM37.4=1 sayma yönünü güncelleştir, SM37.3=1 sayma yönü ileri) tanımları yapılır.

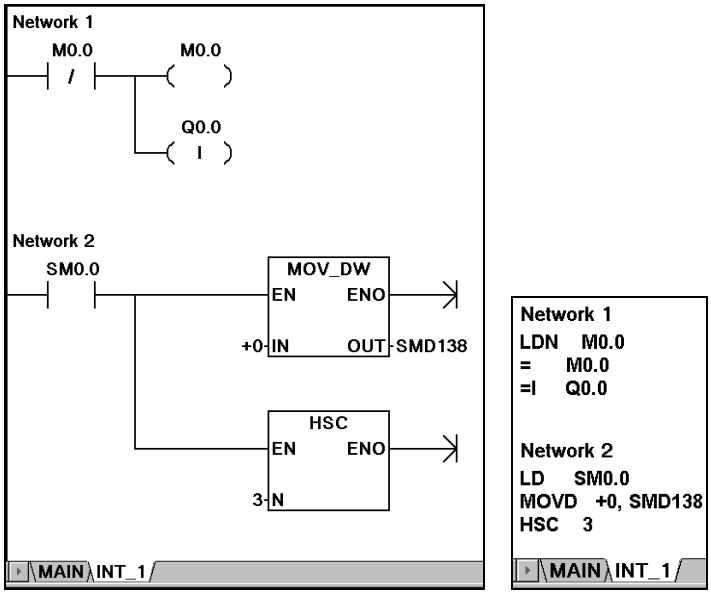
HDEF 0,0 komutu ile HSC0 hızlı sayıcısının 0 sayılı çalışma biçiminde işletileceği,

MOVD 0, SMD38 komutu ile sayıcı içeriğinin başlangıçta 0,

MOVD 500, SMD42 komutu ile karşılaştırma değerinin 500 olacağı belirtilir. HSC 0 komutu ile tanımlanan değerlere göre sayıcı işler duruma getirilir. Sayıcı işler duruma alındığında HSC0 sayıcısı I0.0 girişine gelen darbeleri sayar. Sayıcı içeriği 500 olduğunda 12 sayılı kesme işareti üretilir ve bu kesme olayı ile ilişkili INT_0 kesme altprogramı yürütülür. INT_0 altprogramının her işletilmesinde M0.0 bitinin değeri değiştirilir ve ivedi atama komutu ile anında Q0.0 çıkışına gönderilir.



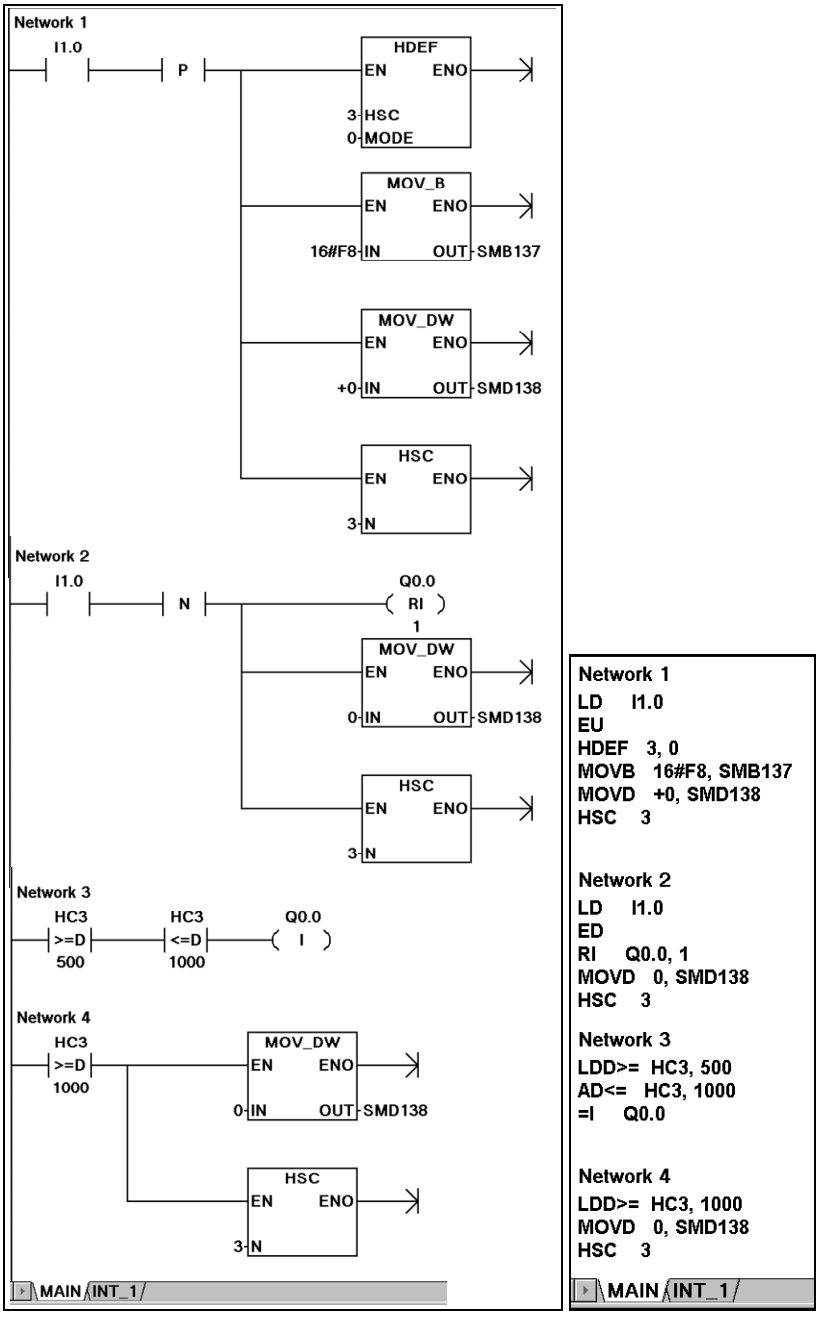
Şekil 7. 5 HSC3 hızlı sayıcısı ile 0 sayılı çalışma biçimine ilişkin program örneği (1/2)



Şekil 7. 5 HSC3 hızlı sayıcısı ile 0 sayılı çalışma biçimine ilişkin program örneği (2/2)

Bu tür bir uygulama, kesmeli çalışma yapılmadan, Şekil 7.6'da verilen program parçası kullanılarak da gerçeklenebilir. Buradaki programda her tarama çevriminde yüksek hızlı sayıcının içeriği değerlendirilmekte, eğer sayıcı içeriği 500 ile 10000 arasında veya eşit ise Q0.0=1 değilse Q0.0=0 yapılmaktadır. Sayıcı içeriği 1000'e eriştiğinde veya büyük olduğunda sayıcı içeriğine ilişkin SMD138 özel bellek alanına 0 yazılarak sayıcı içeriği sıfırlanmaktadır.

Bu programda, Q0.0 çıkışından elde edilen işaretler PLC tarama çevriminin süresine ve komutun işlendiği ana bağlı olarak değişir ve tarama çevrimi süresinin büyüklüğüne bağlı olarak, zamanlama hatası büyük değerler alabilir.



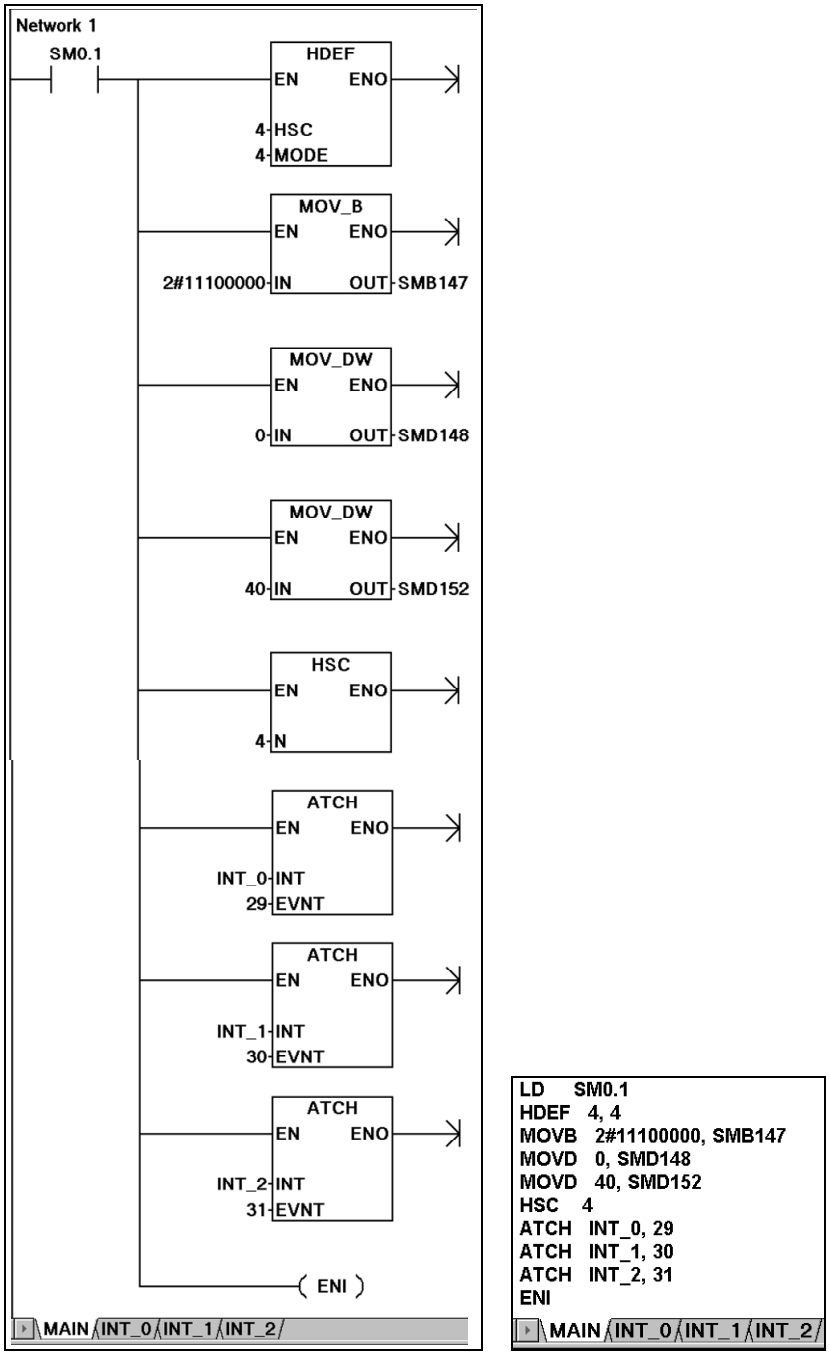
Şekil 7.6 HSC3 hızlı sayıcısı ile kesmesiz çalışmaya ilişkin program örneği

Örnek 7.5 HSC4 hızlı sayıcısının 4 sayılı çalışma biçimine göre işletilmesini sağlayan bir program yazınız.

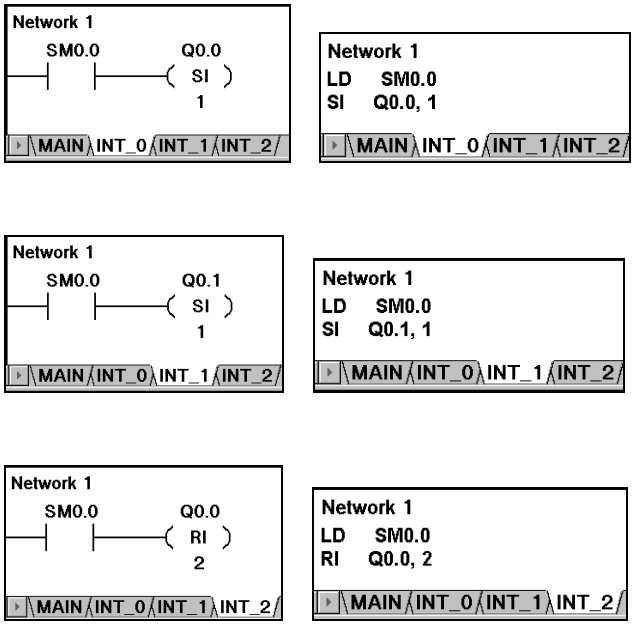
HSC4 ile gerçeklenen bu çalışma biçiminde I0.3 sayma girişi, I0.4 yön girişi ve I0.5 silme girişi olarak kullanılır. Bu programa göre I0.4=1 iken, I0.3 girişine gelen darbeler sayılır, sayıcı içeriği 40 olduğunda 29 olay sayılı kesme işareti üretilerek ilgili kesme altprogramı INT_0 yürütülür ve Q0.0 anında 1'e kurulur.

I0.4=0 yapıldığında yön bilgisi değiştirilir ve ilk I0.3 girişine gelen darbe işareti ile birlikte 30 olay sayılı kesme işareti üretilir, ilgili kesme altprogramı INT_1 yürütülür ve anında Q0.1 çıkışı 1'e kurulur.

I0.5=1 yapıldığında sayıcı içeriği silinir, 31 olay sayılı kesme işareti üretilir, ilgili kesme altprogramı INT_2 yürütülür ve anında Q0.0 ve Q0.1 çıkışları sıfırlanır.



Şekil 7.7 HSC4 hızlı sayıcısı ile 4 sayılı çalışma biçimine ilişkin program örneği (1/2)

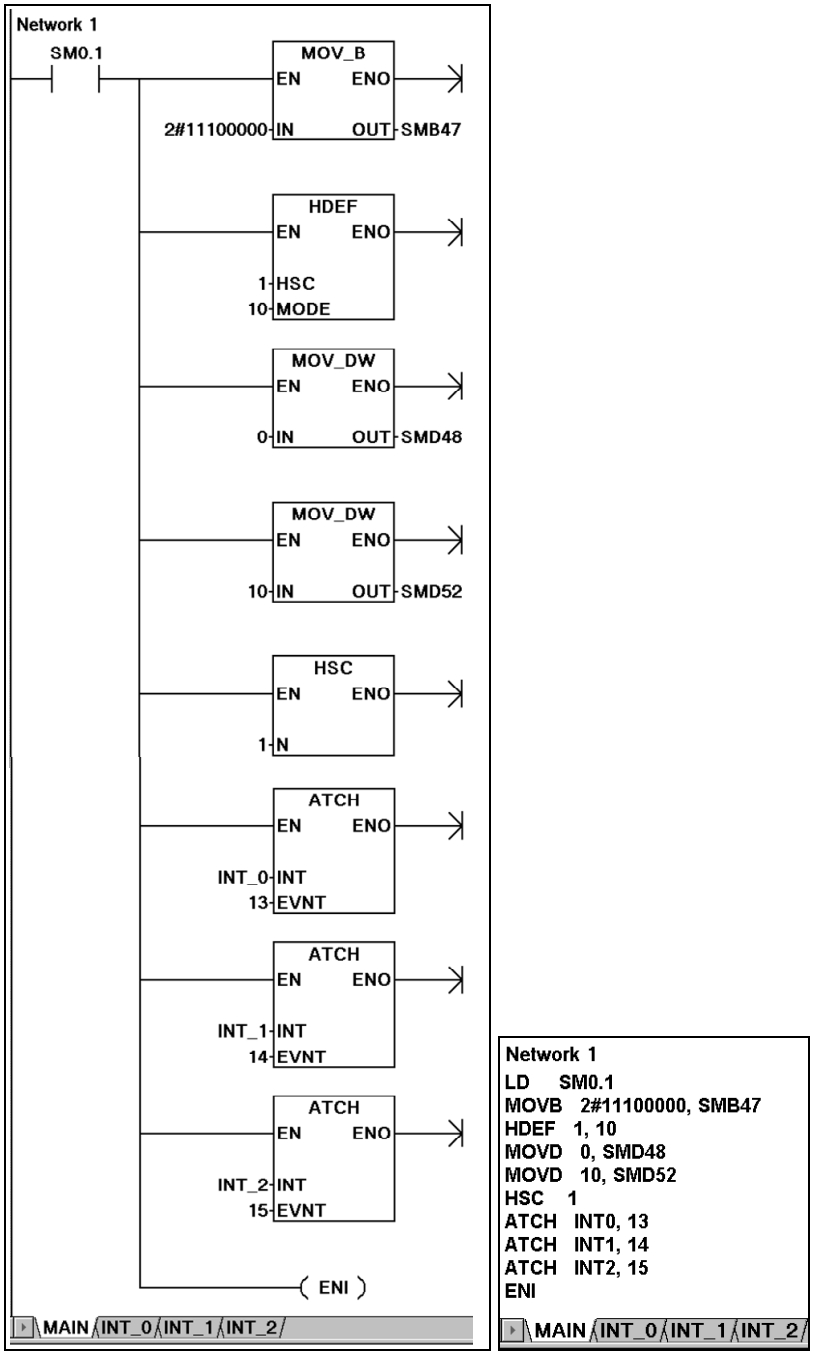


Şekil 7.7 HSC4 hızlı sayıcısı ile 4 sayılı çalışma biçimine ilişkin program örneği (2/2)

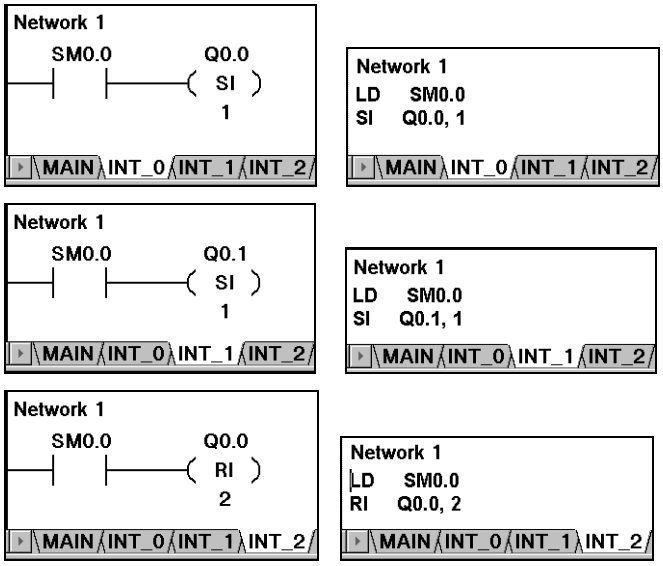
Örnek 7.6 HSC1 hızlı sayıcısının 10 sayılı çalışma biçimine göre işletilmesini sağlayan bir program yazınız.

Bu çalışma biçiminde enkoder A ve B uçlarından alınan işaretlerin değişim sırasına göre sayıcı içeriği değiştirilir. Eğer önce A sonra B işareti 0'dan 1'e değişirse sayıcı içeriği artar, tersi olursa sayıcı içeriği azalır. Bu iş için HSC1 hızlı sayıcısının kullanılması durumunda A işareti I0.6, B işareti I0.7 ve silme girişi I1.0 girişlerine bağlanır.

Bu uygulamanın çözümüne ilişkin ana program ve kesme altprogramları Şekil 7.8'de verilmiştir.



Şekil 7.8 HSC1 hızlı sayıcısının 10 sayılı çalışma biçimine ilişkin program örneği (1/2)



Şekil 7.8 HSC1 hızlı sayıcısının 10 sayılı çalışma biçimine ilişkin program örneği (2/2)

Şekil 7.8'deki programa göre I0.6 (A) ve I0.7 (B) girişlerine gelen işaretlerin art arda 0'dan 1'e değişimi 10 olduğunda 13 olay sayılı kesme işareti üretilir ve ilgili kesme altprogramı INT_0 işletilerek Q0.0 çıkışı anında 1'e kurulur. Eğer önce B sonra A işareti değişmeye başlarsa (dönüş yönünün değişmesi durumu) 14 olay sayılı kesme işareti üretilir ve ilgili kesme altprogramı INT_1 işletilerek Q0.1 çıkışı anında 1'e kurulur. I1.0 girişi 1 yapılısa sayıcı içeriği sıfırlanır, 15 olay sayılı kesme işareti üretilir ve ilgili kesme altprogramı INT_2 işletilerek anında Q0.0 ve Q0.1 çıkışları sıfırlanır.

7.7 YÜKSEK HIZLI ÇIKIŞ İŞARETİ ÜRETİLMESİ

Yüksek hızlı çıkış işaretleri, darbe-boşluk oranı eşit değişken frekanslı kare dalga işaretler (PTO) ve darbe-genişlik oranı değişken sabit frekanslı darbe genişlik modülasyonlu (PWM) işaretler biçiminde elde edilebilir. Bu işaretler Q0.0 ve Q0.1 çıkış noktalarından alınır. Yüksek hızlı çıkış işaretleri üretilmesi ile ilgili durum ve denetim yazmaçları Çizelge 7.7'de verilmiştir.

Çizelge 7.7

Q0.0	Q0.1	Çıkışlar
SM67.0	SM77.0	PTO / PWM periyot süresini güncelleştir=1
SM67.1	SM77.1	PWM darbe süresini güncelleştir=1
SM67.2	SM77.2	PTO darbe sayısını güncelleştir=1
SM67.3	SM77.3	PTO / PWM süre artımı : (1μs = 0, 1ms = 1)
SM67.4	SM77.4	PWM güncelleme yöntemi : (asenkron=0 ,senkron=1)
SM67.5	SM77.5	PTO tek parçalı / çok parçalı işlem : (tekli=0, çoklu= 1)
SM67.6	SM77.6	PTO / PWM modu seçimi : (PTO=0, PWM=1)
SM67.7	SM77.7	PTO / PWM çalışma izni (izin var=1)
SMW68	SMW78	PTO / PWM periyot süresi değeri : { 2, 3, …, 65535}
SMW70	SMW80	PWM darbe genişliği değeri: { 0,1, …, 65535}
SMD72	SMD82	PTO darbe sayısı değeri : { 1,2,…., 4294967295}
SMB166	SMB176	İşletimde olan PTO parçasının numarası
SMW168	SMW178	PTO profil tablosunun başlangıç adresi

Değişken frekanslı kare dalga işaretler, periyot süresi ve darbe sayısı; darbe genişlik modülasyonlu işaretler, periyot süresi ve darbe genişlik değeri belirtilerek tanımlanır. Bu işlem, Çizelge 7.6'da verilen ilgili özel denetim yazmaçlarına uygun değerler yazılarak yapılır.

Yüksek frekanslı işaretlerin Q0.0 çıkış noktasından üretilmesi durumunda çizelgenin birinci sütunundaki yazmaçlar, Q0.1 çıkış noktasından üretilmesi durumunda ise ikinci sütundaki yazmaçlar kullanılır.

7.7.1 YÜKSEK FREKANSLI KARE DALGA ÜRETİLMESİ

Yüksek frekanslı kare dalgaların üretilmesi tek ve çok parçalı olmak üzere iki şekilde yapılır. Tek parçalı işlemde denetim yazmaçlarına yazılan değerlere göre belirli frekans ve darbe sayısında bir darbe katarı üretilir ve eğer tanımlanmışsa darbe sayısının bitiminde bir kesme işareti üretilir. Darbe katarlarının art arda sürekli üretilmesi için bir darbe katarı bitmeden, sonraki darbe katarına ilişkin tanımlamalar yapılır. Çok parçalı işlemde ise belirli frekans eğrilerine göre değişen darbe katarları elde edilebilir.

7.7.1.1 Tek Parçalı İşleme İlişkin Bir Darbe Katarı Üretilmesi

Darbe katarının Q0.0 çıkış noktasından üretilmesine ilişkin veriler için SMB67 denetim yazmacı, Q0.1 noktasından üretilmesine ilişkin veriler için SMB77 denetim yazmacı kullanılır. Darbenin Q0.0 çıkış noktasından üretilmesi için SMB67 denetim yazmacına

SM67.0=1 // periyot değerini güncelleştir,

SM67.2=1 // darbe sayısını güncelleştir,

SM67.3=1 // zaman artımı seçimi 1= ms, 0=μs ,

SM67.5=0 // tek parçalı işlemi seç,

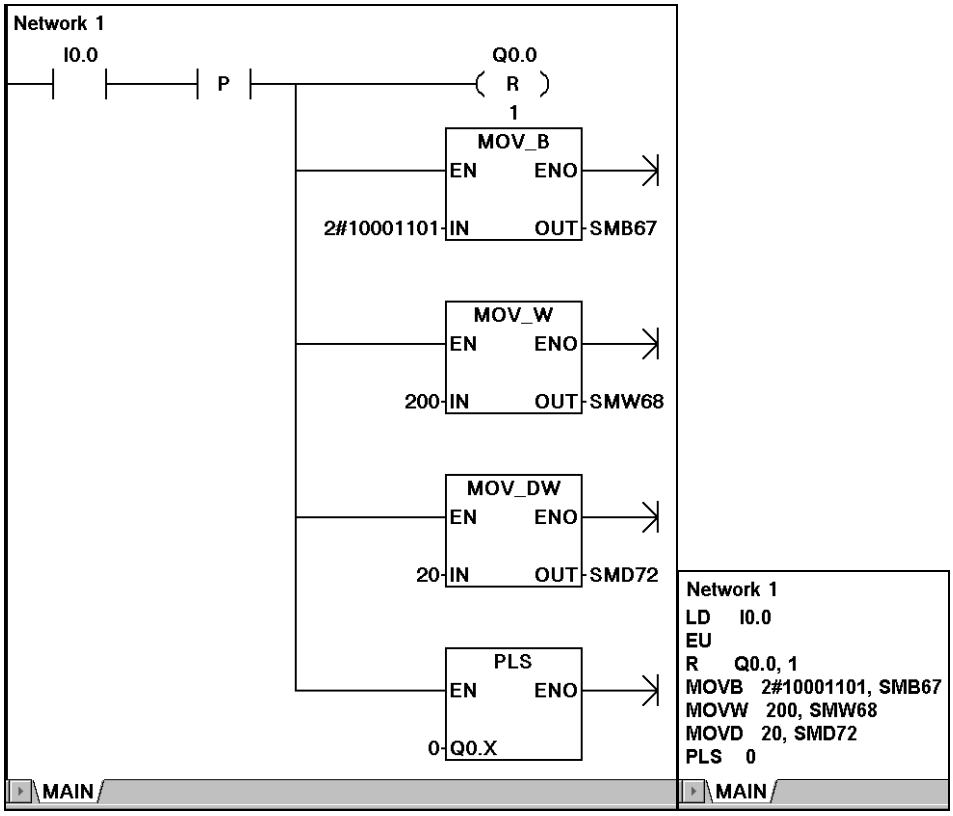
SM67.6=0 // PTO modunu seç,

SM67.7=1 // izin ver.

değerleri yazılır. Darbe işaretinin periyodu SMW68 denetim yazmacına, darbe sayısı ise SMD72 denetim yazmacına yazılır.

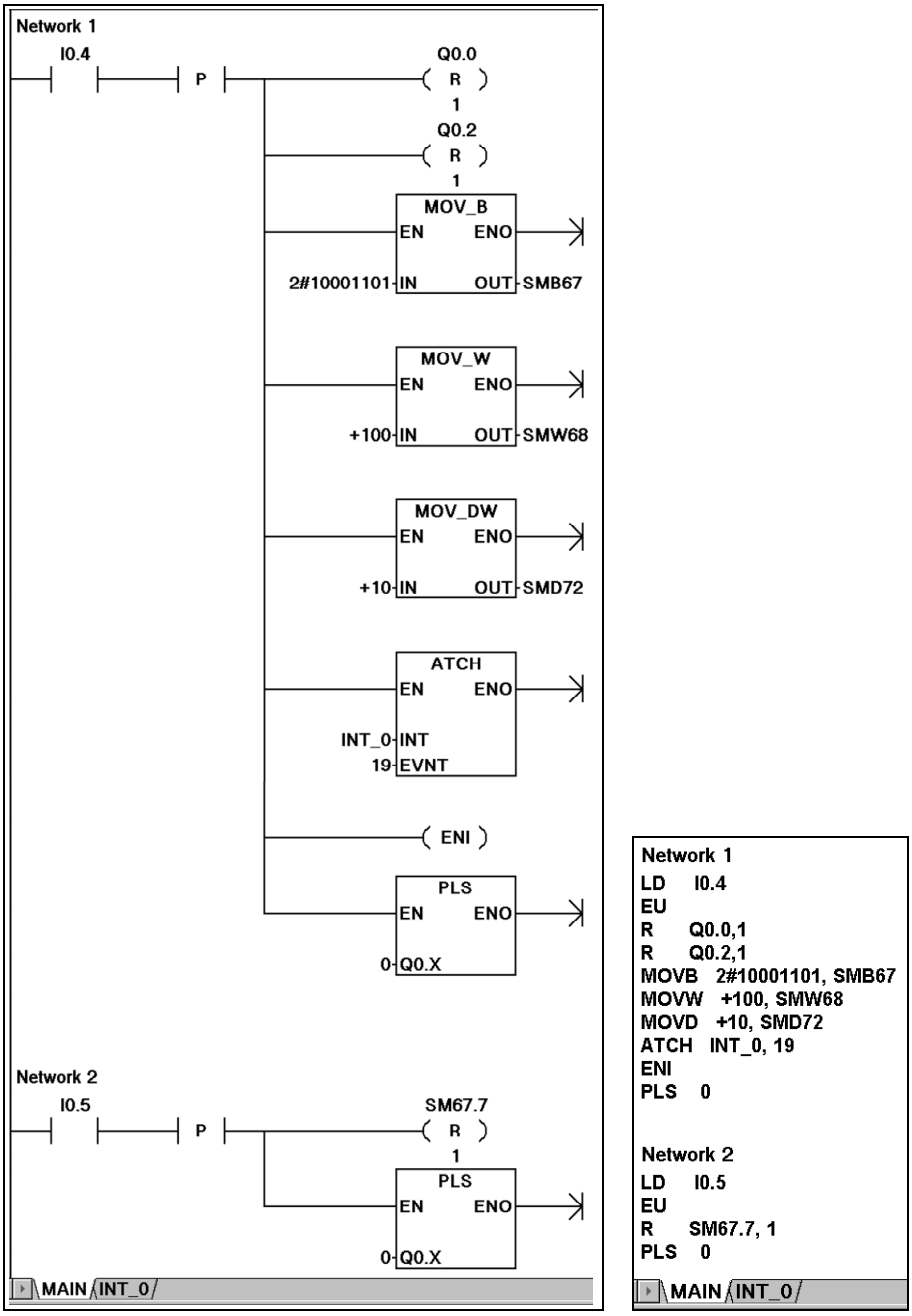
PTO/PWM işaretlerinin üretildiği Q0.0 veya Q0.1 çıkış noktalarına ilişkin ayarlar güncelleştirilmeden önce bu noktaların görüntü belleğindeki değerleri sıfırlanır. Bu işlemlerden sonra PLS 0 komutunun işletilmesi ile darbe üretilmeye başlanır.

Örneğin, PLC I0.0 girişinin 1 yapılması ile birlikte Q0.0 çıkışından periyodu 200ms ve darbe sayısı 20 olan bir darbe katarı üretmek için Şekil 7.9'daki program yazılabilir.

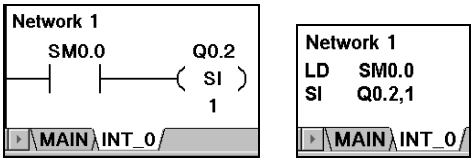


Şekil 7.9 Q0.0 çıkışından periyodu 200ms darbe sayısı 20 olan darbe katarını üretmeye ilişkin program

Örnek 7.7 PLC I0.4 girişine normalde açık olarak bağlı bir düğmeye basıldığı anda Q0.0 çıkışından 10 darbeli 100 ms periyotlu bir darbe katarı üreten ve darbe sayısı tamamlandığı anda Q0.2 çıkışını anında 1'e kuran, PLC I0.5 girişi 1 yapıldığı anda darbe katarının durdurulmasını sağlayan bir program yazınız.



Şekil 7.9 Q0.0 çıkışından 100ms periyodu 10 darbe üreten uygulamaya ilişkin program örneği (1/2)



Şekil 7.9 Q0.0 çıkışından 100ms periyodu 10 darbe üreten uygulamaya ilişkin program örneği (2/2)

Şekil 7.9'da verilen programda

MOVB 2#10001101, SMB67 komutu ile periyot değerini güncelleştir (SM67.0=1), darbe sayısını güncelleştir (SM67.2=1), zaman artım boyutu ms (SM67.3=1), tek parçalı işlemi seç (SM67.5=0)

PTO modu seç (SM67.6=0) ve izin ver (SM67.7=1) koşulları tanımlanır.

MOVW +100, SMW68 komutu ile periyot değerinin 100 ms,

MOVD +10, SMD72 komutu ile darbe sayısının 10,

ATCH 0,19 komutu ile darbe sayısı tamamlandığında oluşan 19 0lay sayılı kesme işareti ile INT_0 kesme alt programının işletileceği, ENI komutu ile kesmeli çalışma durumuna izin verildiği belirtilir.

PLS 0 komutu ile süreç başlar ve tanımlanan biçimde darbe katarları üretilir.

I0.5 işaretinin çıkan kenarında darbe katarının durdurulması için SM67.7=0 yapılır ve PLS 0 komutu işletilir (Network 2).

7.7.1.2 Çok Parçalı İşleme İlişkin Bir Darbe Katarı Üretilmesi

Bu tür çalışmada birden çok parçadan oluşan eğrilere ilişkin değişken frekanslı darbe katarları üretilir. Bunun için her eğri parçasının başlangıç ve son noktalarına ilişkin zaman ve frekans değerlerinden hareketle ilk periyot değeri, periyot artımı ve darbe sayısı belirtilir. Bu değerler başlangıç adresi SMB168 (Q0.0) veya SMB178 (Q0.1) denetim yazmaçlarında belirtilen V bellek alanına sıralı olarak,

VBx parça sayısı,

VW(x+1) ilk periyot değeri,

VW(x+3) periyot artım değeri,

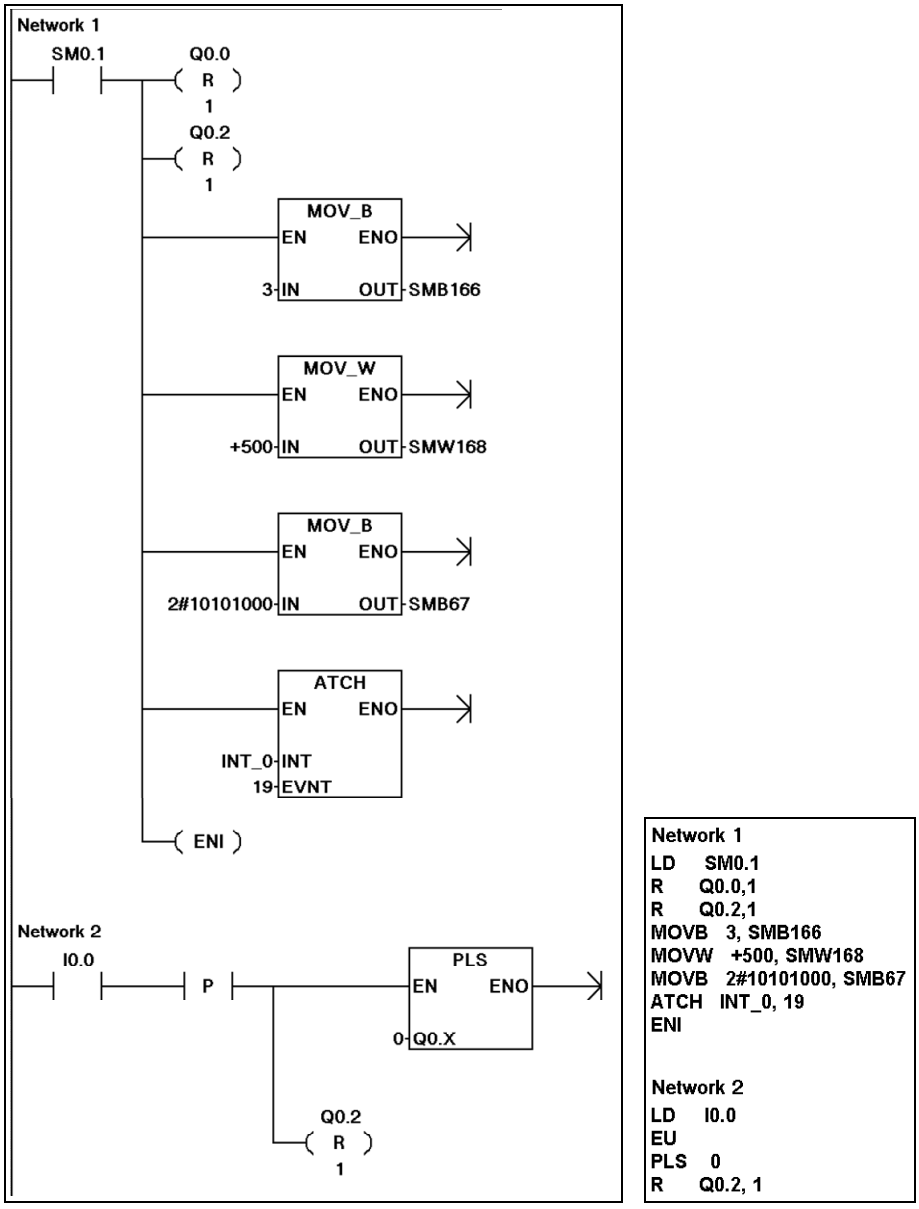
VD(x+5) darbe sayısı,

VW(x+9) ikinci parçanın ilk periyot değeri,

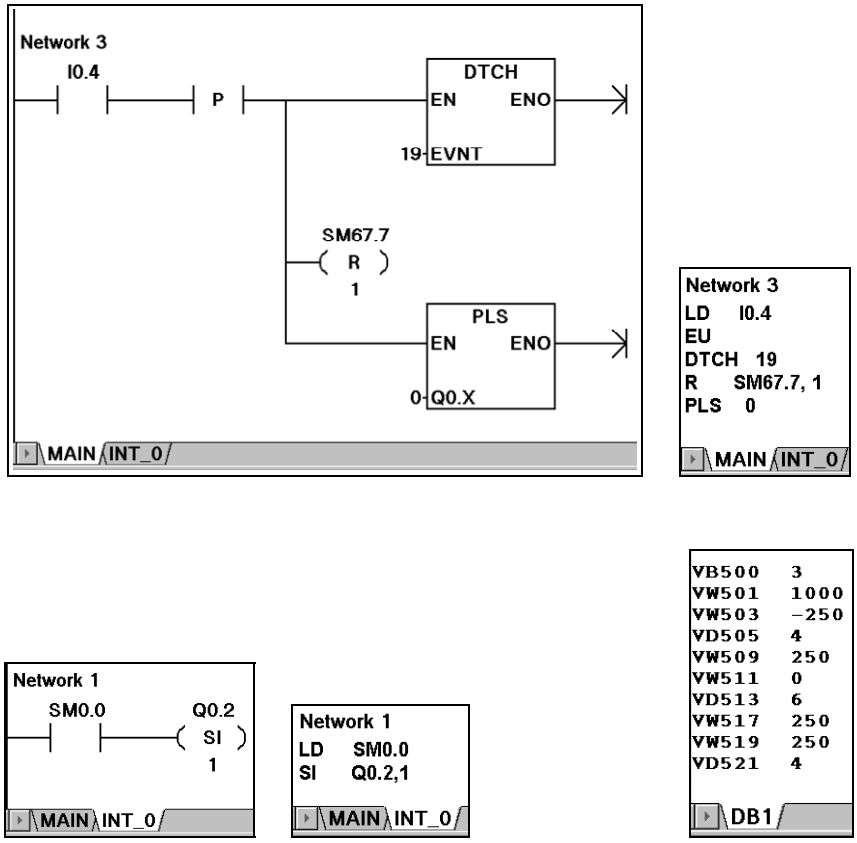
VW(x+11) ikinci parçanın periyot artım değeri,

VD(x+13) ikinci parçanın darbe sayısı biçiminde parça sayısı kadar yazılır.

Şekil 7.10'da 3 parçalı değişken frekanslı darbe katarı üreten bir program örneği verilmiştir.



Şekil 7.10 Çok parçalı darbe katarı üretilmesine ilişkin program örneği (1/2)



Şekil 7.10 Çok parçalı darbe katarı üretilmesine ilişkin program örneği (2/2)

Şekil 7.10'daki programda

MOVB 3, SMB166 komutu ile parça sayısının 3,

MOVW +500, SMW168 komutu ile V tablosu başlangıç adresinin VB500,

MOVB 2#10101000, SMB67 komutu ile süre artım boyutunun ms (SM67.3=1), çok parçalı işlem yapılacağı (SM67.5=1) ve izin verildiği (SM67.7=1),

ATCH INT_0, 19 komutu ile darbe katarı tamamlandığında INT_0 kesme altprogramının işletileceği belirtilir. Bu tanımlamalardan sonra PLS 0 komutu yürütüldüğünde VB500 veri tablosunda belirtilen değerlere göre darbe katarı üretilir. Veri tablosu DATA BLOCK alanına yazılarak veya doğrudan program içinde atama yapılarak oluşturulabilir. Verilen örnekte veri tablosu DATA BLOK alanına yazılmıştır. Bu veri tablosuna göre parça sayısı 3 (VB500), ilk parçanın başlangıç periyodu 1000ms (VW501), artım değeri -250ms (VW503), darbe sayısı 4 (VD505), ikinci parçanın sırayla başlangıç

periyodu, artım değeri ve darbe sayısı 250ms (VW509), 0 (VW511) ve 6 (VD513), üçüncü parçanın ise başlangıç periyodu 250ms (VW517), artım değeri 250ms (VW519) ve darbe sayısı 4 (VD521) olarak tanımlanmıştır.

Bu program göre I0.0 girişi 1 yapıldığında darbe katarı üretilmeye başlanır ve süreç tamamlandığında Q0.2 çıkışı 1'e kurulur. Eğer darbe katarı tamamlanmadan I0.4 girişi 1 yapılırsa darbe üretimi sonlandırılır.

7.7.2 DARBE GENİŞLİK MODÜLASYONLU YÜKSEK HIZLI İŞARET ELDE EDİLMESİ

Darbe genişlik modülasyonlu işaretlerin üretilmesi için ayrıntıları Çizelge 7.7'de verilen SMB67 (Q0.0) ve SMB77 (Q0.1) denetim yazmaçları kullanılır. PWM işaretinin Q0.0 çıkışından üretilmesi için

SM67.0 (periyot süresini güncelleştirme=1)

SM67.1 (darbe süresini güncelleştirme=1

SM67.3 (süre artım boyutu: 1μs = 0, 1ms = 1)

SM67.4 (PWM güncelleme yöntemi asenkron=0, senkron=1)

SM67.6 (PWM modu=1)

SM67.7 (PWM çalışma izni var = 1)

adreslerine belirtilen değerler yazılır.

Darbe süresi SMW68 {2, 3, …, 65535} darbe genişliği değeri SMW70 {0,1, .., 65535}

adreslerine yazılır.

PWM işaretinin Q0.1 çıkışından üretilmesi için

SM77.0 (periyot süresini güncelleştirme=1)

SM77.1 (darbe süresini güncelleştirme=1

SM77.3 (süre artım boyutu: 1μs = 0, 1ms = 1)

SM77.4 (PWM güncelleme yöntemi asenkron=0, senkron=1)

SM77.6 (PWM modu = 1)

SM77.7 (PWM çalışma yetkisi var=1)

adresleri kullanılır.

Darbe periyodu süresi SMW78 {2, 3, …, 65535} darbe genişliği süresi SMW80 {0,1, .., 65535} adreslerine yazılır.

PWM işaret üretme sürecinin başlatılması için

Q0.0 noktası için PLS 0

Q0.1 noktası için PLS 1

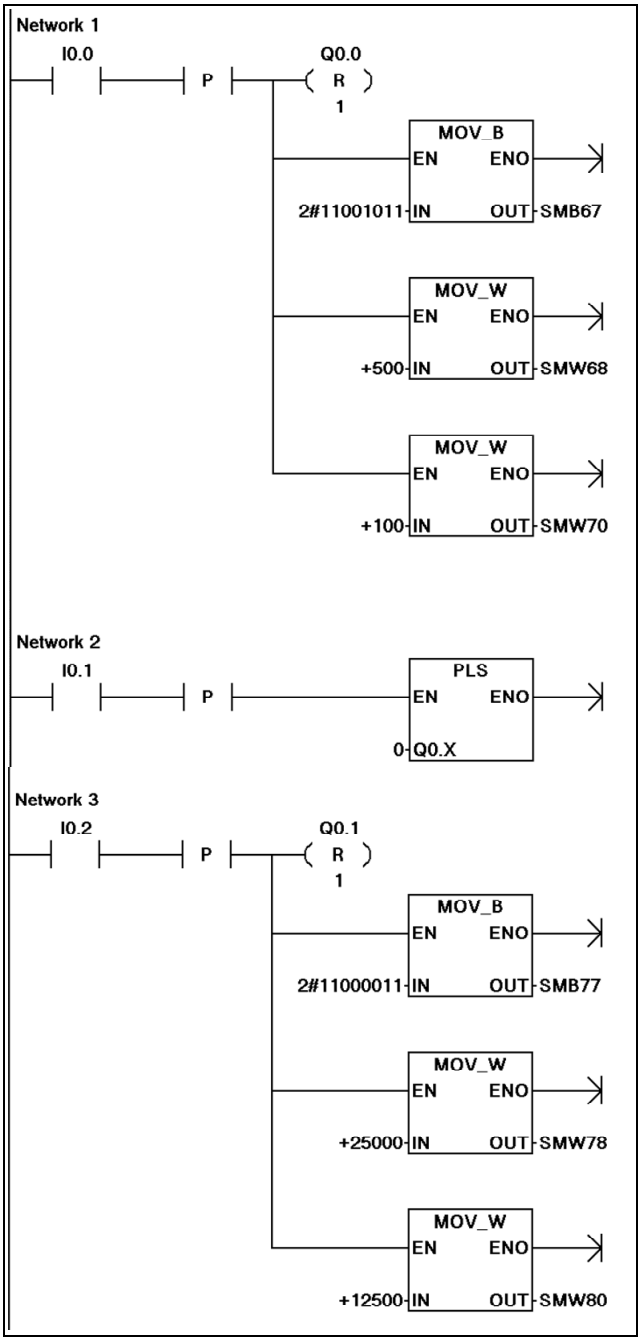
komutu bir kez işletilir.

PWM çalışmada periyot süreleri 10µs ile 65535µs, darbe süreleri ise 2ms ile 65535ms değerleri arasında değişir. Eğer darbe süresi periyot süresinden büyük seçilmişse ilgili çıkış her zaman 1 değerinde, periyot süresi 0 seçilmişse ilgili çıkış her zaman 0 değerinde olur.

Şekil 7.11'de, Q0.0 ve Q0.1 çıkışlarından PWM işaretleri elde edilmesine ilişkin bir program örneği verilmiştir. Bu programa göre,

I0.0 ve I0.1 girişleri 0'dan 1'e değiştirildiğinde Q0.0 çıkışından periyodu 500ms ve darbe süresi 100ms olan PWM işaretleri üretilir. I0.4 girişi 0'dan 1'e değiştirildiğinde PWM işaretinin üretilmesine son verilir.

Eğer I0.2 girişi 0'dan 1'e, daha sonra I0.3 girişi 0'dan 1'e değiştirilirse Q0.1 çıkışından periyodu 25000µs ve darbe süresi 12500µs olan PWM işaretleri üretilir. I0.5 girişi 0'dan 1'e değiştirildiğinde PWM işaretinin üretilmesi durdurulur.

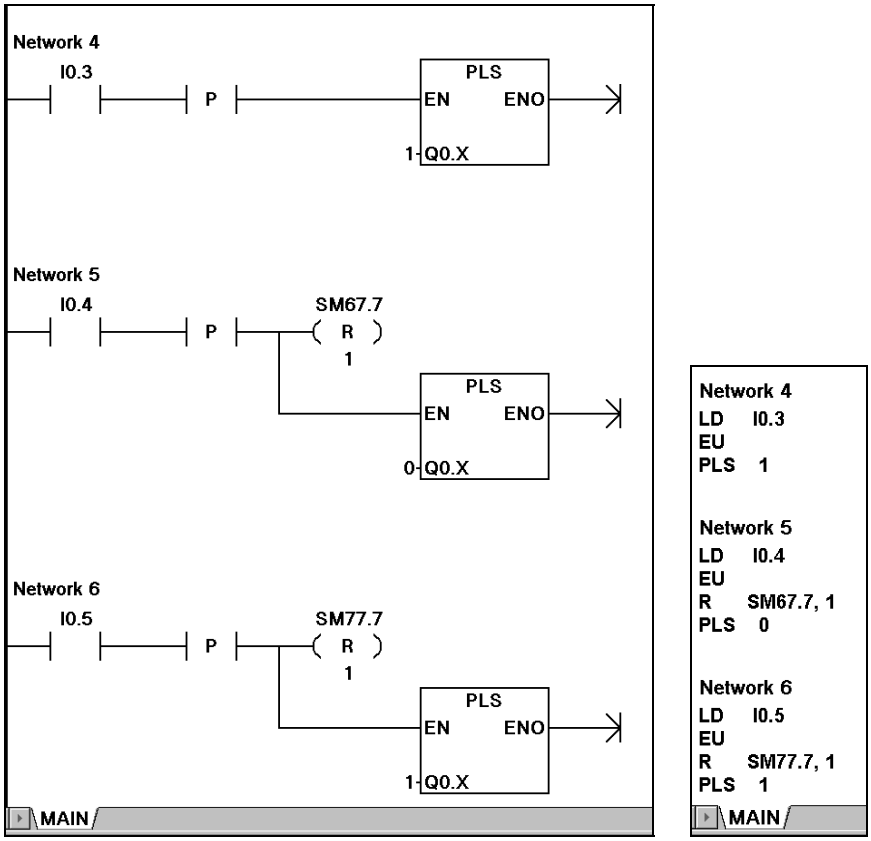



```
Network 1
LD    I0.0
EU
R     Q0.0, 1
MOVB  2#11001011, SMB67
MOVW  +500, SMW68
MOVW  +100, SMW70

Network 2
LD    I0.1
EU
PLS   0

Network 3
LD    I0.2
EU
R     Q0.1, 1
MOVB  2#11000011, SMB77
MOVW  +25000, SMW78
MOVW  +12500, SMW80
```

Şekil 7.11 Q0.0 ve Q0.1 çıkışlarından PWM işaretleri elde edilmesine ilişkin bir program örneği (1/2)



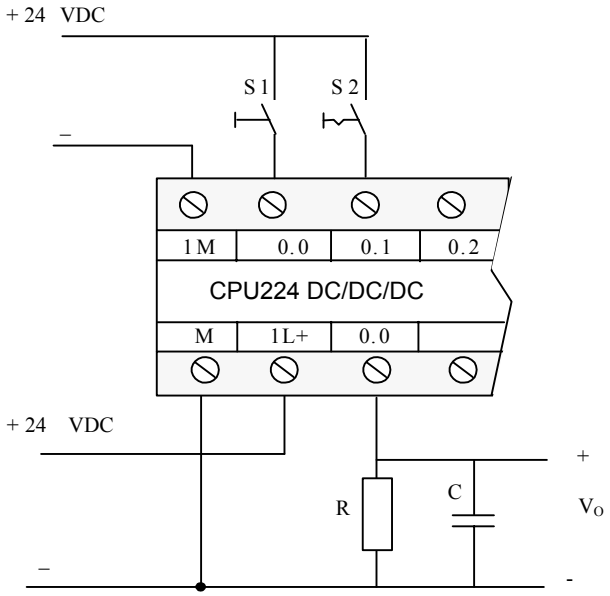
Şekil 7.11 Q0.0 ve Q0.1 çıkışlarından PWM işaretleri üretilmesine ilişkin program örneği (2/2)

Örnek 7.8 Şekil 7.12'deki düzenek kullanılarak PLC PWM çıkış üreteci ile 2 ayrı gerilim düzeyi elde edilecektir. Gerilim düzeyini belirlemek için S2 anahtarı, üreteci başlatmak için S1 kumanda düğmesi kullanılacaktır. Gerilim düzeyleri ve üretecin çalışmasına ilişkin koşullar şu şekilde verilmiştir:

S2 anahtarı kapalı iken (S2=1), S1 düğmesine basıldığında (S1=1) V_0=12V

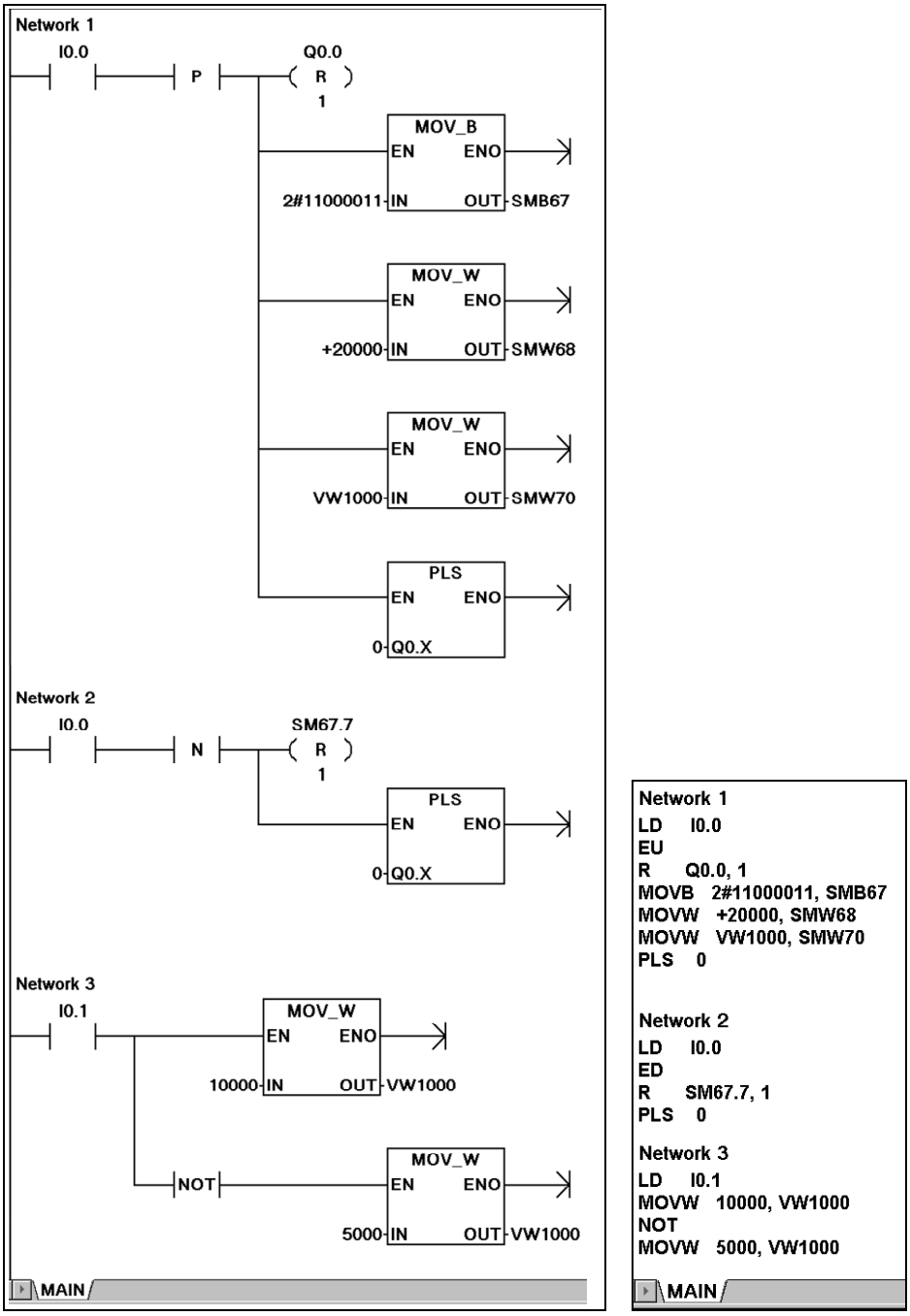
S2 anahtarı açık iken (S2=0), S1 düğmesine basıldığında (S1=1) V_0=6V

S1 anahtarı açıldığında (S1=0) üretecin çalışması durdurulacak ve V_0=0V olması sağlanacaktır. Bu özellikleri sağlayan bir program yazınız.



Şekil 7.12 PLC Q0.0 çıkışından değişken gerilim elde edilmesine ilişkin bağlantı şeması

PWM çıkış işareti periyodunun 20000μs seçilmesi durumunda bu özellikleri sağlayan program Şekil 7.13'teki gibi yazılabilir.



Şekil 7.12 PLC Q0.0 çıkışından değişken gerilim elde edilmesine ilişkin program örneği

Şekil 7.12'de verilen programa göre,

I0.0 girişi 0'dan 1'e değiştirildiğinde; Q0.0 çıkışı sıfırlanır, SMB67 denetim yazmacına 2#11000011 değeri yazılarak

SM67.0 (periyot süresini güncelleştir=1)

SM67.1 (darbe süresini güncelleştir=1

SM67.3 (süre artım boyutunu μs seç= 0)

SM67.4 (PWM güncelleme yöntemini asenkron seç=0)

SM67.6 (PWM modunda çalıştır=1)

SM67.7 (PWM çalışma izni var = 1)

tanımları yapılır.

periyot süresi (SMW68) ve darbe süresi (SMW70) adreslerine yazılır ve Q0.0 çıkış noktasından PWM işareti üretilir. (Network 1).

I0.0=0 yapıldığında, PWM işaretinin üretilmesi durdurulur (Network 2).

I0.1=1 iken VW1000 adresine 10000, I0.1=0 iken VW1000 adresine 5000 değeri yazılır (Network 3).

Giriş geriliminin genliği 24V olduğundan I0.1=1 iken darbe genişlik oranı (10000μs/20000μs)=1/2, çıkış gerilimi V_0=12V,

I0.1=0 iken darbe genişlik oranı (5000μs/20000μs)=1/4, çıkış gerilimi V_0=6V olur.

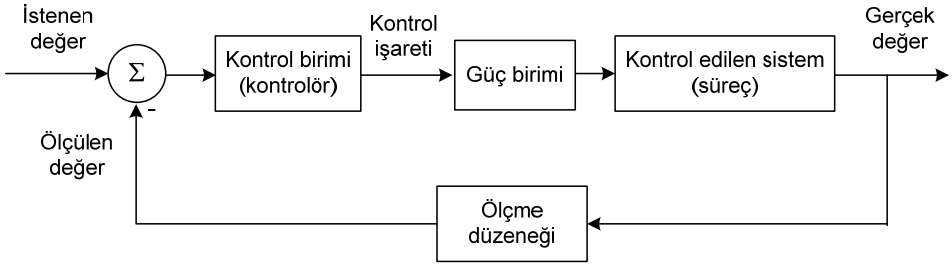
BÖLÜM 8
KONTROL SİSTEMLERİNDE CPU 22X UYGULAMALARI

8.1 GİRİŞ

Analog giriş-çıkış birimleri ile donatılmış ve matematiksel işlem özelliği olan herhangi bir sayısal işlemci ile kontrol sistemleri gerçeklenebilir. Günümüzde üretilen çoğu PLC bu özellikleri taşıdığından geribeslemeli kontrol sistemlerinin gerçeklenmesinde yaygın olarak kullanılır. Kontrol edilen sistemden alınan sürekli işaretler analog giriş birimi (ADC) ile sayısal işaretlere dönüştürülür, kontrol kuralına ilişkin matematiksel işlemlerden sonra elde edilen sayısal işaret analog çıkış birimi üzerinden sürekli işarete dönüştürülerek sisteme uygulanır.

Kontrol biriminin temel işlevi, istenen büyüklük ile bu büyüklüğe ilişkin gerçek değer arasındaki farkın en kısa sürede ortadan kaldırılmasını sağlayan kontrol işaretini üretmektir. PLC ile gerçeklemede kontrol işareti, bir kontrol kuralına ilişkin matematiksel ifadenin gerçek zamanda hesaplanması ile elde edilir.

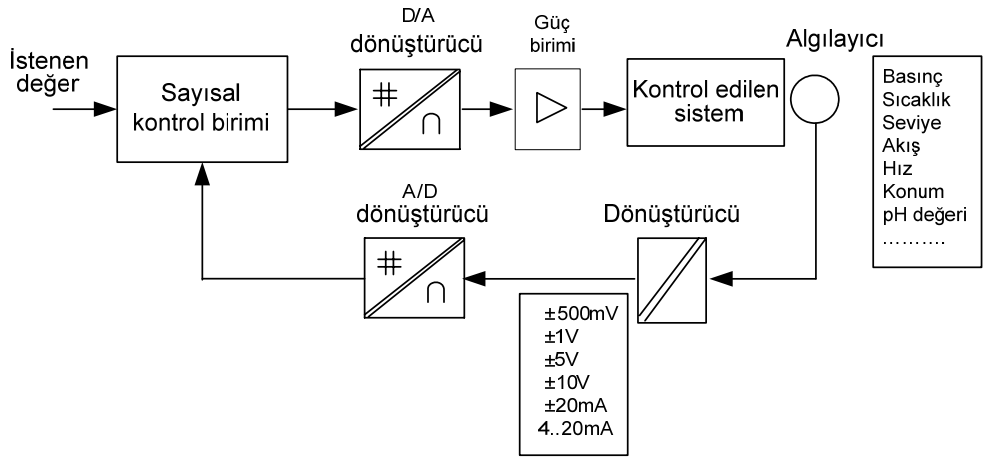
Endüstriyel uygulamalarda yaygın olarak Şekil 8.1'de gösterilen bir giriş ve bir çıkışlı geri beslemeli kontrol sistemleri kullanılır. Bu tür sistemlerde kontrol edilmek istenen fiziksel büyüklüğün gerçek değeri ölçülür, istenen değer ile karşılaştırılır ve hata olarak tanımlanan iki büyüklük arasındaki fark kontrol birimine iletilir. Kontrol birimi bu hatayı en kısa sürede gidermek için gerekli kontrol işaretini üretir.



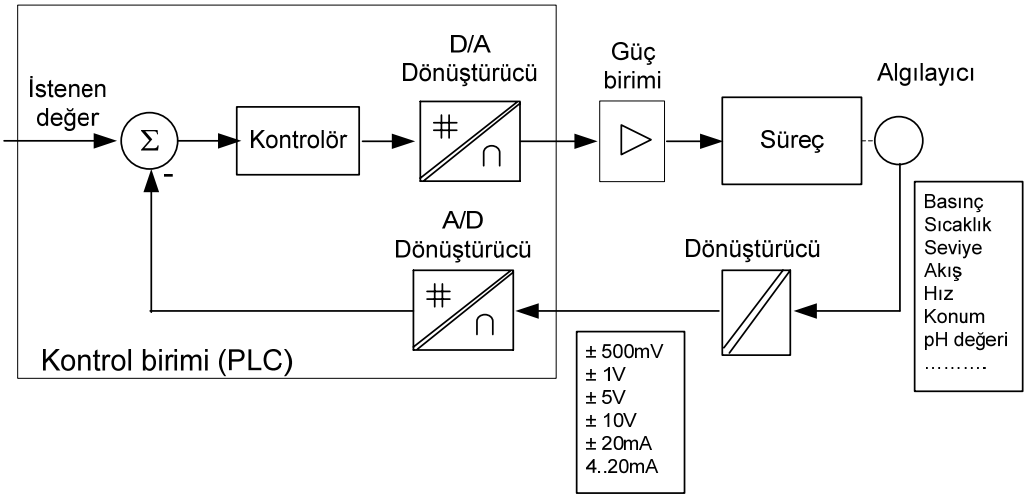
Şekil 8.1 Bir girişli ve bir çıkışlı geri beslemeli kontrol sistemi

Kontrol birimi ya da kontrolör, kontrol edilen sistemin istenen biçimde çalışması için elde edilen kontrol kuralının işlendiği temel birimdir. İstenen değer ile ölçülen değer arasındaki farkın en kısa sürede en aza indirilmesini sağlayan kontrol kuralı, çeşitli kontrol modelleri veya yaklaşımları kullanılarak elde edilebilir. Genellikle matematiksel olarak ifade edilen kontrol kuralı, günümüz endüstriyel kontrol sistemlerinde, sayısal kontrol birimleri kullanılarak gerçeklenir. Sayısal kontrol birimi ile gerçeklenen kontrol sistemleri, sayısal (dijital) kontrol sistemleri olarak tanımlanır. Bu tür sistemlere ilişkin genel yapı Şekil 8.2'deki gibi gösterilebilir. Burada bir algılayıcı üzerinden alınan fiziksel büyüklük, sayısal değere dönüştürülerek istenen değer ile karşılaştırılır, elde edilen hata büyüklüğü sayısal kontrol biriminde kontrol kuralına göre işlenir, hesaplanan sayısal değer analog işarete dönüştürülür ve kontrol işareti olarak güç birimine iletilir. Güç birimi kontrol işareti ile orantılı enerjiyi kontrol edilen sisteme uygular.

Bir giriş ve bir çıkışlı geri beslemeli bir kontrol sisteminde PLC kullanılması durumunda kontrol sistemi Şekil 8.3'teki gibi gösterilebilir. Bu sistemde kontrol edilmek istenen çıkış büyüklüğü bir ölçü düzeneği ile standart bir elektriksel büyüklük düzeyine (0-5V, 0-10V, 0-20mA, 4-20mA gibi) dönüştürülür ve analog giriş birimi üzerinden PLC'ye iletilir. Sistem çıkışının istenen değeri (referans) diğer bir analog giriş noktası üzerinden alınır. Bu iki işaret belirli bir kontrol kuralına göre işlenerek gerekli kontrol işareti üretilir ve analog çıkış birimi (DAC) üzerinden sürücü sisteme uygulanır. Sürücü sistem, kontrol işaretine bağlı olarak sistem için gerekli olan enerjiyi üretir.



Şekil 8.2 Bir sayısal kontrol sistemine ilişkin genel yapı

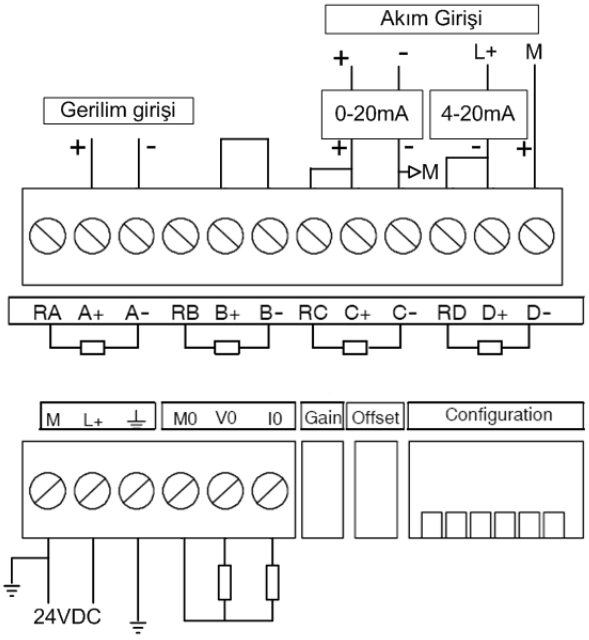


Şekil 8.3 Bir PLC ile gerçeklenen kontrol sistemi

8.2 ANALOG GİRİŞ-ÇIKIŞ BİRİMİ

S7-200 ailesi CPU 22x işlemcili PLC'lerde; EM 231 analog giriş, EM 232 analog çıkış ve EM 235 analog giriş-çıkış birimleri kullanılır. EM 231 birimi, 12 bitlik 4 girişli bir analog birimdir. Analog işaret değerleri, bir kutuplu gerilim girişli olarak 0-5V veya 0-10V, akım girişli olarak 0-20mA; iki

kutuplu gerilim girişli olarak ±5V veya ±2.5V olarak seçilebilir. EM 232 analog çıkış birimi akım çıkışlı olarak 0-20mA, gerilim çıkışlı olarak ±10V değerlerinde kullanılır. EM 235 analog biriminde ise 4 analog giriş kanalı ve 1 analog çıkış kanalı bulunur. Analog işaret değerleri bir kutuplu olarak 0-50mV, 0-100mV , 0-500mV, 0-1V, 0-5V, 0-20mA veya 0-10V veya iki kutuplu olarak ±25mV, ±50mV, ±100mV, ±250mV, ±500mV, ±1V, ±2.5V, ±5V ve ±10V seçilebilir. Şekil 8.4.'te EM 235 analog giriş-çıkış birimine ilişkin bağlantı noktaları ve bu noktalara gerilim girişi, akım girişi; gerilim çıkışı ve akım çıkışı bağlantılarının nasıl yapıldığı gösterilmiştir. Burada (B+, B-) noktalarındaki kısa devre bağlantısı kullanılmayan analog giriş noktaları için önerilen bağlantı biçimidir. Analog birimlere ilişkin diğer teknik ayrıntılar EK'te verilmiştir.



Şekil 8.4 EM 235 analog giriş-çıkış birimi bağlantı şeması

8.2.1 ANALOG VERİLERE ERİŞİM

Analog giriş noktalarına 16 bitlik AIWxx adresleri kullanılarak doğrudan erişilebilir. Analog giriş adreslerinin içerikleri, mantıksal giriş işaretlerinden farklı olarak komut işlendiği anda okunur veya yazılır. Analog giriş birimlerine ilişkin adresler, PLC'ye eklenen ilk analog birimden başlanarak

AIW0: analog birimin A+ ve A-

AIW2: analog birimin B+ ve B-

AIW4: analog birimin C+ ve C-

AIW4: analog birimin D+ ve D-

analog çıkış noktasına ilişkin adres

AQW0 (V0 ve M veya I0 ve M)

olarak, ikinci analog birim için giriş ve çıkış adresleri

AIW8 : (A+ ve A-)

AIW10: (B+ ve B-)

AIW12: (C+ ve C-)

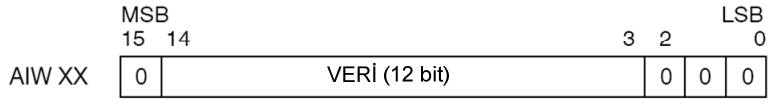
AIW14: (D+ ve D-)

AQW4: (V0 ve M veya I0 ve M)

biçiminde artarak tanımlanır.

Analog işaretler, analog giriş birimi (ADC) üzerinden 12 bitlik sayısal işaretlere dönüştürülür. Ancak, analog giriş noktalarına ilişkin bu sayısal veriler, 16 bitlik AIWxx adreslerine kaydedilir. Analog işarete ilişkin 12 bitlik sayısal veriler 16 bitlik AIWxx adresli veri alanında 2'ye tümleyenli işaretli tamsayılar biçiminde kaydedilir. Bilindiği gibi 2'ye tümleyenli işaretli sayı gösteriminde en anlamlı bit işaret biti olup pozitif değerler için 0, negatif değerler için 1 kullanılır. Pozitif sayıların mutlak değeri, işaret biti dışındaki kalan 2'li sayıların ağırlıklı toplamı, negatif sayıların değeri ise işaret biti dışındaki kalan 2'li sayıların 2'ye tümleyeninin ağırlıklı toplamına eşittir.

Analog işaretler ilişkin veriler tek kutuplu ve çift kutuplu kullanımda değişik biçimde kaydedilir. Tek kutuplu kullanımda (0-20mA, 0-5V, 0-10V gibi) analog işarete ilişkin sayısal veriler 16 bitlik alanda Şekil 8.5'te gösterildiği gibi tutulur. Bu tür kullanımda yalnız pozitif işaretler değerlendirildiğinden 16 bitlik sayının en anlamlı (MSB) biti (15.bit) 0 ve en düşük anlamlı 3 bit (0, 1 ,2) 0 değerindedir. İşaretin genliğine ilişkin sayısal bilgi geriye kalan 12 bitlik (14, 13, ..., 3) alana yazılır.



Şekil 8.5 Tek kutuplu kullanımda 16 bitlik alanda 12 bitlik analog işarete ilişkin verinin yeri

Tek kutuplu kullanımda değer aralığı 0 ile +32760 arasında değişir. 16 bitlik alanda en büyük sayısal değer, ikilik gösterimde, 2#0111_1111_1111_1000 için elde edilir. Bu durumda ondalık sayısal değer

$$1\times2^{14}+1\times2^{13}+1\times2^{12}+1\times2^{11}+1\times2^{10}+1\times2^{9}+1\times2^{8}+1\times2^{7}+1\times2^{6}+1\times2^{5}+1\times2^{4}+1\times2^{3}=+32760$$

veya onaltılık gösterimde (16#7FF8)

$$7\times16^{3}+15\times16^{2}+15\times16^{1}+8\times16^{0}=+32760$$

olur. Bu değer giriş işaretinin en büyük değerine karşı düşer. Örneğin, giriş gerilim değeri [0-10V] olarak seçilen bir analog giriş birimine +10V gerilim uygulandığında AIWxx adresinin içeriği

2#0111_1111_1111_ 1000=16#7FF8=+32760

+5V gerilim uygulandığında AIWxx adresinin içeriği

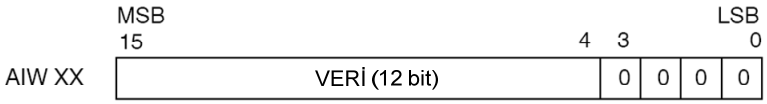
2#0011_1111_1111_ 1000=16#3FF8=+16376

girişe 0V gerilim uygulandığında AIWxx adresinin içeriği

2#0000_0000_0000_ 0000=16#0000=+0

olur.

İki kutuplu kullanımda (±5, ±10 gibi), analog işarete ilişkin 12 bitlik sayısal veri, 16 bitlik alanda Şekil 8.6'da gösterildiği gibi yer alır. İki kutuplu kullanımda en yüksek değerli bit (15.bit) işaret bitini, en düşük anlamlı 4 bit (0,1, 2, 3) 0 diğer bitler verinin sayısal değerini gösterir.



Şekil 8.6 İki kutuplu kullanımda 16 bitlik alanda 12 bitlik analog işarete ilişkin verinin yeri

İki kutuplu kullanımda pozitif sayısal veriler için değer aralığı 0 ile +32752 arasında olur. Bu değer ikilik sayı sisteminde

(2#0000_0000_0000_0000) ile (2#0111_1111_1111_0000)

onaltılık sayı sisteminde

16#0000 ile 16#7FF0

arasında değişir. Örneğin, giriş gerilim değeri aralığı ±10V seçilen bir analog birim için giriş noktasına +10V gerilim uygulandığında AIWxx adresinin içeriği

2#0111_1111_1111_ 0000=16#7FF8=+32752

giriş noktasına +5V gerilim uygulandığında AIWxx adresinin içeriği

2#0011_1111_1111_ 0000=16#3FF8=+16376

giriş noktasına 0V gerilim uygulandığında AIWxx adresinin içeriği

2#0000_0000_0000_ 0000=16#0000=+0

olur.

Negatif sayısal veriler için değer aralığı -1 ile -32768 arasında olur. Bu değer ikili sayı sisteminde

(2#1111_1111_1111_1111) ile (2#1000_0000_0000_0000)

onaltılık sayı sisteminde

16#FFFF ile 16#8000

arasında değişir.

İki kutuplu kullanımda, işaret biti dışındaki sayısal değerin pozitif gösterimdeki karşılığı N olmak üzere negatif sayının mutlak değeri (2^{15}-N) olur. Örneğin, giriş gerilim değeri aralığı ±10V seçilen bir analog birim için giriş noktasına -10V gerilim uygulandığında AIWxx adresinin içeriği

2#1000_0000_0000_ 0000=-(2^{15}-N)= -(32768-0)= -32768

giriş noktasına –5V gerilim uygulandığında AIWxx adresinin içeriği

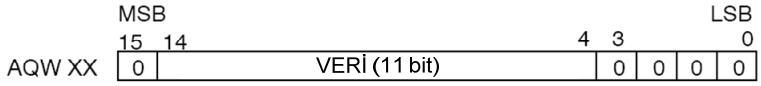
2#1100_0000_0000_ 0000 =- (2^{15}-N)= -(2^{15}-2^{14}) = -(32768-16384)= -16384

olur.

Analog çıkış adreslerine ilişkin ayısal veriler, akım ve gerilim çıkışlı olma durumlarına göre 16 bitlik AQWxx adresinde farklı biçimde tutulur.

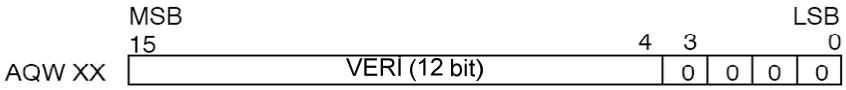
Akım çıkışı için yalnız pozitif işaret üretildiğinden, işaret biti 0 (15. bit), işaret bitini izleyen 11 bitlik veri alanında işaretin 2'li kodlanmış sayısal değeri bulunur (Şekil 8.7). Buna göre analog akım çıkış noktasından 0-20mA değerleri arasında akım üretmek için AQWx analog çıkış adresine, +16 ile

+32752 işaretli tamsayıları arasında bir değer yazılır. Analog çıkış adresine +16'dan küçük işaretli bir tamsayı ya da negatif bir tamsayı yazıldığında analog çıkış noktasından I_0=0mA, değeri +32752 olan bir işaretli tamsayı yazıldığında I_0=20mA akım üretilir.



Şekil 8.7 Akım çıkışlı analog işarete ilişkin sayısal verinin 16 bitlik alandaki yeri

Gerilim çıkış işaretine ilişkin 12 bitlik sayısal veri ise Şekil 8.8'de gösterildiği gibi 16 bitlik alanın en yüksek anlamlı 12 bitlik bölümüne yazılır.



Şekil 8.8 Gerilim çıkışlı analog işarete ilişkin sayısal verinin 16 bitlik alandaki yeri

Analog gerilim çıkış noktasından (-10V) ile (+10V) değerleri arasında gerilim üretmek için AQWx analog çıkış adresine, -32768 (16#8000) ile +32752 (16#7FF0) işaretli tamsayılar arasında bir değer yazılır. AQWx analog çıkış adresine -32768 işaretli tamsayısı yazıldığında çıkış noktasından -10V, +32752 işaretli tamsayısı yazıldığında çıkış noktasından +10V değerinde gerilim üretilir. Eğer analog çıkış adresine +32767 işaretli tamsayısından büyük bir değer yazılırsa, analog çıkış noktasından negatif bir gerilim üretilir. Bu durum, 16 bitlik verilerin 32 bitlik alanlarda (akümülatör veya VDxx adreslerinde) hesaplanıp, sonuçların 16 bitlik analog çıkış adresine aktarılmasında ortaya çıkabilir. Çünkü 32 bitlik alanda işareti pozitif olan bir tamsayı 16 bitlik alanda, işaret bitinin değerine bağlı olarak işareti negatif bir sayıya karşı düşebilir. Örneğin, 32 bitlik alanda ondalık sayısal değeri +65536 olan 16#0000FFFF onaltılık sayısı, 16 bitlik analog çıkış adresine yazıldığında, 16#FFFF onaltılık sayısı biçiminde yer alır. Bu sayısal değerin işaret biti 1 olduğundan, negatif işaretli bir tamsayı olarak 2#1111_1111_1111_1111=-1 olur.

8.2.2 ANALOG VERİLER ÜZERİNDE İŞLEMLER

Analog işaretlere ilişkin 16 bitlik işaretli tamsayı biçimindeki veriler üzerinde çeşitli matematiksel işlemler yapmak için, bu veriler reel sayı biçimine dönüştürülür. Bunun için Alt Bölüm 5.6'da verilen komutlardan yararlanılır.

Analog veriye ilişkin AIWxx adresindeki 16 bitlik işaretli tamsayıyı 32 bitlik işaretli tamsayıya dönüştürmek için

ITD AIWxx, OUT

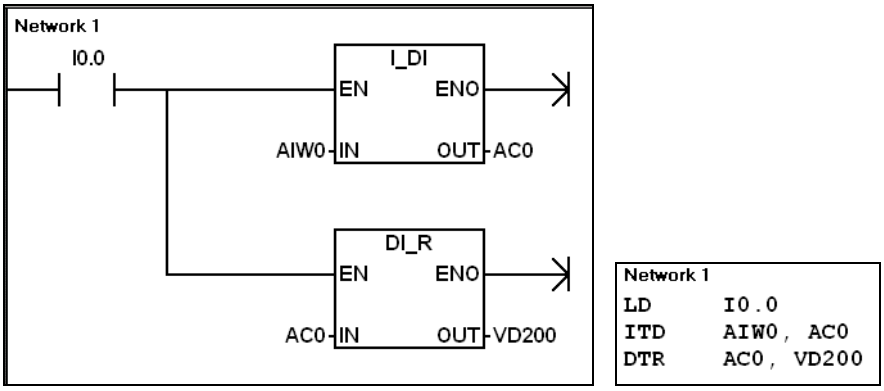
biçiminde yazılan komut kullanılır. Bu komut yürütüldüğünde AIWxx adresindeki 16 bitlik işaretli tamsayı, OUT adresine 32 bitlik işaretli tamsayı biçiminde yazılır.

32 bitlik işaretli tamsayıyı reel sayıya dönüştürmek için

DTR IN, OUT

biçiminde yazılan komut kullanılır. Bu komut yürütüldüğünde IN adresindeki 32 bitlik işaretli tamsayı 32 bitlik OUT adresine reel sayı biçiminde yazılır.

Örneğin, AIW0 adresindeki analog veriye ilişkin 16 bitlik işaretli tamsayıyı VD200 adresine reel sayı biçiminde yazmak için Şekil 8.9'daki program kullanılabilir.



Şekil 8.9 16 bitlik işaretli tamsayıyı reel sayıya dönüştürme işlemi

Bir reel sayıyı 16 bitlik işaretli tamsayıya dönüştürmek için kırpma (TRUNCATION) veya yuvarlama (ROUND) komutları kullanılır. Kırpma komutu

TRUNC IN, OUT

biçiminde yazılır. Bu komutun yürütülmesi ile IN adresindeki reel sayının kesirli bölümü atılır ve kalan ondalık bölümü 32 bitlik işaretli tamsayı biçiminde OUT adresine yazılır.

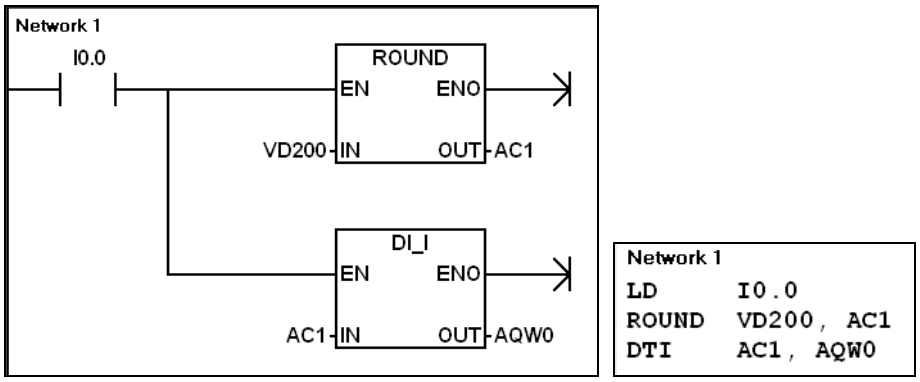
Yuvarlama komutu

ROUND IN, OUT

biçiminde yazılır. Bu komutun işletilmesi ile IN adresindeki reel sayının kesirli bölümü 5 ve daha büyükse ondalık bölümü 1 arttırılarak 32 bitlik işaretli tamsayı biçiminde, OUT adresine yazılır. Eğer

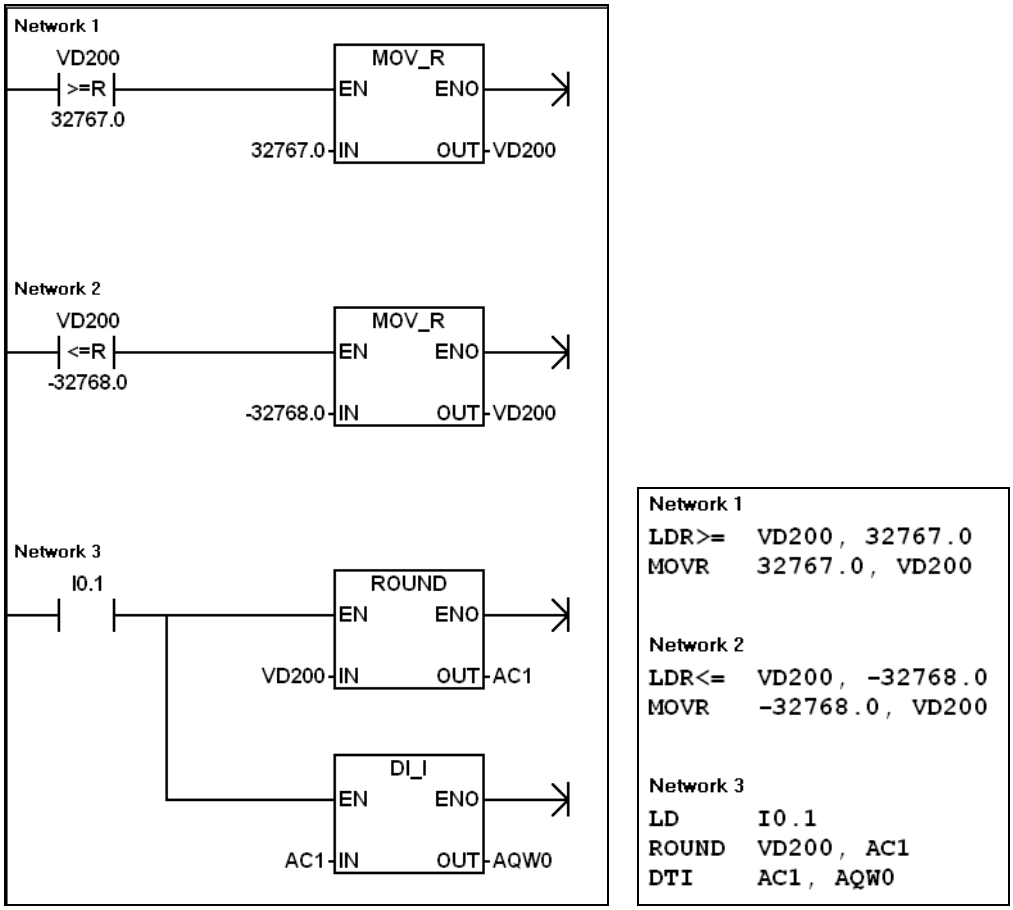
kesirli bölüm 5'ten küçükse kırpma komutu gibi kesirli bölüm atılır ve ondalık bölüm işaretli tamsayı olarak OUT adresine yazılır.

Şekil 8.10'da ROUND komutu kullanılarak VD200 adresindeki bir reel sayıyı 16 bitlik işaretli tamsayıya dönüştüren ve bu değeri AQW0 analog çıkış adresine yazan bir program örneği verilmiştir.



Şekil 8.10 Reel sayıyı işaretli tamsayıya dönüştürüp AQW0 analog çıkış adresine yazma işlemi

Şekil 8.10'da verilen komutlar ile VD200 adresindeki reel sayı 32 bitlik işaretli tamsayıya yuvarlatılarak 32 bitlik AC1 akümülatör adresine, AC1 adresindeki 32 bitlik veri 16 bitlik işaretli tamsayıya dönüştürülerek AQW0 adresine yazılır. AQW0 adresine yazılan işaretli tamsayı biçimindeki verinin, 16 bitlik işaretli tamsayılar için geçerli değer aralığı olan {-32768, +32767} sayıları dışında olması durumunda sonuç yanlış olur. Örneğin, VD200 adresindeki +40000.0 reel sayısı bu işlem sonucu -25536 değerindeki işaretli tamsayıya, -40000.0 reel sayısı +25536 değerindeki işaretli tamsayıya dönüşür. Bu durum, analog çıkış biriminden yanlış işaretlerin alınmasına neden olur. Bu sorunun giderilmesi için hesaplanan reel sayının alt sınırı -32768.0, üst sınırı +32767.0 değerleri ile sınırlandırılır. Bu amaçla Şekil 8.11'deki program yazılabilir.



Şekil 8.11 Reel sayıların sınırlandırılmasına ilişkin program örneği

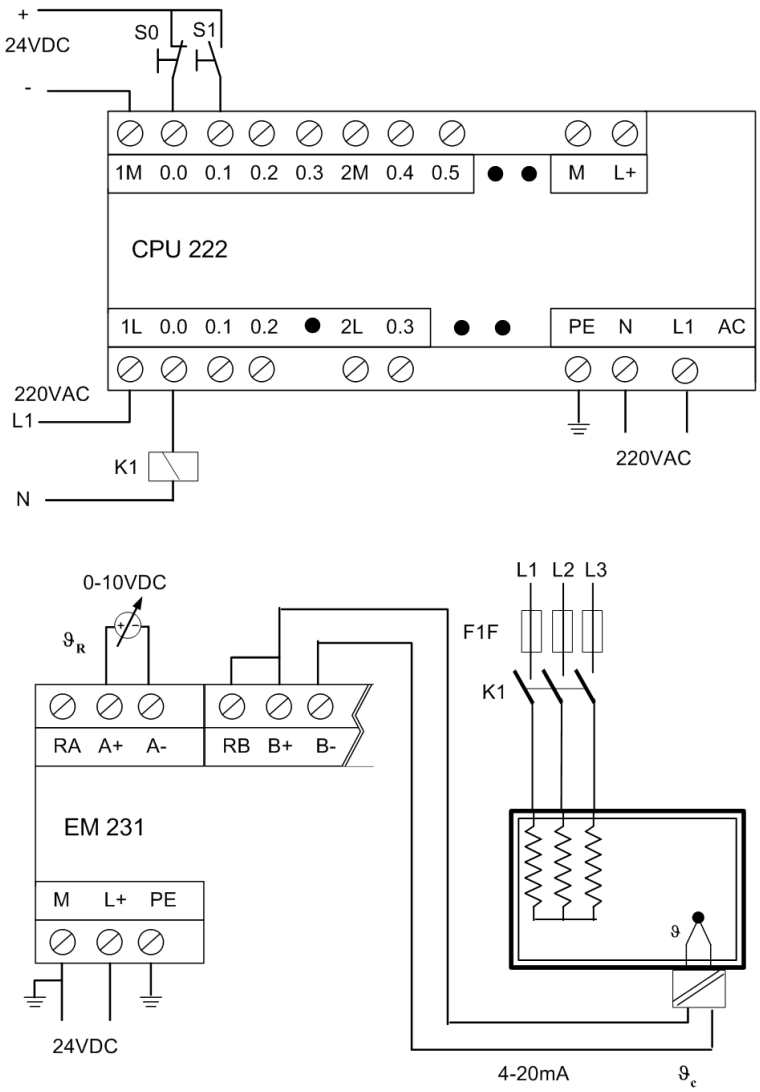
8.3 KONTROL UYGULAMALARI

Kontrol sistemleri amaç ölçütüne bağlı olarak değişik biçim ve yapılarda gerçeklenebilir. Kontrol sistemin gerçeklenmesinde, amaç ölçütünü sağlayan en basit ve ucuz çözümler aranır. Bir kontrol sisteminin seçiminde kontrolör, sürücü sistem ve ölçü düzeneği maliyetleri birlikte değerlendirilir. PLC ile gerçeklenen kontrolörün işlevi yazılım ile sağlandığından, değişik kontrol kuralları maliyeti etkilemez. Ancak, kontrol işaretlerini sisteme uygulanabilir enerjiye dönüştüren sürücü sistem ve ölçü düzeneğinin maliyeti, seçilecek kontrol yapısı ve türüne göre değişir.

8.3.1 İKİ KONUMLU KONTROL

İki konumlu kontrol (two position, bang-bang, on/off control), gerçeklenmesi kolay basit ve ucuz bir kontrol sistemidir. Bu tür bir kontrol için gerekli olan sürücü sistem de basit ve ucuzdur. Örneğin, bir ısıtma sisteminde, ısıtıcının sürücü devresi olarak bir kontaktör ya da yarıiletken bir anahtarlama elemanı kullanılabilir. Bu sistemde iki değerli (0,1) kontrol işareti kullanılır ve kontrol edilen büyüklükten alınan bilgiye göre, sistem devreye alınır ya da devreden çıkartılır. Sistem çıkışına ilişkin bilgi iki değerli bir işaret olarak da alınabilir. Bu bilgi, sistem istenen çıkış değerine erişmiştir ya da erişmemiştir bilgisini taşıyan 1 bitlik mantıksal bir işaret olabilir. Bu şekilde yapılan kontrol işlemi giriş bölümünde tanımlandığı gibi bir kumanda işlemidir. Kontrol kuralı basit mantıksal kurallara bağlı olarak elde edilir.

İki konumlu kontrol sistemine örnek olarak bağlantı şeması Şekil 8.12'de gösterilen sıcaklık kontrol düzeneği verilebilir. Bu sistemde oda sıcaklığı bir ısılçift (thermocouple) elemanı ile ölçülür, işaret koşullandırma ve dönüştürme devresi yardımıyla sıcaklıkla orantılı 4-20mA arasında değişen akım değerleri üretilir. İstenen sıcaklık değeri bir potansiyometre üzerinden 0-10V gerilim değeri olarak PLC'ye iletilir. Oda sıcaklığına ilişkin bilgi, AIW2 adresli analog giriş birimi (-B, +B, RB), istenen sıcaklık değeri ise AIW0 adresli analog giriş birimi (-A, +A) üzerinden alınır. Bu düzenekte başlatma bilgisi I0.1 girişine bağlanan S1 düğmesi, durdurma bilgisi I0.0 girişine bağlanan S0 düğmesi üzerinden alınır. Isıtıcıya enerji verilmesini sağlayan K1 kontaktörünü sürmek için Q0.0 çıkışı kullanılır.

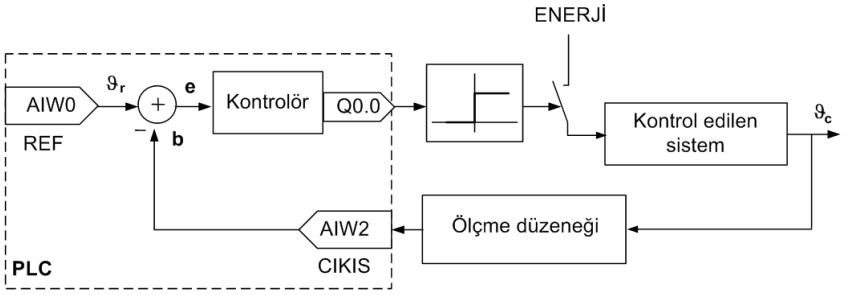


Şekil 8.12 İki konumlu kontrol sistemine ilişkin bağlantı şeması

Bu sisteme ilişkin kontrol şeması Şekil 8.13'teki gösterilebilir. Bu şemaya göre istenen sıcaklık değerinden (ϑ_r) oda sıcaklığı (ϑ_c) değeri çıkartılarak bir hata işareti (e) elde edilir.

$$e = \vartheta_r - \vartheta_c \qquad (8.1)$$

İki konumlu kontrol kuralına göre eğer hata sıfırdan büyükse Q0.0=1, hata sıfırdan küçük veya eşit ise Q0.0=0 yapılır.



Şekil 8.13 İki konumlu kontrol sistemine ilişkin uygulama şeması

Şekil 8.12'de gösterilen kontrol sistemine ilişkin bağlantı şemasında, oda sıcaklık değerini ölçmek için kullanılan algılayıcı, oda sıcaklığı 0^0C olduğunda I=4mA, oda sıcaklığı 200^0C olduğunda I=20mA değerinde akım üretir. İstenen sıcaklık bilgisinin 0-10V gerilim değeri ile PLC'ye iletilmesi durumunda; 0^0C sıcaklık değeri V=0V, 200^0C sıcaklık değeri V=10V değerlerine karşı düşer. Ara değerlerin doğrusal değiştiği varsayılarak sıcaklık ile akım ve gerilim arasındaki ilişki Şekil 8.14'teki gibi gösterilebilir.

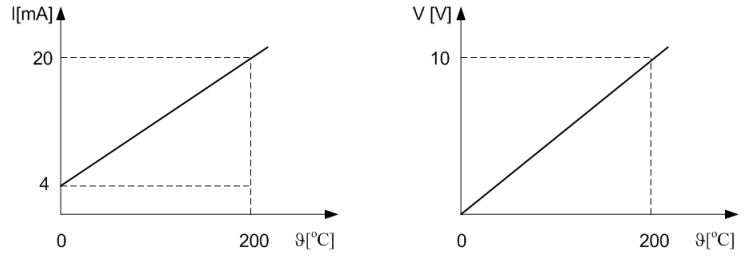
Şekil 8.14'te verilen eğrilere göre, akım ile sıcaklık arasındaki ilişki

$$I = 4 + \frac{20-4}{200}\vartheta_c \Rightarrow \vartheta_c = \frac{25}{2}(I-4) \tag{8.2}$$

gerilim ile sıcaklık arasındaki ilişki

$$V = \frac{10}{200}\vartheta_r \quad \Rightarrow \quad \vartheta_r = 20V \tag{8.3}$$

ifadeleri ile verilebilir.



Şekil 8.14 Akım ve Gerilim ile sıcaklık arasındaki değişim eğrileri

Analog giriş birimi akım girişindeki değer 0mA olduğunda adresin içeriği 16#0000=0, 20mA olduğunda adresin içeriği 16#7FF8=32760 olur. Aynı değerler tek kutuplu gerilim girişi kullanımı için de geçerlidir. Buna göre $I_{\#}$ akım girişinden okunan sayısal değer olmak üzere

$$I_{\#} = I \times 32760/20mA \Longrightarrow I = I_{\#} \times 20mA/32760$$

ilişkisi yazılabilir. Bu bağıntı (8.2) ifadesinde kullanılırsa akım ile sıcaklık arasındaki ilişkiyi gösteren bağıntı

$$\vartheta_c = \frac{25}{2}(I-4) = \frac{25}{2}(\frac{20}{32760}I_{\#} - 4) = \frac{25}{3276}I_{\#} - 50 = 0.00763I_{\#} - 50 \qquad (8.4)$$

biçiminde elde edilir.

Referans gerilimi ile gerilim girişinden okunan sayısal değer $V_{\#}$ arasındaki ilişki

$$V = 10 \times V\#/32760$$

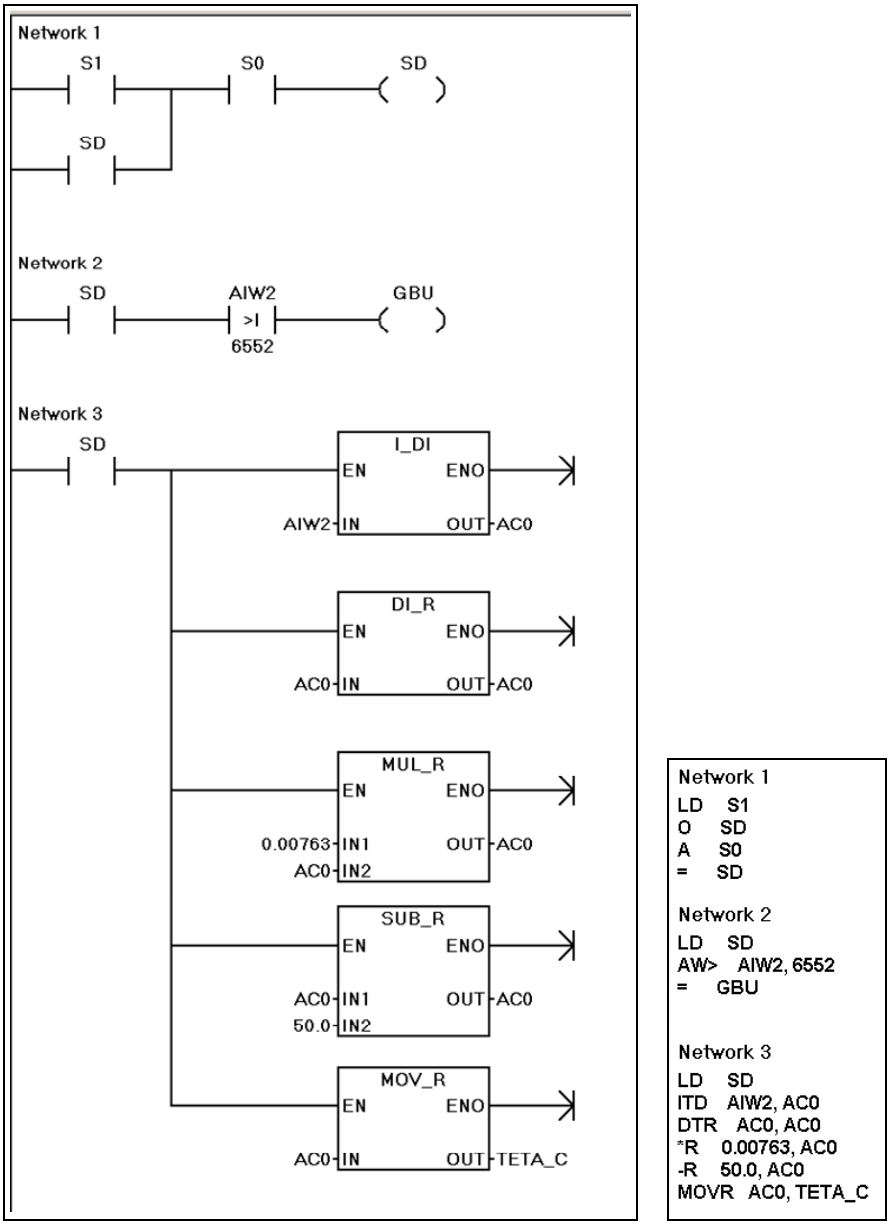
biçiminde yazılabilir. Bu ifade (8.3) bağıntısında kullanılırsa referans sıcaklığı ile $V_{\#}$ sayısal değeri arasındaki bağıntı

$$\vartheta_r = 20\frac{10}{32760}V_{\#} = \frac{200}{32760}V_{\#} = 0.0061V_{\#} \qquad (8.5)$$

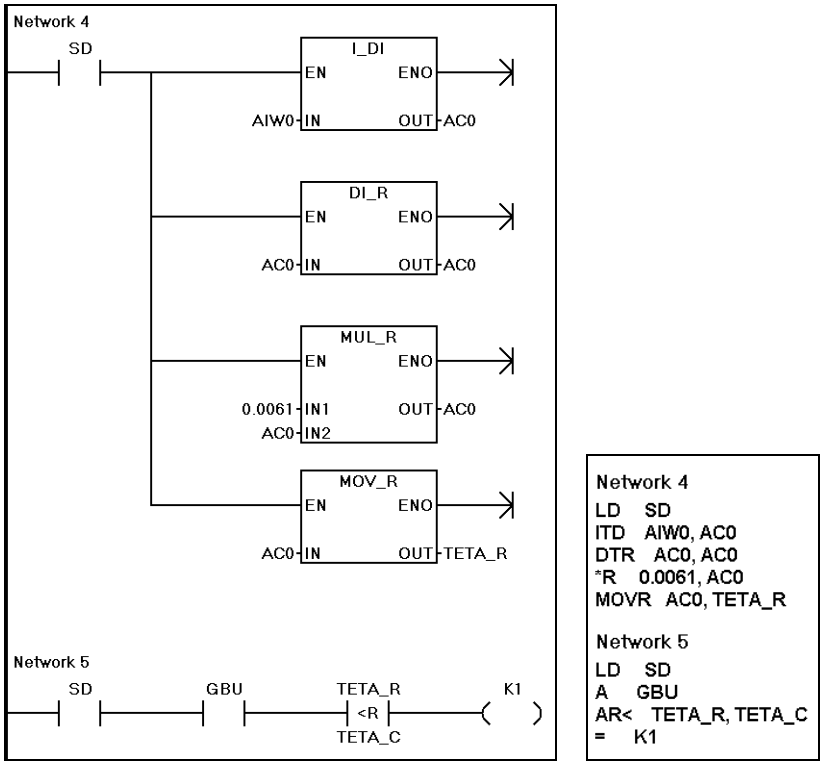
biçiminde elde edilir.

Bu sisteme ilişkin program Şekil 8.15'teki gibi verilebilir. Bu programda "Network 1" sistemin devreye alınması ve devreden çıkartılmasına ilişkin koşulları, "Network 2" geribesleme iletkeninin durumuna ilişkin koşulları (AIW2 adresinin içeriği 32760/20*4=6552 değerinden küçük ise geribeslemede sorun vardır) sağlar; "Network 3" (8.4) ifadesini, "Network 4" (8.5) ifadesini gerçekler ve "Network 5" kontrol kuralına göre ısıtıcının devreye girmesi veya çıkmasını sağlar.

Symbol	Address	Comment
S1	I0.1	BASLA
S0	I0.0	DUR
SD	M0.0	SISTEM DEVREDE
GBU	M0.1	GERI BESLEME UYGUN
K1	Q0.0	ISITICI DEVREDE
TETA_R	VD20	ISTENEN SICAKLIK
TETA_C	VD24	GERÇEK SICAKLIK



Şekil 8.15 İki konumlu kontrol kuralına ilişkin program örneği (1/2)



Şekil 8.15 İki konumlu kontrol kuralına ilişkin program örneği (2/2)

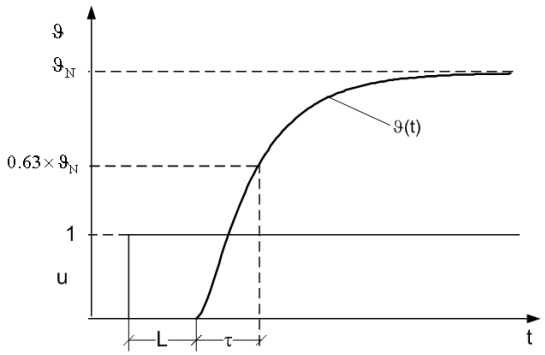
Bu kontrol kuralına göre elde edilen kumanda işaretinin anahtarlama sıklığı, bazı sürücü düzenekleri için uygun olmayan değerlere çıkabilir. Uygulamada daha düşük anahtarlama sıklığı elde etmek için çeşitli yöntemler kullanılır. Bu amaçla, kumanda işaretinin bir histerizis eğrisine göre veya darbe genişlik modülasyonu yöntemine göre üretilmesi en yaygın kullanılan yöntemlerdir.

İki konumlu kontrol yöntemi, belirli büyüklükte aşım ve salınıma izin verilebilen sistemlere uygulanabilir. Bu tür sistemlere örnek olarak birinci mertebeden sistemler, aşırı veya kritik sönümlü yüksek dereceden sistemler, integral etkili birinci mertebeden sistemler veya integral etkili yüksek dereceden kritik sönümlü veya aşırı sönümlü sistemler gösterilebilir. Ancak, endüstriyel süreçlerde genellikle karşılaşılan çeşitli büyüklüklerdeki ölçme gecikmeleri nedeniyle aşım ve salınım değerleri kabul edilemez değerlere çıkabilir. Bir sisteme iki konumlu kontrol uygulandığında oluşacak aşım ve salınım değerleri bulunarak, bu kontrol yönteminin başarımı değerlendirilebilir.

Şekil 8.12'de verilen ısıl sürece enerji verildiğinde (u=1), odanın ortam üstü sıcaklığı Şekil 8.16'da gösterildiği gibi değişir. Basamak girişe yanıtı bu biçimde değişen süreçler, birinci mertebeden ölü zamanlı sistemler olarak tanımlanır ve s-tanım bölgesi transfer fonksiyonu

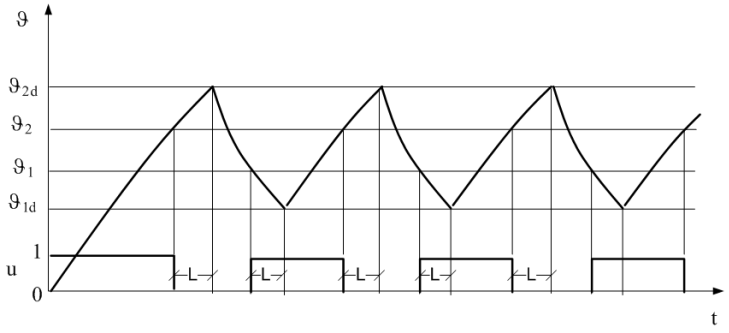
$$G(s) = \frac{K}{\tau s + 1} e^{-sL} \tag{8.6}$$

biçiminde ifade edilir. Burada K kazanç, L ölü zaman ve τ zaman sabitidir.



Şekil 8.16 Birinci mertebeden ölü zamanlı sistemleri basamak girişe yanıtı

Birinci mertebeden ölü zamanlı sistem olarak tanımlanan bir ısıl sürece iki konumlu kontrol uygulandığında, sistem çıkışı Şekil 8.17'de gösterildiği gibi değişir. Burada, ortam üstü sıcaklığının ϑ_2 değerini aşması durumunda ısıl sürece verilen enerjinin kesildiği (u=0), ortam üstü sıcaklığının ϑ_1 değerinin altına inmesi durumunda ısıl sürece enerji verildiği (u=1) varsayılmıştır.



Şekil 8.17 Isıl sisteme iki konumlu kontrol uygulanması durumunda sistem çıkışının değişimi

Isıl sistemin u=1 için ulaştığı son sıcaklık değeri ϑ_N ve sıcaklığın başlangıç değeri ϑ_0 olmak üzere, ısıl sürece enerji uygulandığında sıcaklığın zaman göre değişimi

$$\vartheta(t) = \vartheta_N (1 - e^{-(t-L)/\tau}) u(t-L) + \vartheta_0 e^{-(t-L)/\tau} \tag{8.7}$$

ifadesi ile verilebilir. Bu ilişkiden ve Şekil 8.17'deki eğrilerden hareketle aşım değerleri için

$$\vartheta_{1d} = \vartheta_1 e^{-L/\tau}$$

$$\vartheta_{2d} = \vartheta_N (1 - e^{-L/\tau}) + \vartheta_2 e^{-L/\tau}$$

ilişkileri yazılabilir. Eğer, Şekil 8.15'te verilen programda gerçeklendiği gibi $\vartheta_r = \vartheta_1 = \vartheta_2$ alınırsa, çıkışın en yüksek değeri ile en düşük değeri arasındaki fark

$$\Delta\vartheta = \vartheta_{2d} - \vartheta_{1d} = \vartheta_N (1 - e^{-L/\tau}) \qquad (8.8)$$

biçiminde elde edilir. Bu ifadeye göre, örneğin, $L/\tau = 0.5$ olan bir ısıl süreç için

$$\Delta\vartheta = \vartheta_N (1 - e^{-0.5}) = 0.393 \times \vartheta_N$$

$L/\tau = 0.1$ olan bir süreç için

$$\Delta\vartheta = \vartheta_N (1 - e^{-0.1}) = 0.095 \times \vartheta_N$$

değerleri bulunur.

8.3.2 PID KONTROL

Endüstriyel süreçlere ilişkin kontrol sorunlarının çözümünde PID kontrol, özel bir yere ve öneme sahiptir. Endüstriyel uygulamalara ilişkin kontrol yapıları genellikle PID kontrol çevrimlerinden oluşur. PID kontrol kuralı, hata işaretinin oransal değeri (P), zamana göre integrali (I) ve zamana göre değişimine (D) bağlı olarak ifade edilir. PID (Oransal, integral, türev) kontrol kuralına ilişkin matematiksel ifade

$$u(t) = K_C \left(e(t) + \frac{1}{T_I} \int e(t)dt + T_D \frac{de(t)}{dt} \right) \qquad (8.9)$$

biçiminde ifade edilebilir. Burada,

K_C: toplam kontrolör kazancı

T_I: integral zaman sabiti

T_D: türev zamanı sabiti

e(t): hata işareti

u(t): kontrol işareti

olarak tanımlanır.

PID kontrol kuralına göre u(t) kontrol işareti

$$P(t) = K_C e(t) \text{ oransal terim,}$$

$$I(t) = \frac{K_C}{T_I}\int e(t)dt \text{ integral terimi,}$$

$$D(t) = K_C T_D \frac{de(t)}{dt} \text{ türev terimi}$$

olmak üzere

$$u(t) = P(t) + I(t) + D(t) \qquad (8.10)$$

biçiminde yazılabilir.

PID kontrol kuralına ilişkin bu ifadeler matematiksel işlem komutları kullanılarak PLC'de kolayca gerçeklenebilir. Çoğu PLC'de PID işlevini yerine getiren komutlar veya hazır fonksiyonlar da bulunur. S7-200 ailesi PLC'lerde bu amaçla PID komutu kullanılır.

PID komutu

PID TBL, LOOP

biçiminde yazılır. Burada TBL 36 byte boyutundaki bir veri belleği alanının başlangıç adresi, LOOP 0-7 arasında bir tamsayıdır. Her tamsayı ayrı bir PID çevrimini temsil eder. Buna göre en çok 8 PID fonksiyonu gerçeklenebilir. Her PID çevrimi için ayrı bir TABLO Şekil 8.18'deki gibi tanımlanır ve düzenlenir.

PID komutu, PID işlevini sayısal gerçeklediğinden işlem sonucunun doğru yapılabilmesi için, bu komutun belirli bir örnekleme süresi ile yürütülmesi gerekir. Bunu sağlamak için PID komutunun PID tablosunda belirtilen Ts değerine göre yürütülmesini sağlayan bir düzenleme yapılmalıdır. Bunun için eğer örnekleme zamanı 255ms değerinden küçük ise zaman bağlı kesmeli çalışma özelliğinden yararlanılabilir. Örnekleme zamanının 255ms'den büyük olduğu uygulamalar için zamanlayıcılar kullanılarak, altprogramın belirli zaman dilimlerinde işletilmesi sağlanır.

PID Tablosu

VBx	Veri	V. biçimi
0	y	#.#
4	r	#.#
8	u	#.#
12	Kc	#.#
16	Ts	#.#[s]
20	T_I	#.#[dak]
24	T_D	#.#[dak]
28	I_1	#.#
32	y_1	#.#

Şekil 8.18 PID veri tablosu

Şekil 8.18'deki veri tablosunda

y simgesi ile gösterilen 32 bitlik alana kontrol edilen büyüklüğün bağıl değeri,

r simgesi ile gösterilen 32 bitlik alana kontrol edilen büyüklüğün istenen değerine ilişkin bağıl değer,

u simgesi ile gösterilen 32 bitlik alana hesaplanan kontrol işaretinin bağıl değeri,

Kc simgesi ile gösterilen 32 bitlik alana oransal katsayı,

Ts simgesi ile gösterilen 32 bitlik alana saniye boyutunda örnekleme zamanı değeri,

T_I simgesi ile gösterilen 32 bitlik alana dakika boyutunda integral zamanı değeri,

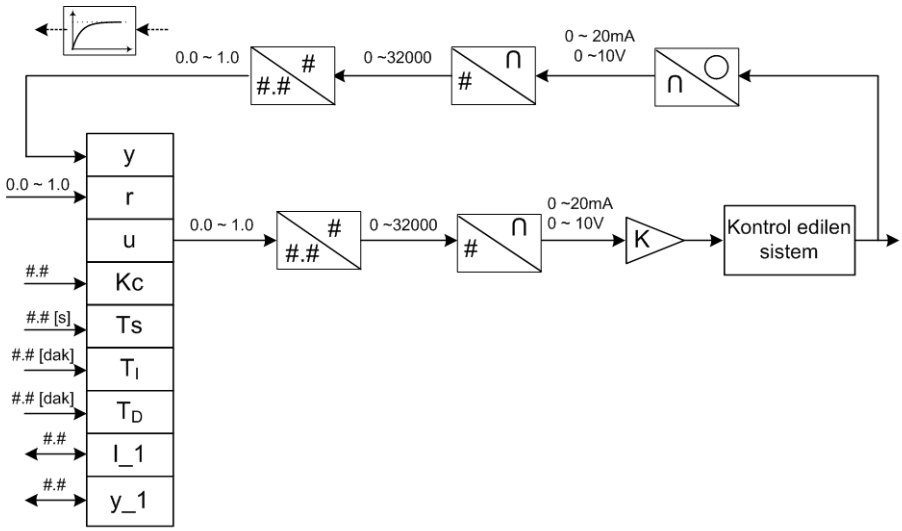
T_D simgesi ile gösterilen 32 bitlik alana dakika boyutunda türev zamanı değeri,

I_1 simgesi ile gösterilen 32 bitlik alana integral işleminin bir örnekleme zamanı önceki bağıl değeri

y_1 simgesi ile gösterilen 32 bitlik alana kontrol edilen büyüklüğün bir örnekleme zamanı önceki bağıl değeri yazılır.

PID veri tablosu ile kontrol edilen sistem arasındaki ilişki Şekil 8.19'da gösterilmiştir. Bu şemada # simgesi 16 bitlik işaretli tamsayıyı, #.# simgesi reel sayıyı ve ⋂ simgesi analog işareti göstermektedir.

Tüm bağıl değerler 0.0 ile 1.0 arasında değiştiğinden kontrol edilen sisteme ilişkin ölçülen analog işaretlerin işaretli tamsayı olarak okunan değerleri, bu değerlere karşı düşürülür.



Şekil 8.19 PID veri tablosu ve kontrol çevrimi

PID komutu

$$
\begin{aligned}
&P(k) = Kc\,[\, r(k) - y(k)]\\
&I(k) = I(k-1) + Kc\,(Ts / TI\,)\,[\, r(k) - y(k)]\\
&D(k) = Kc\,(T_D\ / Ts\,)\,[\, y(k-1) - y(k)]\\
&u(k) = P(k) + I(k) + D(k)\\
&I(k-1) = 1.0 - (\, P(k) + D(k)\,) \quad , \qquad I(k-1) > 1.0\\
&I(k-1) = -(\, P(k) + D(k)\,) \quad , \qquad I(k-1) < 0.0
\end{aligned}
\tag{8.11}
$$

fark denklemlerine göre u kontrol işaretini üretir. Üretilen u kontrol işareti 0.0 ile 1.0 değerleri arasında değişir. Bu değerin analog çıkış birimi adresine yazılması için 16 bitlik işaretli tamsayıya dönüştürülmesi gerekir.

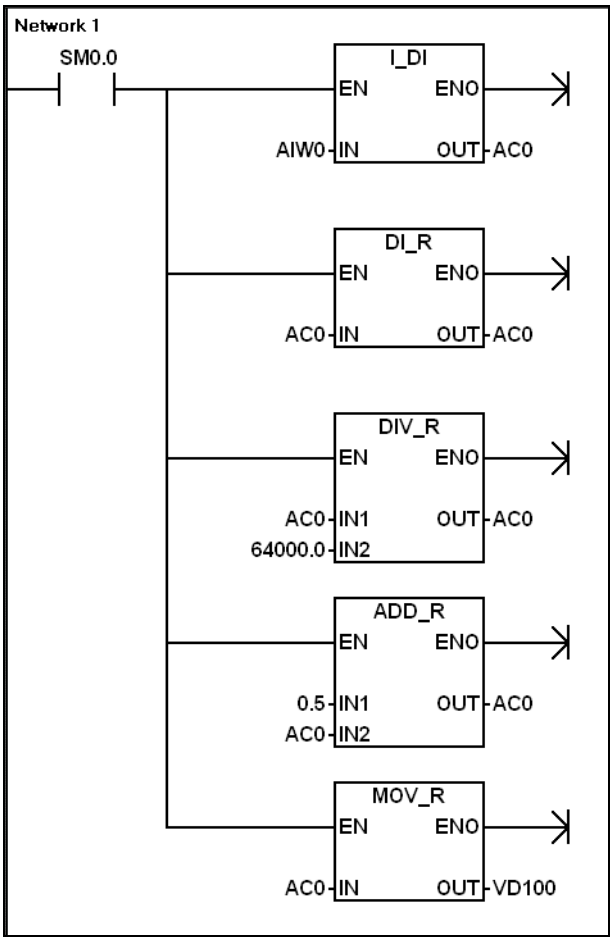
Çıkış ve kontrol işaretlerinin 0.0 ile 1.0 arasında değişen bağıl değerlere dönüştürülmesi için

$$x_r = \frac{x_{\#}}{x_{\#N}} + x_0 \tag{8.12}$$

ifadesi kullanılabilir. Burada x_r : 0.0 ile 1.0 arasında değişen bağıl değer, $x_{\#}$ işaretin sayısal değeri, $x_{\#N}$ temel sayısal değerdir. Tek kutuplu analog işaretler için temel sayısal değer, analog işaretin tam değerine karşı düşen 16 bitlik sayı (örneğin, 32000), iki kutuplu analog işaretlerde ise analog işaretin

tam değerine karşı düşen 16 bitlik sayının 2 katı (örneğin, 64000) alınır. x_0 ise düzeltme (offset) değeridir: Tek kutuplu analog işaretler için 0.0, çift kutuplu analog işaretler için 0.5 alınır.

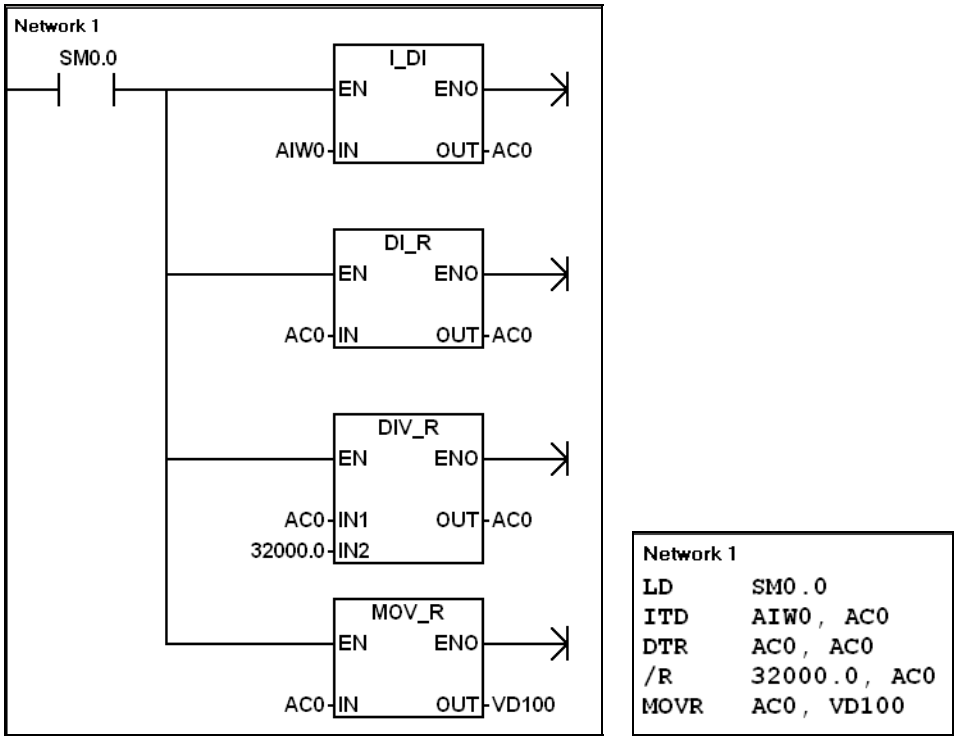
Değer aralığı –10 ile +10V arasında değişen ve AIW0 adresinden okunun çift kutuplu bir analog işaretin bağıl değerini hesaplayarak VD100 adresine yazan bir program örneği Şekil 8.20'deki gibi yazılabilir.




```
Network 1
LD      SM0.0
ITD     AIW0, AC0
DTR     AC0, AC0
/R      64000.0, AC0
+R      0.5, AC0
MOVR    AC0, VD100
```

Şekil 8.20 Çift kutuplu bir analog işaretin bağıl değerinin bulunmasına ilişkin program örneği

Değer aralığı 0 ile +10V arasında değişen tek kutuplu bir analog işaretin bağıl değerini hesaplamak için Şekil 8.21'deki program parçası yazılabilir.

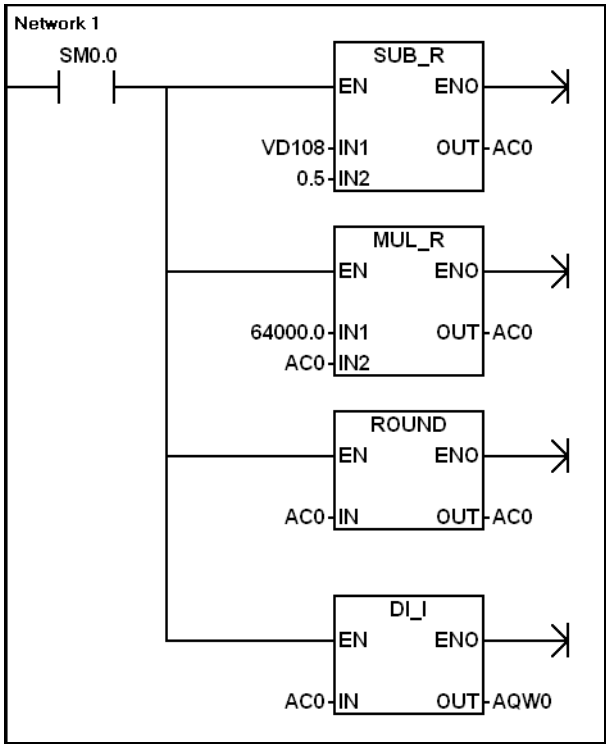


Şekil 8.21 Tek kutuplu bir analog işaretin bağıl değerinin bulunmasına ilişkin program örneği

PID komutu ile hesaplanan bağıl kontrol işaretinin işaretli tamsayıya dönüştürülmesi için (8.12) ifadesinden yararlanılabilir. Bu ifade düzenlenirse

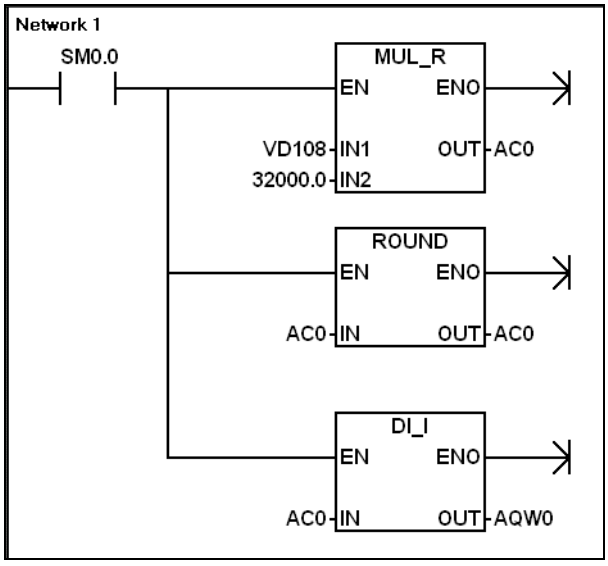
$$x_{\#} = (x_r - x_0)x_{\#N} \tag{8.13}$$

elde edilir. Bu ifade programlanarak işaretli tamsayı hesaplanır. İşaretli tamsayının iki kutuplu olması durumunda $x_{\#N}$=64000 ve x_0=0.5 değerleri, tek kutuplu olması durumunda $x_{\#N}$=32000 ve x_0=0 değerleri kullanılır. Örneğin, PID tablosundaki u kontrol işaretine ilişkin bağıl değerin yazıldığı VD108 adresindeki reel sayıyı, değer aralığı –10 ile +10V arasında değişen iki kutuplu 16 bitlik işaretli tamsayıya dönüştüren program parçası Şekil 8.22'deki gibi, değer aralığı 0 ile 20mA arasında değişen tek kutuplu işaretli tamsayıya dönüştüren program parçası Şekil 8.23'teki gibi yazılabilir.




```
Network 1
LD      SM0.0
MOVR    VD108, AC0
-R      0.5, AC0
*R      64000.0, AC0
ROUND   AC0, AC0
DTI     AC0, AQW0
```

Şekil 8.22 Bağıl değerin, iki kutuplu işaretin sayısal değerine dönüştürülmesi işlemi




```
Network 1
LD      SM0.0
MOVR    VD108, AC0
*R      32000.0, AC0
ROUND   AC0, AC0
DTI     AC0, AQW0
```

Şekil 8.23 Bağıl değerin, tek kutuplu işaretin sayısal değerine dönüştürülmesi işlemi

Örnek 8.1 Bir kontrol sisteminde istenen değer çift kutuplu gerilim girişi olarak adresi AIW0 olan analog giriş noktası, sistem çıkışına ilişkin analog işaret çift kutuplu olarak adresi AIW2 olan analog giriş noktası üzerinden alınmakta ve hesaplanan kontrol işareti çift kutuplu olarak adresi AQW0 analog çıkış noktası üzerinden sisteme uygulanmaktadır. Bu kontrol sisteminde kullanılan PID kontrolörü için örnekleme zamanı Ts=100ms, kontrolör oransal katsayısı Kc=0.5, integral zamanı T_I=6s olarak verilmiştir. Sistem, PLC I0.0 girişinin 1 yapılması ile devreye girmekte ve I0.1 girişinin 0 yapılması ile devreden çıkmaktadır. Bu sistem için gerekli PLC programını yazınız.

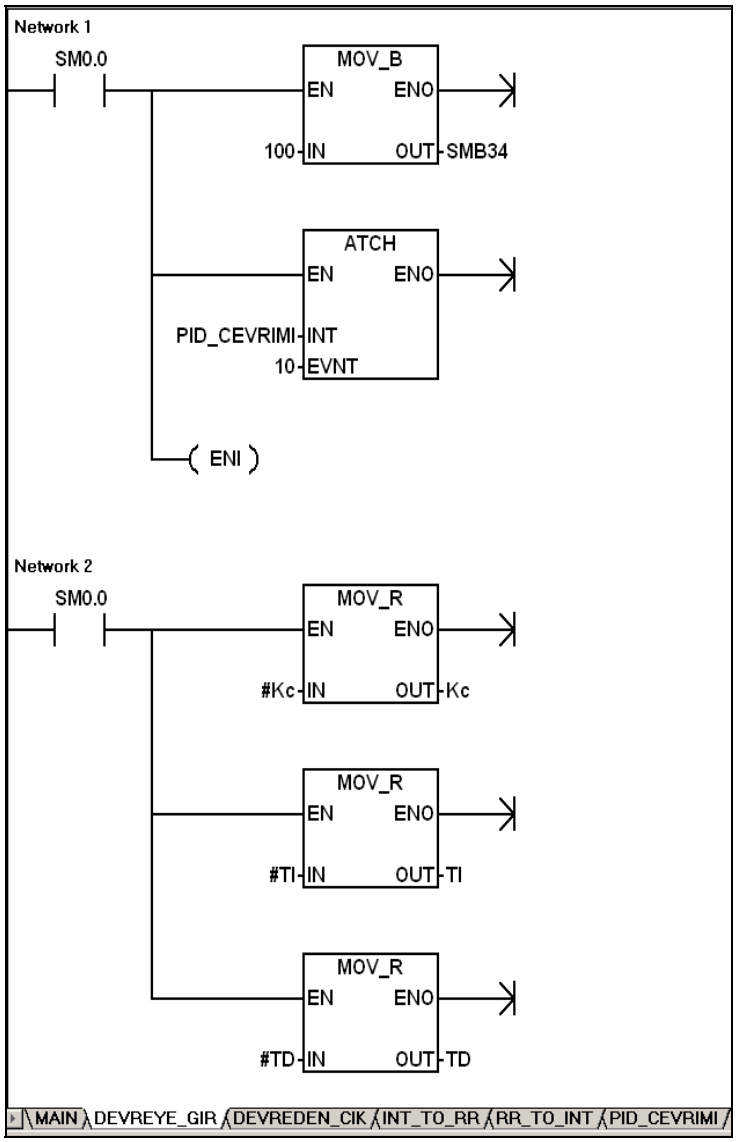
Bu örneğe ilişkin sembol tablosu, Şekil 8.24'teki gibi düzenlenebilir.

Symbol	Address	Comment
y_r	VD200	CIKIS [0.0, 1.0]
r_r	VD204	REFERANS [0.0, 1.0]
u_r	VD208	KONTROL ISARETI [0.0, 1.0]
Kc	VD212	ORANSAL KATSAYI [##.##]
Ts	VD216	ORNEKLEME ZAMANI [s]
TI	VD220	INTEGREAL SABITI [dakika]
TD	VD224	TUREV SABITI [dakika]
I_1	VD228	ONCEKI INTEGRAL DEGERI [#.#]
y_1	VD232	ONCEKI CIKIS DEGERI [#.#]
BASLA	I0.0	
DUR	I0.1	
Y_INT	AIW0	CIKIS [INT]
R_INT	AIW2	REFERANS[INT]
U_INT	AQW0	KONTROL ISARETI INT]

Şekil 8.24 Sembol tablosu

Sistemin devreye alınmasına ilişkin koşullar göz önüne alınarak, Şekil 8.25'teki "DEVREYE_GIR" altprogramı yazılabilir. Burada, "PID_CEVRIMI" başlıklı kesme altprogramının işletilmesine ilişkin koşullar tanımlanır ve PID parametrelerine ilişkin katsayılar atanır.

	Symbol	Var Type	Data Type
	EN	IN	BOOL
LD0	Kc	IN	REAL
LD4	TI	IN	REAL
LD8	TD	IN	REAL

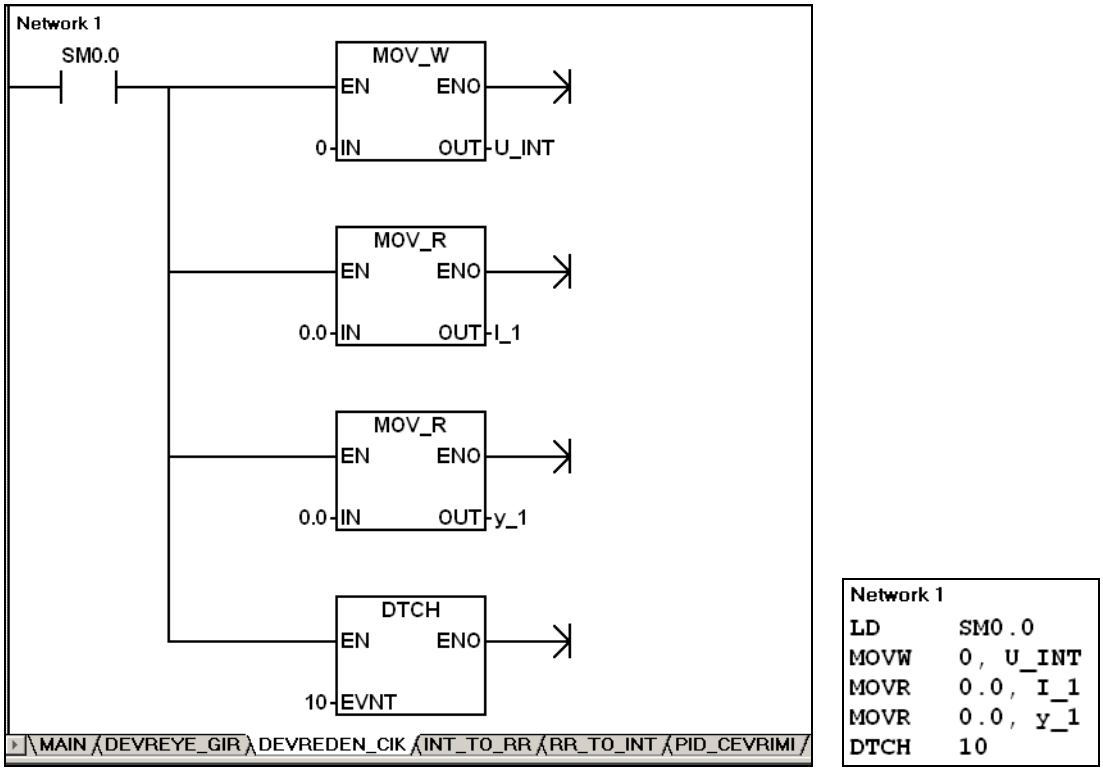



```
Network 1
LD      SM0.0
MOVB    100, SMB34
ATCH    PID_CEVRIMI, 10
ENI

Network 2
LD      SM0.0
MOVR    #Kc, Kc
MOVR    #TI, TI
MOVR    #TD, TD
```

Şekil 8.25 Sistemi devreye almaya ilişkin altprogam

Sistemi devreden çıkarma koşulları göz önüne alınarak, Şekil 8.26'da verilen "DEVREDEN_CIK" altprogramı yazılabilir.



Şekil 8.25 Sistemi devreden çıkartmaya ilişkin altprogam

PID komutuna ilişkin veri tablosunda giriş ve çıkışlara ilişkin büyüklükler bağıl değer olarak tutulduğundan analog girişten okunan ve analog çıkışa yazılan işaretli tamsayıların bağıl değerleri hesaplanır. İşaretli tamsayıyı bağıl değere dönüştürmek için Şekil 8.26'da verilen "INT_TO_RR" başlıklı altprogram yazılabilir. Bu altprogram hem iki kutuplu hem de tek kutuplu işaretlerin bağıl değerlere dönüştürülmesi için kullanılabilir. İki kutuplu işaretlerin bağıl değerlere dönüştürülmesinde "BIPOLAR" girişi mantıksal 1, tek kutuplu işaretlerin bağıl değerlere dönüştürülmesinde "BIPOLAR" girişi mantıksal 0 yapılır.

	Symbol	Var Type	Data Type
	EN	IN	BOOL
LW0	X_INT	IN	INT
L2.0	BIPOLAR	IN	BOOL
		IN_OUT	
LD3	X_RR	OUT	REAL
LD7	OFFSET	TEMP	REAL
LD11	TD_REAL	TEMP	REAL

```
Network 1
LD      #BIPOLAR
JMP     0
MOVR    0.0, #OFFSET
MOVR    32000.0, #TD_REAL
JMP     1

Network 2
LBL     0

Network 3
LD      SM0.0
MOVR    0.5, #OFFSET
MOVR    64000.0, #TD_REAL

Network 4
LBL     1

Network 5
LD      SM0.0
ITD     #X_INT, AC0
DTR     AC0, AC0
/R      #TD_REAL, AC0
+R      #OFFSET, AC0
MOVR    AC0, #X_RR
```

MAIN | DEVREYE_GIR | DEVREDEN_CIK | INT_TO_RR | RR_TO_INT | PID_CEVRIMI

Şekil 8.26 İşaretli tamsayıyı bağıl değere dönüştüren altprogam

PID komutu ile hesaplanan bağıl değeri işaretli tamsayıyı dönüştürmek için Şekil 8.27'de verilen "RR_TO_INT" başlıklı altprogram yazılabilir. Bu altprogram hem iki kutuplu hem de tek kutuplu işaretlerin işaretli tamsayıya dönüştürülmesi için kullanılabilir. Bağıl değerin iki kutuplu işaretli

tamsayıya dönüştürülmesinde "BIPOLAR" girişi mantıksal 1, tek kutuplu işaretli tamsayıya dönüştürülmesinde "BIPOLAR" girişi mantıksal 0 yapılır.

	Symbol	Var Type	Data Type
LD0	X_RR	IN	REAL
L4.0	BIPOLAR	IN	BOOL
		IN_OUT	
LW5	X_INT	OUT	INT
LD7	OFFSET	TEMP	REAL
LD11	TD_REAL	TEMP	REAL

```
Network 1
LD      #BIPOLAR
JMP     0
MOVR    0.0, #OFFSET
MOVR    32000.0, #TD_REAL
JMP     1


Network 2
LBL     0


Network 3
LD      SM0.0
MOVR    0.5, #OFFSET
MOVR    64000.0, #TD_REAL

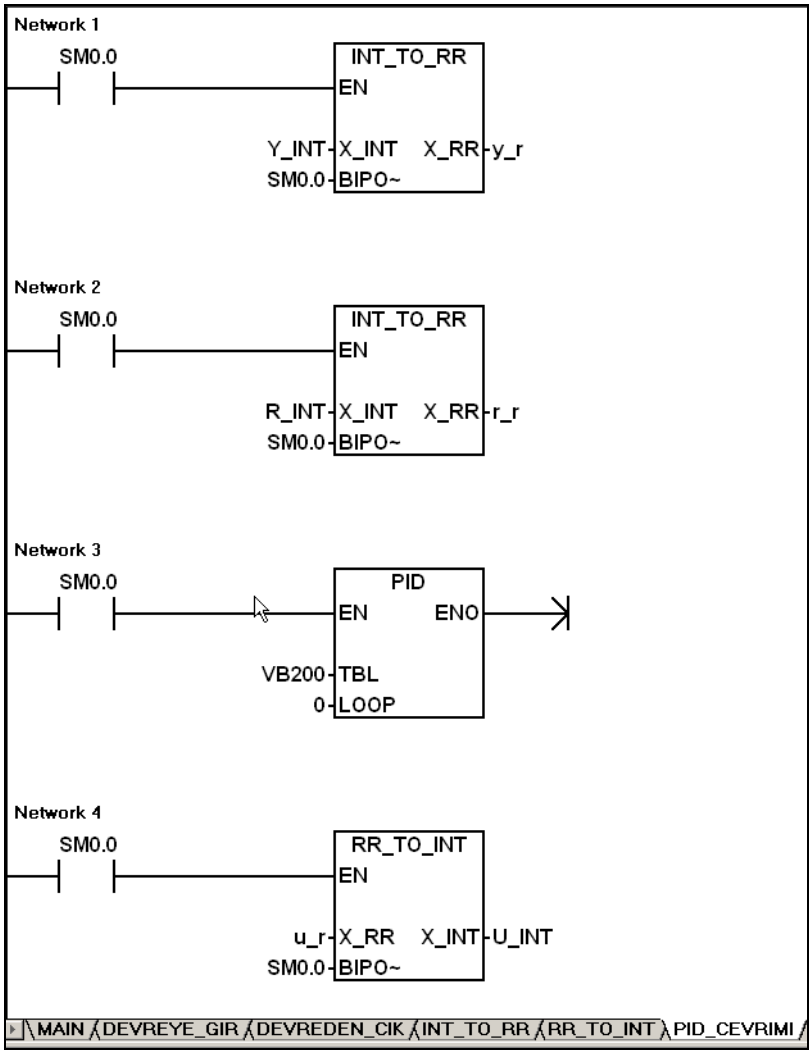

Network 4
LBL     1


Network 5
LD      SM0.0
MOVR    #X_RR, AC0
-R      #OFFSET, AC0
*R      #TD_REAL, AC0
ROUND   AC0, AC0
DTI     AC0, #X_INT
```

MAIN | DEVREYE_GIR | DEVREDEN_CIK | INT_TO_RR | RR_TO_INT | PID_CEVRIMI

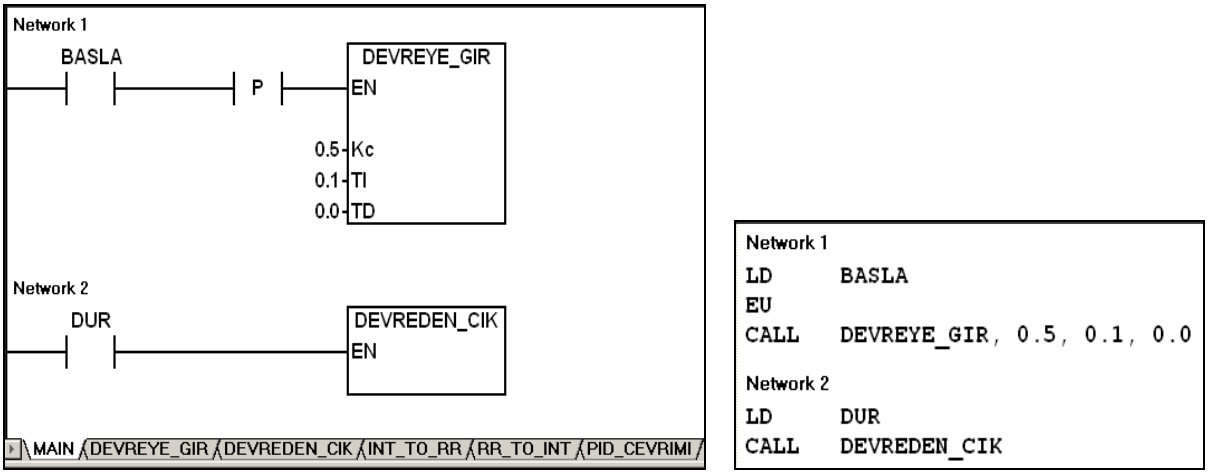
Şekil 8.27 Bağıl değeri işaretli tamsayıyı dönüştüren altprogam

PID komutunun yürütüleceği zamana bağlı kesmeli çalışan altprogram Şekil 8.28'deki gibi düzenlenebilir.



Şekil 8.28 PID komutunun yürütülmesine ilişkin kesme altprogramı

Son olarak ana program Şekil 8.29'daki gibi düzenlenerek bu uygulamaya ilişkin PLC programı için gerekli işlemler tamamlanır.

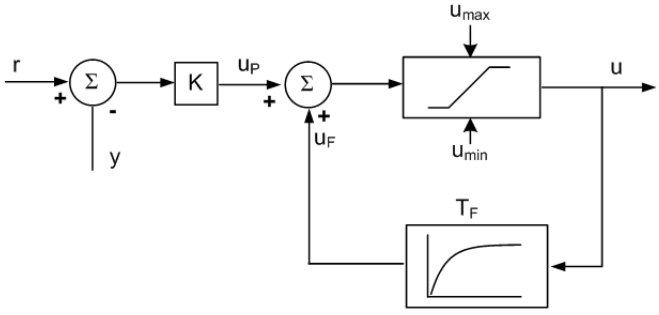


Şekil 8.29 Ana program

Bu uygulama, PID komutu yerine aynı işlevi sağlayan bir PI kontrol altprogramı yazılarak da gerçeklenebilir. Bu altprogramda giriş ve çıkışlar işaretli tamsayı olarak tanımlanırsa, bağıl değer işaretli tamsayı dönüştürülmesi işlemlerine gerek kalmaz.

8.3.3 BİR PI KONTROLÖR ALTPROGRAMI GERÇEKLEME

PI kontrol kuralını gerçeklemek üzere, integral sarması sorununu da kendiliğinden gideren Şekil 8.30'daki blok diyagramı kullanılabilir.



Şekil 8.30 PI kontrolör blok diyagramı

Burada r istenen değer, y kontrol edilen büyüklük değeri, $\alpha = \exp(-T/T_I)$ biçiminde tanımlanan bir katsayı, u sınırlandırılmış kontrol işareti büyüklüğüdür.

Bu blok diyagramında, geri besleme yolu üzerindeki alçak geçiren filtreye ilişkin s-tanım bölgesi transfer fonksiyonu

$$G_F(s) = \frac{U_F(s)}{U(s)} = \frac{1}{T_F s + 1} \tag{8.14}$$

olmak üzere, blok diyagramına ilişkin transfer fonksiyonu

$$G_{PI}(s) = \frac{U(s)}{E(s)} = K(1 + \frac{1}{T_F}s), \quad E(s)=R(s)-Y(s) \tag{8.15}$$

biçiminde yazılabilir. Bu ifadeye göre filtre zaman sabitinin PI kontrol kuralına ilişkin T_I integral zaman sabitine karşı düştüğü görülür.

Sayısal gerçekleme için fitreye ilişkin z-tanım bölgesi transfer fonksiyonundan yararlanılabilir. s-tanım bölgesi G(s) olarak verilen bir sistemin ayrık zaman modeline ilişkin z-tanım bölgesi transfer fonksiyonu

$$G_{ZOH}(s) = \frac{1 - e^{-sT}}{s}$$

$G_{ZOH}(s)$ sıfırıncı mertebeden tutucuya ilişkin transfer fonksiyonu ve T örnekleme zamanı olmak üzere

$$G(z) = Z\{G_{zoh}(s)G(s)\}, \quad z = e^{-sT}$$

ifadesinden elde edilebilir. Buna göre

$$G_F(z) = Z\{G_{zoh}(s)G_F(s)\} = Z\{\frac{1 - e^{-sT}}{s}\frac{1}{T_F s + 1}\} = \frac{(1 - e^{-sT})}{1} Z\{\frac{1/T_F}{s(s + 1/T_F)}\}$$

$$G_F(z) = \frac{(1 - e^{-sT})}{1} Z\{\frac{1}{s} - \frac{1}{s + 1/T_F}\}$$

$$G_F(z) = (\frac{z-1}{z})(\frac{z}{z-1} - \frac{z}{z - e^{-T/T_F}}), \qquad \alpha = e^{-T/T_F}$$

$$G_F(z) = (1 - \frac{z-1}{z-\alpha}) = \frac{1-\alpha}{z-\alpha} = \frac{(1-\alpha)z^{-1}}{1-\alpha z^{-1}}$$

$$G_F(z) = \frac{U_F(z)}{U(z)} = \frac{(1-\alpha)z^{-1}}{1-\alpha z^{-1}} \tag{8.16}$$

filtreye ilişkin z-tanım bölgesi transfer fonksiyonu bulunur ve bu ifadeden hareketle fark denklemi

$$u_F(k) = \alpha u_F(k-1) + (1-\alpha)u(k-1) \tag{8.17}$$

elde edilir.

Blok diyagramına göre,

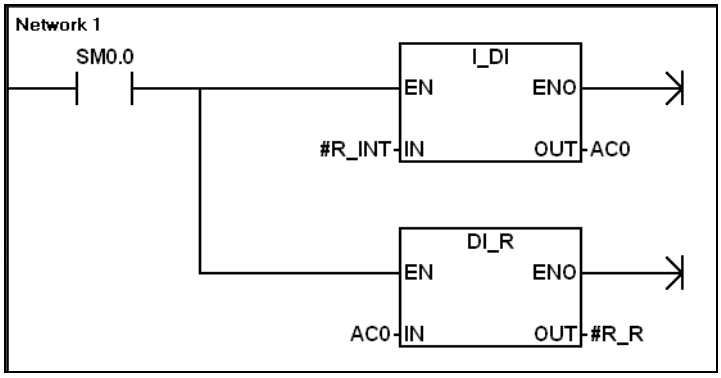
$$u_p(k) = K[r(k) - y(k)]$$

$$u_F(k) = \alpha u_F(k-1) + (1-\alpha)u(k-1) \tag{8.18}$$

$$u(k) = u_p(k) + u_F(k)$$

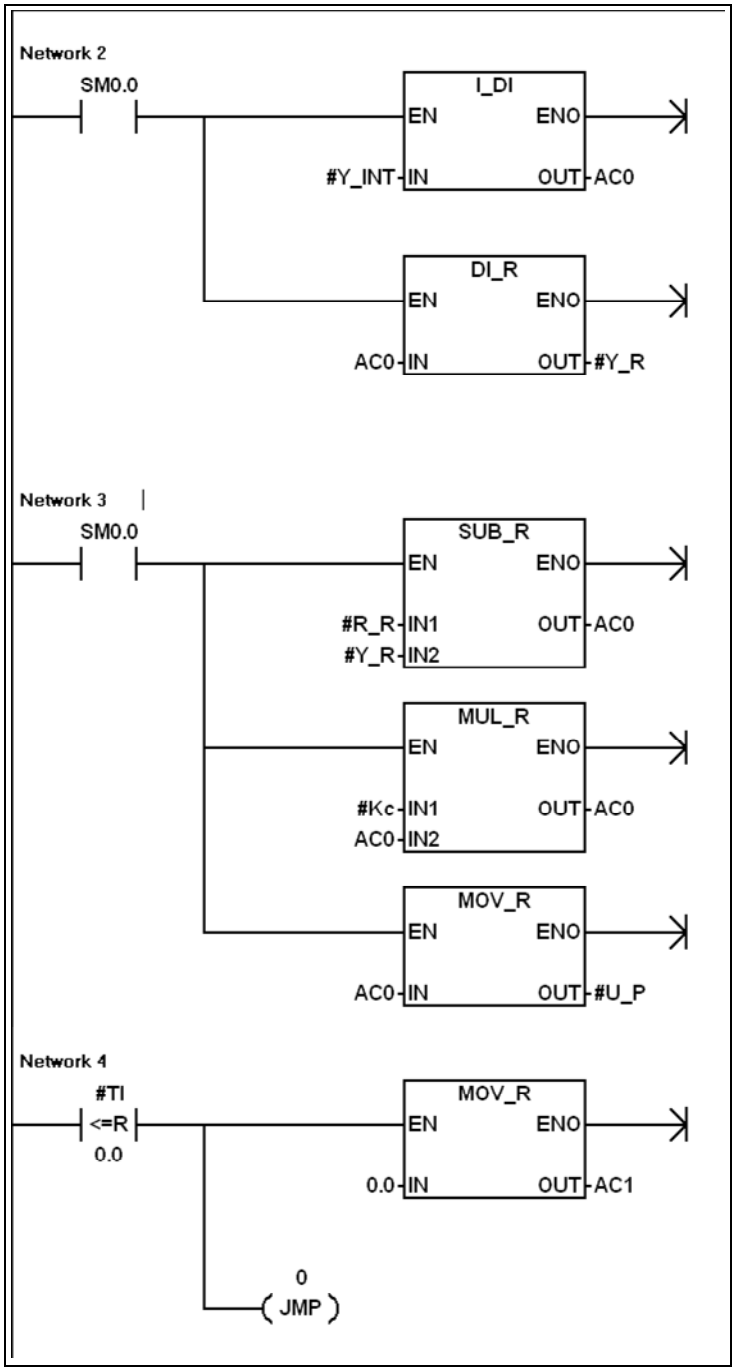
fark denklemleri yazılır ve programlanırsa PI kontrolör altprogramı gerçeklenir. Buna göre"PI" başlıklı altprograma ilişkin yerel (LOCAL) değişkenler ve program Şekil 8.31'deki gibi yazılabilir.

	Symbol	Var Type	Data Type	Comment
	EN	IN	BOOL	
LW0	R_INT	IN	INT	r: ISTENEN DEGER
LW2	Y_INT	IN	INT	y: SISTEM CIKISI
LD4	Kc	IN	REAL	ORANSAL KATSAYI
LD8	TI	IN	REAL	INTEGRAL KATSAYISI
LD12	T	IN	REAL	ORNEKLEME ZAMANI
LD16	Umin	IN	REAL	EN KUCUK DEGER
LD20	Umax	IN	REAL	EN BUYUK DEGER
LD24	U_1	IN_OUT	REAL	ONCEKI KONTROL ISARETI
LD28	UF_1	IN_OUT	REAL	ONCEKI FILTRE CIKISI
LW32	U_INT	OUT	INT	KONTROL ISARETI
LD34	R_R	TEMP	REAL	R REEL DEGERI
LD38	Y_R	TEMP	REAL	Y REEL DEGERI
LD42	U_P	TEMP	REAL	ORANSAL DEGER
LD46	U_R	TEMP	REAL	INTEGRAL DEGERI
LD50	ALFA	TEMP	REAL	EXP(-T/TI)




```
Network 1
LD      SM0.0
ITD     #R_INT, AC0
DTR     AC0, #R_R
```

Şekil 8.31 PI altprogramı (1/4)

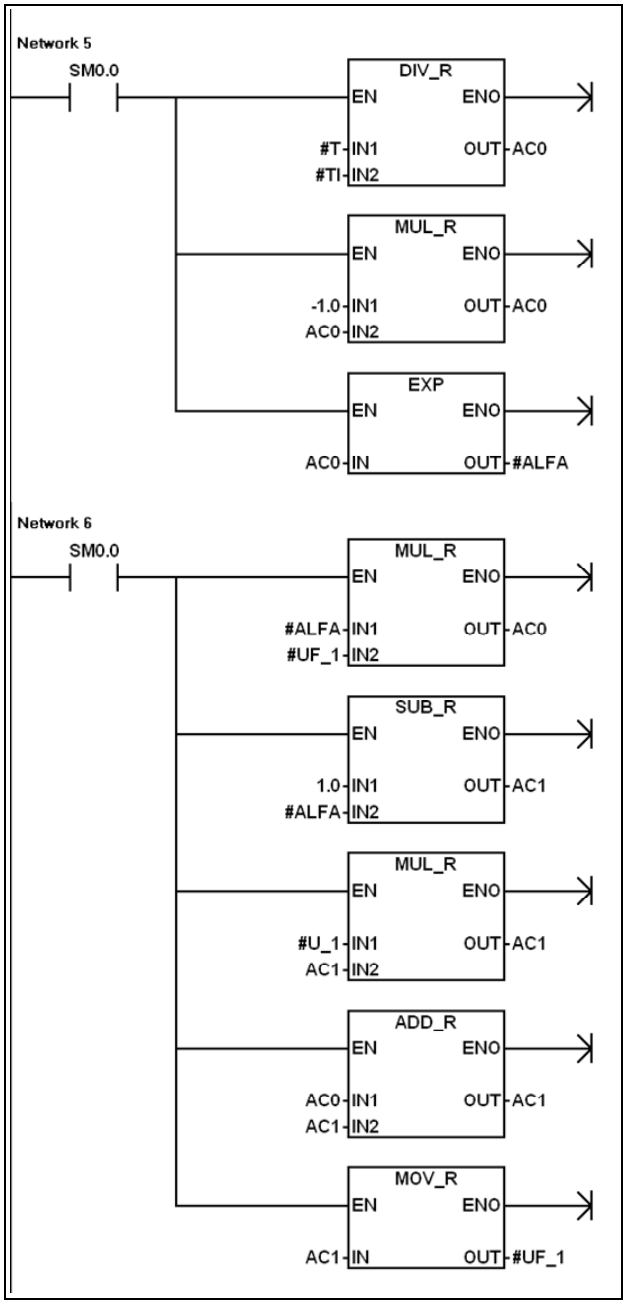



```
Network 2
LD      SM0.0
ITD     #Y_INT, AC0
DTR     AC0, #Y_R

Network 3
LD      SM0.0
MOVR    #R_R, AC0
-R      #Y_R, AC0
*R      #Kc, AC0
MOVR    AC0, #U_P

Network 4
LDR<=   #TI, 0.0
MOVR    0.0, AC1
JMP     0
```

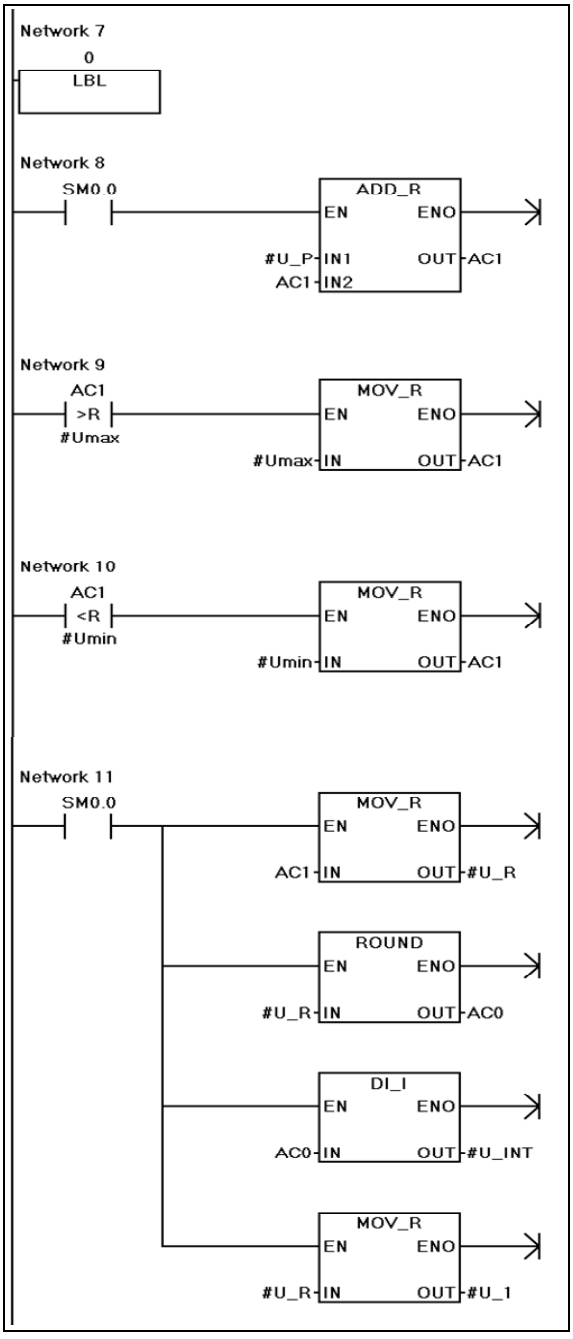
Şekil 8.31 PI altprogramı (2/4)




```
Network 5
LD      SM0.0
MOVR    #T, AC0
/R      #TI, AC0
*R      -1.0, AC0
EXP     AC0, #ALFA

Network 6
LD      SM0.0
MOVR    #ALFA, AC0
*R      #UF_1, AC0
MOVR    1.0, AC1
-R      #ALFA, AC1
*R      #U_1, AC1
+R      AC0, AC1
MOVR    AC1, #UF_1
```

Şekil 8.31 PI altprogramı (3/4)




```
Network 7
LBL     0

Network 8
LD      SM0.0
+R      #U_P, AC1

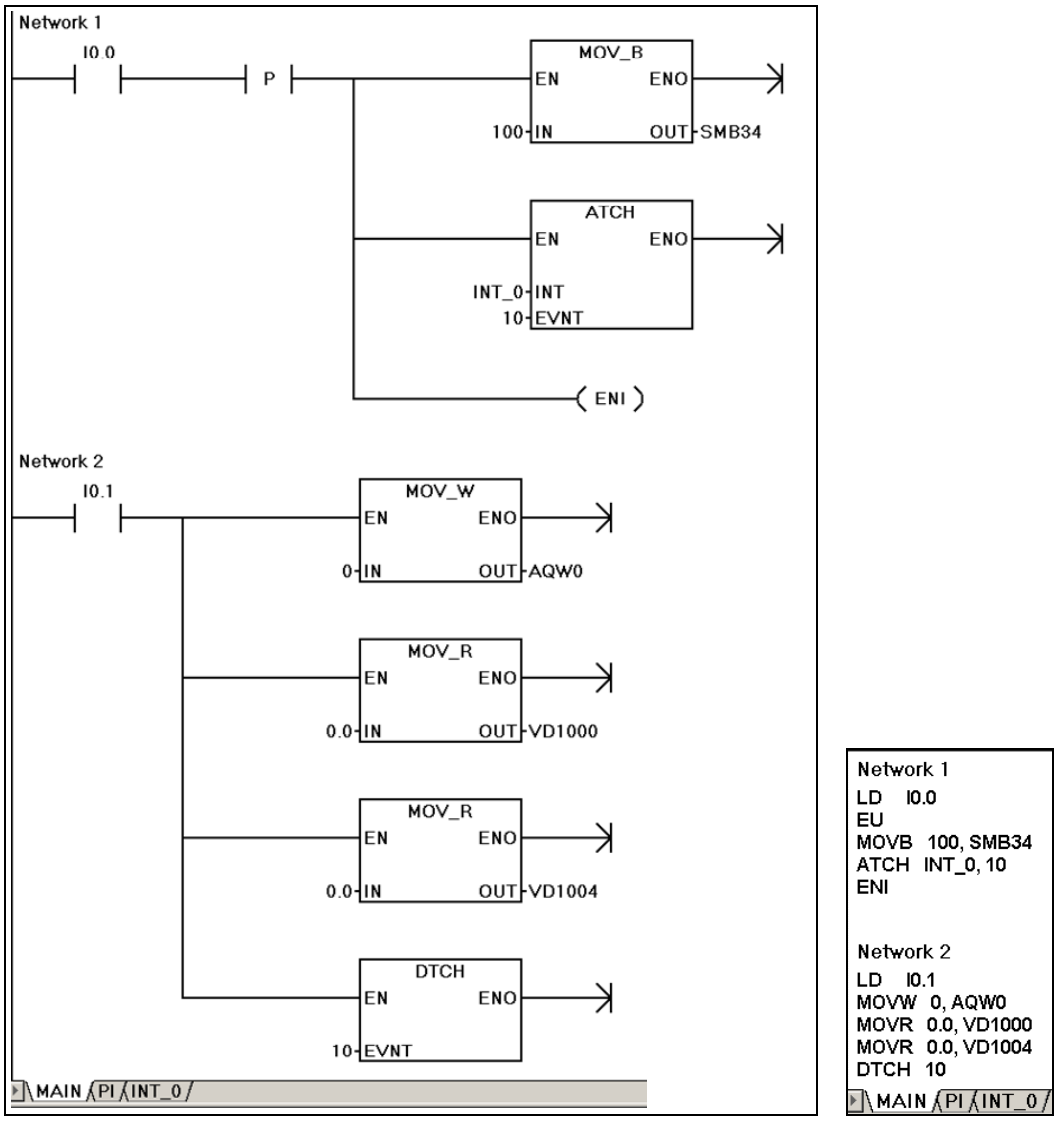
Network 9
LDR>    AC1, #Umax
MOVR    #Umax, AC1

Network 10
LDR<    AC1, #Umin
MOVR    #Umin, AC1

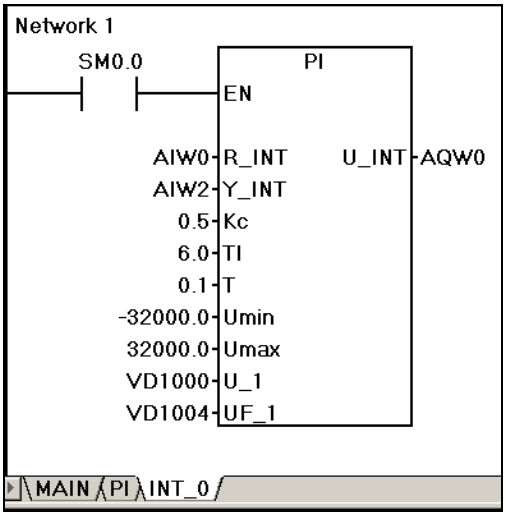
Network 11
LD      SM0.0
MOVR    AC1, #U_R
ROUND   #U_R, AC0
DTI     AC0, #U_INT
MOVR    #U_R, #U_1
```

Şekil 8.31 PI altprogramı (4/4)

PID komutu ile gerçeklenen Örnek 8.1'e ilişkin uygulama Şekil 8.31'deki PI altprogramından yararlanılarak Şekil 8.32'deki ana program ve Şekil 8.33'teki INT_0 zamana bağlı kesme altprogramı yazılarak gerçeklenebilir.



Şekil 8.32 Anaprogram



Şekil 8.33 Kesme altprogramı

8.3.4 BİRİNCİ MERTEBEDEN ALÇAK GEÇİREN FİLTRE GERÇEKLEME

Köşe frekansı ω_c olan birinci mertebeden alçak geçiren bir fitreye ilişkin s-tanım bölgesi transfer fonksiyonu

$$G_F(s) = \frac{Y(s)}{X(s)} = \frac{\omega_c}{s + \omega_c} \tag{8.19}$$

biçiminde, sıfırıncı mertebeden tutucu ile elde edilen ayrık zaman modeline ilişkin z-tanım bölgesi transfer fonksiyonu, T örnekleme zamanı olmak üzere

$$G_F(z) = Z\{G_{zoh}(s)G_F(s)\} = Z\{\frac{1-e^{-sT}}{s}\frac{\omega_c}{s+\omega_c}\} = \frac{1-z^{-1}}{1}Z\{\frac{\omega_c}{s(s+\omega_c)}\} = \frac{1-z^{-1}}{1}Z\{\frac{1}{s} - \frac{1}{s+\omega_c}\}$$

$$G_F(z) = \frac{z-1}{z}(\frac{z}{z-1} - \frac{z}{z-\exp(-\omega_c T)}) = 1 - \frac{z-1}{z-A_{WC}} = \frac{1-A_{WC}}{z-A_{WC}}, \quad A_{WC} := \exp(-\omega_c T)$$

$$G_F(z) = \frac{Y(z)}{X(z)} = \frac{(1-A_{WC})z^{-1}}{1-A_{WC}z^{-1}}$$

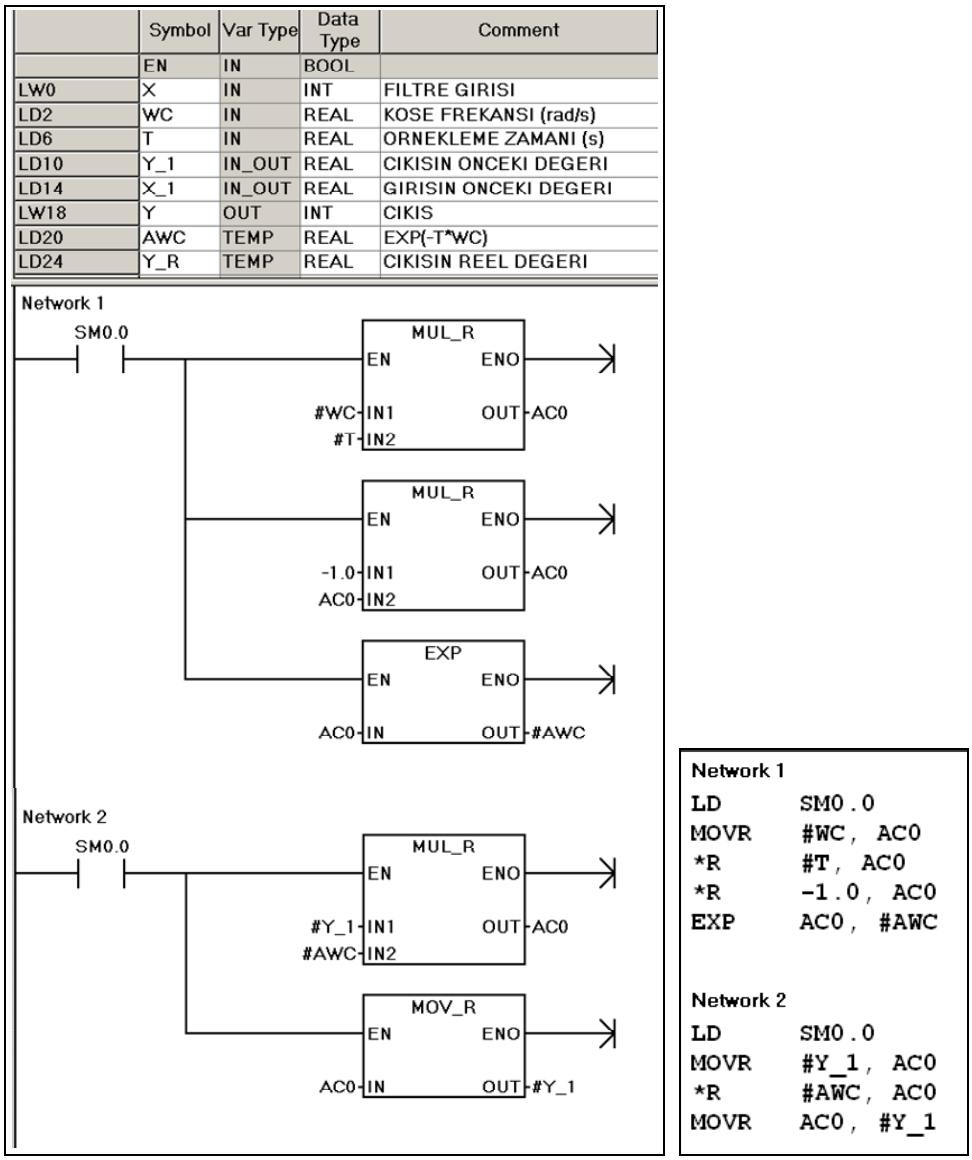
biçiminde ve fark denklemi

$$Y(z)(1-A_{WC}z^{-1}) = (1-A_{WC})z^{-1}X(z)$$

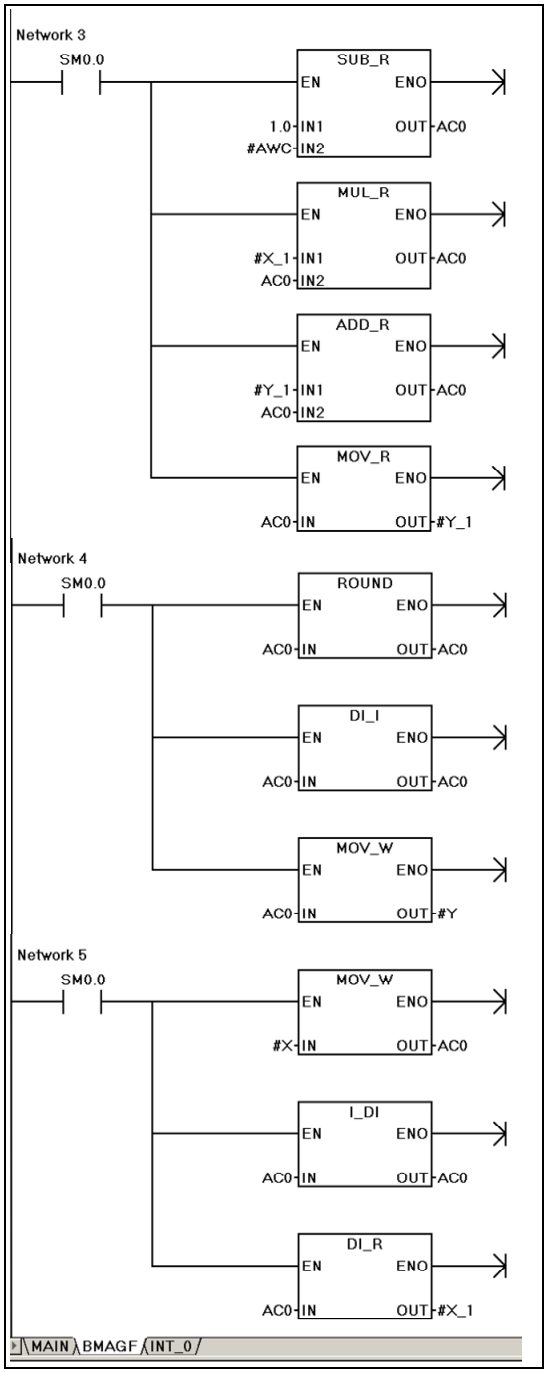
$$Y(z) = A_{WC} z^{-1} Y(z) + (1 - A_{WC}) z^{-1} X(z)$$

$$y(k) = A_{WC} y(k-1) + (1 - A_{WC}) x(k-1) \qquad (8.20)$$

biçiminde yazılabilir. Elde edilen fark denklemine ilişkin "BMAGF" başlıklı altprogram Şekil 8.34'te gösterildiği gibi programlanabilir.



Şekil 8.34 Alçak geçiren filtreye ilişkin altprogram (1/2)




```
Network 3
LD      SM0.0
MOVR    1.0, AC0
-R      #AWC, AC0
*R      #X_1, AC0
+R      #Y_1, AC0
MOVR    AC0, #Y_1

Network 4
LD      SM0.0
ROUND   AC0, AC0
DTI     AC0, AC0
MOVW    AC0, #Y


Network 5
LD      SM0.0
MOVW    #X, AC0
ITD     AC0, AC0
DTR     AC0, #X_1
```

Şekil 8.34 Alçak geçiren filtreye ilişkin altprogram (2/2)

Örnek 8.2 Köşe frekansı 0.2 rad/s olarak verilen bir alçak geçiren filtre ile AIW0 adresinden okunan bir analog işarete ilişkin verilerin süzülerek VW300 adresine yazılmasını sağlayan programı yazınız.

Birinci mertebeden alçak geçiren bir filtrenin köşe frekansındaki genliği, köşe frekansının 10 katı bir değerde %10 değerine (20dB zayıflama), 100 katı bir değerde %1 değerine (40dB zayıflama) düşer. İşaret genliğinin, 40dB zayıfladığı frekansın en az 2 katı örnekleme frekansı olarak seçilirse, örnekleme frekansı ω_s

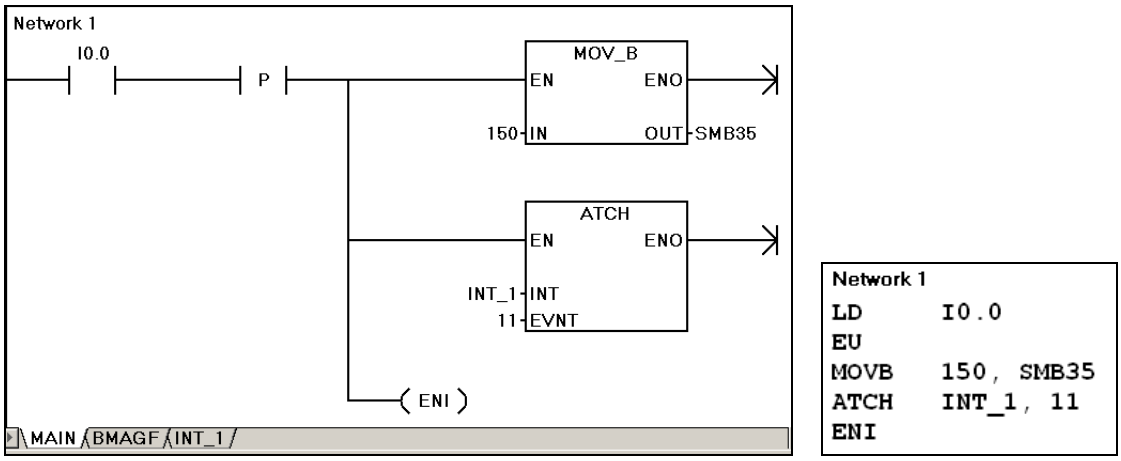
$$\omega_s \geq 2 \times 100 \times \omega_c = 2 \times 100 \times 0.2 = 40 \text{rad/s}$$

bulunur. Buna göre örnekleme zamanı

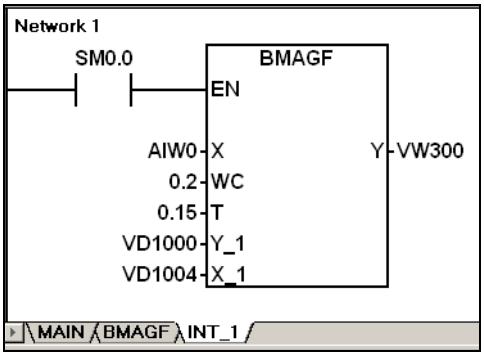
$$T \leq \frac{2\pi}{\omega_s} = \frac{2\pi}{40} = 0.157 \text{s}$$

koşulunu sağlayan bir değer seçilebilir.

Örnekleme zamanının 150ms alınması durumunda, ana program Şekil 8.35 ve kesme altprogramı Şekil 8.36'daki gibi düzenlenebilir.



Şekil 8.35 Alçak geçiren filtre uygulamasına ilişkin anaprogram



Şekil 8.36 Alçak geçiren filtre uygulamasına ilişkin kesme altprogramı

8.4 SAYISAL BENZETİM

Sayısal benzetim, bir sistemin dinamik davranışına ilişkin sayısal verilerin üretilmesi olarak tanımlanabilir. Bir sisteme ilişkin sayısal benzetimden yararlanarak, bir kontrol yönteminin amaca uygunluğu ya da sistem davranışı üzerindeki etkisi yazılım aşamasında değerlendirilebilir. Bu bölümde, bir sisteme ilişkin benzetim yazılımının gerçek zamanlı olarak PLC'de gerçeklenmesi ve bu programların kullanımına ilişkin uygulama örnekleri verilecektir.

Bir sistemin dinamik davranışını tanımlayan matematiksel ifadelerin sayısal çözümünde, birçok sayısal çözümleme yöntemi kullanılır. Sayısal kontrol sistemlerinin modellenmesinde kullanılan z-tanım bölgesi transfer fonksiyonlarından elde edilen fark denklemleri de bu amaçla kullanılabilir.

Genel olarak s-tanım bölgesi transfer fonksiyonu G(s) olan bir sisteme ilişkin z-tanım bölgesi transfer fonksiyonu G(z), $G_{ZOH}(s)$ sıfırıncı mertebeden tutucuya ilişkin transfer fonksiyonu olmak üzere,

$$G(z) = Z\left\{G_{ZOH}(s)G(s)\right\} \tag{8.21}$$

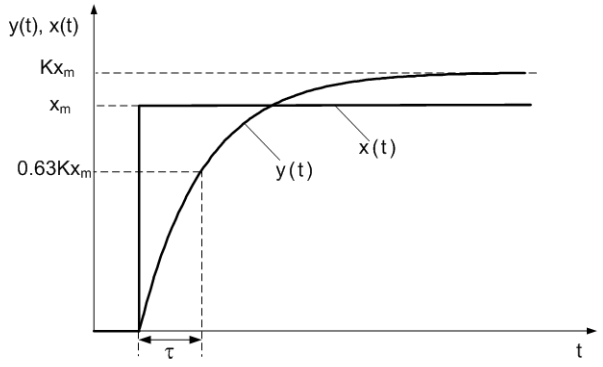
ifadesinden elde edilebilir. Bu ifadeden elde edilen fark denklemi programlanarak sayısal benzetim gerçeklenebilir.

8.4.1 BİRİNCİ MERTEBEDEN SİSTEMLER

Birinci mertebeden sistemler s-tanım bölgesi transfer fonksiyonu

$$G(s) = \frac{Y(s)}{X(s)} = \frac{K}{\tau s + 1} \tag{8.22}$$

biçiminde verilen sistemler olarak tanımlanır. Burada K sistem kazancı ve τ sistem zaman sabitidir. Bu büyüklükler, Şekil 8.37'de gösterilen sistemin basamak girişe yanıtından elde edilebilir.



Şekil 8.37 Birinci mertebeden sistemin basamak girişe yanıtı

Bu sistemin sıfırıncı mertebeden tutucu ile elde edilen ayrık zaman modeline ilişkin z-tanım bölgesi transfer fonksiyonu, T örnekleme zamanı olmak üzere

$$G(z) = Z\{G_{zoh}(s)G(s)\} = Z\{\frac{1-e^{-sT}}{s}\frac{K}{\tau s+1}\} = \frac{1-z^{-1}}{1}Z\{\frac{K/\tau}{s(s+1/\tau)}\} = K\frac{1-z^{-1}}{1}Z\{\frac{1}{s}-\frac{1}{s+1/\tau}\}$$

$$G(z) = K\frac{z-1}{z}(\frac{z}{z-1}-\frac{z}{z-\exp(-T/\tau)}) = K(1-\frac{z-1}{z-A}) = \frac{K(1-A)}{z-A},\quad A := \exp(-T/\tau)$$

$$G(z) = \frac{Y(z)}{X(z)} = \frac{K(1-A)z^{-1}}{1-Az^{-1}}$$

biçiminde ve bu ifadeden, fark denklemi

$$Y(z)(1-Az^{-1}) = K(1-A)z^{-1}X(z)$$

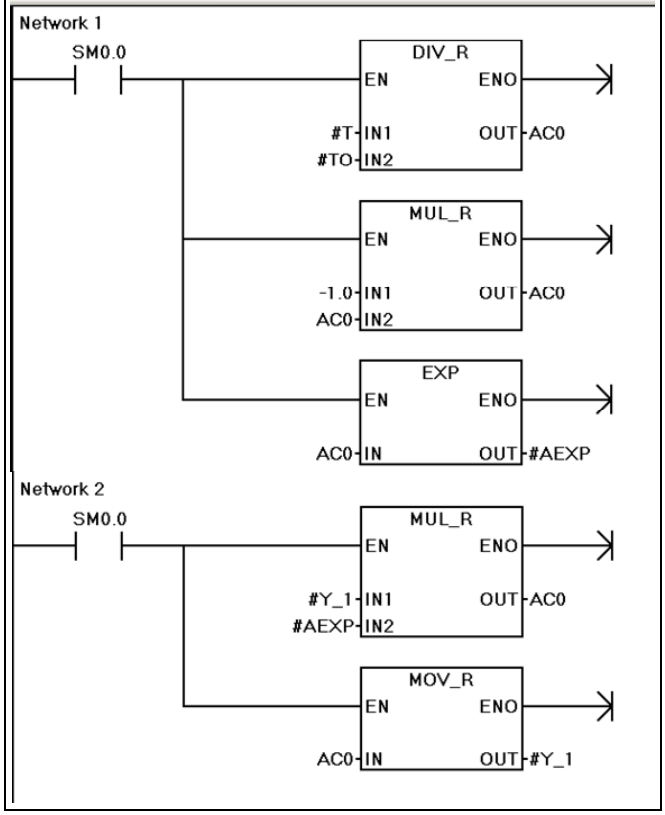
$$Y(z) = Az^{-1}Y(z) + K(1-A)z^{-1}X(z)$$

$$y(k) = Ay(k-1) + K(1-A)x(k-1) \qquad (8.23)$$

elde edilir. Burada A=exp(-T/τ) olarak tanımlanan bir katsayıdır. Bu fark denklemi programlanarak birinci mertebeden bir sisteme ilişkin PLC benzetim yazılımı elde edilir.

Şekil 8.38'de, birinci mertebeden bir sistemin, (8.23) fark denklemi kullanılarak elde edilen"BDS" başlıklı bir benzetim altprogramı verilmiştir.

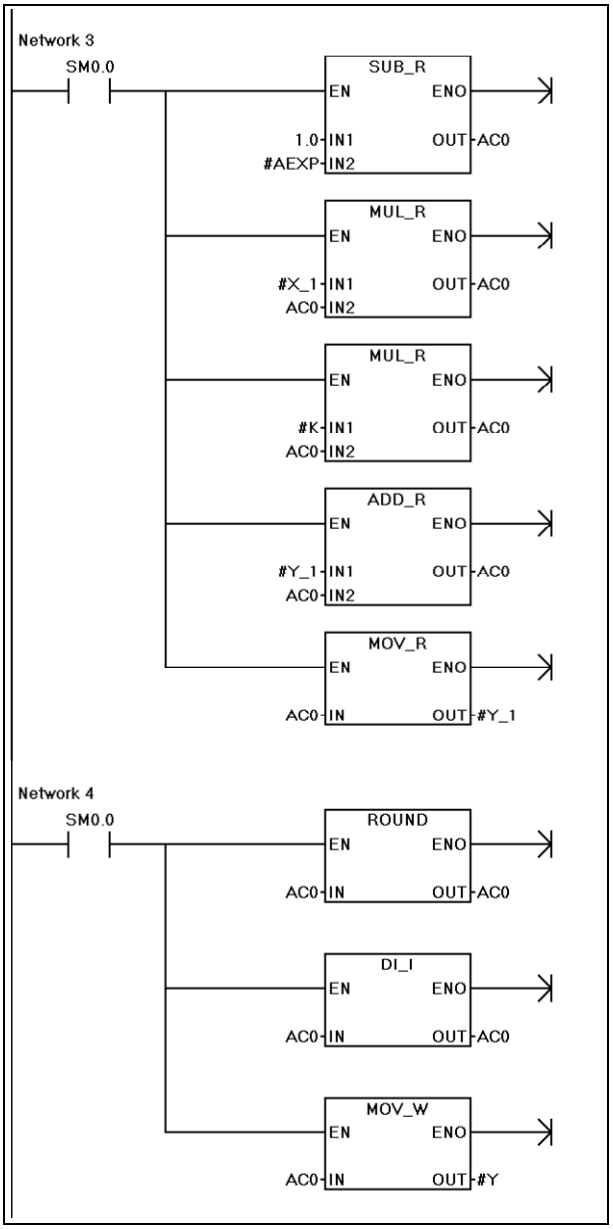
	Symbol	Var Type	Data Type	Comment
	EN	IN	BOOL	
LW0	X	IN	INT	SISTEM GIRISI
LD2	K	IN	REAL	KAZANC
LD6	TO	IN	REAL	ZAMAN SABITI [s]
LD10	T	IN	REAL	ORNEKLEME ZAMANI (s)
LD14	Y_1	IN_OUT	REAL	CIKISIN ONCEKI DEGERI
LD18	X_1	IN_OUT	REAL	GIRISIN ONCEKI DEGERI
LW22	Y	OUT	INT	CIKIS
LD24	AEXP	TEMP	REAL	EXP(-T/TO)




```
Network 1
LD      SM0.0
MOVR    #T, AC0
/R      #TO, AC0
*R      -1.0, AC0
EXP     AC0, #AEXP

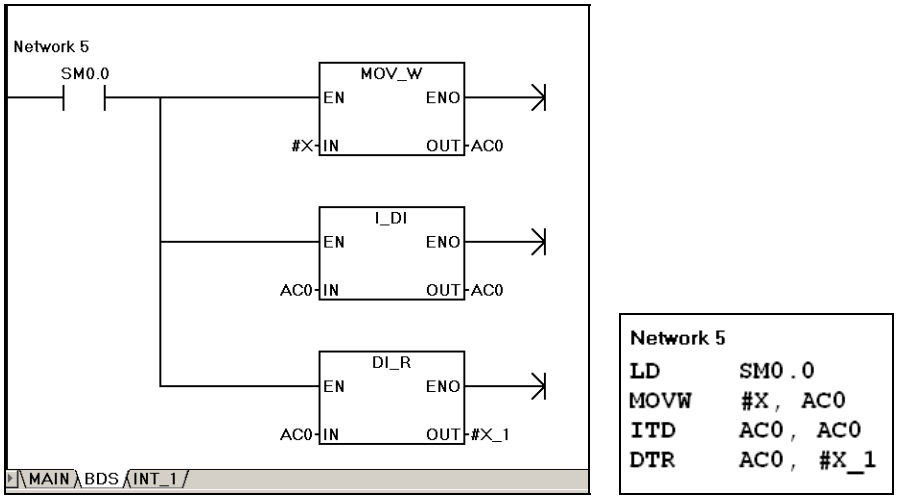
Network 2
LD      SM0.0
MOVR    #Y_1, AC0
*R      #AEXP, AC0
MOVR    AC0, #Y_1
```

Şekil 8.38 Birinci mertebeden bir sisteme ilişkin benzetim altprogramı (1/3)




```
Network 3
LD      SM0.0
MOVR    1.0, AC0
-R      #AEXP, AC0
*R      #X_1, AC0
*R      #K, AC0
+R      #Y_1, AC0
MOVR    AC0, #Y_1
Network 4
LD      SM0.0
ROUND   AC0, AC0
DTI     AC0, AC0
MOVW    AC0, #Y
```

Şekil 8.38 Birinci mertebeden bir sisteme ilişkin benzetim altprogramı (2/3)



Şekil 8.38 Birinci mertebeden bir sisteme ilişkin benzetim altprogramı (3/3)

Örnek 8.3 s-tanım bölgesi transfer fonksiyonu

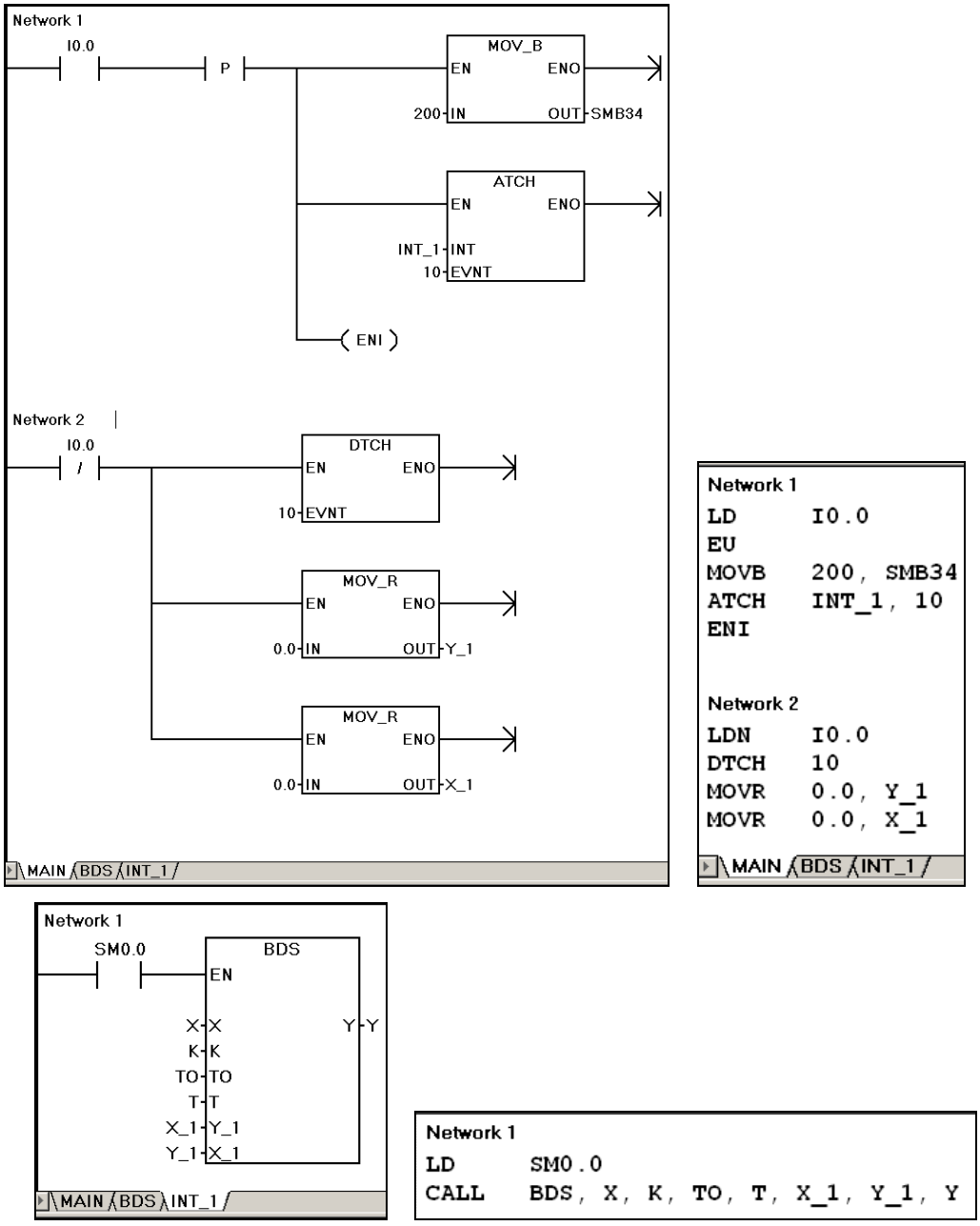
$$G(s) = \frac{1.5}{5s+1}$$

biçiminde verilen bir sistemin, PLC benzetim programını yazınız.

Bu sistemin 200ms örnekleme zamanı ile benzetimini gerçeklemek için Şekil 8.39'da gösterilen sembol tablosu oluşturulur, Şekil 8.40'ta gösterilen "MAIN" başlıklı ana program ve "INT_1" başlıklı zamana bağlı kesme altprogramı ve Şekil 8.41'de gösterilen "BDS" başlıklı benzetim altprogramı yazılabilir.

Symbol	Address	Comment
X	VW400	GIRIS
Y	VW402	CIKIS
X_1	VD800	GIRISIN ONCEKI DEGERI
Y_1	VD804	CIKISIN ONCEKI DEGERI
K	VD808	KAZANC
TO	VD812	ZAMAN SABITI
T	VD816	ORNEKLEME ZAMANI

Şekil 8.39 Benzetim programına ilişkin sembol tablosu



Şekil 8.40 Benzetim ana programı ve kesme altprogramı

	Symbol	Var Type	Data Type	Comment
LW0	X	IN	INT	SISTEM GIRISI
LD2	K	IN	REAL	KAZANC
LD6	TO	IN	REAL	ZAMAN SABITI [s]
LD10	T	IN	REAL	ORNEKLEME ZAMANI (s)
LD14	Y_1	IN_OUT	REAL	CIKISIN ONCEKI DEGERI
LD18	X_1	IN_OUT	REAL	GIRISIN ONCEKI DEGERI
LW22	Y	OUT	INT	CIKIS
LD24	AEXP	TEMP	REAL	EXP(-T/TO)
		TEMP		

```
Network 1
LD      SM0.0
MOVR    #T, AC0
/R      #TO, AC0
*R      -1.0, AC0
EXP     AC0, #AEXP

Network 2
LD      SM0.0
MOVR    #Y_1, AC0
*R      #AEXP, AC0
MOVR    AC0, #Y_1

Network 3
LD      SM0.0
MOVR    1.0, AC0
-R      #AEXP, AC0
*R      #X_1, AC0
*R      #K, AC0
+R      #Y_1, AC0
MOVR    AC0, #Y_1

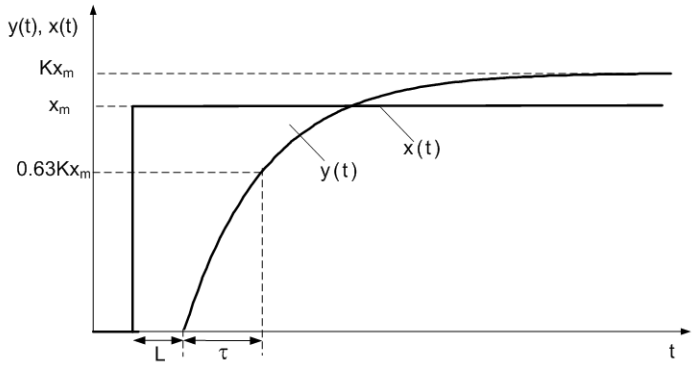
Network 4
LD      SM0.0
ROUND   AC0, AC0
DTI     AC0, AC0
MOVW    AC0, #Y

Network 5
LD      SM0.0
MOVW    #X, AC0
ITD     AC0, AC0
DTR     AC0, #X_1
```

MAIN | BDS | INT_1

Şekil 8.41 Birinci mertebe sistem için benzetim altprogramı

Birçok endüstriyel sürecin basamak girişe yanıtı, ölçme veya sürücü düzeneklerinden kaynaklanan gecikmeler nedeniyle Şekil 8.42'de gösterildiği gibi değişir. Bu tür sistemler, birinci mertebeden ölü zamanlı sistem olarak tanımlanır.



Şekil 8.42 Birinci mertebeden ölü zamanlı sistemin basamak girişe yanıtı

Birinci mertebeden ölü zamanlı sistemlere ilişkin s-tanım bölgesi transfer fonksiyonu

$$G(s)=\frac{Y(s)}{X(s)}=\frac{K}{\tau s+1}e^{-sL} \tag{8.24}$$

biçiminde verilir. Burada K sistem kazancı τ sistem zaman sabiti ve L ölü zaman değeridir. Bu sistemin sıfırıncı mertebeden tutucu ile elde edilen ayrık zaman modeline ilişkin z-tanım bölgesi transfer fonksiyonu, T örnekleme zamanı olmak üzere

$$G(z)=Z\{G_{zoh}(s)G(s)\}=Z\{\frac{1-e^{-sT}}{s}\frac{K/\tau}{s+1/\tau}e^{-sL}\}=\frac{(1-e^{-sT})e^{-sL}}{1}Z\{\frac{K/\tau}{s(s+1/\tau)}\}$$

ifadesinden

$$G(z)=\frac{Y(z)}{X(z)}=\frac{K(1-A)}{z-A}z^{-d},\quad A=e^{-T/\tau}\ ,\ d=L/T$$

biçiminde elde edilir. Bu ifade düzenlenirse

$$Y(z)=\frac{K(1-A)}{z-A}z^{-d}X(z)=\frac{K(1-A)}{1-Az^{-1}}z^{-d-1}X(z)$$

$$Y(z)=Az^{-1}Y(z)+K(1-A)z^{-d-1}X(z)$$

$$y(k)=Ay(k-1)+K(1-A)x(k-1-d) \tag{8.25}$$

fark denklemi elde edilir. Bu ifade programlanarak birinci mertebeden ölü zamanlı bir sisteme ilişkin benzetim yazılımı elde edilir.

Örnek 8.4 s tanım bölgesi transfer fonksiyonu

$$G(s) = \frac{Y(s)}{X(s)} = \frac{1.25}{8s+1} e^{-0.4s}$$

biçiminde verilen sisteme ilişkin bir benzetim programı yazınız.

Örnekleme zamanı T=0.2s seçilirse

$$A = e^{-T/\tau} = e^{-0.2/8} = 0.9753\,,\ d = L/T = 0.4/0.2 = 2$$

olur ve bu sisteme ilişkin (2.25) ifadesi ile verilen fark denklemi

$$y(k) = 0.9753y(k-1) + 1.25(1-0.9753)x(k-1-2)$$

$$y(k) = 0.9753y(k-1) + 0.03086x(k-3)$$

elde edilir. Bu uygulamaya ilişkin sembol tablosu ve ana program Şekil 8.42, fark denklemine ilişkin "BMOZS" başlıklı benzetim altprogramı ve zaman bağlı kesme altprogramı Şekil 8.43'teki gibi yazılabilir.

Symbol	Address	Comment
X	VW100	GIRIS
X_1	VD200	GIRISIN (k-1) DEGERI
X_2	VD204	GIRISIN (k-2) DEGERI
X_3	VD208	GIRISIN (k-3) DEGERI
Y_1	VD212	CIKISIN (k-1) DEGERI
Y	VW102	CIKIS

```
Network 1
LD   I0.0
EU
MOVB  200, SMB34
ATCH  INT_1, 10
ENI


Network 2
LDN  I0.0
DTCH  10
MOVR  0.0, Y_1
MOVR  0.0, X_1
MOVR  0.0, X_2
MOVR  0.0, X_3
```

MAIN | BMOZS | INT_1

Şekil 8.42 Sembol tablosu ve ana program

	Symbol	Var Type	Data Type	Comment
LW0	X	IN	INT	SISTEM GIRISI
LD2	X_1	IN_OUT	REAL	GIRISIN (k-1) DEGERI
LD6	X_2	IN_OUT	REAL	GIRISIN (k-2) DEGERI
LD10	X_3	IN_OUT	REAL	GIRISIN (k-3) DEGERI
LD14	Y_1	IN_OUT	REAL	CIKISIN (k-1) DEGERI
LW18	Y	OUT	INT	CIKIS
		TEMP		

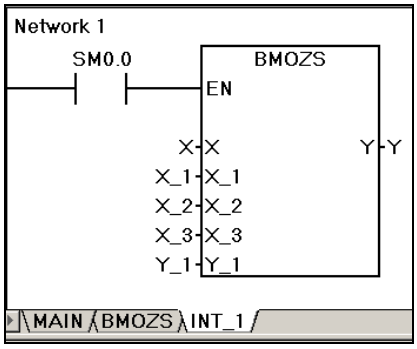
```
Network 1      y(k)=0.9753y(k-1)+0.03086x(k-3)
LD    SM0.0
MOVR  #Y_1, AC0
*R    0.9753, AC0

Network 2
LD    SM0.0
MOVR  #X_3, AC1
*R    0.03086, AC1
+R    AC1, AC0
MOVR  AC0, #Y_1

Network 3
LD    SM0.0
MOVR  #X_2, #X_3
MOVR  #X_1, #X_2
ITD   #X, AC0
DTR   AC0, #X_1

Network 4
LD    SM0.0
MOVR  #Y_1, AC0
ROUND AC0, AC0
DTI   AC0, #Y
```

MAIN | BMOZS | INT_1



Şekil 8.43 "BMOZS" başlıklı benzetim altprogramı ve kesme altprogramı

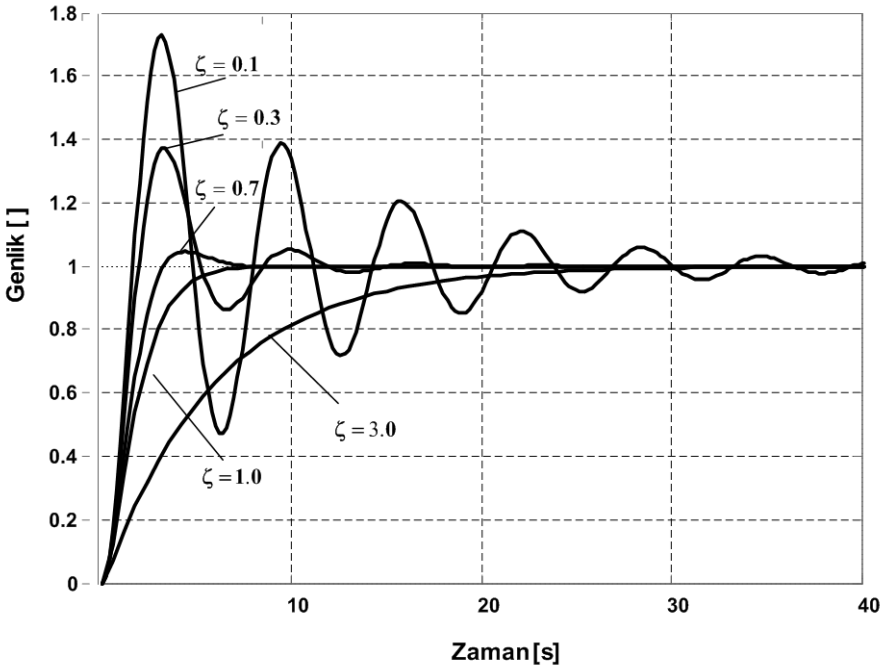
8.4.2 İKİNCİ MERTEBEDEN SİSTEMLER

İkinci mertebeden sistemler, s-tanım bölgesi transfer fonksiyonu

$$G(s) = \frac{Y(s)}{X(s)} = \frac{K\omega_n^2}{s^2 + 2\zeta\omega_n s + \omega_n^2} \tag{8.26}$$

biçiminde verilen sistemler olarak tanımlanır. Burada ζ sönüm oranı, ω_n doğal frekans olarak tanımlanır. Bu tür sistemler ζ parametresine göre az sönümlü, kritik sönümlü, aşırı sönümlü ve sönümsüz sistemler olarak sınıflandırılır. Sönüm oranı $0 < \zeta < 1$ değerinde olan sistemler az sönümlü,

sönüm oranı ζ=1 değerinde olan sistemler kritik sönümlü, sönüm oranı ζ>1 değerinde olan sistemler aşırı sönümlü ve sönüm oranı ζ=0 değerinde olan sistemler sönümsüz sistem olarak tanımlanır. Şekil 8.44'te sönüm oranının değişik değerlerinde, basamak girişe sistem yanıtları gösterilmiştir.



Şekil 8.44 Doğal frekansı $\omega_n = 1 rad/s$ olan ikinci mertebe sistemlerin sönüm oranına göre birim basamak girişine yanıtı

8.4.2.1 AZ SÖNÜMLÜ SİSTEMLER

Basamak girişe yanıtı Şekil 8.45'te gösterilen bir süreç, ikinci mertebeden az sönümlü bir sistem olarak tanımlanır ve sistem yanıtından alınan Kx_m, t_p ve y_m değerleri kullanılarak K, ζ ve ω_n parametreleri bulunabilir.

Aşım

$$c = \frac{y_m - Kx_m}{Kx_m}$$

olarak tanımlanır ve bu büyüklüğe bağlı olarak

$$\delta = \ln(c), \qquad 0 < Kx_m < y_m < 2Kx_m$$

sönüm oranı

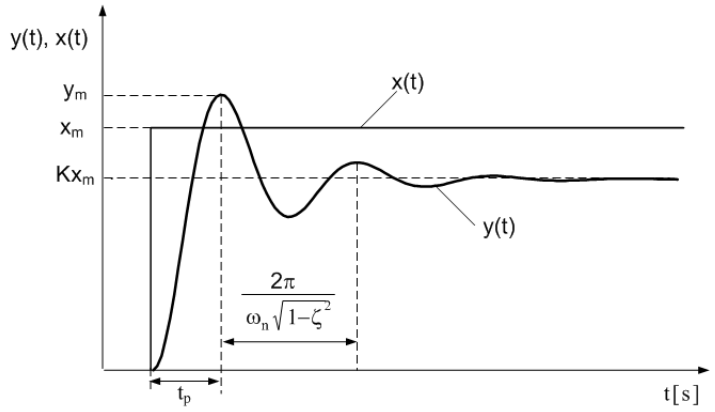
$$\zeta = \sqrt{\frac{\delta^2}{\pi^2 + \delta^2}} \tag{8.27}$$

ifadesi kullanılarak bulunur.

Doğal frekans

$$\omega_n = \frac{\pi}{t_p\sqrt{1-\zeta^2}}, \quad 0 < \zeta < 1, \quad \omega = \omega_n\sqrt{1-\zeta^2} \tag{8.28}$$

ilişkisinden, K kazanç büyüklüğü ise çıkış işaretinin son değerinin basamak girişin genliğine oranından elde edilir.



Şekil 8.45 Az sönümlü sistemin basamak girişe yanıtı

İkinci mertebeden az sönümlü bir sistemin ayrık zaman modeline ilişkin z-tanım bölgesi transfer fonksiyonu

$$G(z) = Z\{G_{ZOH}(s)G(s)\} = Z\{\frac{1-e^{-sT}}{s}\frac{K\omega_n^2}{s(s^2+2\zeta\omega_n s+\omega_n^2)}\} = (1-z^{-1})Z\{\frac{K\omega_n^2}{s(s^2+2\zeta\omega_n s+\omega_n^2)}\}$$

ifadesinin Z-dönüşümü alınarak

$$G(z) = K\frac{(1-a\cos\theta-ab\sin\theta)z+a^2-a\cos\theta+ab\sin\theta}{z^2-(2a\cos\theta)z+a^2} \tag{8.28}$$

biçiminde elde edilir. Burada a, b ve θ büyüklükleri

$$a = e^{-\zeta\omega_n T},\ b = \frac{\zeta}{\sqrt{1-\zeta^2}},\ \theta = \omega T,\ \omega = \omega_n\sqrt{1-\zeta^2} \tag{8.29}$$

olarak tanımlanır.

(8.28) ifadesi

$$G(z) = K(1 - a\cos\theta - ab\sin\theta)\frac{z + \dfrac{a^2 - a\cos\theta + ab\sin\theta}{1 - a\cos\theta - ab\sin\theta}}{z^2 - (2a\cos\theta)z + a^2}$$

biçiminde yazılır ve

$$\begin{aligned} b_0 &= \frac{a^2 - a\cos\theta + ab\sin\theta}{1 - a\cos\theta - ab\sin\theta} \\ K_Z &= K(1 - a\cos\theta - ab\sin\theta) \\ a_1 &= -2a\cos\theta \\ a_0 &= a^2 \end{aligned} \tag{8.30}$$

tanımları yapılırsa

$$G(z) = \frac{K_Z(z + b_0)}{z^2 + a_1 z + a_0} \tag{8.31}$$

transfer fonksiyonu elde edilir.

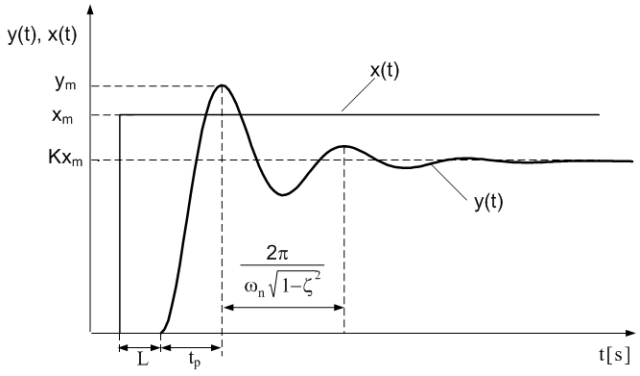
Basamak girişe yanıtı Şekil 8.46'da gösterilen az sönümlü ölü zamanlı sistemlere ilişkin s-tanım bölgesi transfer fonksiyonu

$$G(s) = \frac{Y(s)}{X(s)} = \frac{K\omega_n^2}{s^2 + 2\zeta\omega_n s + \omega_n^2}e^{-sL}$$

biçiminde ifade edilir. Ölü zaman değerinin örnekleme zamanına oranı $d = L/T$ olan bir sistem için z-tanım bölgesi transfer fonksiyonu, (8.31) ifadesinin elde edilmesindeki gibi benzer işlemler yapılır ve (8.30) da verilen parametre tanımları kullanılırsa

$$G(z) = \frac{K_Z(z + b_0)}{z^2 + a_1 z + a_0}z^{-d} \tag{8.32}$$

biçimindeki z-tanım bölgesi transfer fonksiyonu elde edilir.



Şekil 8.46 İkinci mertebeden ölü zamanlı az sönümlü sistemin basamak girişe yanıtı

İkinci mertebeden az sönümlü sisteme ilişkin fark denklemi, (8.32) ifadesinden yararlanılarak

$$G(z)=\frac{Y(z)}{X(z)}=\frac{K_Z(z+b_0)}{z^2+a_1z+a_0}z^{-d}=\frac{K_Z(z^{-1}+b_0z^{-2})z^{-d}}{1+a_1z^{-1}+a_0z^{-2}}=\frac{K_Z(z^{-d-1}+b_0z^{-d-2})}{1+a_1z^{-1}+a_0z^{-2}}$$

$$(1+a_1z^{-1}+a_0z^{-2})Y(z)=K_Z(z^{-d-1}+b_0z^{-d-2})X(z)$$

$$Y(z)=-a_1z^{-1}Y(z)-a_0z^{-2}Y(z)+K_Zz^{-d-1}X(z)+b_0z^{-d-2}X(z)$$

$$y(k)=-a_1y(k-1)-a_0y(k-2)+K_Zx(k-d-1)+b_0x(k-d-2) \qquad (8.33)$$

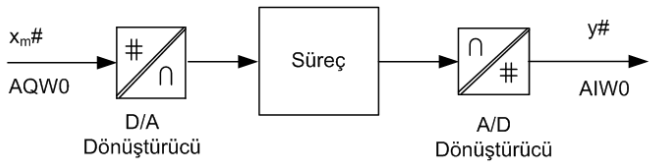
biçiminde bulunur. Bu ifadede d=0 yapılırsa ölü zamansız sistem için geçerli olan fark denklemi elde edilir.

Örnek 8.5 Analog giriş ve çıkış bağlantı biçimi Şekil 8.47'de gösterilen bir sürece, PLC AQW0 analog çıkış noktası üzerinden genliği x_m=10000 olan bir basamak giriş uygulandığında, AIW0 analog giriş adresinden okunan verilerin zamana göre değişimi Şekil 8.48'de verilmiştir. Bu sisteme ilişkin bir benzetim altprogramı yazınız.

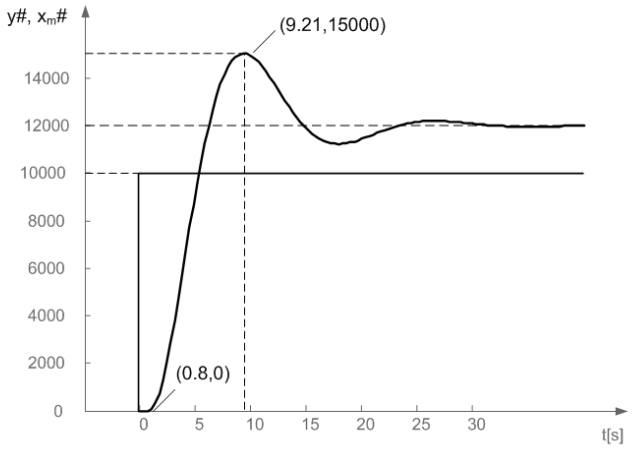
Sistemin basamak girişe yanıtından elde edilen Kx_m , t_p ve y_m değerlerinden (8.27) ve (8.28) ifadeleri kullanılarak sistemin s-tanım bölgesi transfer fonksiyonu bulunur. Eğriden okunan y_m=15000 ve Kx_m=12000 değerlerine göre aşım

$$c=\frac{y_m-Kx_m}{Kx_m}=\frac{15000-12000}{12000}=0.25$$

bulunur.



Şekil 8.47 Süreç analog giriş-çıkış bağlantı şeması



Şekil 8.48 Sürecin basamak girişe yanıtı

$$\delta = \ln(c) = \ln(0.25) = -1.3863$$

sönüm oranı

$$\zeta = \sqrt{\frac{\delta^2}{\pi^2 + \delta^2}} = \sqrt{\frac{(-1.3863)^2}{\pi^2 + (-1.3863)^2}} = 0.4037$$

t_p=9.21-0.8=8.41s olduğundan doğal frekans

$$\omega_n = \frac{\pi}{t_p\sqrt{1-\zeta^2}} = \frac{\pi}{8.41\sqrt{1-(0.4037)^2}} = 0.4083 \text{rad/s}$$

bulunur. Sistemin açık çevrim kazancı

$$K = \frac{Kx_m}{x_m} = \frac{12000}{10000} = 1.2$$

ve ölü zaman değeri L=0.8s olduğundan, sisteme ilişkin s-tanım bölgesi transfer fonksiyonu

$$G(s) = \frac{Y(s)}{X(s)} = \frac{K\omega_n^2}{s^2 + 2\zeta\omega_n s + \omega_n^2} e^{-sL} = \frac{1.2 \times (0.4083)^2}{s^2 + 2 \times (0.4037) \times (0.4083)s + (0.4083)^2} e^{-s0.8}$$

$$G(s) = \frac{0.2}{s^2 + 0.3297s + 0.1667} e^{-s0.8}$$

elde edilir.

Örnekleme frekansı ω_s, ω_n doğal frekansının 20 katı seçilirse

$$\omega_s = 20\omega_n = 20 \times 0.4083 = 8.1661 \text{rad/s}$$

bulunur ve örnekleme zamanı

$$T = \frac{2\pi}{\omega_s} = 0.7694s \sim 0.8s$$

seçilebilir.

Buna göre sisteme ilişkin (8.33) fark denkleminin katsayıları

$$a = e^{-\zeta\omega_n T} = \exp(-0.4037 \times 0.4083 \times 0.8) = 0.8753$$

$$b = \frac{\zeta}{\sqrt{1-\zeta^2}} = \frac{0.4037}{\sqrt{1-(0.4037)^2}} = 0.4413$$

$$\omega = \omega_n \sqrt{1-\zeta^2} = 0.4083\sqrt{1-(0.4037)^2} = 0.3736$$

$$\theta = \omega T = 0.3736 \times 0.8 = 0.2988$$

$$b_0 = \frac{a^2 - a\cos\theta + ab\sin\theta}{1 - a\cos\theta - ab\sin\theta} = \frac{(0.8753)^2 - 0.8753 \times \cos(0.2988) + 0.8753 \times 0.4413 \times \sin(0.2988)}{1 - 0.8753 \times \cos(0.2988) - 0.8753 \times 0.4413 \times \sin(0.2988)} = 0.9156$$

$$K_Z = K(1 - a\cos\theta - ab\sin\theta) = 1.2 \times (1 - 0.8753 \times \cos(0.2988) - 0.8753 \times 0.4413 \times \sin(0.2988)) = 0.0582$$

$$a_1 = -2a\cos\theta = -2 \times 0.8753 \times \cos(0.2988) = -1.6752$$

$$a_0 = a^2 = (0.8753)^2 = 0.7682$$

bulunur. Ölü zaman değerinin örnekleme zamanına oranı d=L/T=0.8/0.8=1 olduğundan (8.33) fark denklemi

$$y(k) = -a_1 y(k-1) - a_0 y(k-2) + K_Z x(k-2) + b_0 x(k-3)$$

olur. Bu fark denkleminin gerçeklenmesine ilişkin "IMS" başlıklı altprogram Şekil 8.49'daki gibi yazılabilir.

	Symbol	Var Type	Data Type	Comment
LW0	X	IN	INT	
LD2	b0	IN	REAL	
LD6	a1	IN	REAL	
LD10	a0	IN	REAL	
LD14	Kz	IN	REAL	
LD18	Ymin	IN	REAL	
LD22	Ymax	IN	REAL	
LD26	Y_1	IN_OUT	REAL	
LD30	Y_2	IN_OUT	REAL	
LD34	X_1	IN_OUT	REAL	
LD38	X_2	IN_OUT	REAL	
LD42	X_3	IN_OUT	REAL	
LW46	Y	OUT	INT	

```
Network 1      y(k)= - a1*y(k-1) - a0*y(k-2)+Kz*[x(k-2)+b0*x(k-3)]
LD      SM0.0
MOVR    #Y_1, AC0
*R      #a1, AC0
MOVR    #Y_2, AC1
*R      #a0, AC1
+R      AC1, AC0
*R      -1.0, AC0

Network 2
LD      SM0.0
MOVR    #X_3, AC1
*R      #b0, AC1
+R      #X_2, AC1
*R      #Kz, AC1

Network 3
LD      SM0.0
MOVR    #Y_1, #Y_2
+R      AC0, AC1
```

MAIN | IMS | INT_0

```
Network 4
LDR>=   AC1, #Ymax
MOVR    #Ymax, AC1

Network 5
LDR<=   AC1, #Ymin
MOVR    #Ymin, AC1

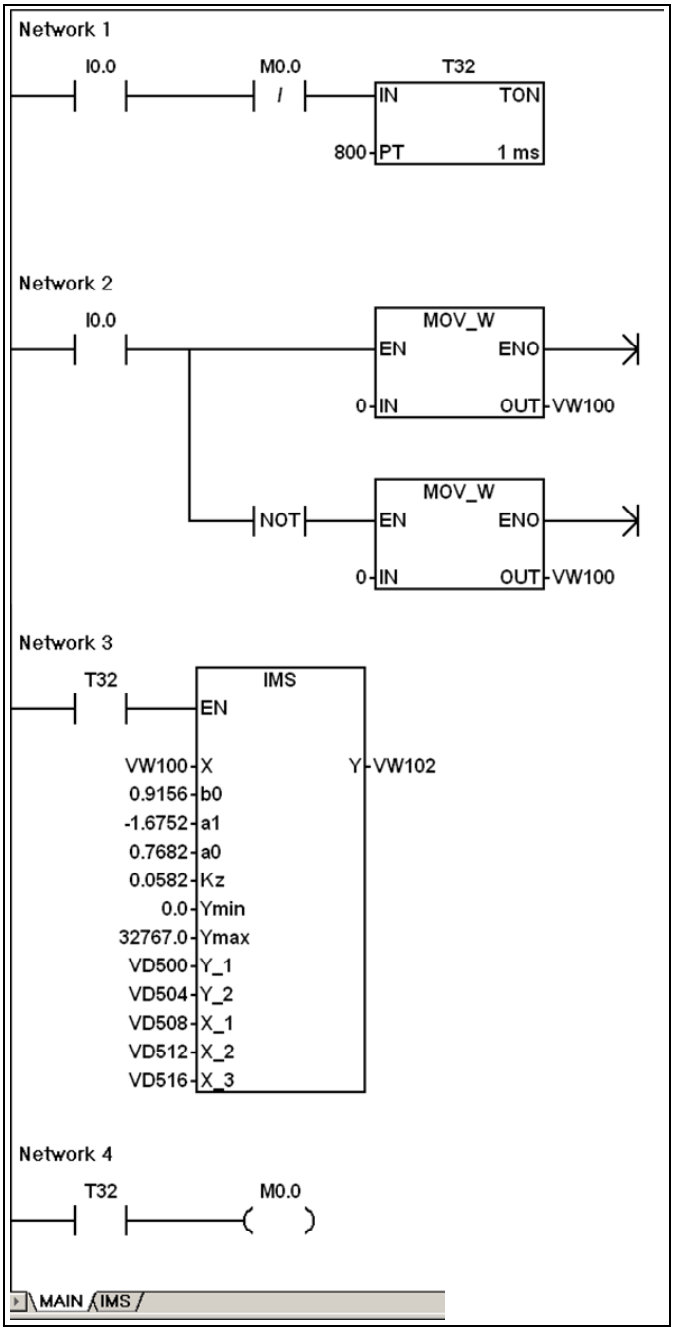
Network 6
LD      SM0.0
MOVR    AC1, #Y_1
MOVR    #X_2, #X_3
MOVR    #X_1, #X_2
ITD     #X, AC0
DTR     AC0, #X_1

Network 7
LD      SM0.0
MOVR    #Y_1, AC0
ROUND   AC0, AC0
DTI     AC0, #Y
```

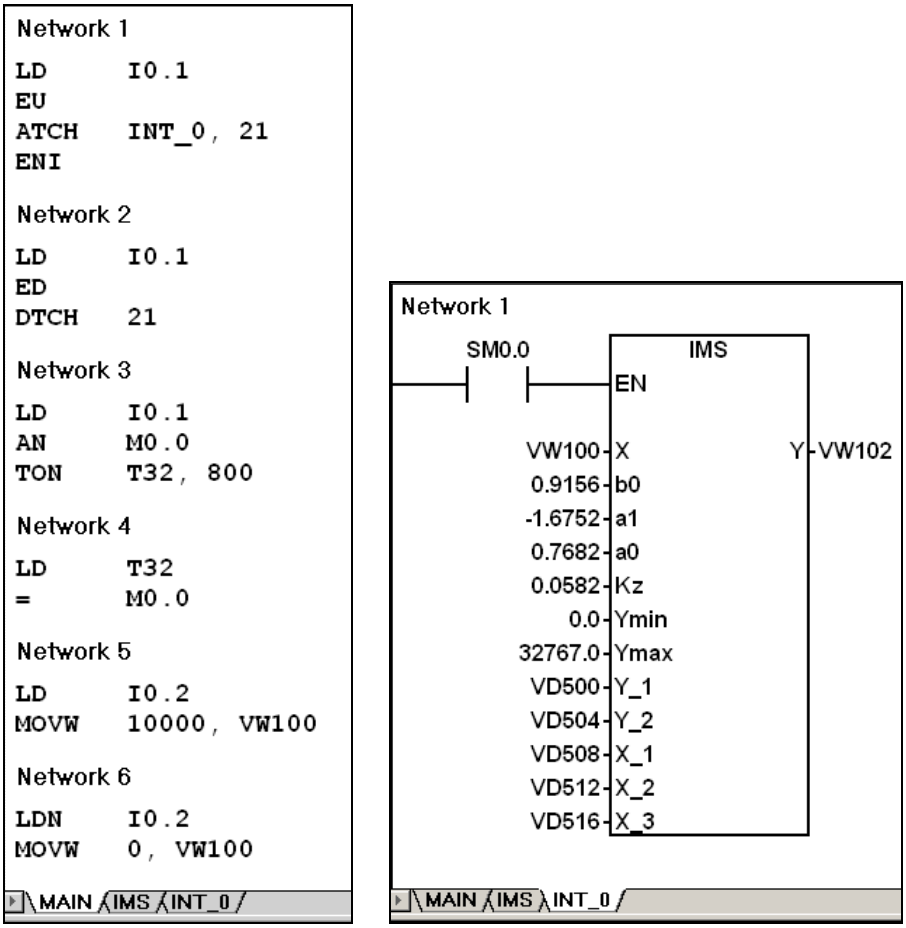
Şekil 8.49 İkinci mertebe sisteme ilişkin benzetim altprogram

Sürece ilişkin benzetimin gerçeklenmesi için Şekil 8.49'da verilen altprogramın 800ms aralıklarla işletilmesini sağlamak gerekir. Bunun için çeşitli çözümler verilebilir.

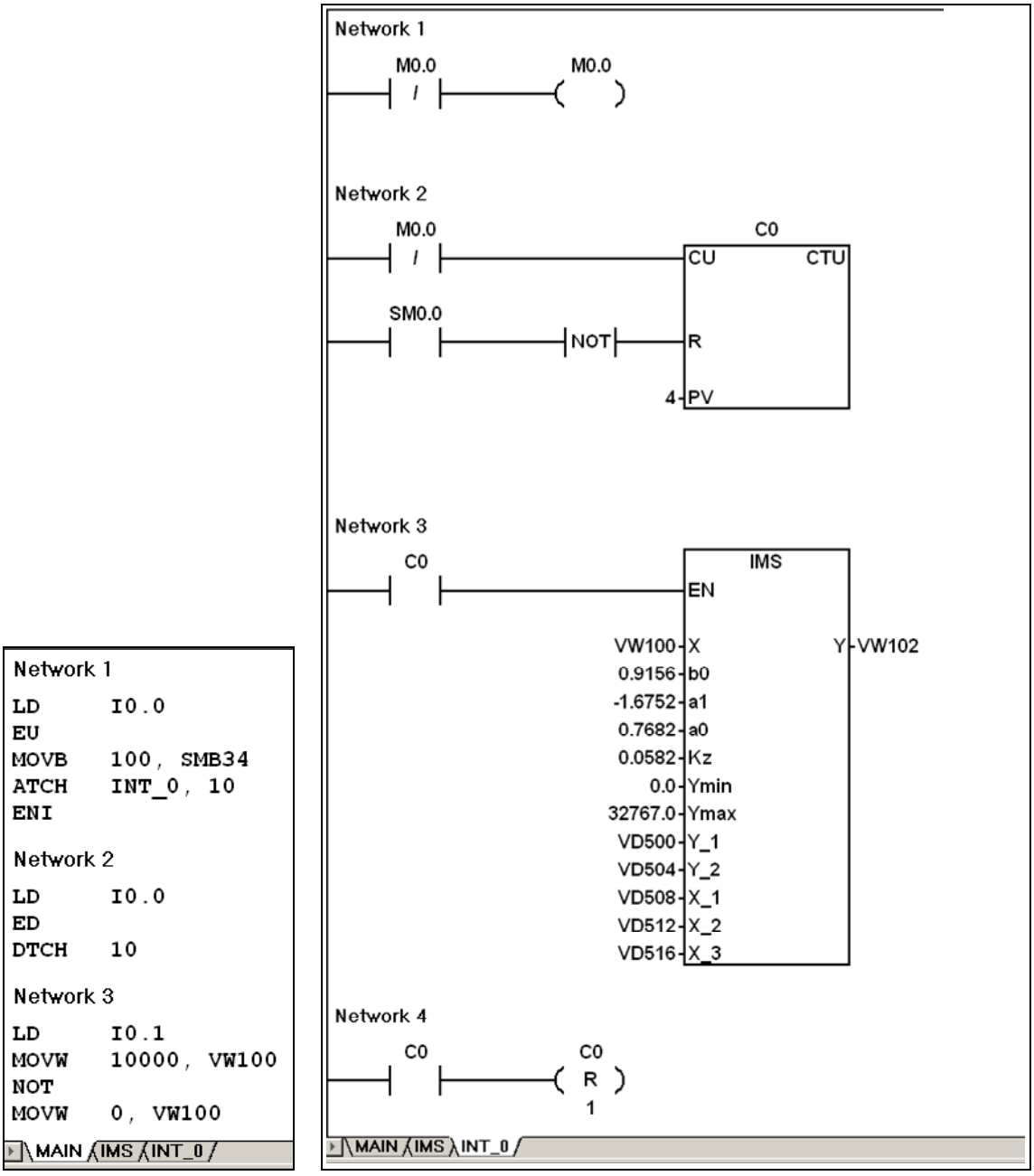
Ana programa Şekil 8.50'deki gibi düzenlenerek benzetim altprogramının 800ms aralıklarla işletilmesi sağlanabilir. Ancak, bu programa göre işletilen benzetim altprogramının yürütülme sıklığı, PLC tarama süresi ve zamanlayıcı komutunun işlendiği ana bağlı olarak, 800ms'den uzun gecikmelere yol açar. Belirli bir örnekleme zamanı için elde edilen fark denkleminin gerçek zamanda tam olarak gerçeklenebilmesi için bu fark denklemine ilişkin benzetim programının, örnekleme süresinin tam katı değerlerinde yürütülmesi gerekir. Bunu sağlamak için kesmeli çalışma özelliğinden yararlanılır. Bölüm 7.4'te değinilen zamanlayıcı kaynaklı kesmeli çalışma veya Bölüm 7.5'te değinilen zamana bağlı kesmeli çalışma özellikleri kullanılarak bu tür uygulamalar gerçeklenebilir. Şekil 8.51'de zamanlayıcı kaynaklı kesmeli çalışma ve Şekil 8.52'de zamana bağlı kesmeli çalışma ile gerçeklenen program örnekleri verilmiştir.



Şekil 8.50 İkinci mertebe sisteme ilişkin benzetim programının ana programdan işletilmesi



Şekil 8.51 Zamanlayıcı kaynaklı kesmeli çalışma ile gerçeklenen program örneği



Şekil 8.52 Zamana bağlı kesmeli çalışma ile gerçeklenen program örneği

8.4.2.2 KRİTİK SÖNÜMLÜ SİSTEMLER

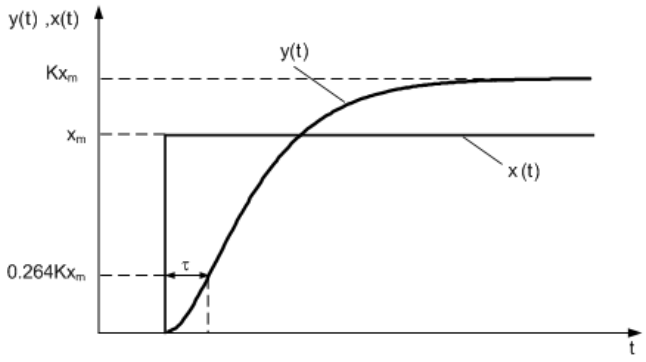
Basamak girişe yanıtı Şekil 8.53'te gösterilen bir süreç, ikinci mertebeden kritik sönümlü sistem olarak tanımlanabilir. Kritik sönümlü sisteme ilişkin transfer fonksiyonu ikinci mertebeden sistemlere ilişkin (8.26) ifadesinde $\zeta = 1$ yazılarak

$$G(s) = \frac{K\omega_n^2}{s^2 + 2\zeta\omega_n s + \omega_n^2} = \frac{K\omega_n^2}{s^2 + 2\omega_n s + \omega_n^2} = \frac{K\omega_n^2}{(s + \omega_n)^2} \tag{8.34}$$

biçiminde veya zaman sabiti ve kazanç parametrelerine bağlı olarak

$$G(s) = \frac{K}{(\tau s + 1)^2} \tag{8.35}$$

biçiminde tanımlanabilir. Burada K ve τ sabitleri sistem yanıtından elde edilebilir. K sabiti çıkış son değerinin x_m giriş işaretine oranından, τ sabiti çıkış işaretinin $0.264Kx_m$ değerine ulaştığı süreden bulunur.



Şekil 8.53 İkinci mertebeden kritik sönümlü sistemin basamak girişe yanıtı

Transfer fonksiyonu (8.35) ifadesi ile verilen ikinci mertebeden kritik sönümlü sistemin ayrık zaman modeline ilişkin z-tanım bölgesi transfer fonksiyonu

$$G(z) = Z\{G_{ZOH}(s)G(s)\} = Z\{\frac{1 - e^{-sT}}{s} K \frac{1}{(\tau s + 1)^2}\}$$

ifadesinden hareketle, az sönümlü sisteme ilişkin (8.31 ifadesinin elde edilmesinde yapılan benzer işlemlerle

$$G(z) = K_Z \frac{z + b_0}{z^2 + a_1 z + a_0}$$

biçiminde elde edilebilir. Burada,

$$A = \exp(-T/\tau)$$

$$K_Z = K(1 - A - AT/\tau)$$

$$b_0 = \frac{A(A - 1 + T/\tau)}{(1 - A - AT/\tau)}$$

$$a_1 = -2A$$

$$a_0 = A^2$$

olarak tanımlanan sabitlerdir. Bu ifade kritik sönümlü ölü zamanlı sistemler için

$$G(z) = K_Z \frac{z + b_0}{z^2 + a_1 z + a_0} z^{-d}$$

biçiminde yazılabilir. Bu ifadeye ilişkin (8.33) fark denklemi K_z, b_0, a_0, a_1 katsayılarına göre düzenlenerek sistemin benzetimi gerçeklenebilir.

Örnek 8.6 Genliği x_m=5000 olan basamak girişe yanıtı Şekil 8.54'te verilen ikinci mertebeden kritik sönümlü sisteme ilişkin bir benzetim programını yazınız.

Sistemin basamak girişe yanıtından K=1200/1000=1.2, sabiti çıkış işaretinin $0.264Kx_m$ değerine ulaştığı $y(L + \tau) = 0.264 \times Kx_m = 0.264 \times 1200 = 316.8$ değerinden bulunur.

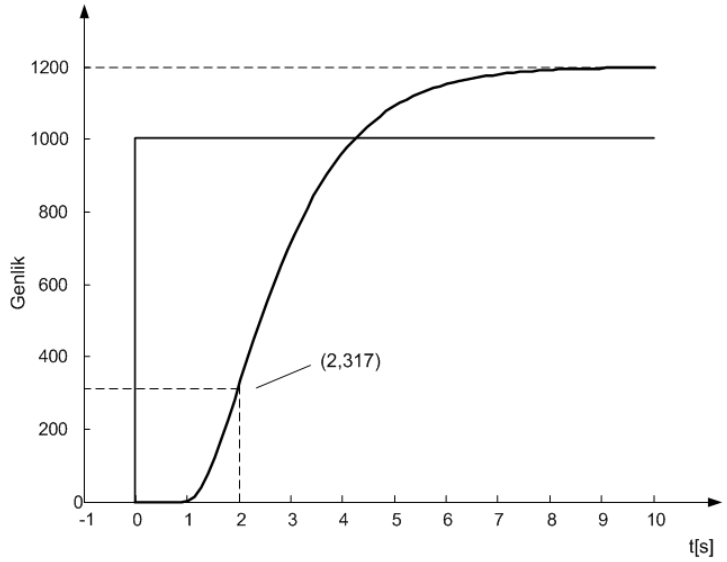
Buna göre $L + \tau = 2s$, L=1s ve τ=1s bulunur. Bu verilere göre s-tanım bölgesi transfer fonksiyonu

$$G(s) = \frac{1200}{(s+1)^2} e^{-s}$$

elde edilir. Bu sistemin 40dB/dec zayıfladığı frekans $\omega = 10 rad/s$, örnekleme frekansı bu değerin iki katından büyük seçilirse $\omega_s \geq 20 rad/s$ ve örnekleme zamanı

$$T \leq \frac{2\pi}{\omega_s} = \frac{2\pi}{20} = 0.314$$

ifadesi yazılabilir. Örnekleme zamanı T=250ms seçilerek zamana bağlı kesmeli çalışma doğrudan kullanılabilir.



Şekil 8.54 ikinci mertebeden kritik sönümlü ölü zamanlı sistemin basamak girişe yanıtı

T=0.25s için d=L/T=1s/0.25s=4 elde edilir ve

$$y(k) = -a_1 y(k-1) - a_0 y(k-2) + K_Z x(k-d-1) + b_0 x(k-d-2)$$

fark denklemi

$$y(k) = -a_1 y(k-1) - a_0 y(k-2) + K_Z x(k-5) + b_0 x(k-6)$$

elde edilir. Bu sistem için

$$A = \exp(-T/\tau) = \exp(-0.25/1) = 0.7788$$

$$K_Z = K(1 - A - AT/\tau) = 1.2 \times (1 - 0.7788 - 0.7788 \times 0.25/1) = 0.0318$$

$$b_0 = \frac{A(A - 1 + T/\tau)}{(1 - A - AT/\tau)} = \frac{0.7788 \times (0.7788 - 1 + 0.25/1)}{(1 - 0.7788 - 0.7788 \times 0.25/1)} = 0.8464$$

$$a_1 = -2A = -2 \times 0.7788 = -1.5576$$

$$a_0 = A^2 = (0.7788)^2 = 0.6065$$

bulunur. Şekil 8.55'te bu sistemin benzetimine ilişkin "IMKSS" başlıklı bir altprogram verilmiştir.

	Symbol	Var Type	Data Type	Comment
LW0	X	IN	INT	
LD2	b0	IN	REAL	
LD6	a1	IN	REAL	
LD10	a0	IN	REAL	
LD14	Kz	IN	REAL	
LD18	Ymin	IN	REAL	
LD22	Ymax	IN	REAL	
LD26	Y_1	IN_OUT	REAL	
LD30	Y_2	IN_OUT	REAL	
LD34	X_1	IN_OUT	REAL	
LD38	X_2	IN_OUT	REAL	
LD42	X_3	IN_OUT	REAL	
LD46	X_4	IN_OUT	REAL	
LD50	X_5	IN_OUT	REAL	
LD54	X_6	IN_OUT	REAL	
LW58	Y	OUT	INT	

Network 1 y(k)= - a1*y(k-1) - a0*y(k-2)+Kz*[x(k-5)+b0*x(k-6)]

```
LD      SM0.0
MOVR    #Y_1, AC0
*R      #a1, AC0
MOVR    #Y_2, AC1
*R      #a0, AC1
+R      AC1, AC0
*R      -1.0, AC0
```

Network 2

```
LD      SM0.0
MOVR    #X_6, AC1
*R      #b0, AC1
+R      #X_5, AC1
*R      #Kz, AC1
```

Network 3

```
LD      SM0.0
MOVR    #Y_1, #Y_2
+R      AC0, AC1
```

Network 4

```
LDR>=   AC1, #Ymax
MOVR    #Ymax, AC1
```

\MAIN \IMKSS /INT_0 /

Network 5

```
LDR<=   AC1, #Ymin
MOVR    #Ymin, AC1
```

Network 6

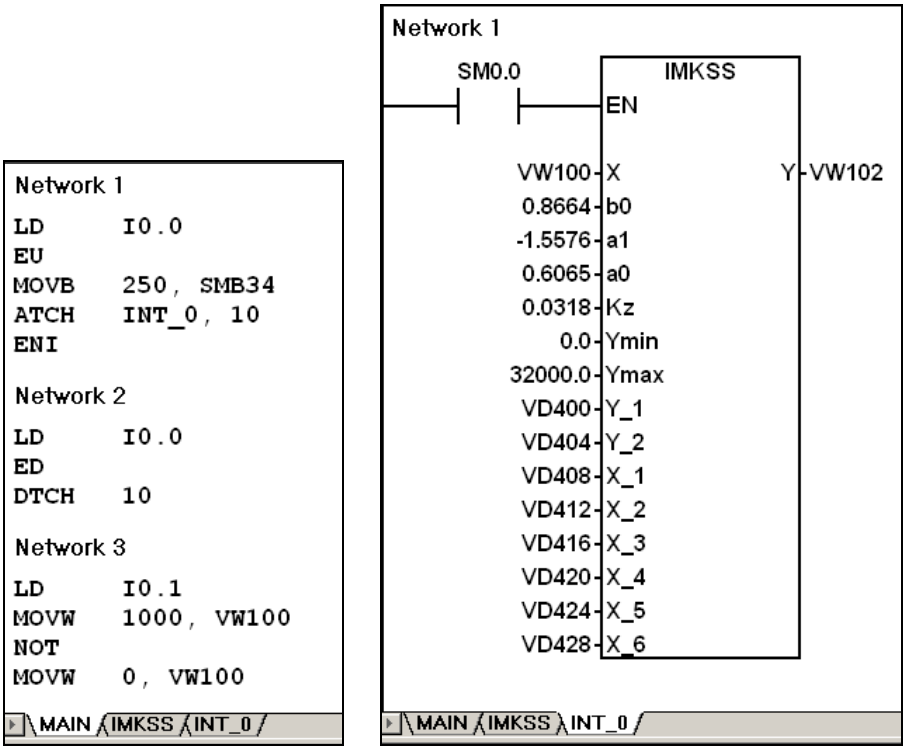
```
LD      SM0.0
MOVR    AC1, #Y_1
MOVR    #X_5, #X_6
MOVR    #X_4, #X_5
MOVR    #X_3, #X_4
MOVR    #X_2, #X_3
MOVR    #X_1, #X_2
ITD     #X, AC0
DTR     AC0, #X_1
```

Network 7

```
LD      SM0.0
MOVR    #Y_1, AC0
ROUND   AC0, AC0
DTI     AC0, #Y
```

Şekil 8.55 “IMKSS” başlıklı benzetim altprogramı

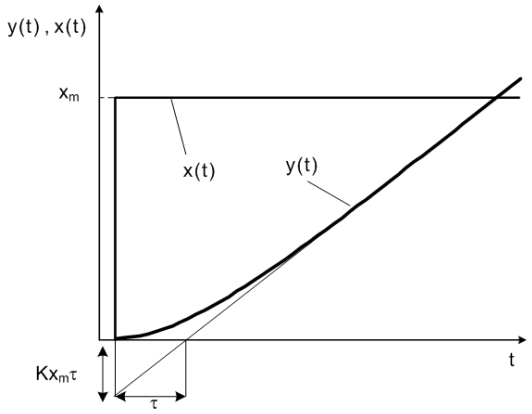
Benzetim işleminin PLC'de gerçeklenmesi için ana program ve kesme altprogramı Şekil 8.56'daki gibi düzenlenebilir.



Şekil 8.56 Benzetime ilişkin ana program ve kesme altprogramı

8.4.3 BİRİNCİ MERTEBEDEN İNTEGRAL ETKİLİ SİSTEMLER

Basamak girişe yanıtı Şekil 8.57'de verilen bir süreç, birinci mertebeden integral etkili sistem olarak tanımlanabilir.



Şekil 8.57 Birinci mertebeden integral etkili sistemin basamak girişe yanıtı

Bu tür sistemlerin matematiksel modeline ilişkin s-tanım bölgesi transfer fonksiyonu

$$G(s) = \frac{Y(s)}{X(s)} = \frac{K}{s(\tau s + 1)} \tag{8.36}$$

biçiminde verilebilir.

Bu sistemin ayrık zaman modeline ilişkin z-tanım bölgesi transfer fonksiyonu, $G_{ZOH}(s)$ sıfırıncı mertebeden tutucuya ilişkin transfer fonksiyonu ve T örnekleme zamanı olmak üzere

$$G(z) = Z\{G_{zoh}(s)G(s)\} = Z\{\frac{1-e^{-sT}}{s}\frac{K}{s(\tau s+1)}\} = \frac{(1-e^{-sT})}{1}Z\{\frac{K/\tau}{s^2(s+1/\tau)}\}$$

ifadesinden elde edilebilir. Bu ifade düzenlenirse

$$G(z) = K\frac{(1-e^{-sT})}{1}Z\{\frac{1}{s^2}-\frac{\tau}{s}+\frac{\tau}{s+1/\tau}\}$$

$$G(z) = K\frac{(1-z^{-1})}{1}Z\{\frac{Tz}{(z-1)^2}-\frac{\tau z}{(z-1)}+\frac{\tau z}{(z-A)}\},\ A = e^{-T/\tau}$$

$$G(z) = K(\frac{T}{z-1} - \tau + \frac{\tau(z-1)}{z-A})$$

$$G(z) = K(\frac{T(z-A) - \tau(z-A)(z-1) + \tau(z-1)^2}{(z-1)(z-A)})$$

$$G(z) = K\frac{z(T-\tau+A\tau) - (TA + A\tau - \tau)}{z^2 - (1+A)z + A})$$

elde edilir. Eğer

$$K_Z = K(T - \tau + A\tau)\,,\ b_0 = -\frac{TA + \tau A - \tau}{T - \tau + \tau A}\,,\ a_1 = -(1+A)\,,\ a_0 = A$$

tanımları yapılırsa

$$G(z) = K_Z \frac{z + b_0}{z^2 + a_1 z + a_0} \tag{8.37}$$

elde edilir.

Bu ifadeden, sisteme ilişkin fark denklemi

$$y(k) = -a_1 y(k-1) - a_0 y(k-2) + K_Z x(k-1) + K_Z b_0 x(k-2)$$

biçiminde elde edilir.

Basamak girişe yanıtı Şekil 8.58'de verilen, birinci mertebeden integral etkili ölü zamanlı sistemler için s-tanım bölgesi transfer fonksiyonu

$$G(s) = \frac{Y(s)}{X(s)} = \frac{Ke^{-sL}}{s(\tau s + 1)} \tag{8.38}$$

ayrık zaman modeline ilişkin z-tanım bölgesi transfer fonksiyonu

$$K_Z = K(T - \tau + A\tau)\,,\ b_0 = -\frac{TA + \tau A - \tau}{T - \tau + \tau A}\,,\ a_1 = -(1+A)\,,\ a_0 = A\,, d = L/T$$

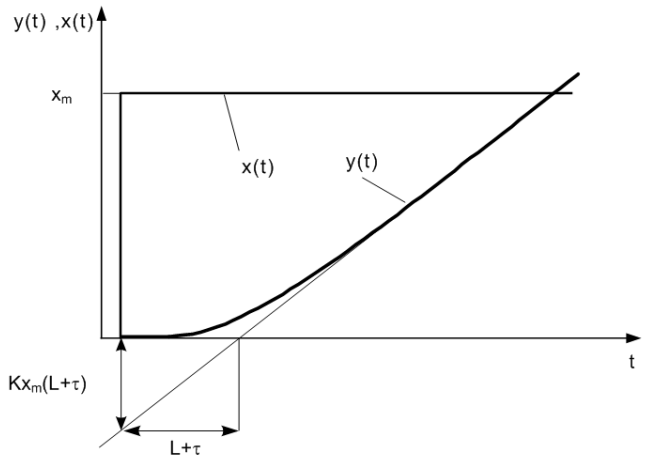
tanımları yapılırsa

$$G(z) = K_Z \frac{z + b_0}{z^2 + a_1 z + a_0} z^{-d} \tag{8.39}$$

biçiminde ve fark denklemi

$$y(k) = -a_1 y(k-1) - a_0 y(k-2) + K_Z x(k-d-1) + K_Z b_0 x(k-d-2)$$

olarak elde edilir.



Şekil 8.58 Birinci mertebeden İntegral etkili ölü zamanlı sisteme ilişkin basamak giriş yanıtı

Benzetim programı, fark denklemine ilişkin K_Z, b_0, a_1, a_0 ve d parametreleri belirlenerek gerçeklenebilir.

Örnek 8.7 Genliği x_m=1000 olan basamak girişe yanıtı Şekil 8.59'da gösterilen sisteme ilişkin bir benzetim programı yazınız.

Sistem yanıtından

$$Kx_m(L+\tau) = 650s,\ L+\tau = 6.5s \text{ ve } L = 1.5s$$

değerlerinden $K = 0.1$, $\tau = 5s$ elde edilir ve sisteme ilişkin s-tanım bölgesi transfer fonksiyonu

$$G(s) = \frac{Y(s)}{X(s)} = \frac{Ke^{-sL}}{s(\tau s+1)} = \frac{0.1}{s(5s+1)}e^{-1.5s}$$

bulunur.

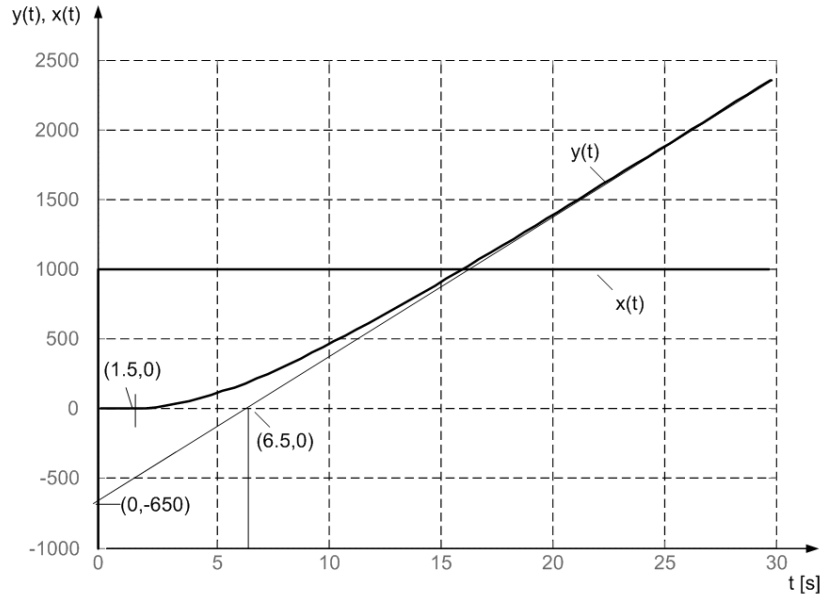
Bu sistemin benzetimi için örnekleme zamanı $T = 0.1 \times \tau = 0.5s$ seçilirse $d = L/T = 1.5/0.5 = 3$

ve fark denklemi

$$y(k) = -a_1 y(k-1) - a_0 y(k-2) + K_Z x(k-d-1) + K_Z b_0 x(k-d-2)$$

$$y(k) = -a_1 y(k-1) - a_0 y(k-2) + K_Z x(k-4) + K_Z b_0 x(k-5)$$

olur.



Şekil 8.59 Sistemin, genliği x_m=1000 olan basamak girişe yanıtı

Fark denklemine ilişkin katsayılar

$$A = e^{-T/\tau} = \exp(-0.5/5) = 0.9048$$

$$K_Z = K(T - \tau + A\tau) = 0.1 \times (0.5 - 5 + 0.9048 \times 0.5) = 0.0024$$

$$b_0 = -\frac{TA + \tau A - \tau}{T - \tau + \tau A} = -\frac{0.5 \times 0.9048 + 5 \times 0.9048 - 5}{0.5 - 5 + 5 \times 0.9048} = 0.9672$$

$$a_1 = -(1 + A) = -(1 + 0.9048) = -1.9048$$

$$a_0 = A = 0.9048,$$

$$d = L / T = 1.5 / 0.5 = 3$$

bulunur. Bu verilere göre, sistemin benzetimine ilişkin "IEBMS" başlıklı altprogram Şekil 8.60, ana program ve 500ms aralıklarla işletilen zamanlayıcı kaynaklı kesme altprogramı Şekil 8.61'deki gibi düzenlenebilir.

	Symbol	Var Type	Data Type	Comment
LW0	X	IN	INT	
LD2	b0	IN	REAL	
LD6	a1	IN	REAL	
LD10	a0	IN	REAL	
LD14	Kz	IN	REAL	
LD18	Ymin	IN	REAL	
LD22	Ymax	IN	REAL	
LD26	Y_1	IN_OUT	REAL	
LD30	Y_2	IN_OUT	REAL	
LD34	X_1	IN_OUT	REAL	
LD38	X_2	IN_OUT	REAL	
LD42	X_3	IN_OUT	REAL	
LD46	X_4	IN_OUT	REAL	
LD50	X_5	IN_OUT	REAL	
LW54	Y	OUT	INT	
		TEMP		

```
Network 1      y(k)= - a1*y(k-1) - a0*y(k-2)+Kz*[x(k-4)+b0*x(k-5)]
LD      SM0.0
MOVR    #Y_1, AC0
*R      #a1, AC0
MOVR    #Y_2, AC1
*R      #a0, AC1
+R      AC1, AC0
*R      -1.0, AC0

Network 2
LD      SM0.0
MOVR    #X_5, AC1
*R      #b0, AC1
+R      #X_4, AC1
*R      #Kz, AC1

Network 3
LD      SM0.0
MOVR    #Y_1, #Y_2
+R      AC0, AC1

Network 4
LDR>=   AC1, #Ymax
MOVR    #Ymax, AC1
```

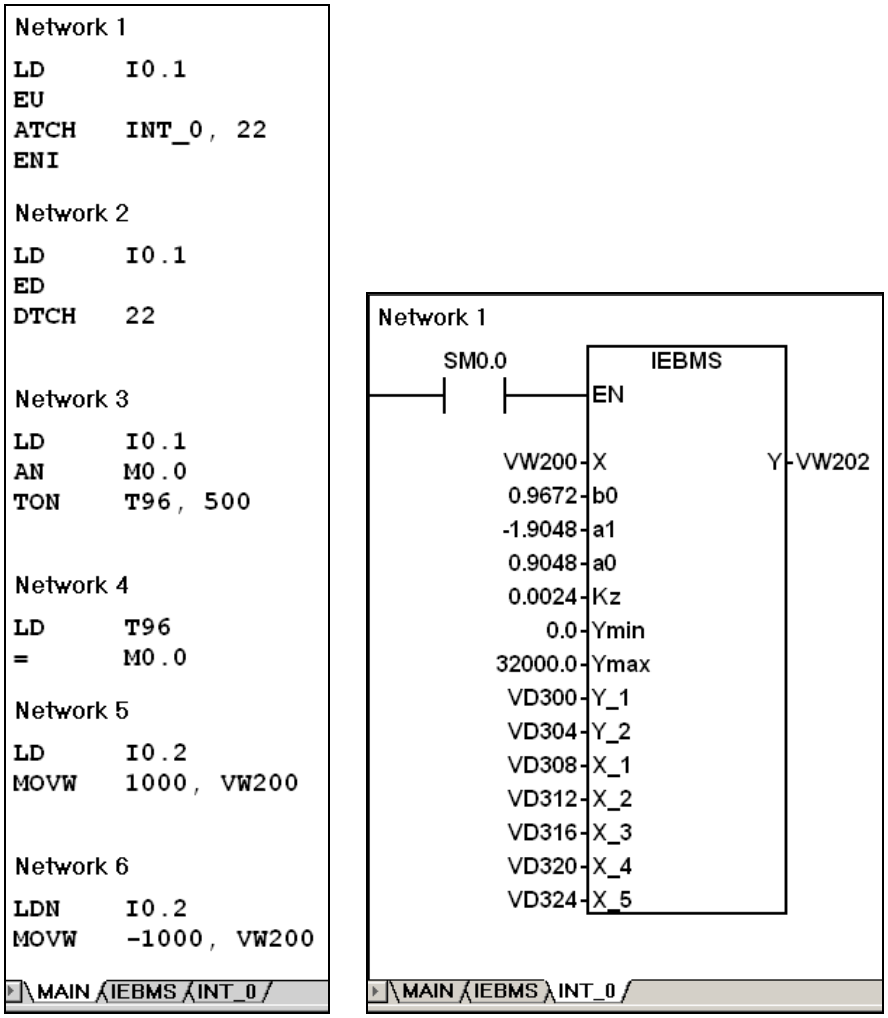
MAIN | IEBMS | INT_0

```
Network 5
LDR<=   AC1, #Ymin
MOVR    #Ymin, AC1

Network 6
LD      SM0.0
MOVR    AC1, #Y_1
MOVR    #X_4, #X_5
MOVR    #X_3, #X_4
MOVR    #X_2, #X_3
MOVR    #X_1, #X_2
ITD     #X, AC0
DTR     AC0, #X_1

Network 7
LD      SM0.0
MOVR    #Y_1, AC0
ROUND   AC0, AC0
DTI     AC0, #Y
```

Şekil 8.60 “IEBMS” başlıklı altprogram



Şekil 8.61 Benzetime ilişkin ana program ve kesme altprogramı

BÖLÜM 9
SIMATIC S7-300/S7-400
PLC AİLESİ VE UYGULAMALARI

9.1 GİRİŞ

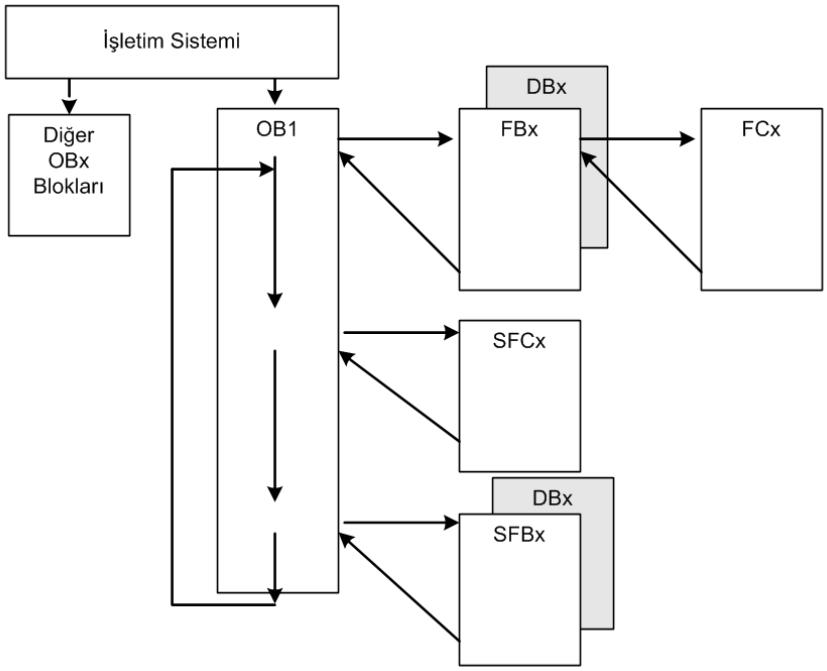
Büyük ölçekli otomasyon sistemlerinin gerçeklenmesinde kullanılan SIMATIC S7-300/S7-400 PLC ailesinde çok sayıda işlemci türü bulunur. Tüm işlemci türlerinin donanım ve iletişim ayarlarının yapılması, programlanması ve program akışının izlenmesi "SIMATIC MANAGER" yazılımı ile sağlanır. "SIMATIC MANAGER" yazılımı altında LAD, FBD ve STL standart programlama dillerinin yanı sıra birçok yüksek düzey metin ve grafiksel programlama dilleri de kullanılabilir. Geniş ölçekli bir otomasyon sisteminin gerektirdiği birçok özelliğe sahip bu yazılımın, kumanda ve kontrol uygulamalarının gerçeklenmesinde kullanılan özellikleri, örnekler verilirken açıklanacaktır.

9.2 PROGRAM YAPISI VE ÇALIŞMA BİÇİMİ

SIMATIC S7-300/S7-400 PLC ailesi PLC'lerinde program,"BLOK" olarak tanımlanan program belleği alanlarına yazılır. Bu alanlar; organizasyon blokları (OBs), fonksiyonlar (FCs), fonksiyon blokları (FBs) ve veri blokları (DBs) olarak tanımlanır.

Organizasyon blokları, PLC işletim sistemi tarafından belirli koşullar oluştuğunda işletilir. Fonksiyon, fonksiyon blok, veri bloğu ve çeşitli işlevleri sağlamak üzere sistem belleğine yüklenmiş hazır fonksiyon (SFCs) ve sistem fonksiyon blokları (SFBs) kullanıcı programına göre işletilir.

OB1 organizasyon bloğu işletim sistemi tarafından sürekli taranır ve bu bloğa yazılan komutlar sürekli yinelenerek işlenir. Bu bloğa yazılan program ana program olarak tanımlanır. OB1 bloğunda yazılan komutların işlenme süresi, tarama çevriminin işlemci özellikleri bölümde tanımlanan süreyi (scan cycle monitoring time) aşması durumunda işletim sistemi PLC'yi durdurur.



Şekil 9.1 Program yapısı

İşletim sistemi tarafından denetlenen bazı OB blokları ile ilişkili olaylar aşağıda verilmiştir. İşlemci türüne bağlı olarak sayısı değişen bu bloklardan bazıları koşulsuz olarak, bazıları kesmeli çalışmaya ilişkin koşullandırmaya bağlı olarak işletilir. Örneğin, OB10-OB17 blokları donanım ayarları veya sistem fonksiyonları (SFC28-SFC31), OB20-OB23 blokları sistem fonksiyonları (SFC32-SFC34) ile etkin duruma getirilir ve ilgili olay gerçeklendiğinde bu bloklar işletilir. OB70 ile OB121 arasındaki bloklar ilgili olay gerçeklendiğinde işletilir.

OB10 - OB17: Tarihe bağlı kesmeli işletilen organizasyon blokları

OB20 - OB23: Zaman gecikmeli kesmeli işletilen organizasyon blokları

OB30 - OB38: Zamana bağlı kesmeli işletilen organizasyon blokları

OB40 - OB47: Donanımsal kesme işareti ile işletilen organizasyon blokları

OB70 - OB87: Hatalı durumlarda işletilen organizasyon blokları

OB100 – OB102: İşlemci devreye alındığında işletilen organizasyon blokları

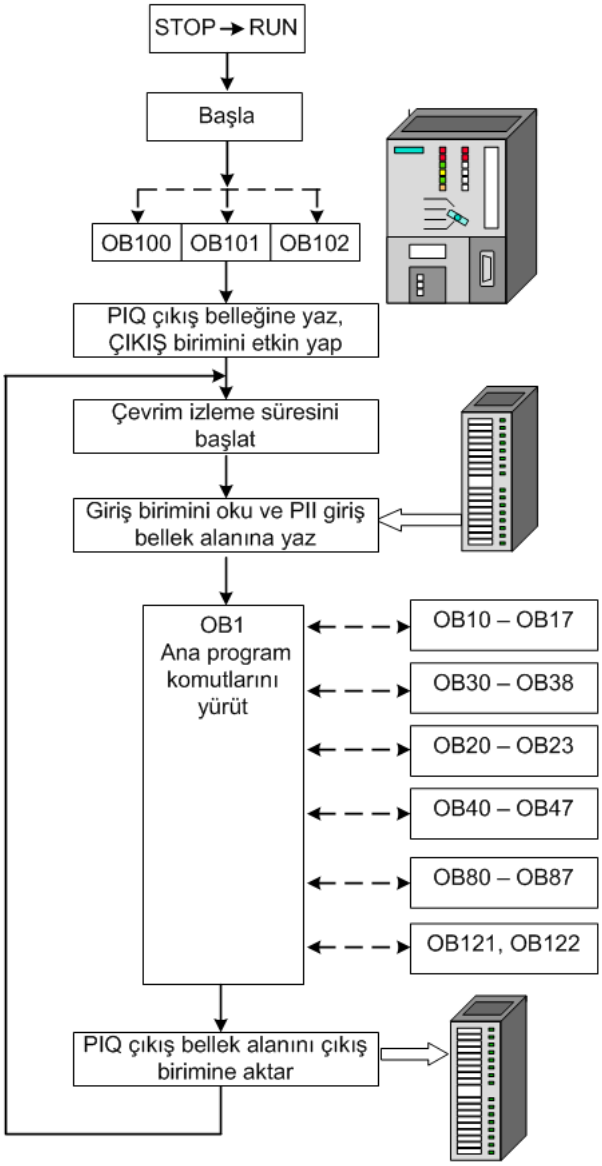
OB121: Programlama hatası oluştuğunda işletilen organizasyon bloğu

OB122: Giriş-çıkış adreslerine erişim hatası olduğunda işletilen organizasyon bloğu

PLC çalışma durumuna getirildiğinde, işlem akışı Şekil 9.2'de gösterilen sıraya göre gerçeklenir. İlk olarak, başlama durumuna göre OB100, OB101 veya OB102 organizasyon bloklarından biri işletilir. Eğer PLC ılık başlama (warm restart) durumuna ayarlı ise OB100 işletilir, kalıcı (retentive) olarak tanımlanmayan M bellek, sayıcı (C) ve zamanlayıcı (T) içerikleri silinir ve OB1'deki ana program baştan yürütülür.

Eğer PLC sıcak başlama (hot restart) durumuna ayarlı ise, OB101 işletilir ve OB1 bloğundaki program, kesildiği noktadan başlanarak yürütülmeye devam edilir.

Eğer PLC soğuk başlama (cold restart) durumuna ayarlı ise OB102 işletilir, tüm M bellek, sayıcı (C) ve zamanlayıcı (T) içerikleri silinir ve OB1'deki ana program baştan yürütülür.



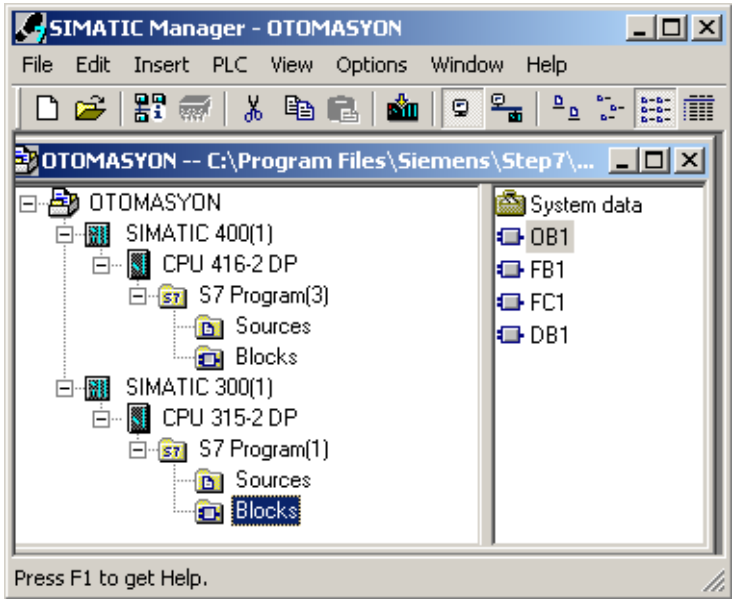
Şekil 9.2 S7-300/S7-400 PLC işlem akışı şeması

Başlama durumlarına ilişkin organizasyon bloklarından biri işletildikten sonra, burada hesaplanan çıkış değerleri, çıkış görüntü belleğine yazılır ve çıkış birimi etkin duruma getirilir. Bir sonraki adımda gözetleyici zamanlayıcısı (watchdog timer) içeriği sıfırlanarak çevrim izleme süresi başlatılır ve OB1

organizasyon bloğu tarama çevrimi süreci başlar. OB1'de yazılan program komutları işlenirken, diğer organizasyon bloklarının işletilmesi için gerekli koşullar gerçeklendiğinde program akışı bu bloklara geçer ve buradaki programların yürütülmesi tamamlandıktan sonra tekrar OB1'deki programa dönülür. Kalan programın yürütülmesi tamamlandıktan sonra hesaplanan değerler çıkış birimine aktarılır, tekrar gözetleyici zamanlayıcısı sıfırlanır ve tarama çevrimi tekrar başlar. Eğer OB1 tarama çevrimi belirli bir süreyi aşarsa, gözetleyici zamanlayıcısı işlemciyi durdurma işareti üretir.

9.3 PROGRAMLAMA

Program, bir proje altında açılan ve istasyon olarak tanımlanan PLC işlemcilerine ilişkin bloklara yazılır. Şekil 9.3'te iki PLC'den oluşan otomasyon başlıklı bir proje görülmektedir.

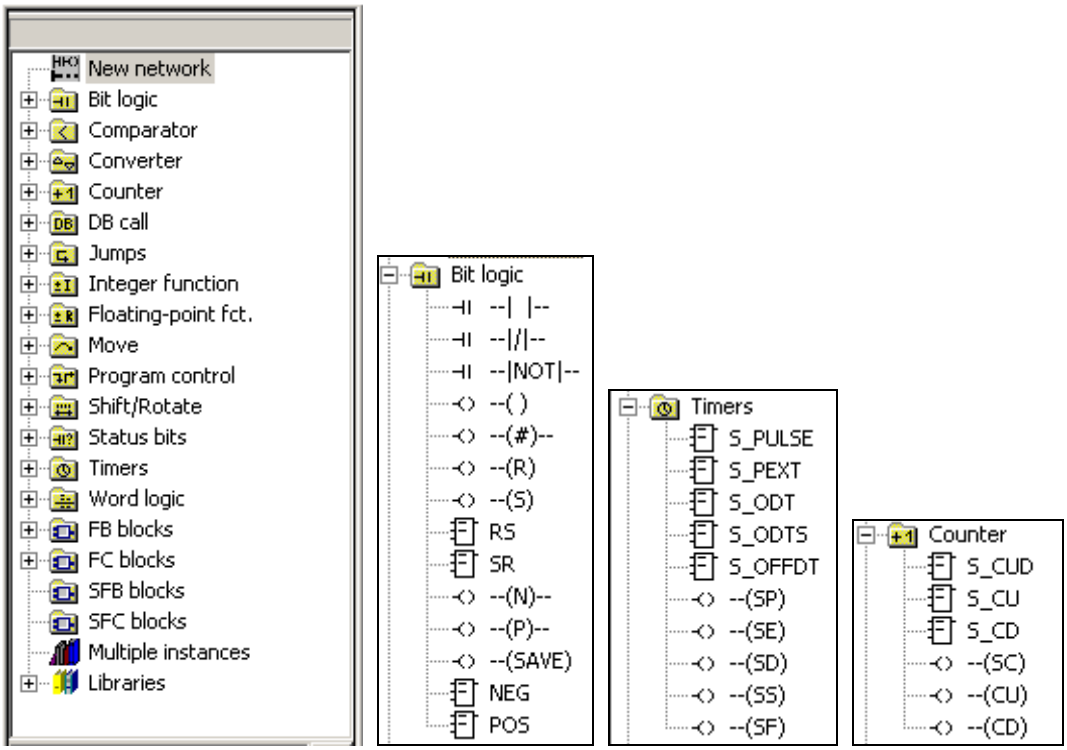


Şekil 9.3 SIMATIC Manager açılış penceresi

Bu bloklardan herhangi biri açılır ve programın yazılacağı menüde "View" başlığı altındaki programlama dillerinden (LAD, STL, FBD) biri seçilerek programlamaya başlanır.

LAD ve FBD ile programlama, Şekil 9.4'te gösterilen ağaç yapısındaki bölümlerin altındaki simgeler taşınarak gerçeklenir. STL ile programlama ise komutların alt alta belirli kurallara göre yazılması ile yapılır.

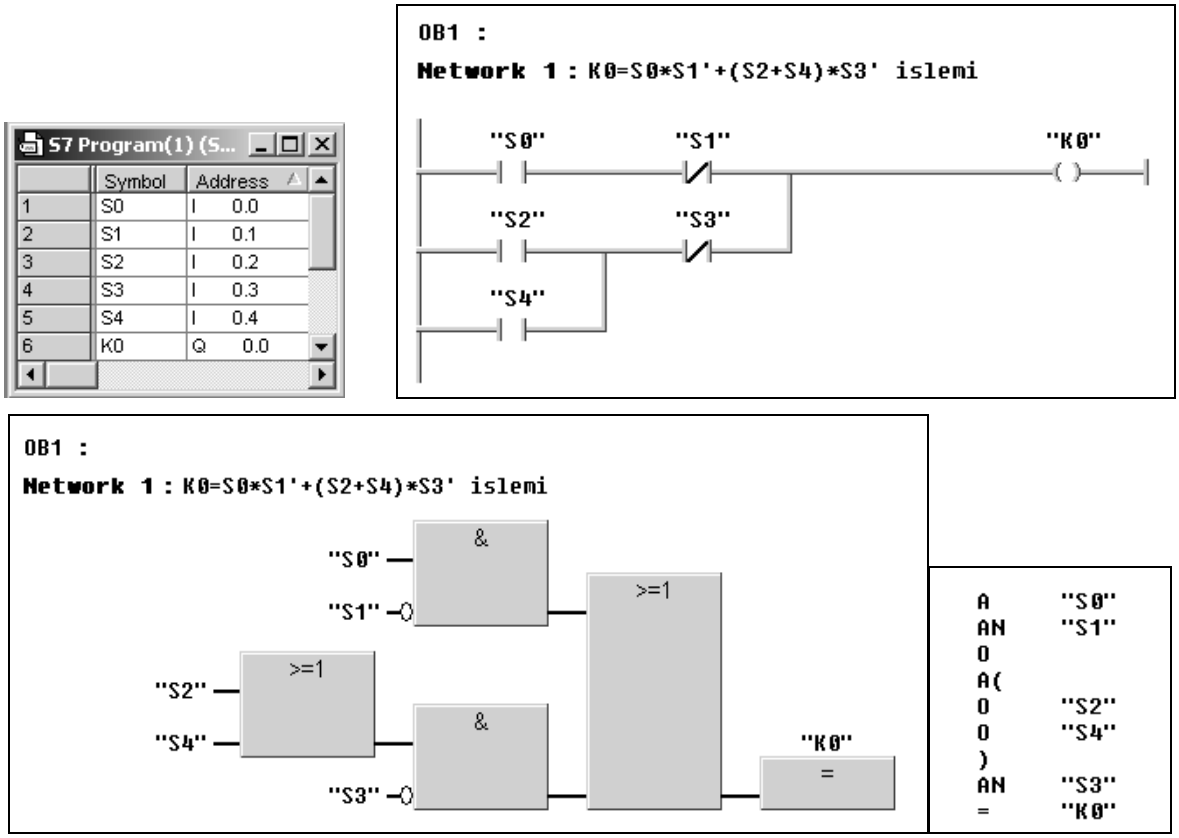
Temel mantıksal işlemlere ilişkin komutlar “Bit logic” başlığı altında, zamanlama işlemlerine ilişkin komutlar “Timers” başlığı altında ve sayma işlemlerine ilişkin komutlar “ Counter” başlığı altında bulunur.



Şekil 9.4 LAD ile programlamada komut simgeleri

9.3.1 TEMEL MANTIKSAL İŞLEMLER

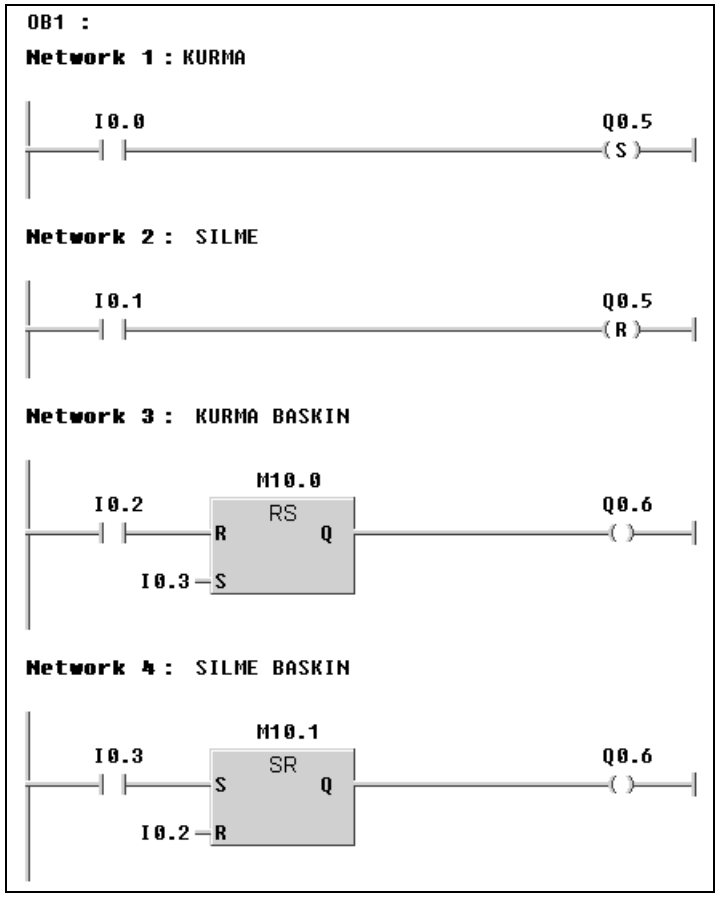
Temel mantıksal işlemler (VE, VEYA, DEĞİL), tüm PLC’lerde yapıldığı gibi, LAD ve FBD ile programlamada kontak veya mantıksal kapı simgelerinin seri ve paralel bağlanması ile gerçeklenir Daha karmaşık kurallara göre gerçeklenen STL ile programlama ayrı bir bölüm altında incelenecektir. Şekil 9.5’te $K0=S0 \bullet \overline{S1}+(S2+S4) \bullet \overline{S3}$ b biçiminde verilen bir mantıksal fonksiyonun gerçeklenmesine ilişkin bir program örneğinin sembol tablosu, LAD, FBD ve STL dilindeki karşılıkları verilmiştir.



Şekil 9.5 Temel mantıksal işlemlere ilişkin LAD, FBD ve STL program örnekleri

9.3.2 KURMA VE SİLME İŞLEMLERİ

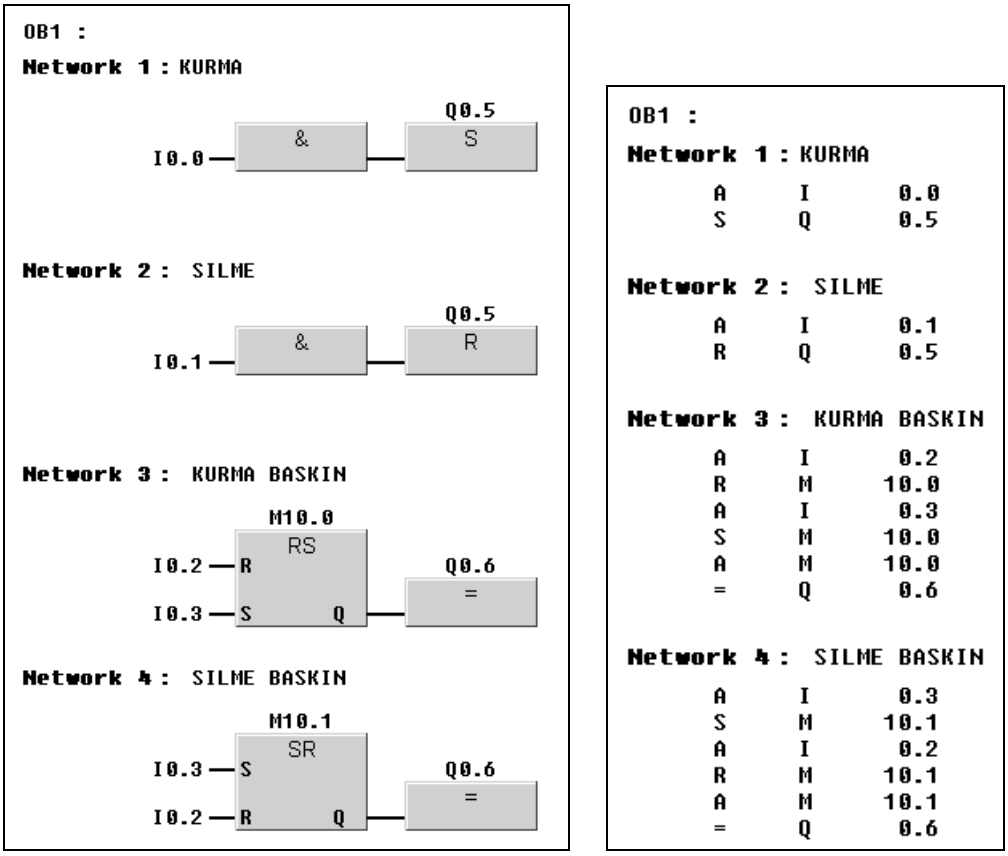
Kurma ve silme işlevlerini sağlayan komutların, LAD dilindeki karşılığı Şekil 9.6, FBD ve STL dilindeki karşılıkları Şekil 9.7'de verilmiş ve işlevleri açıklanmıştır.



Şekil 9.6 Kurma ve silme işlemlerine ilişkin LAD program örneği

Şekil 9.6'da gösterilen programa göre, I0.0=1 yapıldığında Q0.5=1 değerine kurulur (Network 1) ve I0.1=1 yapılıncaya kadar I0.0 değerinden bağımsız olarak bu değerinde kalır (Network 2). Eğer, I0.0 ve I0.1 aynı anda 1 yapılırsa Q0.5=0 olur (Silme baskın).

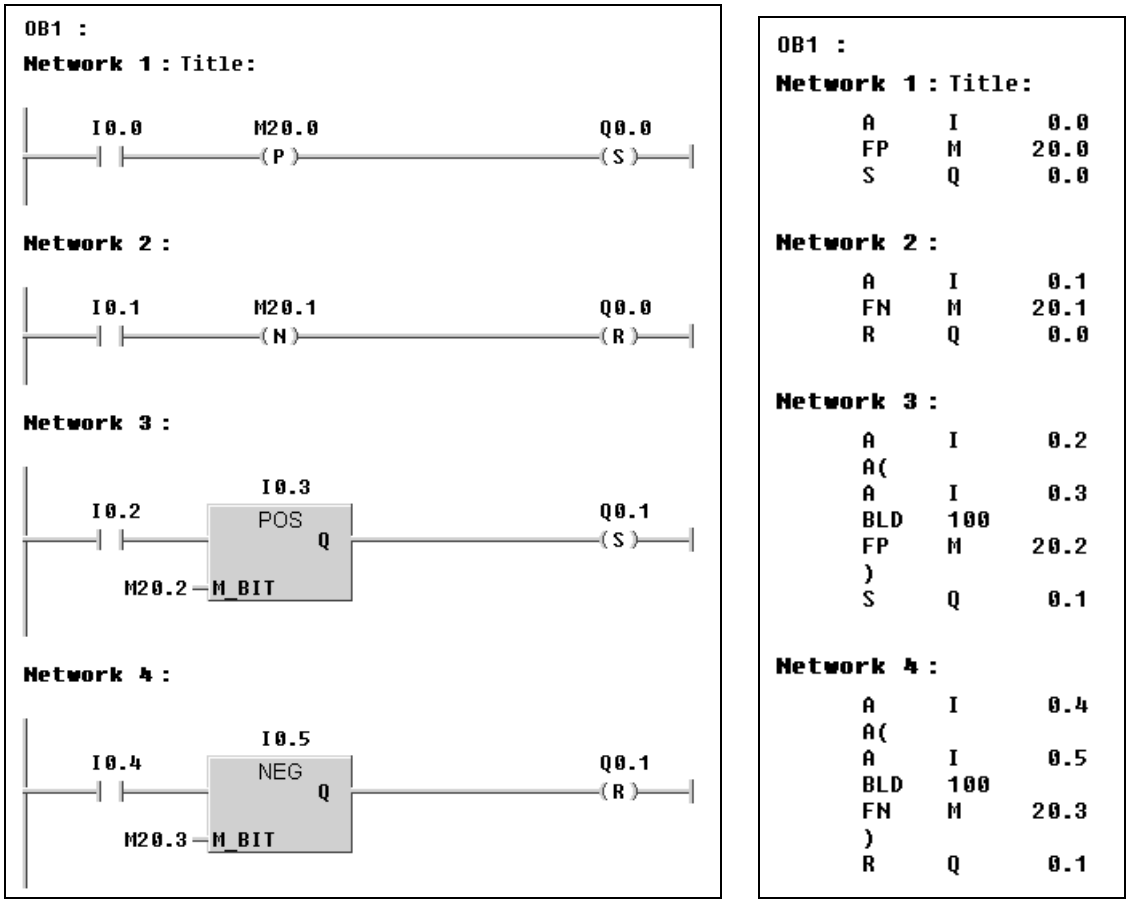
Kurma baskın bir devreyi gerçeklemek için ya (Network 1) ile (Network 2) yer değiştirilir ya da (Network 3)'teki program yazılır. Silme baskın bir devreyi gerçeklemek için (Network 4) programı yazılır.



Şekil 9.7 Kurma ve silme işlemlerine ilişkin FBD ve STL program örnekleri

9.3.3 ÇIKAN VE DÜŞEN KENARI ALGILAMA İŞLEMLERİ

Bir işaretin yalnız çıkan veya düşen kenarında bir tarama süresi etkin olan bir işareti üretmek için LAD ve STL dilindeki karşılıkları Şekil 9.8'da verilen komutlar kullanılır. Buradaki M bit adresleri komutun bir önceki durumu saklaması için kullanılır ve bu adresler komuta ayrıldığından programın başka bir yerinde kullanılmamalıdır.



Şekil 9.8 Çıkan ve düşen kenarı algılama işlemlerine ilişkin FBD ve STL program örnekleri

Bu programa göre, I0.0 girişi 0'dan 1'e değiştirildiğinde Q0.0=1 değerine kurulur. I0.1 girişi 0'dan 1'e değiştirildiğinde durumda bir değişiklik olmaz, ancak I0.1 girişi 1'den 0'a değiştirildiğinde Q0.0=0 olur.

I0.2=1iken I0.3 girişi 0'dan 1'e değiştirildiğinde Q0.1=1 değerine kurulur. Ancak, önce I0.3=1 daha sonra I0.2=1 yapılırsa Q0.1=1 değerine kurulmaz.

I0.4=1 iken I0.5 girişi 1'den 0'a değiştirildiğinde Q0.1=0 olur, bunun dışındaki değişikliklerde durum değişikliği olmaz.

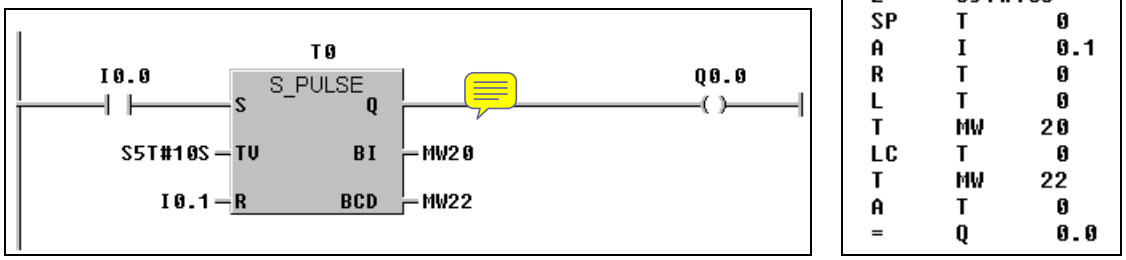
Edited by Foxit Reader
Copyright(C) by Foxit Software Company,2005-2008
For Evaluation Only.

9.3.4 ZAMANLAMA İŞLEMLERİ

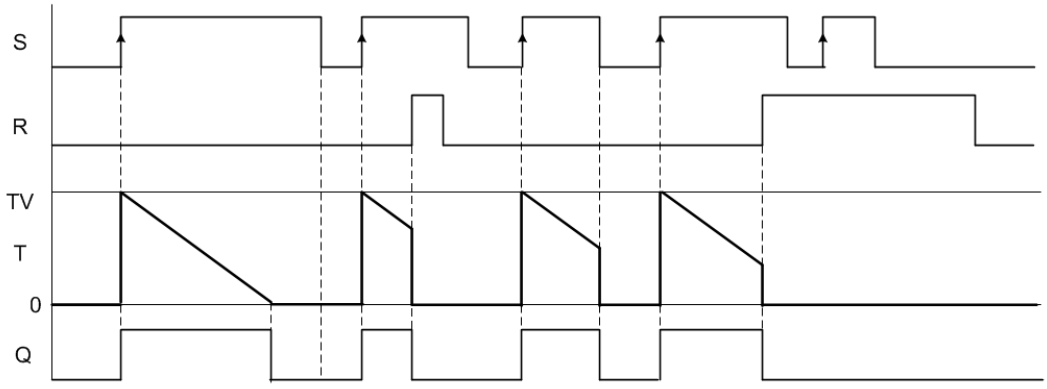
Zamana bağlı kumanda işlemleri; darbe zamanlayıcısı (S_PULSE), uzatılmış darbe zamanlayıcısı (S_PEXT), gecikmeli kapatan zamanlayıcı (S_ODT), kalıcı gecikmeli kapatan zamanlayıcı S_ODTS ve gecikmeli açan zamanlayıcı (S_OFFDT) özelliklerinde 5 tür zamanlayıcı komutu ile gerçeklenebilir.

9.3.4.1 Darbe Zamanlayıcısı (S_PULSE, SP)

Darbe zamanlayıcısının programlanmasına ilişkin komutların LAD ve STL dilindeki karşılıkları Şekil 9.9'da, bu zamanlayıcı türüne ilişkin zamanlama diyagramı Şekil 9.10'da gösterilmiştir.



Şekil 9.9 Darbe zamanlayıcısına ilişkin LAD ve STL programları



Şekil 9.10 Darbe zamanlayıcısına ilişkin zamanlama diyagramı

Bu zamanlayıcıda, S, R ve TV giriş değerlerine bağlı olarak zamanlayıcı içeriği ve Q çıkışının değişimi zamanlama diyagramında gösterilmiştir. Şekilden görüldüğü üzere S girişinin 0'dan 1'e değiştirilmesi ile Q=1 olur; zamanlayıcı içeriği sıfır olduğunda, S=0 veya R=1 yapıldığında Q=0 olur. Zamanlayıcı çıkışındaki BI çıkışına zamanlayıcı içeriğinin kalan değeri 10 bitlik işaretli tamsayı, BCD çıkışına ise zamanlayıcı içeriğinin kalan değeri BCD formatında yazılır. Zamanlayıcıya zaman değeri Şekil 9.11'de gösterildiği gibi S5T zaman formatında yüklenir. Burada, en düşük anlamlı 12 bit zaman sayacının BCD formatında mutlak değeri, t_0 ve t_1 konumundaki bitler sayaç zaman çarpanıdır. Zaman çarpanı t_1t_0=00 için 10ms, t_1t_0=01 için 100ms, t_1t_0=10 için 1s ve t_1t_0=11 için 10s olur. Buna göre

t_1t_0=00 için en uzun süre 999x10ms=9990ms=9.99s değeridir. Bu değer zamanlayıcıya yüklendiğinde zamanlayıcı içeriği 2#0000_1001_1001_1001=16#0999 olur.

t_1t_0=01 için en uzun süre 999x100ms=99900ms=99.9s değeridir. 10s ile 99.9s aralığındaki zaman değeri zamanlayıcıya yüklendiğinde t_1t_0=01 olur.

t_1t_0=10 için en uzun süre 999x1s=999s değeridir. 100s ile 999s aralığındaki zaman değeri zamanlayıcıya yüklendiğinde t_1t_0=10 olur.

t_1t_0=11 için en uzun süre 999x10s=9990s değeridir. 1000s ile 9990s aralığındaki zaman değeri zamanlayıcıya yüklendiğinde t_1t_0=11 olur. En uzun süreye karşı düşen bu değer zamanlayıcıya yüklendiğinde zamanlayıcı içeriği 2#0011_1001_1001_1001=16#2999 olur. Bu değer S5T formatında S5T#2H46M30S biçiminde yazılarak zamanlayıcıya yüklenir.

		13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	t_1	t_0	1	0	0	1	1	0	0	1	1	0	0	1

Şekil 9.11 Zamanlayıcı içeriğinin S5T formatındaki görünümü

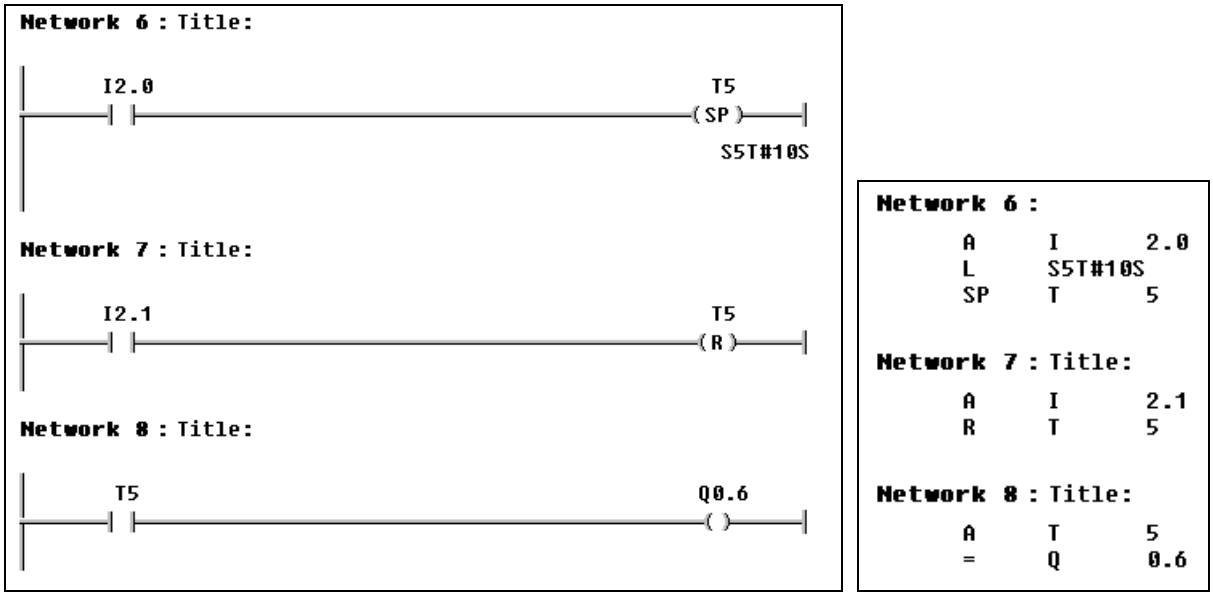
Zamanlayıcı içeriği ile ilgili karşılaştırma veya matematiksel işlem yapmak için L komutu kullanılır. Bu durumda BCD formatındaki zamanlayıcı içeriğinin işaretli tamsayı karşılığı Şekil 9.12'de gösterildiği biçimde elde edilir.

						2^9	2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
0	0	0	0	0	0	1	1	1	1	1	0	0	1	1	1

Şekil 9.12 Zamanlayıcı içeriğinin işaretli tamsayı formatındaki görünümü

Zamanlayıcı içeriğinin kullanımını gerektirmeyen uygulamalar için en yalın biçimde darbe zamanlayıcısı Şekil 9.13'te gösterildiği gibi programlanabilir.

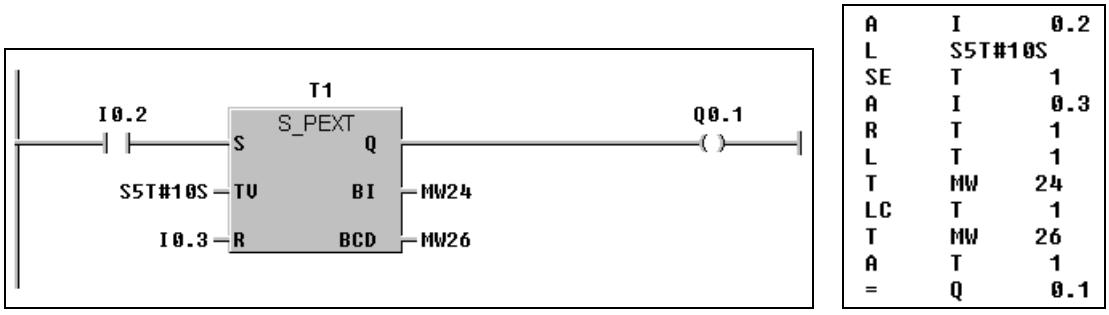
Edited by Foxit Reader
Copyright(C) by Foxit Software Company,2005-2008
For Evaluation Only.



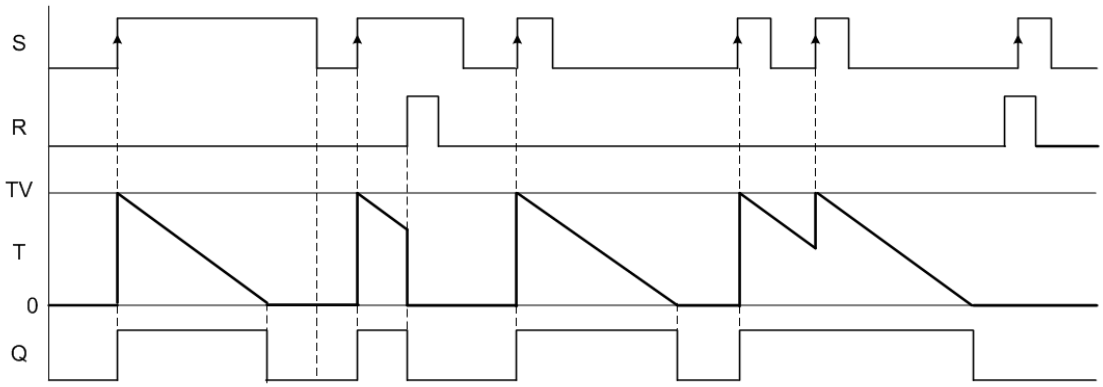
Şekil 9.13 Darbe zamanlayıcısına ilişkin en yalın LAD ve STL programları

9.3.4.2 Uzatılmış Darbe Zamanlayıcısı (S_PEXT, SE)

Uzatılmış darbe zamanlayıcısının LAD ve STL dilinde programlanmasına ilişkin komutlar Şekil 9.14'te, bu zamanlayıcı türüne ilişkin zamanlama diyagramı Şekil 9.15'te gösterilmiştir.



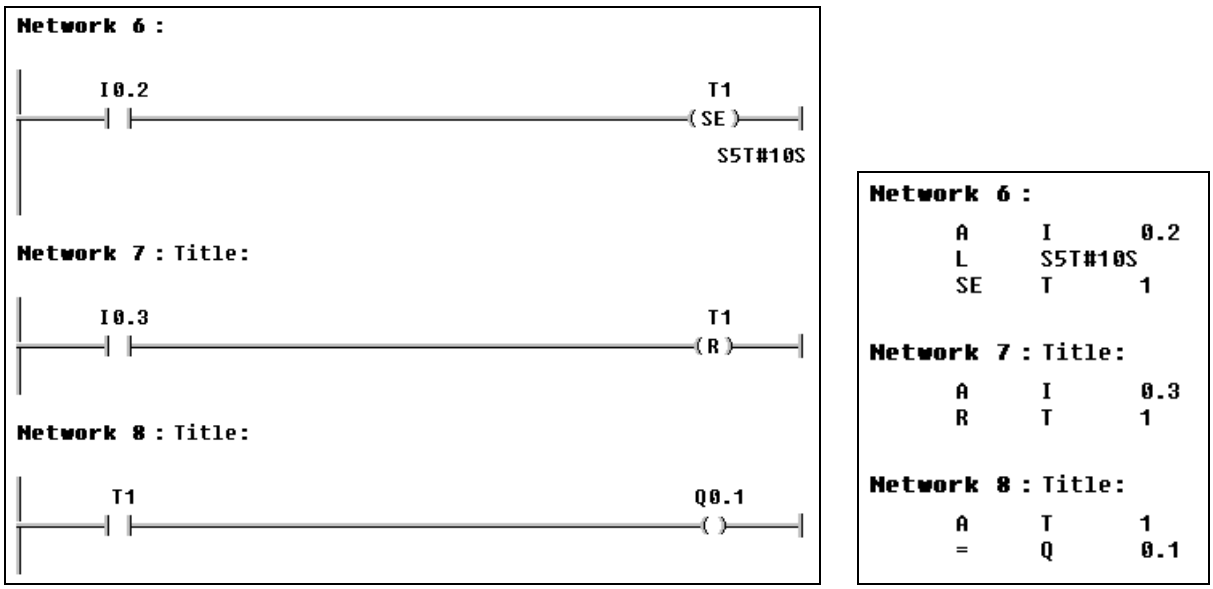
Şekil 9.14 Uzatılmış darbe zamanlayıcısına ilişkin LAD ve STL programları



Şekil 9.15 Uzatılmış darbe zamanlayıcısına ilişkin zamanlama diyagramı

Uzatılmış darbe zamanlayıcısında, S girişinin 0'dan 1'e değiştirilmesi ile Q=1 olur; zamanlayıcı içeriği sıfır olduğunda veya R=1 yapıldığında Q=0 olur. Bu zamanlayıcının darbe zamanlayıcısından farkı, sayma süreci başladıktan sonra S=0 yapılmasının durumu etkilememesidir.

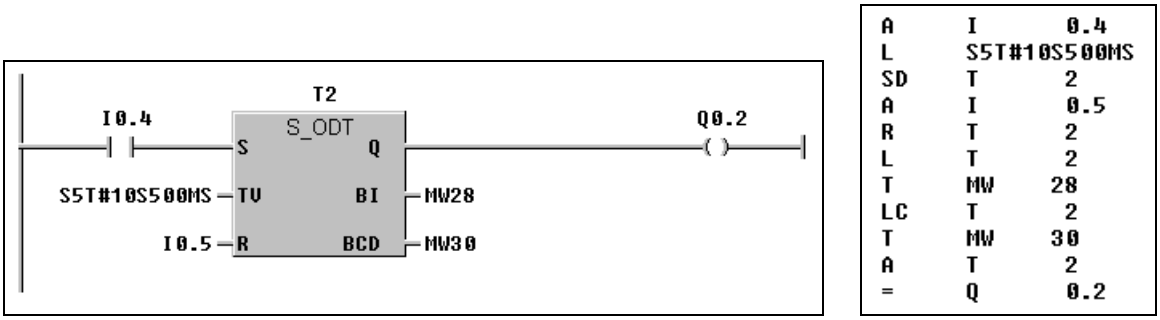
Uzatılmış darbe zamanlayıcısı, zamanlayıcı içeriğinin kullanımını gerektirmeyen uygulamalar için en yalın biçimde Şekil 9.16'da gösterildiği gibi programlanabilir.



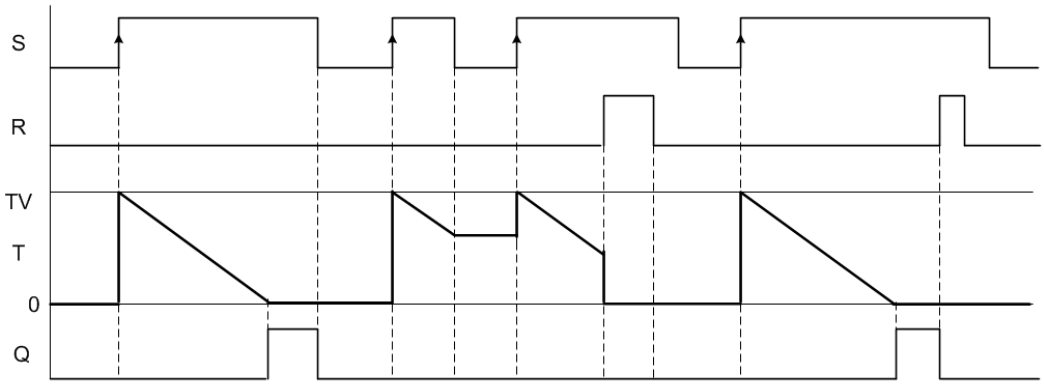
Şekil 9.16 Uzatılmış darbe zamanlayıcısına ilişkin en yalın LAD ve STL programları

9.3.4.3 Gecikmeli Kapatan Zamanlayıcı (S_ODT, SD)

Gecikmeli kapatan zamanlayıcının LAD ve STL dilinde programlanmasına ilişkin komutlar Şekil 9.17'de, bu zamanlayıcı türüne ilişkin zamanlama diyagramı Şekil 9.18'de gösterilmiştir.

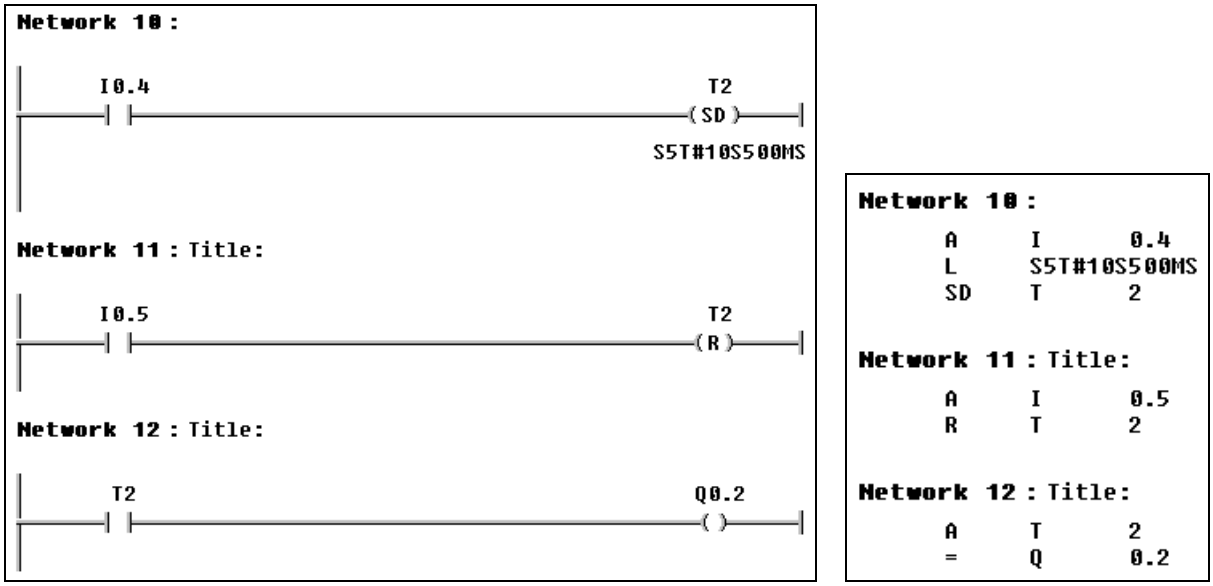


Şekil 9.17 Gecikmeli kapatan zamanlayıcıya ilişkin LAD ve STL programları



Şekil 9.18 Gecikmeli kapatan zamanlayıcıya ilişkin zamanlama diyagramı

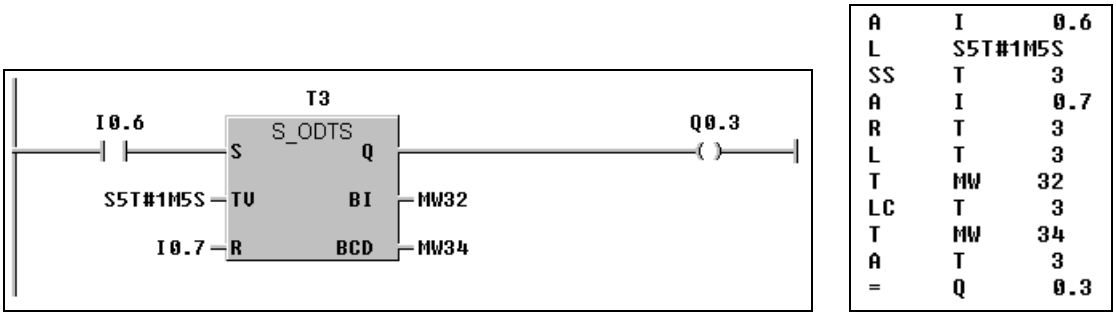
En çok karşılaşılan zamanlı türü olan gecikmeli kapatan zamanlayıcı (ON-DELAY TIMER), zamanlayıcı içeriğinin kullanımını gerektirmeyen uygulamalar için en yalın biçimde Şekil 9.19'da gösterildiği gibi programlanabilir.



Şekil 9.19 Gecikmeli kapatan zamanlayıcıya ilişkin en yalın LAD ve STL programları

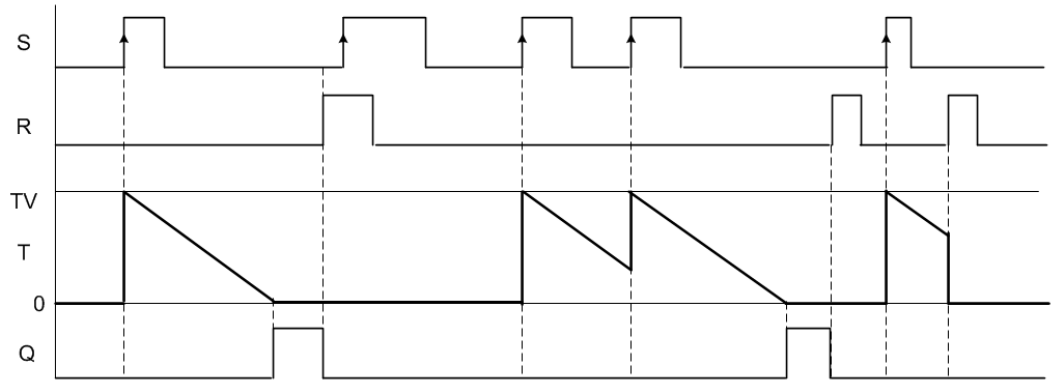
9.3.4.4 Kalıcı Gecikmeli Kapatan Zamanlayıcı (S_ODTS, SS)

Kalıcı gecikmeli kapatan zamanlayıcının LAD ve STL dilinde programlanmasına ilişkin komutlar Şekil 9.20'de, bu zamanlayıcı türüne ilişkin zamanlama diyagramı Şekil 9.21'de gösterilmiştir



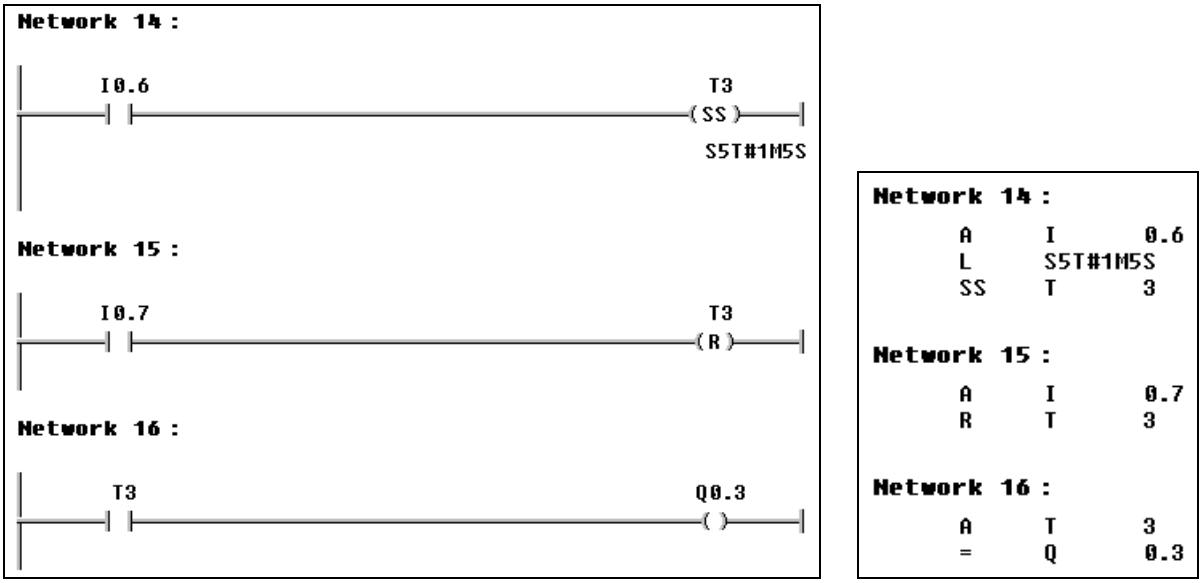
Şekil 9.20 Kalıcı gecikmeli kapatan zamanlayıcıya ilişkin LAD ve STL programları

Bu zamanlamayıcı türünün, gecikmeli kapatan zamanlayıcıdan farkı zamanlayıcı S=0 yapıldığında da sayma sürecinin devam etmesidir.



Şekil 9.21 Kalıcı gecikmeli kapatan zamanlayıcıya ilişkin zamanlama diyagramı

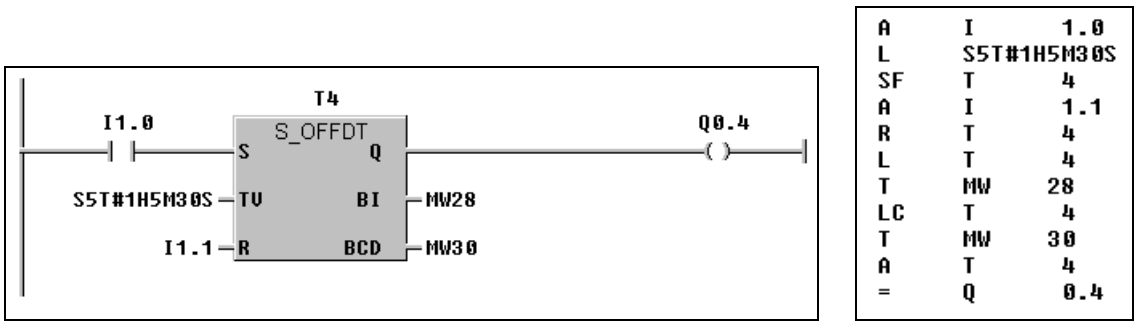
Kalıcı gecikmeli kapatan zamanlayıcı, zamanlayıcı içeriğinin kullanımını gerektirmeyen uygulamalar için en yalın biçimde Şekil 9.22'de gösterildiği gibi programlanabilir.



Şekil 9.22 Kalıcı Gecikmeli kapatan zamanlayıcıya ilişkin en yalın LAD ve STL programları

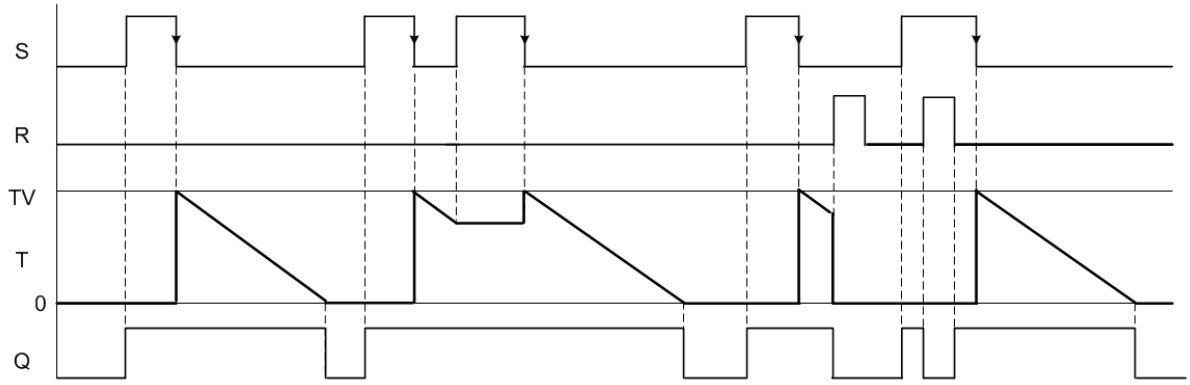
9.3.4.5 Gecikmeli Açan Zamanlayıcı (S_OFFDT, SF)

Gecikmeli açan zamanlayıcının LAD ve STL dilinde programlanmasına ilişkin komutlar Şekil 9.23'te, bu zamanlayıcı türüne ilişkin zamanlama diyagramı Şekil 9.24'te gösterilmiştir




```
A     I     1.0
L     S5T#1H5M30S
SF    T     4
A     I     1.1
R     T     4
L     T     4
T     MW    28
LC    T     4
T     MW    30
A     T     4
=     Q     0.4
```

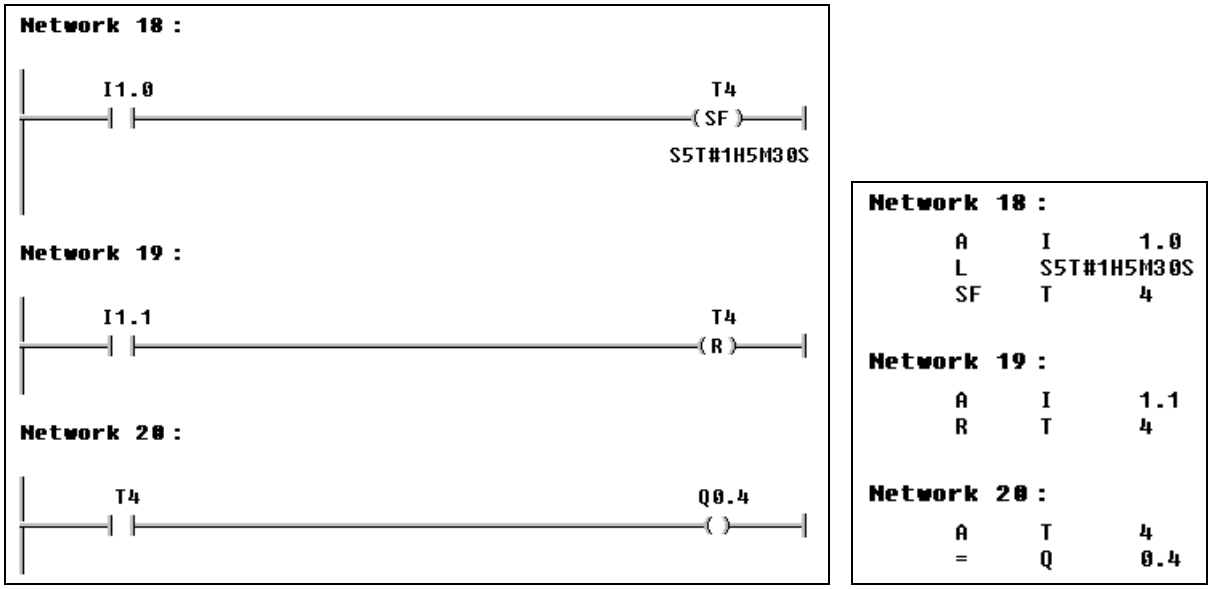
Şekil 9.23 Gecikmeli açan zamanlayıcıya ilişkin LAD ve STL programları



Şekil 9.24 Gecikmeli açan zamanlayıcıya ilişkin zamanlama diyagramı

Bu zamanlayıcıda S=1 yapıldığında Q=0 olmakta, ancak zamanlama süreci S girişinin 1'den 0'a değişmesi ile başlamaktadır.

Gecikmeli açan zamanlayıcı, zamanlayıcı içeriğinin kullanımını gerektirmeyen uygulamalar için en yalın biçimde Şekil 9.22'de gösterildiği gibi programlanabilir.

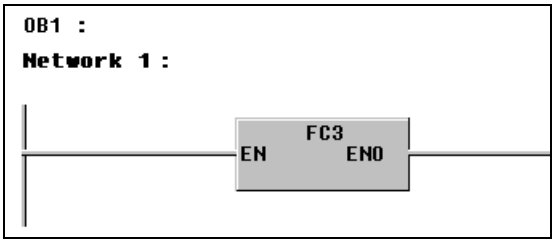


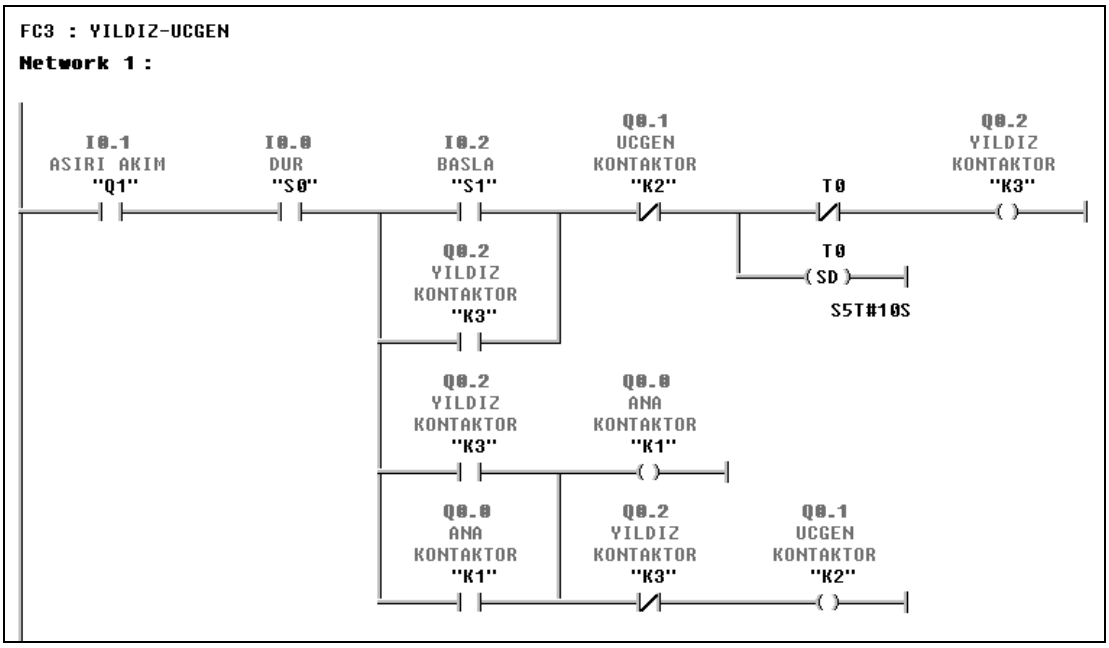
Şekil 9.25 Gecikmeli açan zamanlayıcıya ilişkin en yalın LAD ve STL programları

Örnek 9.1 Kısa devre asenkron motorun kumandasına ilişkin Bölüm 3.4, Şekil 3.13'te verilen röleli kumanda devresi ile aynı işlevi sağlayan ve S7-200 PLC ile çözümü Şekil 3.15'te verilen merdiven mantığı programına karşı düşen S7-300 LAD programını yazınız.

Aynı işlevi sağlayan kumanda devresini gerçeklemek için sembol tablosu, OB1 ve FC3 programları Şekil 9.26'daki gibi düzenlenebilir. Bu program Bölüm 3.4 Şekil 3.13'te verilmiş olan röleli kumanda devresinden yararlanılarak doğrudan yazılabilir.

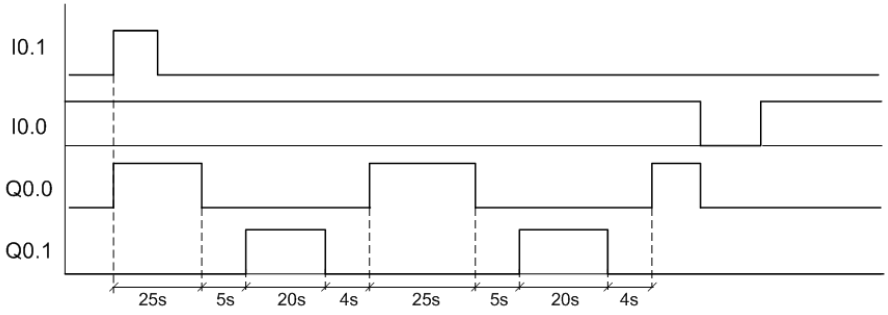
	Status	Symbol	Address	Data type	Comment
1		S0	I 0.0	BOOL	DUR
2		Q1	I 0.1	BOOL	ASIRI AKIM
3		S1	I 0.2	BOOL	BASLA
4		K1	Q 0.0	BOOL	ANA KONTAKTOR
5		K2	Q 0.1	BOOL	UCGEN KONTAKTOR
6		K3	Q 0.2	BOOL	YILDIZ KONTAKTOR





Şekil 9.26 Kısa devre asenkron motorun kumandasına ilişkin S7-300 LAD programı

Örnek 9.2 Zamanlama diyagramı Şekil 9.27'de verilen kumanda sistemine ilişkin programı yazınız.



Şekil 9.27 Zamanlama diyagramı

Zamanlama diyagramına göre

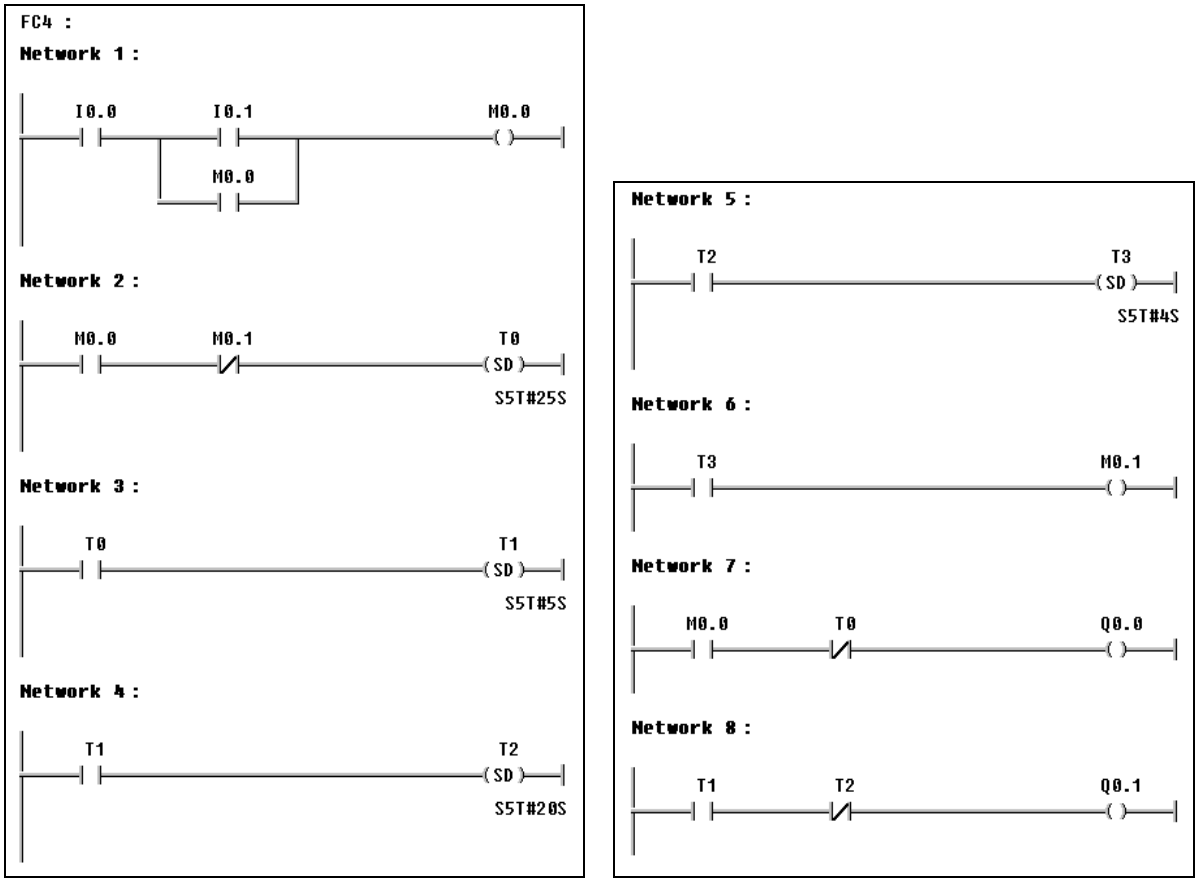
I0.1=1 yapıldığında zamanlama süreci başlamakta,

I0.0=0 yapıldığında süreç sona ermekte,

Sürekli tekrarlanan 54 saniyelik zaman dilimlerinde $t \le 25s$ iken Q0.0=1 ve $30s \le t \le 50s$ iken Q0.1=1 olmaktadır.

Bu uygulama 4 zamanlayıcı veya 1 zamanlayıcı ve karşılaştırma komutları ile gerçeklenebilir.

Şekil 9.28'de 4 zamanlayıcı kullanılarak elde edilen bir çözüme ilişkin program örneği verilmiştir. Burada zamanlayıcılar art arda devreye alınarak 4 ayrı zaman dilimi elde edilmiştir. Şekil 9.29'da zamanlayıcıların art arda devreye alınması ile T0, T1, T2 ve T3 bitlerinin nasıl değiştiği gösterilmiştir. Q0.0 ve Q0.1 çıkışlarının etin olduğu zaman dilimlerindeki zamanlayıcı bitlerinin mantıksal durumu göz önüne alındığında $Q0.0 = M0.0 \bullet \overline{T0}$ ve $Q0.1 = T1 \bullet \overline{T2}$ mantıksal ilişkilerinin yazılabileceği görülür.



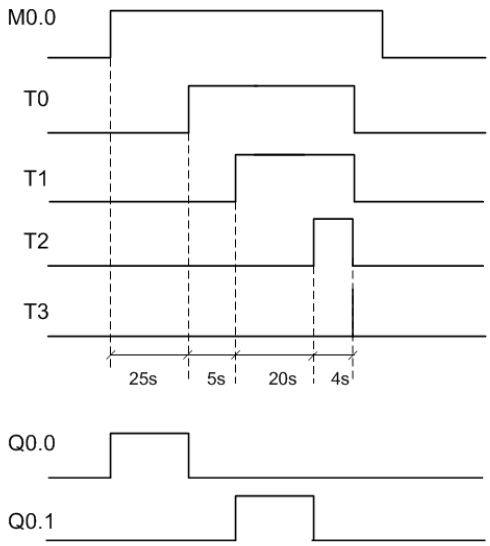
Şekil 9.28 Zamanlayıcılar ile bir çözüme ilişkin program örneği

Bu programda

(Network 1) devreye alma ve devreden çıkarma koşullarını,

(Network 2) - (Network 6) zaman aralıklarını oluşturma

(Network 7) ve (Network 8) $Q0.0 = M0.0 \bullet \overline{T0}$ ve $Q0.1 = T1 \bullet \overline{T2}$ mantıksal ilişkileri gerçekler.

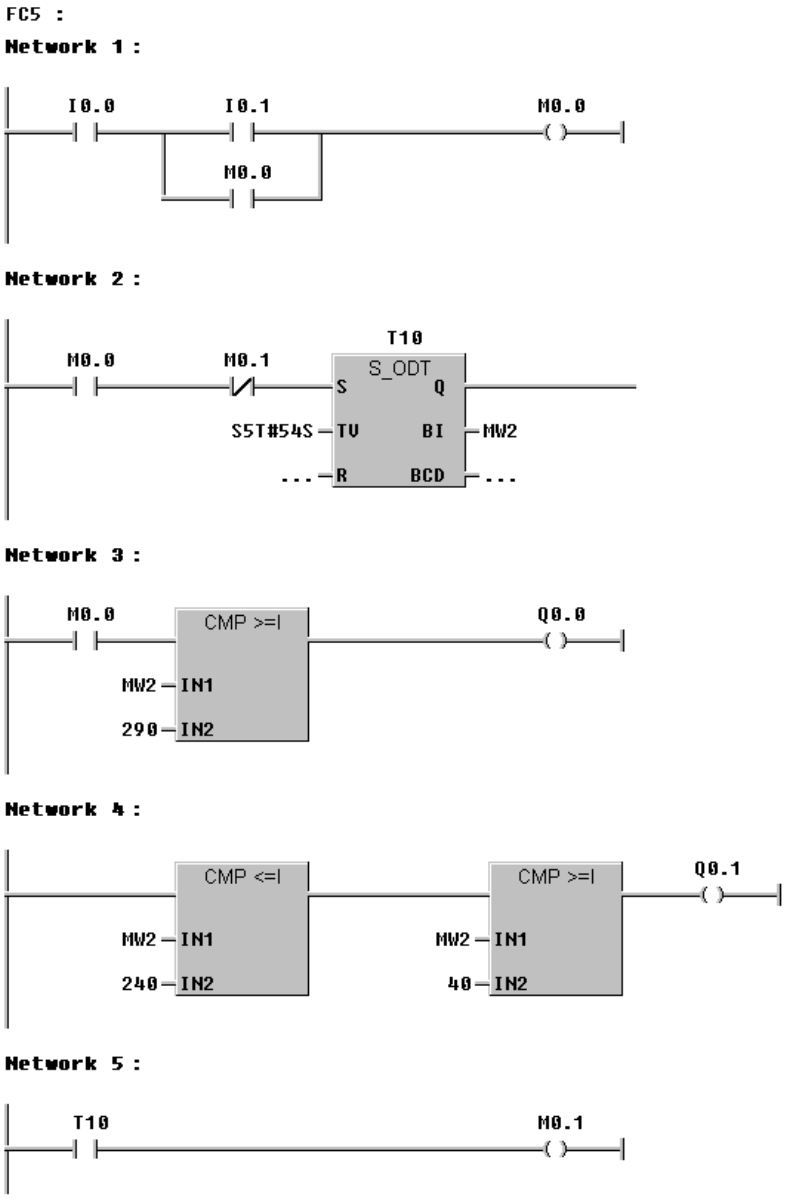


Şekil 9.29 Zamanlayıcıların durum bitleri ve çıkış işaretleri arasındaki ilişki

Bu uygulamanın zamanlayıcı ve karşılaştırma komutları ile çözümüne ilişkin program örneği Şekil 9.30'da verilmiştir. Karşılaştırma komutları S7-200 PLC ailesi karşılaştırma komutları ile benzer özellikler taşır ve aynı biçimde programlanır.

Şekil 9.30'da verilen program örneğinde (Network 2) T10 zamanlayıcısı periyodu 54s olan zamanlama işaretlerini üretmek için kullanılır. Zamanlayıcıya S5T zaman formatında 54s'lik zaman değeri yüklendiğinden zaman çarpanı 100ms olur (Bölüm 9.3.4.1). Buna göre S girişi, 0'dan 1'e değiştirildiğinde zamanlayıcıya tamsayı karşılığı 540 olan değer yüklenir ve 100ms aralıklarla zamanlayıcı içeriği 1 azaltılır. Zamanlayıcı içeriği 0 olduğunda bir tarama çevrimi T10=1 ve buna bağlı olarak S=0 olur, bir tarama sonra zamanlayıcı tekrar sayma sürecine girer ve M0.0=0 yapılıncaya kadar bu durum sürer.

Sayma süreci başladıktan 25s sonra zamanlayıcı içeriği MW2=540-250=290, 30s sonra MW2=540-300=240s ve 50s sonra MW2=540-500=40 olur. Buna göre $MW2 \geq 290$ için Q0.0=1 ve $40 \leq MW2 \leq 240$ için Q0.1=1 koşullarını sağlayan (Network 3) ve (Network 4) program parçaları yazılabilir.



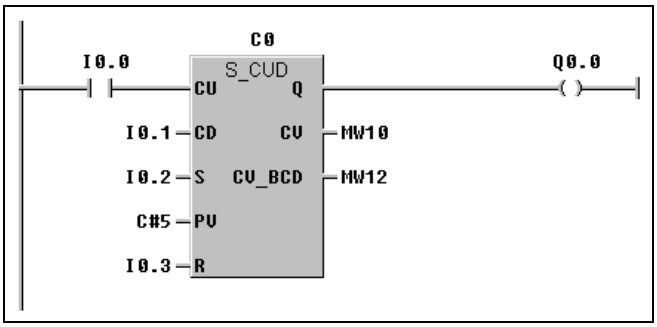
Şekil 9.30 Bir zamanlayıcı ve karşılaştırma komutları ile gerçeklemeye ilişkin program örneği

9.3.5 SAYMA İŞLEMLERİ

Kumanda devrelerinde sayma işlemi gerektiren uygulamalar için sayıcı komutları ve karşılaştırma komutları birlikte kullanılır. Sayma işlemleri ileri-geri sayma, ileri sayma ve geri sayma biçiminde 3 şekilde yapılabilir.

9.3.5.1 İleri-Geri Sayıcı

İleri-geri sayma işlemini gerçeklemeye ilişkin komutların LAD ve STL dilindeki karşılıkları Şekil 9.31'de, sayıcı girişlerinin durumuna bağlı olarak sayıcı içeriği ve çıkışının değişimine ilişkin zamanlama diyagramı Şekil 9.32'de gösterilmiştir.

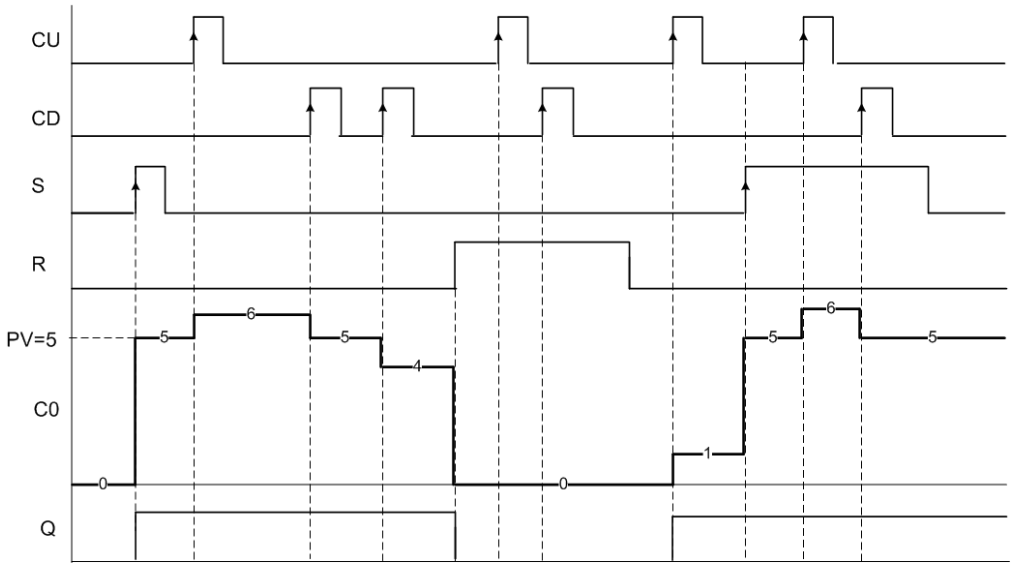



```
A     I     0.0
CU    C     0
A     I     0.1
CD    C     0
A     I     0.2
L     C#5
S     C     0
A     I     0.3
R     C     0
L     C     0
T     MW    10
LC    C     0
T     MW    12
A     C     0
=     Q     0.0
```

Şekil 9.31 İleri-geri sayıcıya ilişkin LAD ve STL programları

Bu sayıcıda CU girişinin 0'dan 1'e değiştirilmesi ile sayıcı içeriği artar, CD girişinin 0'dan 1'e değiştirilmesi ile sayıcı içeriği azalır. S girişinin 0'dan 1'e değiştirilmesi ile PV değeri sayıcıya yüklenir. R girişi 1 yapıldığında sayıcı içeriği sıfırlanır ve bu durumda kalır. Sayıcı içeriği $CV \geq 1$ ise Q=1 olur. Sayıcı içeriği BCD formatında olup geçerli en küçük değer 0, en büyük değer 999 sayısıdır. Sayıcı içeriği 999 ve 0 değerlerini ulaştıktan sonra bu değerlerde kalır.

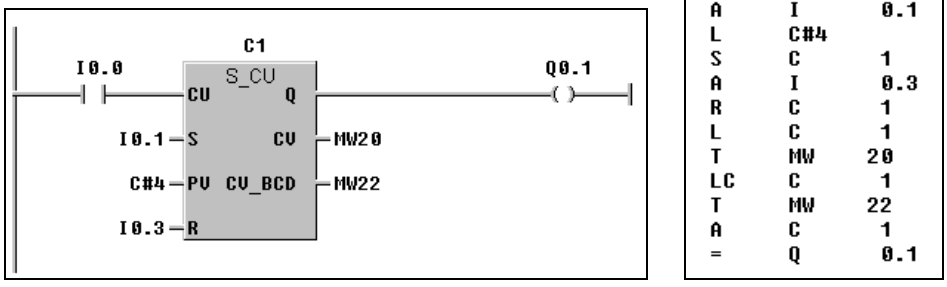
Saymaya ilişkin kumanda işlemlerini gerçeklemek için karşılaştırma komutlarından yararlanılır. Bunun için tamsayı formatındaki CV (MW10) değeri kullanılır.



Şekil 9.32 İleri-geri sayıcıya ilişkin zamanlama diyagramı

9.3.5.2 İleri Sayıcı

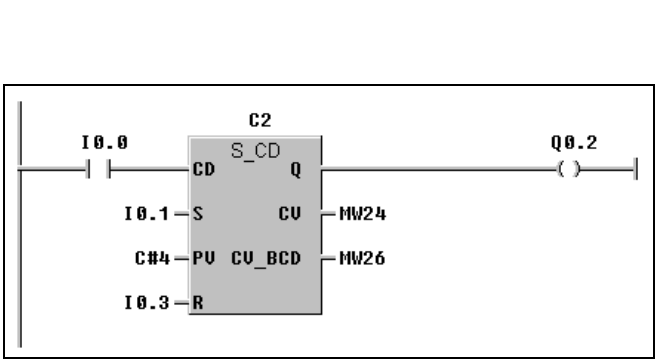
İleri sayma işlemini gerçeklemeye ilişkin komutların LAD ve STL dilindeki karşılıkları Şekil 9.33'te gösterilmiştir. İleri sayıcı ileri-geri sayıcının geri sayma girişi olmayan türüdür. Çalışma biçimi Şekil 9.32'de verilen zamanlama diyagramında görülebilir.



Şekil 9.34 İleri sayıcıya ilişkin LAD ve STL programları

9.3.5.2 Geri Sayıcı

Geri sayma işlemini gerçeklemeye ilişkin komutların LAD ve STL dilindeki karşılıkları Şekil 9.35'te gösterilmiştir. Geri sayıcı ileri-geri sayıcının ileri sayma girişi olmayan türüdür. Çalışma biçimi Şekil 9.32'de verilen zamanlama diyagramında görülebilir.




```
A     I     0.0
CD    C     2
BLD   101
A     I     0.1
L     C#4
S     C     2
A     I     0.3
R     C     2
L     C     2
T     MW    24
LC    C     2
T     MW    26
A     C     2
=     Q     0.2
```

Şekil 9.35 Geri sayıcıya ilişkin LAD ve STL programları

Tüm sayma işlemleri, ileri (CU), geri (CD), sayıcı kurma (SC) ve silme (R) komutlarının kullanımı ile gerçeklenir. Karşılaştırma işlemi için (L) komutu ile alınan sayıcı içeriği kullanılır. Şekil 9.36'da bu komutların kullanımına ilişkin bir program örneği verilmiştir. Bu programa göre

I0.0 girişinin 0'dan 1'e değiştirilmesi ile C10 sayıcı içeriği 1 artar,

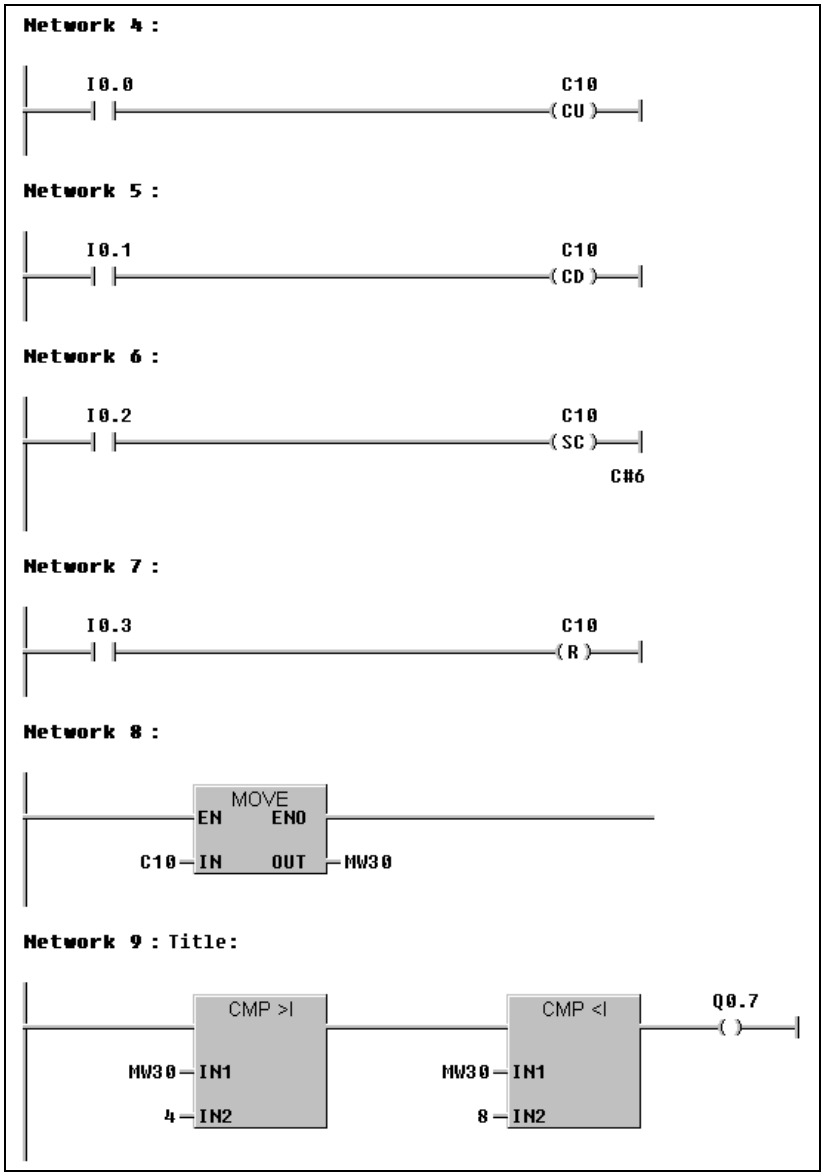
I0.1 girişinin 0'dan 1'e değiştirilmesi ile C10 sayıcı içeriği 1azalır,

I0.2 girişinin 0'dan 1'e değiştirilmesi ile sayıcıya BCD kodlanmış 6 sayısı yüklenir.

I0.3 girişinin 1 yapılması ile sayıcı içeriği sıfırlanır,

Sayıcı içeriği tamsayı formatında MW30 adresine yazılır (Network 8)

Sayıcı içeriği tamsayı olarak karşılaştırılır ve sayıcı içeriği 4'ten büyük ve 8'den küçük ise Q0.7=1 yapılır.




```
Network 4 :
      A     I      0.0
      CU    C      10

Network 5 :
      A     I      0.1
      CD    C      10

Network 6 :
      A     I      0.2
      L     C#6
      S     C      10

Network 7 :
      A     I      0.3
      R     C      10

Network 8 :
      L     C      10
      T     MW     30
      NOP   0

Network 9 : Title:
      A(
      L     MW     30
      L     4
      >I
      )
      A(
      L     MW     30
      L     8
      <I
      )
      =     Q      0.7
```

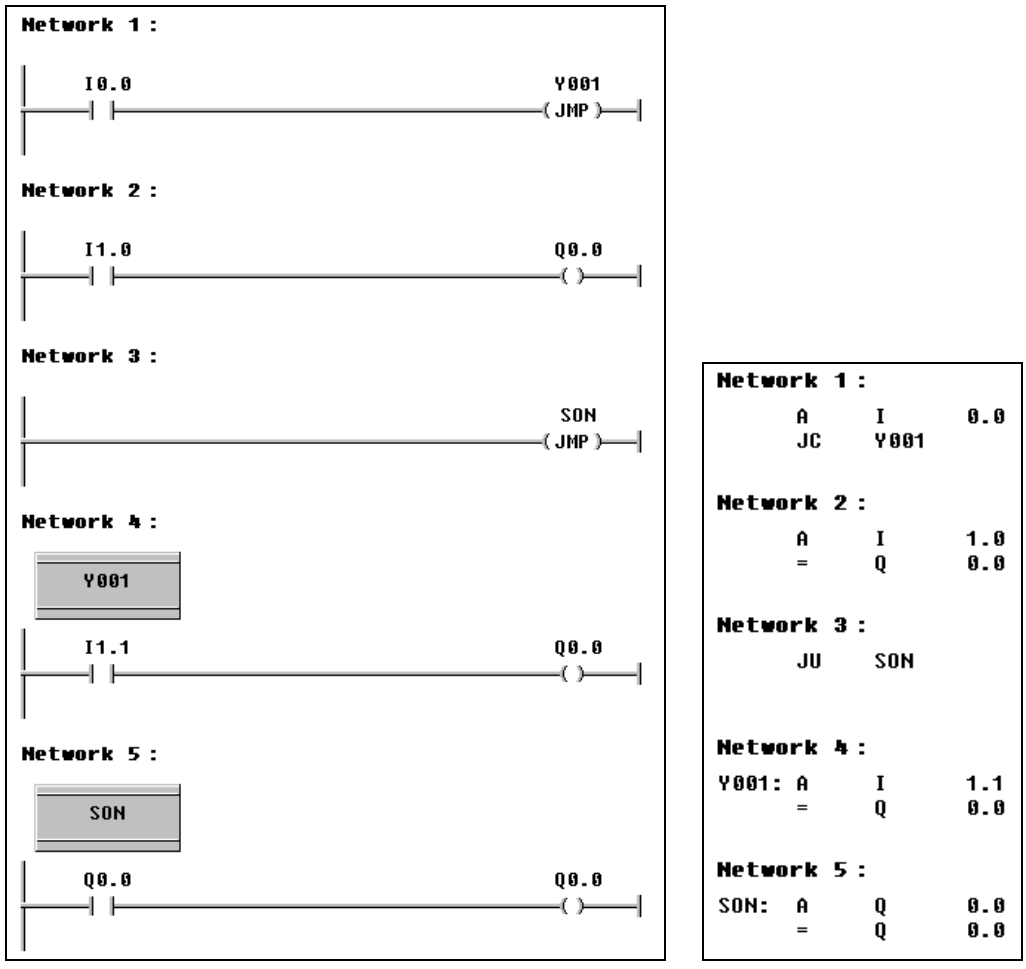
Şekil 9.36 Sayma ve karşılaştırma komutlarının kullanımına ilişkin bir program örneği

9.3.6 PROGRAM DENETİM İŞLEMLERİ

Program denetimi atlama ve ana denetim işlevini sağlayan komutlar ile gerçeklenir.

9.3.6.1 Atlama İşlevini Sağlayan Komutlar

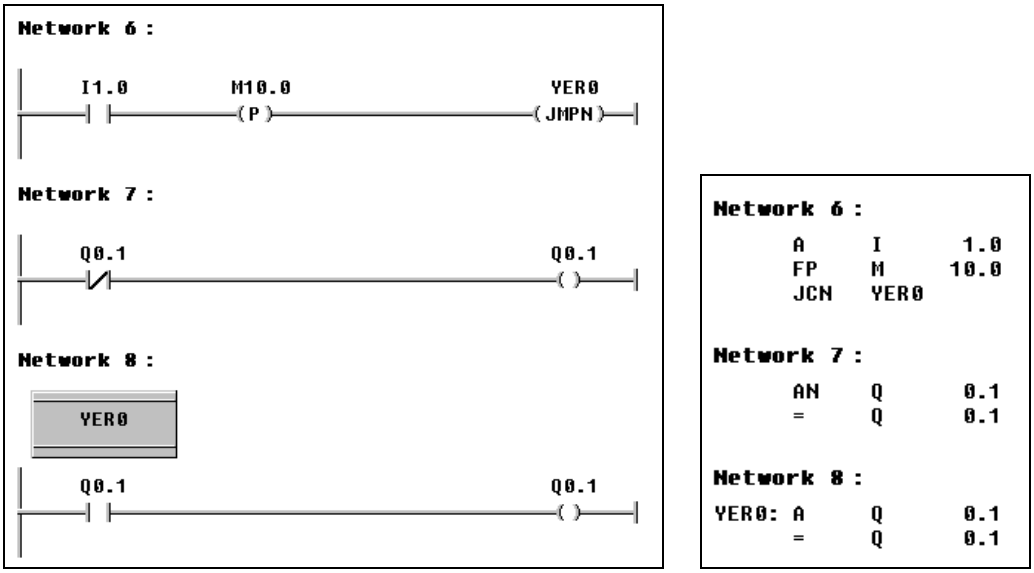
Program akışının denetimi için LAD ve FBD dilinde, koşullu atlama (JC, JCN) koşulsuz atlama (JU) ve atlama adresinin belirtildiği etiket (LABEL) komutları kullanılır. STL dilinde bu amaçla kullanılan çok sayıda komut bulunur. Şekil 9.37 ve Şekil 9.38'de bu komutların kullanımına ilişkin program örnekleri verilmiştir.



Şekil 9.37 Atlama komutlarının kullanımına ilişkin bir program örneği

Şekil 9.37'de verilen programa göre, I0.0=0 iken (Network 2), (Network 3) ve (Network 5) komutları, I0.0=1 yapıldığında (Network 1), (Network 4) ve (Network 5) komutları işlenir. Etiket adresinin bulunduğu alan veya satırın boş bırakılmasına izin verilmez. Bu nedenle, bu alana işlevi olmayan bir komut yazılmalıdır. (Network 5)'te etiket altına yazılan komutlar bu amaçla kullanılmıştır.

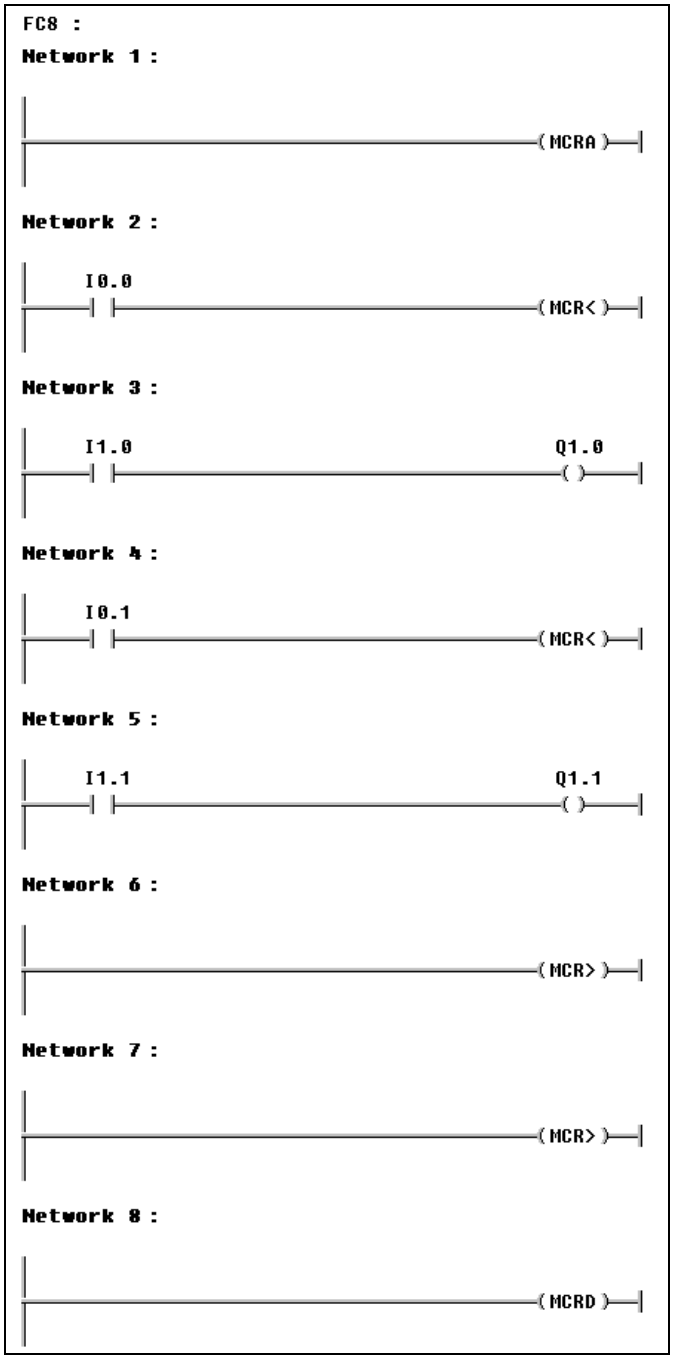
Şekil 9.38'de verilen programa göre, (Network 7)'deki komutlar yalnız I1.0 girişi 0.'dan 1 değiştirildiğinde işletilir. Burada, Q0.1 adresinin tümleyeni alınır ve Q0.1'e yazılır. (Network 8)'te etiket altına yazılan komutlar etiket alanına yazılması gereken işlevsiz komutlardır.



Şekil 9.38 Atlama komutlarının kullanımına ilişkin bir program örneği

9.3.6.2 Ana Denetim İşlevini Sağlayan Komutlar

Ana denetim, programın belirli bir bölgesindeki çıkışların diğer koşullardan bağımsız olarak doğrudan denetim altında tutulması işlemidir. Bu amaçla 4 komut kullanılır. Ana denetimin etkin yapılması için (MCRA) komutu, ana denetimin etkin durumdan çıkartılması için (MCRD), denetim alanı başlangıç noktasını belirtmek için (MCR<) ve denetim alanı son noktasını belirtmek için (MCR>) komutları kullanılır. Şekil 9.39'da ana denetim komutlarının kullanımına ilişkin bir program örneği verilmiştir.




```
FC8 :
Network 1 :
      MCRA

Network 2 :
      A     I     0.0
      MCR(

Network 3 :
      A     I     1.0
      =     Q     1.0

Network 4 :
      A     I     0.1
      MCR(

Network 5 :
      A     I     1.1
      =     Q     1.1

Network 6 :
      )MCR

Network 7 :
      )MCR

Network 8 :
      MCRD
```

Şekil 9.39 Ana denetim komutlarının kullanımına ilişkin bir program örneği

Şekil 9.39'daki programa göre (Network 1)' de yazılan komut ile ana denetim işlevi etkin duruma getirilir. I0.0=1 yapıldığında (Network 2) ile (Network 7) arasındaki çıkışlar, I0.1=1 yapıldığında (Network 4) ile (Network 6) arasındaki çıkışlar programdaki mantıksal ilişkilere göre işletilir. Ana denetim işlevi (Network 8)'e yazılan komutla sonlandırılır. Bu programa göre

I0.0=1 ve I0.1=1 ise Q1.0=I1.0 ve Q1.1=I1.1

I0.0=0 ve I0.1=1 ise Q1.0=0 ve Q1.1=0

I0.0=1 ve I0.1=0 ise Q1.0=I1.0 ve Q1.1=0

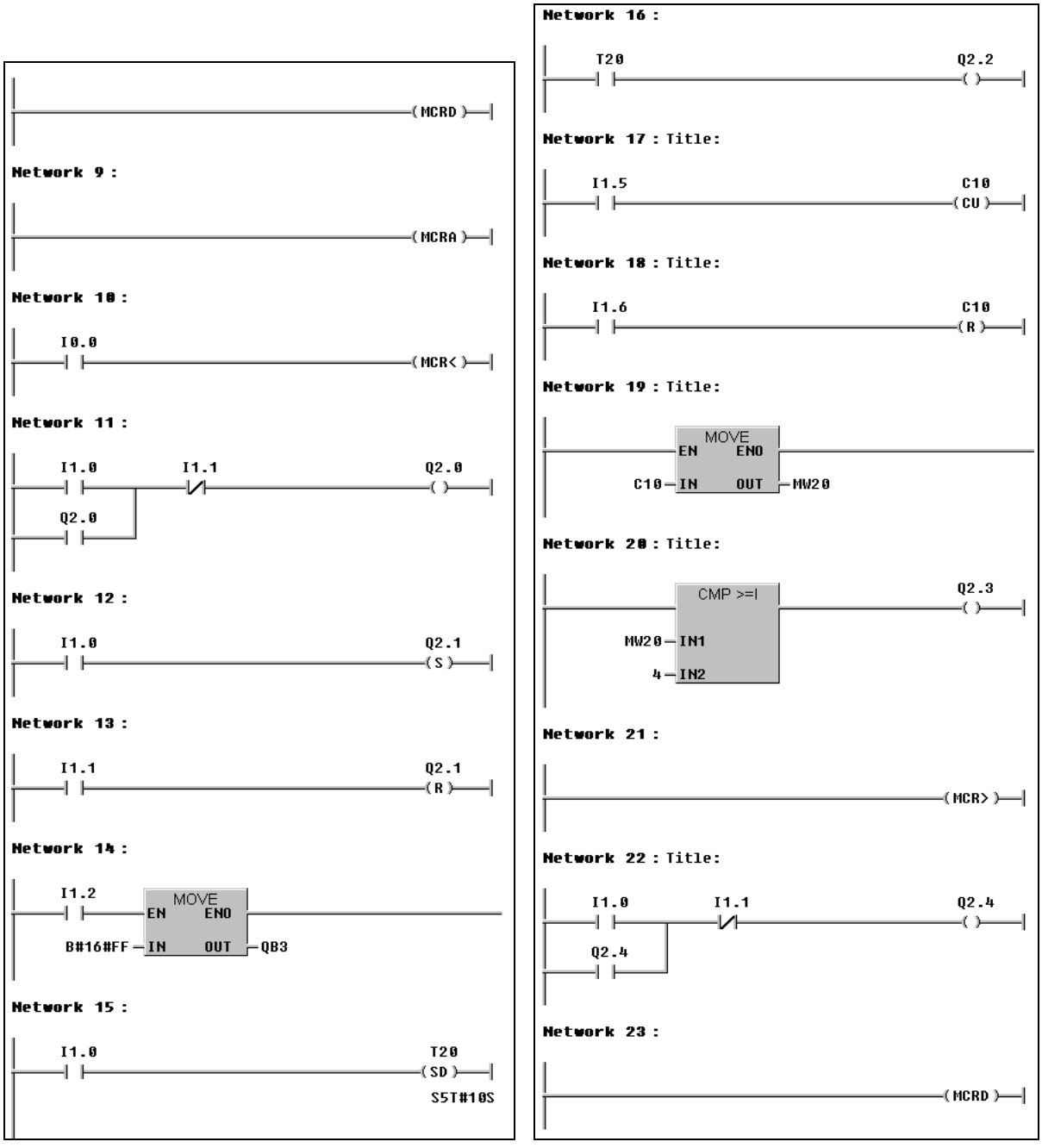
I0.0=0 ve I0.1=0 ise Q1.0=0 ve Q1.1=0

olur.

Ana denetim bölgesi etkin olmadığı zaman bölge içindeki kurma (S), silme (R) ve atama (MOVE, =) komutları işlevini sağlamaz. Zamanlayıcı ve sayıcılar için sayma süreci başlar. Ancak, atama komutları işlevsiz olduğundan zamanlayıcı ile ilgili çıkış etkin olamaz.

Ana denetim bölgesi etkin olduğu zaman bölge içindeki tüm komutlar işlevlerini sağlar. Ana denetim bölgesi etkin durumdan etkin olmayan duruma alınırsa, atama (=, MOVE) komutları kullanılan tüm çıkışlar sıfırlanır, Kurma (S) komutu kullanılan tüm çıkışlar durumunu korurlar. Şekil 9.40'ta ana denetim, S, R, atama, zamanlayıcı ve sayıcı komutlarının kullanımına ilişkin bir program örneği verilmiştir.

Bu programa göre, I0.0=0 iken yalnız zamanlayıcıya ilişkin (Network 15), sayıcıya ilişkin (Network 17-18) ve denetim bölgesi dışındaki (Network 22) komutları işlenir. Buna göre I0.0=0 iken I1.0=1 yapılırsa 10s sonra T20=1 olur. Eğer I1.5 girişi 5 kez 0'dan 1'e değiştirilirse C10=5, I1.6=1 yapılırsa C10=0 olur. I0.0=0 iken atama komutları işlevsiz olduğundanT20=1 olmasına karşın Q2.2=1, MW20>4 koşulu gerçeklenmesine karşın Q2.3=1 olamaz. Ancak, bu durumda iken I0.0=1 yapılırsa Q2.0=1, Q2.1=1 Q2.2=1 ve Q2.3=1 ve Q2.4=1 olur ve tüm komutlar işlenir. Eğer tekrar I0.0=0 yapılırsa Q2.1=1, Q2.4=1 durumlarında kalır, diğer çıkışlar sıfırlanır.



Şekil 9.40 Ana denetim, S, R, atama, zamanlayıcı ve sayıcı komutlarının kullanımına ilişkin bir program örneği

9.3.7 MATEMATİKSEL İŞLEMLER

İşaretli tamsayı ve reel biçimdeki sayısal verilere ilişkin matematiksel işlemler için farklı komutlar kullanılır. 16 ve 32 bitlik işaretli tamsayılar üzerinde dört işlem, reel sayılar üzerinde dört işlem ve bazı karmaşık matematiksel işlemlerin yapılmasına olanak sağlayan komutlar bulunur.

9.3.7.1 İşaretli Tamsayılar Üzerinde İşlemler

İşaretli tamsayılar, değer aralığı {-32768 (H8000), +32767 (H7FFF)} olan 16 bitlik işaretli tamsayılar ve değer aralığı {-2,147,483,654 (H8000,0000), 2,147,483,653 (H7FFF,FFFF)} olan 32 bitlik işaretli tamsayılardan oluşur.

16 bitlik işaretli tamsayılar üzerinde toplama, çıkarma, çarpma ve bölme işlemlerine ilişkin komutların LAD ve STL dilindeki kullanımına ilişkin bir program örneği Şekil 9.41'de verilmiştir.

Bu programa göre I0.0=1 yapılırsa; toplama (Network 1), çıkarma (Network 2), çarpma (Network 3) ve bölme (Network 4) işlemleri yapılır ve sonuç OUT adresine yazılır. Eğer, sonuç değer aralığı içinde değilse taşma biti OV=1 ve buna bağlı olarak ENO=0 olur. Bu durum sonucun geçersiz olduğunu gösterir.

Sonucun geçersiz olduğu durumlara örnek olarak aşağıdaki işlemler verilebilir:

MW20= +20000, MW22= +12768 için toplama işlemi yapılırsa MW24 =-32768 ve M0.0=0 olur,

MW20= -20000, MW22= -12769 için toplama işlemi yapılırsa MW24 =-32767 ve M0.0=0 olur,

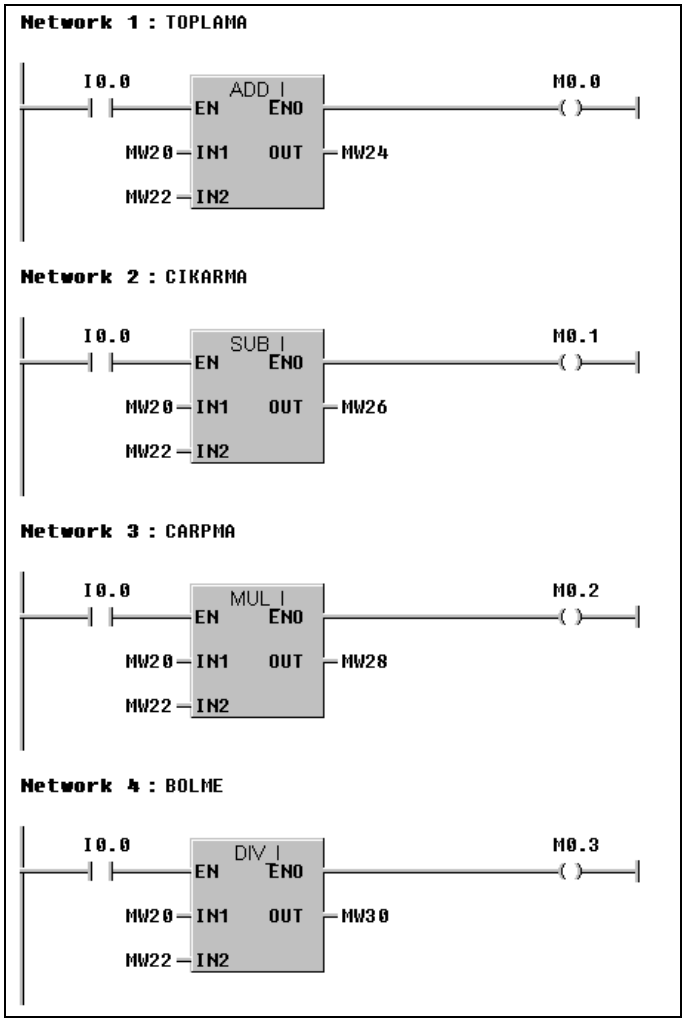
MW20= -20000, MW22= +12769 için çıkarma işlemi yapılırsa MW26 = 32767 ve M0.1=0 olur,

MW20= +16384, MW22= +2 için çarpma işlemi yapılırsa MW28 = -32768 ve M0.2=0 olur,

MW20= -32768, MW22= -1 için bölme işlemi yapılırsa MW30 =-32768 ve M0.3=0 olur,

MW20= 16384, MW22= 0 için bölme işlemi yapılırsa MW30 = 0 ve M0.3=0 olur.

32 bitlik işaretli tamsayılar üzerinde toplama, çıkarma, çarpma ve bölme işlemlerine ilişkin komutların LAD ve STL dilindeki kullanımına ilişkin bir program örneği Şekil 9.42'de verilmiştir. Bu programa göre I0.0=1 yapılırsa; toplama (Network 5), çıkarma (Network 6), çarpma (Network 7) ve bölme (Network 8) işlemleri yapılır ve sonuç OUT adresine yazılır. Eğer, sonuç değer aralığı içinde değilse taşma biti OV=1 ve buna bağlı olarak ENO=0 olur. Bu durum sonucun geçersiz olduğunu gösterir.



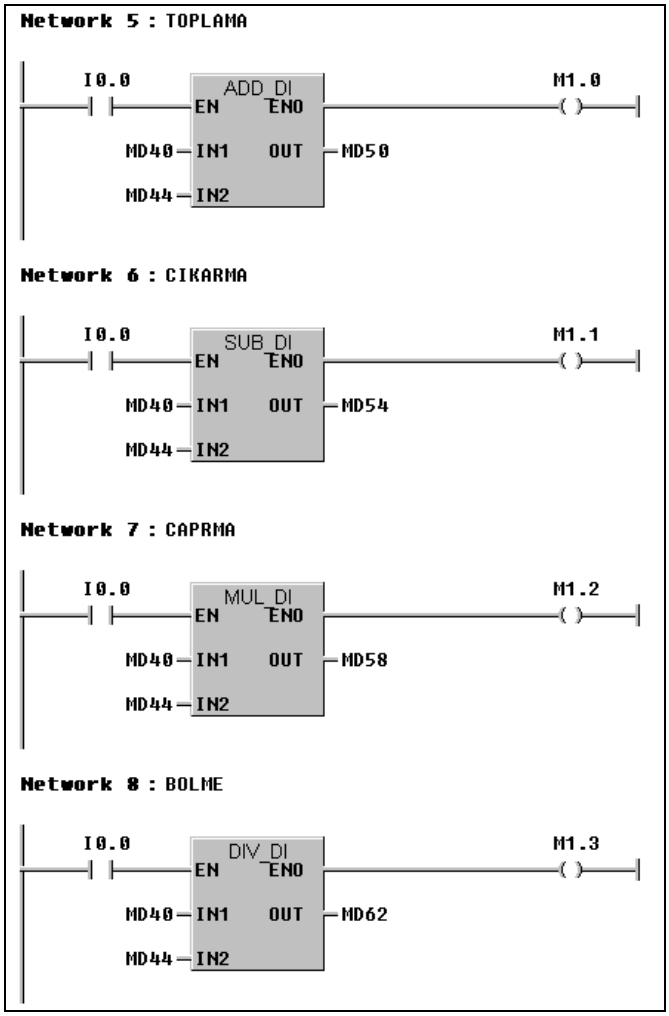

```
Network 1 : TOPLAMA
      A     I      0.0
      JNB   _001
      L     MW     20
      L     MW     22
      +I
      T     MW     24
      AN    OV
      SAVE
      CLR
_001: A     BR
      =     M      0.0

Network 2 : CIKARMA
      A     I      0.0
      JNB   _002
      L     MW     20
      L     MW     22
      -I
      T     MW     26
      AN    OV
      SAVE
      CLR
_002: A     BR
      =     M      0.1

Network 3 : CARPMA
      A     I      0.0
      JNB   _003
      L     MW     20
      L     MW     22
      *I
      T     MW     28
      AN    OV
      SAVE
      CLR
_003: A     BR
      =     M      0.2

Network 4 : BOLME
      A     I      0.0
      JNB   _004
      L     MW     20
      L     MW     22
      /I
      T     MW     30
      AN    OV
      SAVE
      CLR
_004: A     BR
      =     M      0.3
```

Şekil 9.41 16 bitlik işaretli tamsayılar üzerine matematiksel işlem komutları




```
Network 5 : TOPLAMA

      A     I      0.0
      JNB   _005
      L     MD     40
      L     MD     44
      +D
      T     MD     50
      AN    OV
      SAVE
      CLR
_005: A     BR
      =     M      1.0

Network 6 : CIKARMA
      A     I      0.0
      JNB   _006
      L     MD     40
      L     MD     44
      -D
      T     MD     54
      AN    OV
      SAVE
      CLR
_006: A     BR
      =     M      1.1

Network 7 : CAPRMA
      A     I      0.0
      JNB   _007
      L     MD     40
      L     MD     44
      *D
      T     MD     58
      AN    OV
      SAVE
      CLR
_007: A     BR
      =     M      1.2

Network 8 : BOLME
      A     I      0.0
      JNB   _008
      L     MD     40
      L     MD     44
      /D
      T     MD     62
      AN    OV
      SAVE
      CLR
_008: A     BR
      =     M      1.3
```

Şekil 9.42 32 bitlik işaretli tamsayılar üzerine matematiksel işlem komutları

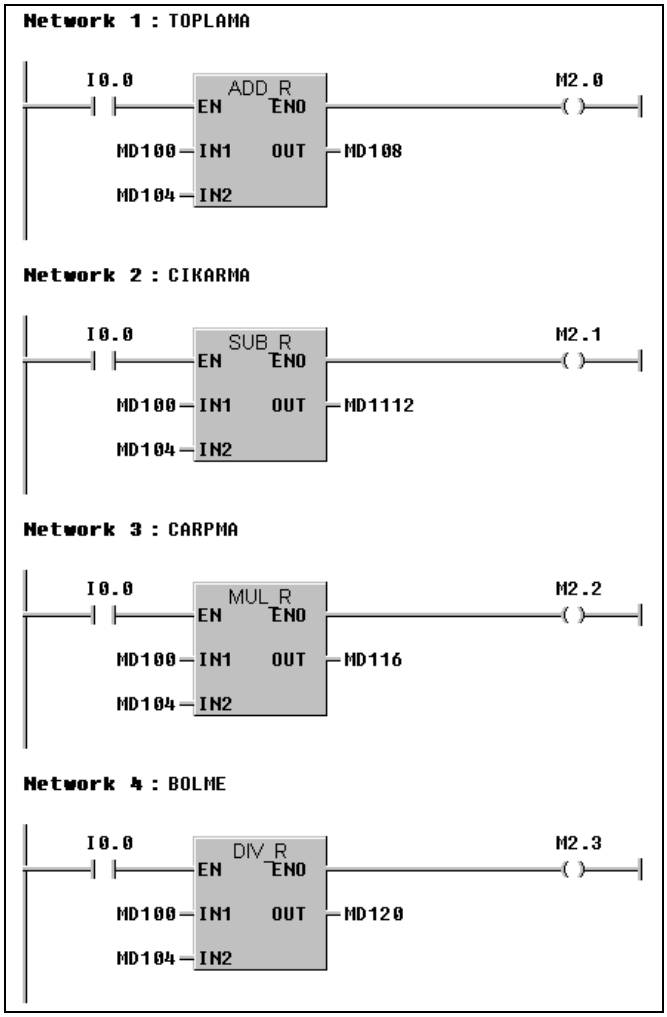
9.3.7.2 Reel Sayılar Üzerinde İşlemler

Reel sayılar, $\{-3.402823\times10^{38}$, -1.175495×10^{-38}, $+1.175495\times10^{-38}$, $+3.402823\times10^{38}\}$ aralığında değişen sayılardır. Reel sayılar üzerinde toplama, çıkarma, çarpma ve bölme işlemlerine ilişkin komutların LAD ve STL dilindeki kullanımına ilişkin bir program örneği Şekil 9.43'te verilmiştir.

Bu programa göre I0.0=1 yapılırsa; toplama (Network 1), çıkarma (Network 2), çarpma (Network 3) ve bölme (Network 4) işlemleri yapılır ve sonuç OUT adresine yazılır. Eğer sonuç, değer aralığı içinde değilse taşma biti OV=1 ve buna bağlı olarak ENO=0 olur. Bu durum sonucun geçersiz olduğunu gösterir. Örneğin, sıfıra bölme veya işlem sonucu elde edilen sayı $+3.402823\times10^{38}$ değerinden büyük veya -3.402823×10^{38} değerinden küçük ise taşma biti OV=1 olur ve buna bağlı olarak ENO=0 olur.

Reel sayılar üzerinde yapılan diğer matematiksel işlemlerden mutlak değer (ABS), karekök (SQRT), kare (SQR), doğal logaritma (LN) ve üstel fonksiyon (EXP) alma işlemlerine ilişkin bir program örneği Şekil 9.44'te, trigonometrik fonksiyonlara ilişkin bir program örneği Şekil 9.45'te ve ters trigonometrik fonksiyonlara ilişkin bir program örneği Şekil 9.46'da verilmiştir.

Matematik işlem komutlarının işlenmesi ile elde edilen sonucun geçerli bir reel sayı olmaması durumunda ENO=0 olur. Örneğin, negatif bir reel sayının karekökü veya logaritmasının alınması durumunda veya {-1.0, 1.0} aralığı dışında bir reel sayının "ASIN" ve "ACOS" işlemlerinde ENO=0 olur.




```
Network 1 : TOPLAMA

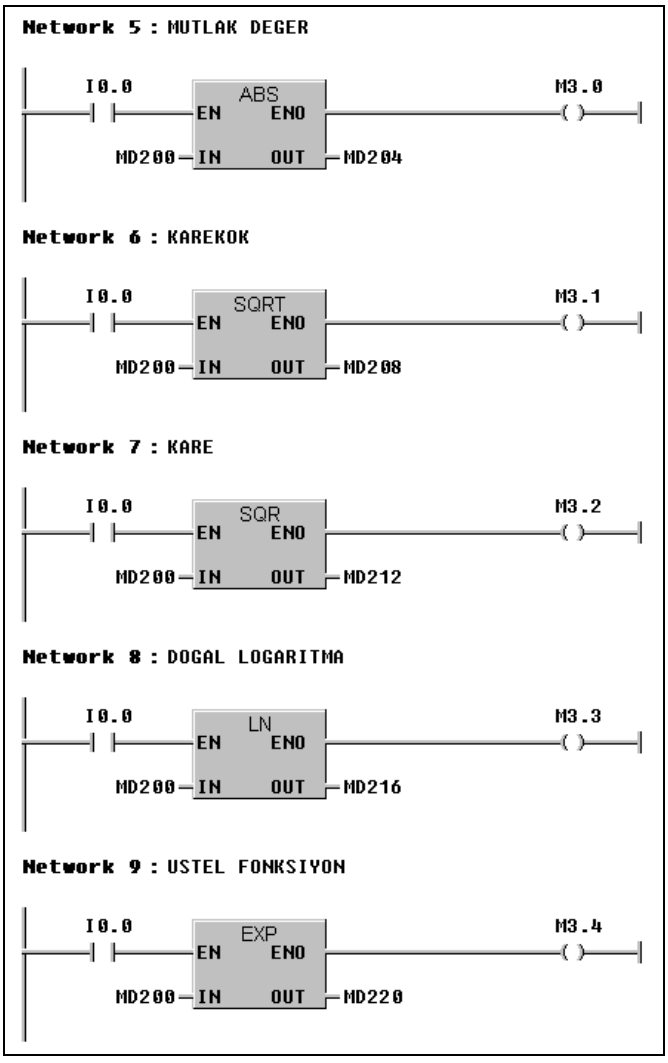
      A     I      0.0
      JNB   _001
      L     MD   100
      L     MD   104
      +R
      T     MD   108
      AN    OV
      SAVE
      CLR
_001: A     BR
      =     M      2.0

Network 2 : CIKARMA
      A     I      0.0
      JNB   _002
      L     MD   100
      L     MD   104
      -R
      T     MD  1112
      AN    OV
      SAVE
      CLR
_002: A     BR
      =     M      2.1

Network 3 : CARPMA
      A     I      0.0
      JNB   _003
      L     MD   100
      L     MD   104
      *R
      T     MD   116
      AN    OV
      SAVE
      CLR
_003: A     BR
      =     M      2.2

Network 4 : BOLME
      A     I      0.0
      JNB   _004
      L     MD   100
      L     MD   104
      /R
      T     MD   120
      AN    OV
      SAVE
      CLR
_004: A     BR
      =     M      2.3
```

Şekil 9.43 Reel sayılar üzerine matematiksel işlem komutları




```
Network 5 : MUTLAK DEGER

      A     I      0.0
      JNB   _005
      L     MD   200
      ABS
      T     MD   204
      SET
      SAVE
      CLR
_005: A     BR
      =     M      3.0

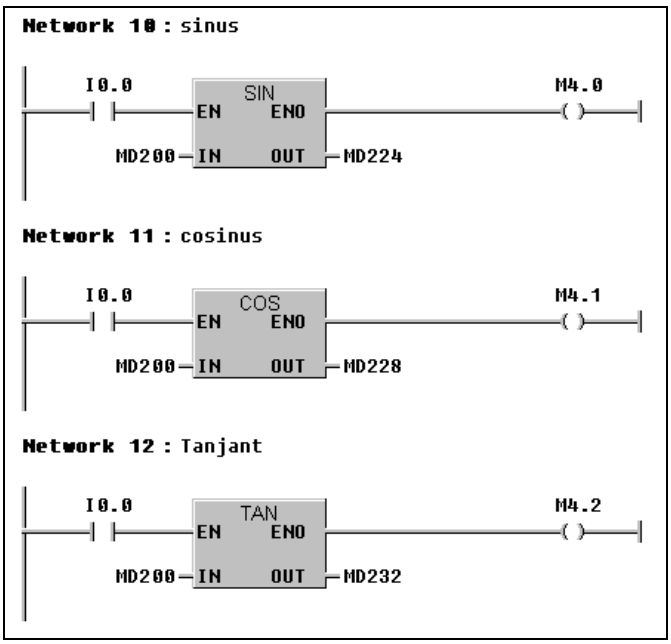
Network 6 : KAREKOK
      A     I      0.0
      JNB   _006
      L     MD   200
      SQRT
      T     MD   208
      AN    OV
      SAVE
      CLR
_006: A     BR
      =     M      3.1

Network 7 : KARE
      A     I      0.0
      JNB   _007
      L     MD   200
      SQR
      T     MD   212
      AN    OV
      SAVE
      CLR
_007: A     BR
      =     M      3.2

Network 8 : DOGAL LOGARITMA
      A     I      0.0
      JNB   _008
      L     MD   200
      LN
      T     MD   216
      AN    OV
      SAVE
      CLR
_008: A     BR
      =     M      3.3

Network 9 : USTEL FONKSIYON
      A     I      0.0
      JNB   _009
      L     MD   200
      EXP
      T     MD   220
      AN    OV
      SAVE
      CLR
_009: A     BR
      =     M      3.4
```

Şekil 9.44 Reel sayılar üzerine diğer matematiksel işlem komutları



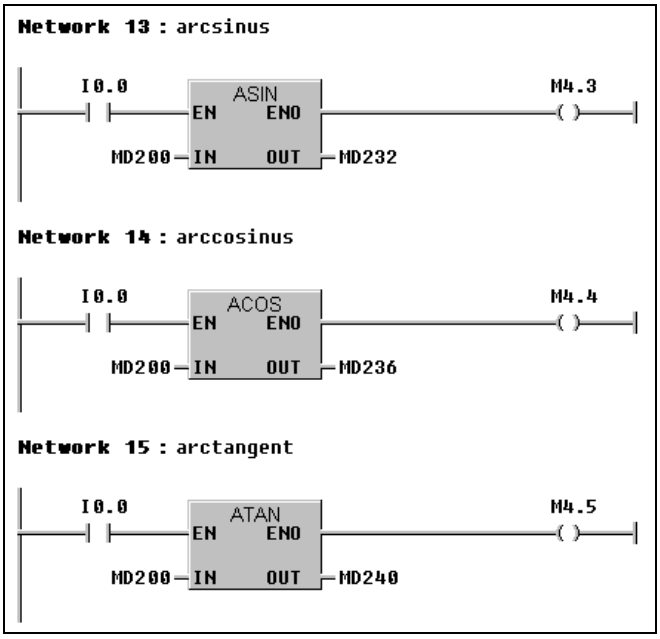

```
Network 10 : sinus
      A     I      0.0
      JNB   _00a
      L     MD   200
      SIN
      T     MD   224
      AN    OV
      SAVE
      CLR
_00a: A     BR
      =     M      4.0

Network 11 : cosinus
      A     I      0.0
      JNB   _00b
      L     MD   200
      COS
      T     MD   228
      AN    OV
      SAVE
      CLR
_00b: A     BR
      =     M      4.1

Network 12 : Tanjant
      A     I      0.0
      JNB   _00c
      L     MD   200
      TAN
      T     MD   232
      AN    OV
      SAVE
      CLR
_00c: A     BR
      =     M      4.2
```

Şekil 9.45 Reel sayılar üzerinde trigonometrik fonksiyonlara ilişkin komutlar

Şekil 9.45'teki yazılan komutlara göre, I0.0=1 yapıldığında MD200 adresine yazılan reel sayıya (radyan) sinüs, kosinüs ve tanjant fonksiyonları uygulanır ve sonuçlar MD224, MD228 ve MD332 adreslerine yazılır.




```
Network 13 : arcsinus
      A     I      0.0
      JNB   _00d
      L     MD   200
      ASIN
      T     MD   232
      AN    OV
      SAVE
      CLR
_00d: A     BR
      =     M      4.3

Network 14 : arccosinus
      A     I      0.0
      JNB   _00e
      L     MD   200
      ACOS
      T     MD   236
      AN    OV
      SAVE
      CLR
_00e: A     BR
      =     M      4.4

Network 15 : arctangent
      A     I      0.0
      JNB   _00f
      L     MD   200
      ATAN
      T     MD   240
      AN    OV
      SAVE
      CLR
_00f: A     BR
      =     M      4.5
```

Şekil 9.46 Reel sayılar üzerinde ters trigonometrik fonksiyonlara ilişkin komutlar

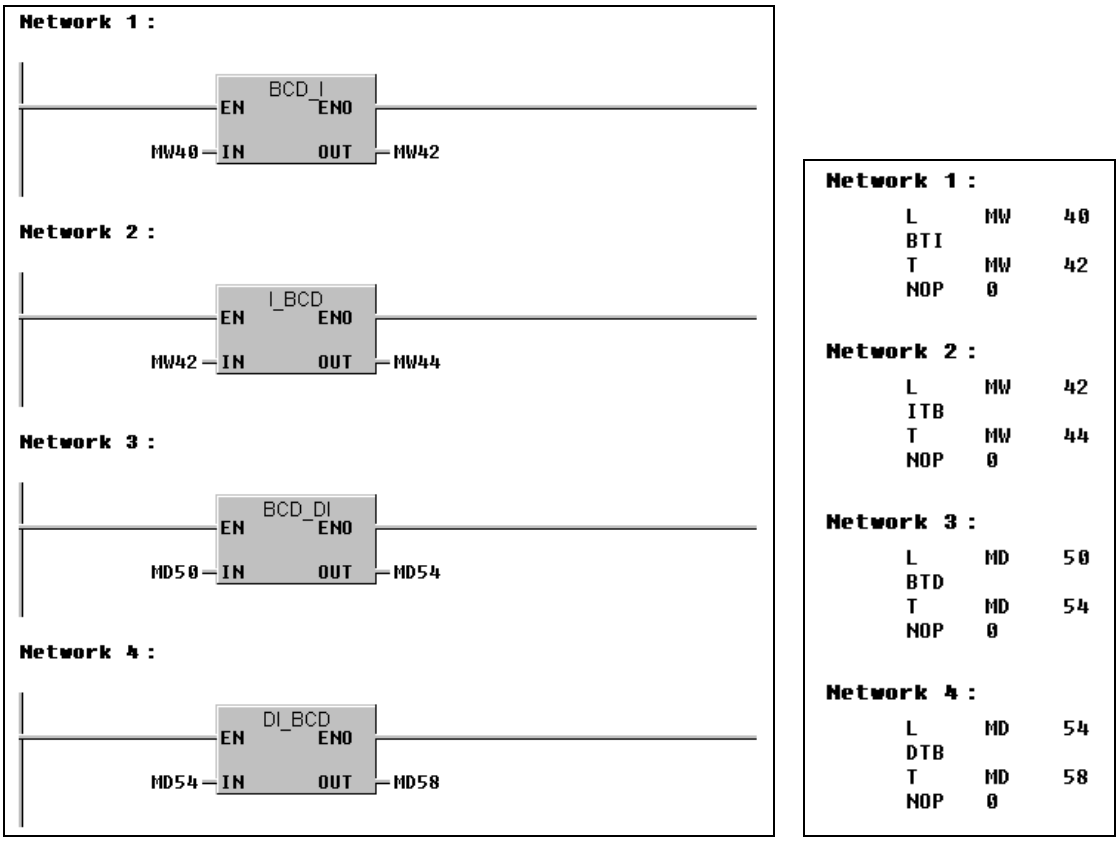
Şekil 9.46'da yazılan komutlara göre, I0.0=1 yapıldığında MD200 adresine yazılan reel sayıların (radyan) ter sinüs, ters kosinüs ve ters tanjant fonksiyonları uygulanır ve sonuçlar MD232, MD236 ve MD240 adreslerine yazılır. Eğer MD200 adresindeki reel sayı {-1.0, 1.0} aralığı dışında ise M4.3=0 ve M4.4=0 olur.

9.3.8 DÖNÜŞTÜRME İŞLEMLERİ

İkilik kodlanmış ondalık sayı (BCD), 16 bitlik işaretli tamsayı (INT), 32 bitlik işaretli tamsayı (DINT) ve reel (REAL) sayılar arasında dönüşümü gerektiren uygulamaların gerçeklenmesinde dönüşüm komutları kullanılır.

9.3.8.1 BCD Türünde Sayılar ile Tamsayılar Arasında Dönüşüm İşlemleri

BCD türündeki sayılar ile işaretli tamsayılar arasında dönüşümü sağlayan komutların LAD ve STL dilindeki kullanımına ilişkin bir program örneği Şekil 9.47'de verilmiştir.



Şekil 9.47 BCD sayılar ile işaretli tamsayılar arasında dönüşümü sağlayan LAD ve STL komutları

Bu programa göre (Network 1)'de MW40 adresinden okunan 16 bitlik BCD türündeki bir sayı, 16 bitlik işaretli tamsayıya dönüştürülür ve sonuç MW42 adresine yazılır. Eğer MW40 adresinin içeriği

BCD formatında değilse, BCD dönüştürme hatası nedeniyle sistem hatası oluşur ve PLC "STOP" durumuna geçer.

Şekil 9.47'deki program parçasında; 16 bitlik işaretli tamsayıyı 16 bitlik BCD sayıya dönüştürmek için (Network 2), 32 bitlik BCD türünde bir sayının 32 bitlik işaretli tamsayıya dönüştürülmesi için (Network 3) ve 32 bitlik işaretli tamsayının 32 bitlik BCD sayıya dönüştürülmesi (Network 4) komutları ile sağlanır. Eğer, MD50 adresinin içeriği BCD formatında değilse, BCD dönüştürme hatası nedeniyle PLC "STOP" durumuna geçer.

9.3.8.2 İşaretli Tamsayılar ile Reel Sayılar Arasında Dönüşüm İşlemleri

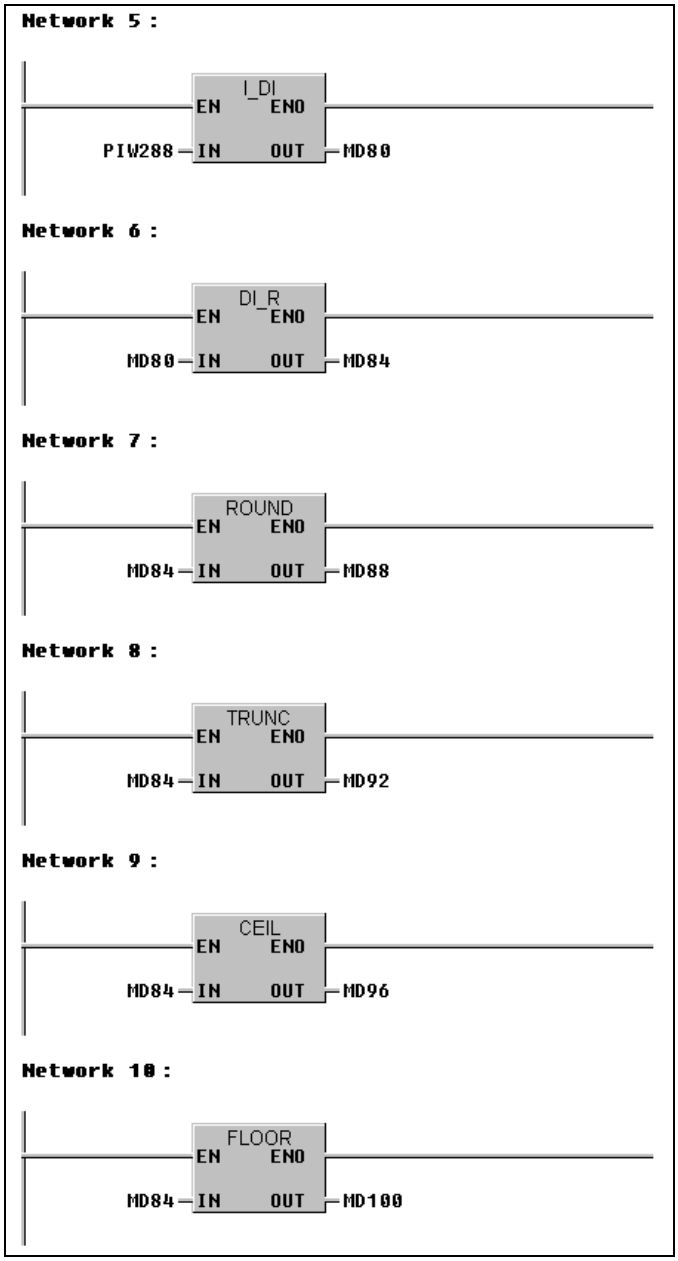
İşaretli tamsayılar ile reel sayılar arasında dönüşümü sağlayan komutların LAD ve STL dilindeki kullanımına ilişkin bir program örneği Şekil 9.48'de verilmiştir. Bu programa göre analog giriş biriminden işaretli tamsayı olarak okunan PIW288 adresindeki sayısal veri, 32 bitlik işaretli tamsayıya dönüştürülür ve MD80 adresine yazılır. MD80 adresindeki 32 bitlik işaretli tamsayı reel sayıya dönüştürülerek MD84 adresine yazılır. Kontrol uygulamalarında, analog giriş biriminden okunan 16 bitlik işaretli tamsayı biçimindeki veriler, bu şekilde reel sayılara dönüştürülür ve kontrol kuralına ilişkin işlemlerin reel sayılarla yapılması sağlanır.

Reel sayı olarak elde edilen bir sayısal verinin analog çıkış birimine yazılması için bu sayı 16 bitlik işaretli tamsayıya dönüştürülür. Bu amaçla; yuvarlama (ROUND, RND), kırpma (TRUNC, TRUNC), yukarı yuvarlama (CEIL, RND+) ve aşağı yuvarlama (FLOOR, RND-) komutları kullanılır. RND komutu reel sayıyı en yakın işaretli tamsayıya yuvarlar. Örneğin,

101.4 sayısı RND, TRUNC ve RND- komutları ile 101, RND+ komutu ile 102 tamsayısına dönüştürülür.

101.51 reel sayısı RND ve RND+ komutu ile 102, TRUNC ve RND- komutu ile 101 sayısına dönüştürülür.

Eğer reel sayı -32768.0 ile 32767.0 aralığında ise "MOVE" komutu ile reel sayı 16 bitlik işaretli tamsayıya dönüştürülür. "MOVE" komutunun kullanımı bir sonraki alt bölümde açıklanacak ve bu konu ile ilgili bir örnek verilecektir.




```
Network 5 :
      L     PIW  288
      ITD
      T     MD    80
      NOP   0

Network 6 :
      L     MD    80
      DTR
      T     MD    84
      NOP   0

Network 7 :
      L     MD    84
      RND
      T     MD    88
      NOP   0

Network 8 :
      L     MD    84
      TRUNC
      T     MD    92
      NOP   0

Network 9 :
      L     MD    84
      RND+
      T     MD    96
      NOP   0

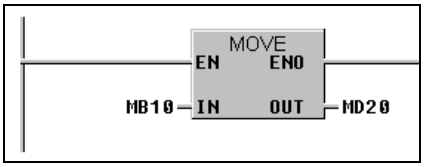
Network 10 :
      L     MD    84
      RND-
      T     MD   100
      NOP   0
```

Şekil 9.48 İşaretli tamsayılar ve reel sayılar arasında dönüşümü sağlayan LAD ve STL komutları

9.3.9 VERİ AKTARMA İŞLEMLERİ

Verilerin bir adresten başka bir adrese aktarılması için yükleme (L) ve aktarma (T) komutlarından oluşan "MOVE" komutu kullanılır. Bu komutlar ile değişik boyutlardaki adresler arasında veri aktarımı gerçeklenebilir.

8 bitlik bir verinin 32 bitlik bir alana aktarılması için MOVE komutu, LAD ve STL dilinde



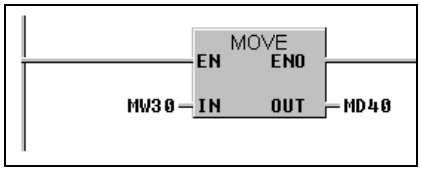

```
L     MB    10
T     MD    20
NOP   0
```

biçiminde yazılır ve komutların yürütülmesi ile 8 bitlik MB10 adresinin içeriği 32 bitlik MD32 adresine

MB 10 B#16#EF ⟶ MD 20 DW#16#000000EF

düzeninde kaydedilir.

16 bitlik bir verinin 32 bitlik bir alana aktarılması için MOVE komutu,



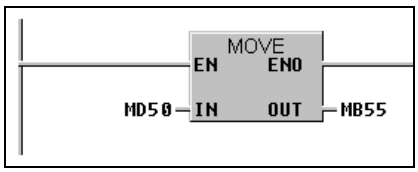

```
L     MW    30
T     MD    40
NOP   0
```

biçiminde yazılır ve 16 bitlik MW30 adresinin içeriği 32 bitlik MD40 adresine

MW 30 W#16#ABCD ⟶ MD 40 DW#16#0000ABCD

olarak kaydedilir.

32 bitlik bir verinin 8 bitlik bir adrese kaydedilmesi için



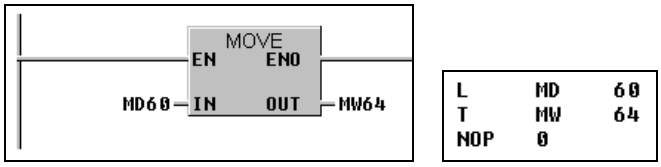

```
L     MD    50
T     MB    55
NOP   0
```

komutları yazılır ve bu durumda MD50 adresinin içeriğinin MB55 adresine kaydedilmesi

MD 50 DW#16#ABCDEF12 ⟶ MB 55 B#16#12

biçiminde gerçeklenir.

32 bitlik bir verinin 16 bitlik bir adrese kaydedilmesi için



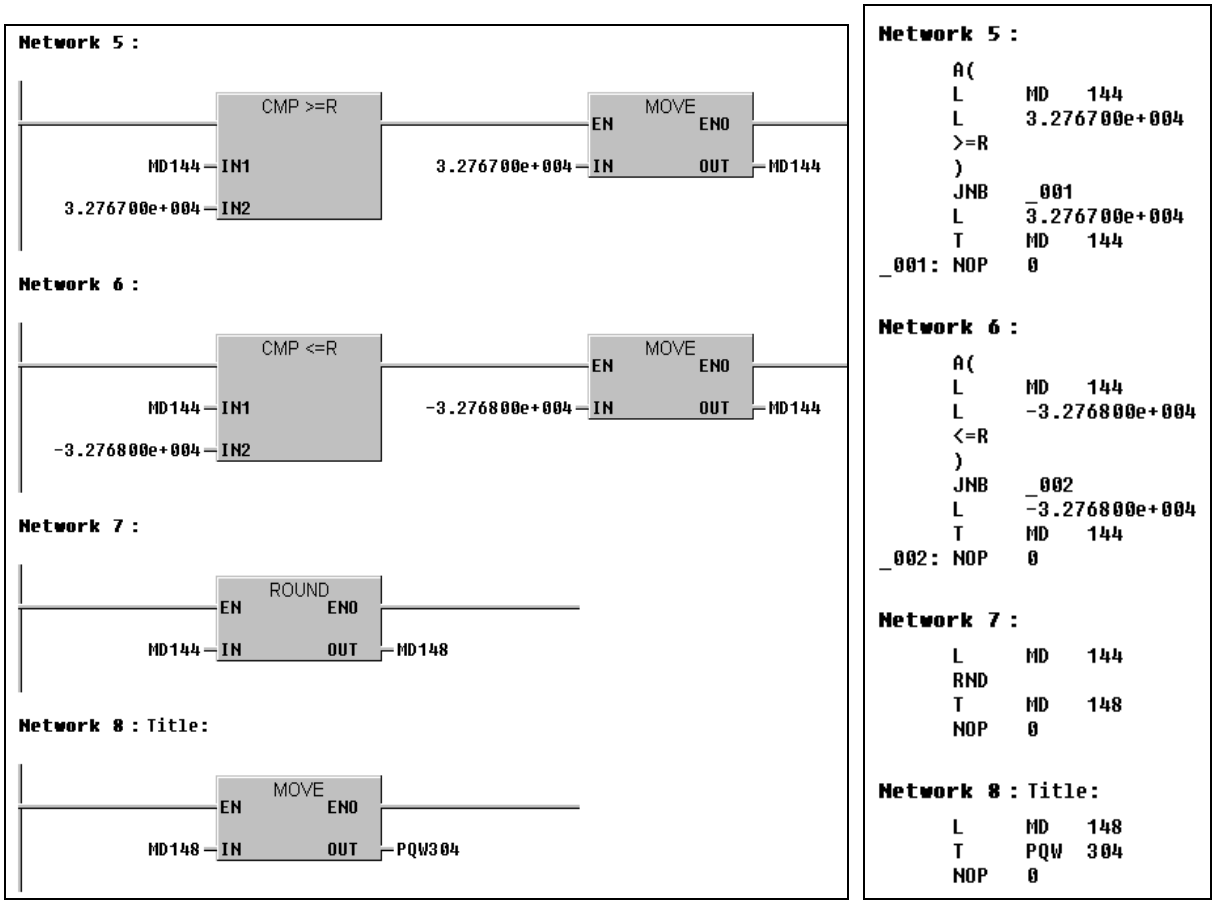
komutları yazılır ve bu durumda MD60 adresinin içeriği MW64 adresine

MD 60 DW#16#ABCD1234 → MW 64 W#16#1234

biçiminde kaydedilir.

Örnek 9.3 MD144 adresindeki bir reel sayının, işaretli tamsayı değer aralığı {-32768, 32767} içindeki bir tamsayıya dönüştürülüp analog çıkış birimine ilişkin PQW304 adresine yazılmasını sağlayan bir program parçası yazınız.

Bu özellikleri sağlayan bir program örneği Şekil 9.49'daki gibi verilebilir. Bu programa göre, eğer reel sayı işaretli tamsayı değer aralığı dışında ise, reel sayıya işaretli tamsayı sınır değerleri atanır, yuvarlama işlemi ile tamsayıya dönüştürülür ve aktarma komutu ile analog çıkış biriminin PQW304 adresine işaretli tamsayı olarak yazılır.

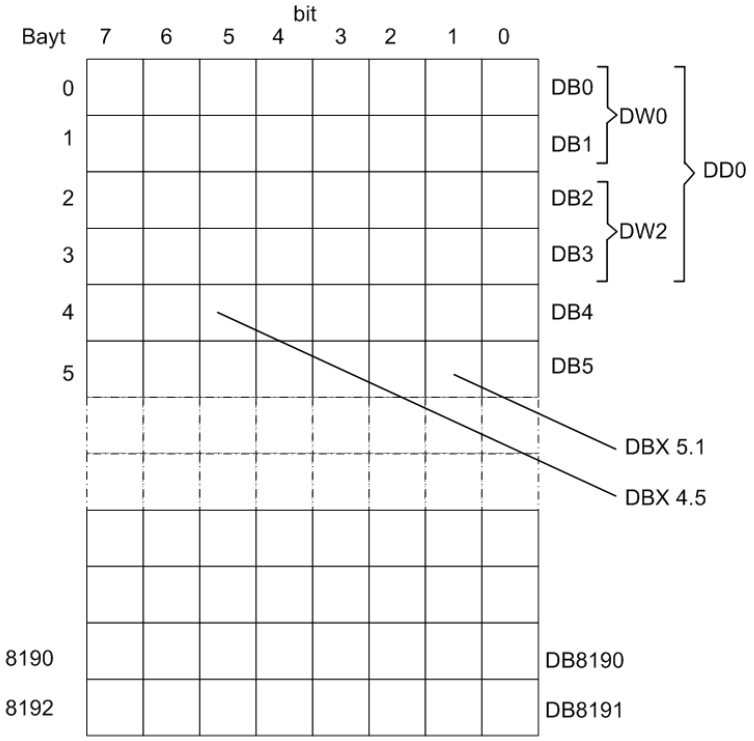


Şekil 9.49 Reel bir sayının işaretli tamsayıya dönüştürülmesine ilişkin program örneği

9.4 VERİ BLOKLARI VE BELLEK ALANLARINA ERİŞİM

Veriler, veri blokları (DBx) olarak tanımlanan bellek alanlarında saklanır. Bir veri bloğunun adreslenmesi, diğer bellek alanlarında olduğu gibi Şekil 9.50'deki gibi yapılır.

Bir veri bloğunun; bit boyutundaki veriye ilişkin adres DBX byte.bit, bayt boyutundaki veriye ilişkin adres DBB bayt, 16 bitlik veriye ilişkin adres DBW bayt ve 32 bitlik veriye ilişkin adres DBD bayt olarak tanımlanır.

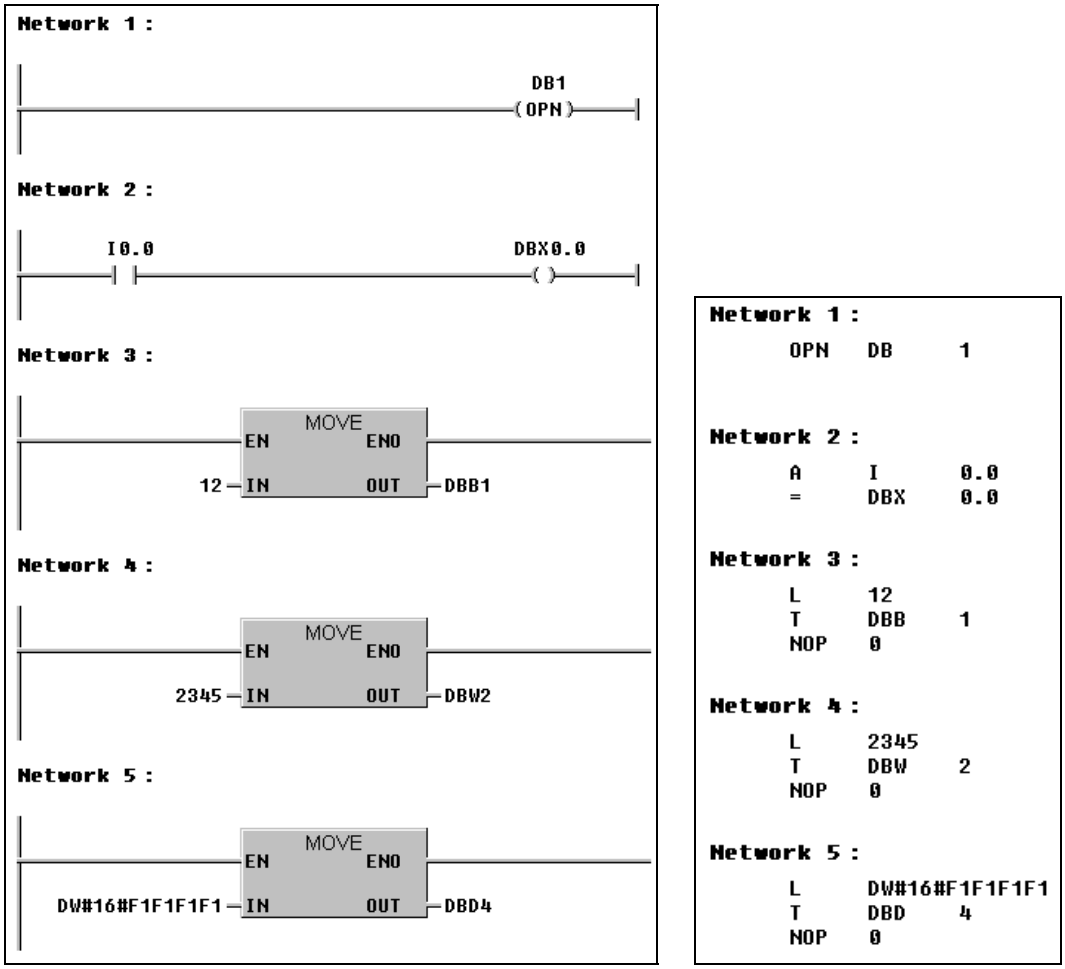


Şekil 9.50 Bir veri bloğuna ilişkin adreslerin kapsadığı alanlar

Bir veri bloğuna veri yazmak ya da okumak için önce "OPN" komutu ile veri bloğu açılır ve tanımlı adreslere erişilir. Veri bloğu üzerinde işlem yapmadan önce bu bloğa ilişkin boyut ve veri türleri tanımlanır. Örneğin, Şekil 9.51'deki gibi tanımlanan DB1 veri bloğundaki çeşitli türdeki verilere, Şekil 9.52'de yazılan programla erişilir.

Address	Name	Type	Initial value
0.0		STRUCT	
+0.0	K1	BOOL	FALSE
+1.0	B1	BYTE	B#16#0
+2.0	W1	INT	0
+4.0	D1	DWORD	DW#16#0
=8.0		END_STRUCT	

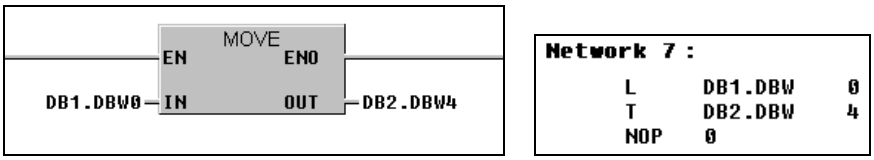
Şekil 9.51 DB1 veri bloğuna ilişkin tanımlamalar



Şekil 9.52 Veri bloğundaki çeşitli türdeki verilere erişim örnekleri

Bir veri bloğu OPN komutu ile açılır ve yeni bir blok açılıncaya kadar bu blok üzerinde işlemler yapılır. Yeni bir blok açıldığında eski blok kapanır.

Veri bloklarına, veri bloğu ve adres birleşik yazılarak da erişilebilir. Örneğin,



komutları ile DB1 veri bloğunun DBW0 adresindeki veri, DB2 veri bloğunun DBW4 adresine yazılır.

9.5 STL İLE PROGRAMLAMA

LAD ve FBD ile programlamaya göre daha çok komut kullanma olanağı sağlayan STL diliyle programlama karmaşık, kullanımı ve izlenmesi güç bir programlama türüdür. Bu programlama biçiminin etkin kullanılması için, komut durum sözcüğü (status word) olarak adlandırılan kaydedicinin yapısı, işlevi ve komutlarla etkileşiminin bilinmesi gerekir.

KOMUT DURUM SÖZCÜĞÜ KAYDEDİCİSİ: Komut durum sözcüğü, 9 biti anlamlı 16 bitlik bir bellek alanıdır. Şekil 9.53'te gösterilen komut durum sözcüğünde yer alan her bitin anlamı, işlevi ve komutlarla etkileşimi aşağıda açıklanmıştır.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
							BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC

Şekil 9.53 Komutu durum sözcüğü ve durum bitleri

/FC: İlk komut değil biti (NOT first check bit): Herhangi bir mantıksal ifadenin hesaplanmasına ilişkin mantıksal komutun ilk komut olup olmadığı bilgisi bu alana kaydedilir. Eğer /FC=1 ise işlenen komut ilk komut değildir. Bu bit değeri işlenen komutlara bağlı olarak 0 veya 1 değerine kurulur. Örneğin,

A [VE]

AN [VE DEĞİL]

O [VEYA]

ON [VEYA DEĞİL]

X [ÖZELVEYA]

XN [ÖZELVEYA DEĞİL]

) [parantez kapatma]

FP [çıkan kenar]

FN [düşen kenar]

karşılaştırma [==I,<>I, <I, >I, >=I, ==D,<>D, <D, >D, >=D, ==R,<>R, <R, >R, >=R]

mantıksal komutları yürütüldüğünde /FC biti 1 değerine kurulur.

= [atama]

S [kur, set]

R [sil, reset]

A([parantez içine VE işlemi]

AN([parantez içine VE DEĞİL işlemi]

O [işlenen adres belirtilmeyen VEYA işlemi]

O([parantez içine VEYA işlemi]

ON([parantez içine VEYA DEĞİL işlemi]

X([parantez içine ÖZEL VEYA işlemi]

XN([parantez içine ÖZEL VEYA DEĞİL işlemi]

CLR [sil]

SET [kur]

Zamanlayıcı: (SP, SE, SD, SS, SF, FR)

Sayıcı: (CU, CD)

Blok sonu : (BE, BEU, BEC)

Fonksiyon çağırma: (CALL, UC, CC)

Bazı atlama: (JC, JCN, JCB, JNB, JBI, JNBI)

komutları işletildiğinde /FC biti sıfırlanır.

RLO biti: Mantıksal işlem sonucunun kaydedildiği bellek alanıdır. Eğer /FC biti 0 ise mantıksal işlem uygulanan adresin içeriği buraya yazılır. Eğer /FC biti 1 ise mantıksal işlem uygulanan adresin içeriği ile RLO bitinin içeriğine ilgili işlem uygulanır ve sonuç yine RLO biti olarak kaydedilir. Atama işleminde RLO'nun içeriği işlenen adrese yazılır.

STA biti: İşlenen adresin mantıksal durumunun kaydedildiği bellek bitidir.

OR biti: İşlenen adres içermeyen O [VEYA] komutunun yürütülmeden önceki VE ya da VEDEĞİL işlemleri sonucu oluşan RLO değerinin saklanması amacıyla kullanılır. OR biti ile RLO bitine VEYA işlemi uygulanır ve sonuç RLO bitinde tutulur. Mantıksal işlem komutlarından O, ON, X, XN, FP, FN,

S, R, =; doğrudan durum sözcüğünü etkileyen CLR ve SET komutları, zamanlama (SP, SE, SD, SS, SF, FR) ve sayma (CU, CD) komutları işletildiğinde OR biti sıfırlanır.

OV biti: Matematiksel işlem sonucu taşma olduğu anda 1 değerini alır.

OS biti: Matematiksel işlem sonucu taşma olduğunda 1 değerine kurulur ve blok sonuna kadar bu değerde kalır.

CC0, CC1 bitleri: Matematiksel, karşılaştırma veya bit kaydırma işlemleri akümülatörler üzerinde yapılır. Bir işlem sonucunda oluşan ACCU1 ve ACCU2 akümülatörlerinin içeriklerine bağlı olarak Şekil 9.54'teki değerler elde edilir.

CC1	CC0	Koşul	Sonuç
0	0	=0	ACCU2=ACCU1
1	0	> 0	ACCU2 > ACCU1
0	1	< 0	ACCU2 < ACCU1
1	1	UO	Taşma

Şekil 9.54 CC0 ve CC1 durum bitlerinin aldığı değerler

BR biti: İkilik sonuç (binary result): RLO bitinin yedeklendiği bellek alanıdır. RLO bitinin içeriği SAVE komutu ile BR bitine kaydedilir. Bu bit fonksiyon ya da fonksiyon blok kutularının ENO çıkışını etkin yapmada kullanılabilir. LAD veya FBD ile programlamada çağrılan fonksiyonun ENO çıkışı, bu bit 1 değerinde ise etkin görünür.

Durum bitlerini doğrudan değiştiren komutlar: Komut durum sözcüğündeki bazı bitler SET, CLR, NOT ve SAVE komutları ile doğrudan değiştirilebilir.

Durum bitleri, **SET** komutu işletildiğinde

BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC
-	-	-	-	-	0	1	1	0

CLR komutu işletildiğinde

BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC
-	-	-	-	-	0	0	0	0

NOT komutu işletildiğinde

BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC
-	-	-	-	-	-	1	(RLO)'	-

SAVE komutu işletildiğinde

BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC
RLO	-	-	-	-	-	-	-	-

biçiminde değişir. Burada (-) işareti etkilenmeyen bitleri göstermektedir.

Yığın belleği: RLO, OR, BR ve işlem kodu bilgisi bitlerinin saklandığı ve derinliği 7 olan yığın yapısındaki bir bellek alanıdır. Parantezli mantıksal ve iç içe VEYA işlemlerinde yığın belleği kullanılır. Örneğin; A(komutu yürütüldüğünde RLO, OR ve işlem kodu bilgileri yığına konur,) komutu yürütüldüğünde bu bilgiler yığından alınır. Yığından veri alınması son giren ilk çıkar (LIFO) ilkesine göre yapılır.

9.5.1 TEMEL MANTIKSAL İŞLEMLER

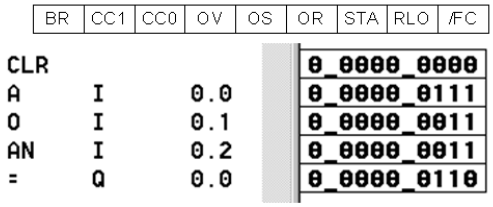
Temel mantıksal işlemlere ilişkin komutlar ile komut durum sözcüğü arasındaki etkileşim aşağıdaki örnekler üzerinde açıklanmıştır.

Örnek 9.4 Mantıksal ilişkisi Q0.0=(I0.0+I0.1)•I0.2' biçiminde verilen ifadeye ilişkin programın Şekil 9.55'teki gibi yazılabilir. Bu programda, her komutun yürütülmesinden sonraki /FC biti ve RLO bitinin hesaplanmasında geçerli olan mantıksal ilişkiler gösterilmiştir.

KOMUT	/FC	RLO
CLR	0	0
A I0.0	1	I0.0
O I0.1	1	I0.0+I0.1
AN I0.2	1	I0.2'·(I0.0+I0.1)
= Q0.0	0	I0.2'·(I0.0+I0.1)

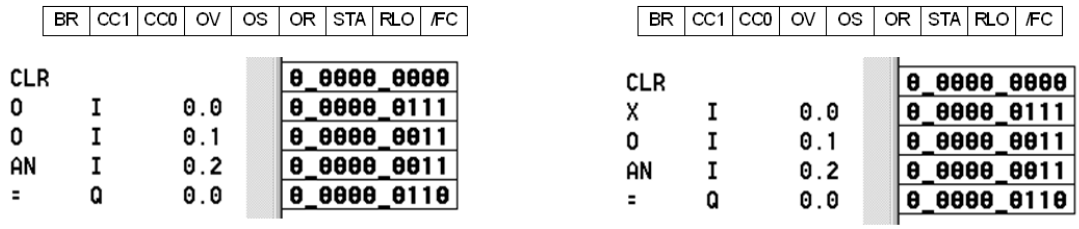
Şekil 9.55 Temel mantıksal işlem komutları ile /FC ve RLO bitleri arasındaki etkileşim

Bu programın yürütülmesi sürecinde, Eğer I0.0=1 yapılır ve durum sözcüğü izlenirse Şekil 9.56'da verilen değerler elde edilir.



Şekil 9.56 I0.0=1 için durum sözcüğünün değişimi

STL ile yazılan programlarda, /FC=0 iken A, O ve X komutları yalnız işlenen adresin içeriğini RLO alanına yazma işlevini görür. Bu özellik nedeniyle, ilk komutun değiştirilmesi mantıksal ilişkiyi değiştirmez. Örneğin, CLR komutunu izleyen A I0.0 komutu yerine O I0.0 veya X I0.0 komutları yazılabilir. Şekil 9.57'de O ve X komutları ile başlayan program örnekleri ve program yürütülmesi sürecindeki durum sözcüğünün değişimi gösterilmiştir.



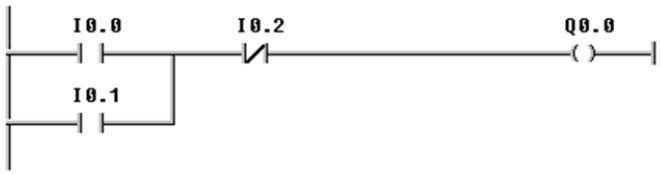
Şekil 9.57 I0.0=1 için O ve X ilk komutları ile başlayan programlar ve durum sözcüğü

Şekil 9.57'de verilen program LAD veya FBD biçimine dönüştürülemez. Aynı mantıksal ilişkiyi gerçeklemek üzere program, Şekil 9.58'deki gibi yazılabilir. Bu şekilde yazılan programın LAD veya FBD biçimindeki karşılığı vardır.

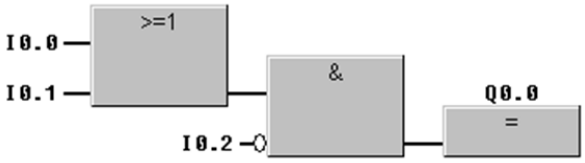
KOMUT	/FC	RLO
A(	0	1
O I0.0	1	I0.0
O I0.1	1	I0.0+I0.1
)	1	1 (I0.0+I0.1)
AN I0.2	1	I0.2' (I0.0+I0.1)
= Q0.0	0	I0.2' (I0.0+I0.1)

Şekil 9.58 Q0.0= (I0.0+I0.1)•I0.2'•mantık fonksiyonuna ilişkin STL programı

Şekil 9.58'deki programın LAD biçimi Şekil 9.59'da, FBD biçimi Şekil 9.60'ta gösterilmiştir



Şekil 9.59 Q0.0= (I0.0+I0.1)•I0.2'•mantık fonksiyonuna ilişkin LAD programı



Şekil 9.60 Q0.0= (I0.0+I0.1)•I0.2'•mantık fonksiyonuna ilişkin FBD programı

Şekil 9.58'de verilen programda /FC=0 iken işlenen O I0.0 komutu yerine A I0.0 veya X I0.0 komutları yazılabilir. Ancak, bu değişiklik sonucu STL dilinde yazılan program LAD veya FBD biçimine dönüşmeyebilir. Örneğin, O I0.0 komutu yerine A I0.0 komutu yazılırsa STL programı LAD veya FBD biçimine dönüşür, X I0.0 komutu yazılırsa dönüşmez.

Örnek 9.5 Üç girişli bir devrede, yalnız tek sayıda (bir veya üç) giriş 1 yapıldığında çıkış işareti 1 değerini alan programı STL dilinde yazınız.

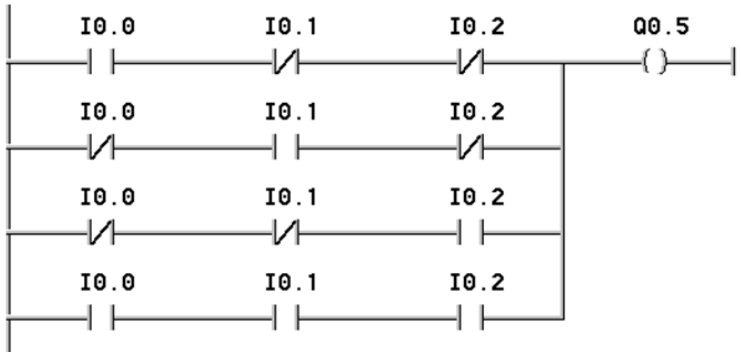
Bu koşulları sağlayan mantıksal ifade; girişler x=I0.0, y=I0.1 ve z=I0.2; çıkış K=Q0.5 olmak üzere

$$K=x\bullet y'\bullet z'+ x'\bullet y\bullet z'+ x'\bullet y'\bullet z+ x\bullet y\bullet z$$

biçiminde yazılabilir. Çarpımların toplamı biçimindeki bu mantıksal ifade, doğrudan Şekil 9.61'deki gibi programlanabilir. Bu programa karşı düşen LAD program biçimi Şekil 9.62'de verilmiştir.

			BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC
A	I	0.0						0_0000_0001			
AN	I	0.1						0_0000_0101			
AN	I	0.2						0_0000_0001			
O								0_0000_0100			
AN	I	0.0						0_0000_0011			
A	I	0.1						0_0000_0111			
AN	I	0.2						0_0000_0011			
O								0_0000_1111			
AN	I	0.0						0_0000_1011			
AN	I	0.1						0_0000_1111			
A	I	0.2						0_0000_1011			
O								0_0000_1111			
A	I	0.0						0_0000_1011			
A	I	0.1						0_0000_1111			
A	I	0.2						0_0000_1011			
=	Q	0.5						0_0000_0110			

Şekil 9.61 K=x•y'•z'+ x'•y•z'+ x'•y'•z+ x•y•z mantık fonksiyonuna ilişkin STL programı ve I0.1=1 için durum sözcüğünün değişimi



Şekil 9.62 K=x•y'•z'+ x'•y•z'+ x'•y'•z+ x•y•z mantık fonksiyonuna ilişkin LAD programı

Bu örneğe ilişkin program, X (ÖZELVEYA) komutu ile daha az komut kullanılarak gerçeklenebilir. Ancak, X veya XN komutu içeren STL programlarının LAD veya FBD karşılıkları yoktur. ÖZELVEYA (XOR) işlemi x,y giriş Q çıkış olmak üzere Q=x•y'+x'•y biçiminde tanımlanır. Bu ilişkiden yararlanılarak Şekil 9.62'deki program yazılabilir.

KOMUT	/FC	RLO
A I0.0	0	I0.0
X I0.1	1	I0.0•I0.1'+ I0.0'•I0.1
X I0.2	1	I0.2•(I0.1'+ I0.0'•I0.1)'+ I0.2'•(I0.1'+ I0.0'•I0.1)
= Q0.5	0	I0.2•(I0.1'+ I0.0'•I0.1)'+ I0.2'•(I0.1'+ I0.0'•I0.1)

ŞEKİL 9.62 K=x•y'•z'+x'•y•z'+x'•y'•z+x•y•z mantık fonksiyonunun X komutu ile gerçeklenmesi

x=I0.0, y=I0.1 ve z=I0.2; çıkış K=Q0.5 olmak üzere birinci komut (A I0.0) komut yürütüldüğünde /FC=0 olduğundan x değeri RLO' ya yazılır. İkinci komut yürütüldüğünde /FC=1 olduğundan RLO=x değeri ile y değerine ÖZELVEYA işlemi uygulanır ve sonuç RLO'ya yazılır: RLO=x•y'+ x'•y

Üçüncü komut yürütüldüğünde /FC=1 olduğundan RLO ile z değişkenine ÖZELVEYA işlemi uygulanır ve sonuç RLO'ya yazılır. Buna göre RLO'da gerçeklenen mantıksal ifade

RLO=z•(x•y'+x'•y)'+z'•(x•y'+ x'•y)

olur. De Morgan teoremine göre ifade edilen (x+y)'=x'•y' eşitliği kullanılarak, Bole cebri işlemleri ile

RLO =z•(x•y'+x'•y)'+z'•(x•y'+ x'•y)=z•((x•y')'•(x'•y)')+z'•(x•y'+x'•y)

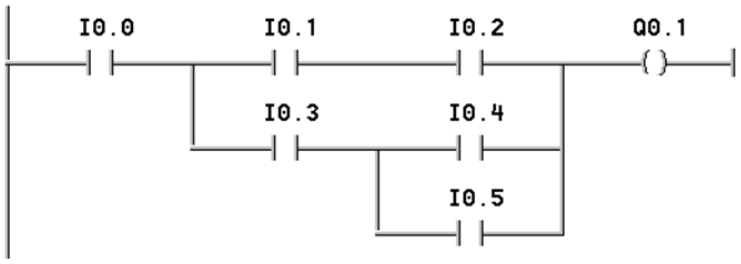
=z•((x'+y)•(x+y'))+z'•(x•y'+x'•y)=z•(x'•y'+x•y)+z'•(x•y'+x'•y)

=x'•y'•z +x•y•z+x•y'•z'+x'•y•z'

mantıksal ilişkisi elde edilir.

Örnek 9.6 K=x•(y•z+u•(v+w)) biçiminde verilen mantık fonksiyonuna ilişkin programı yazınız.

Giriş değişkenleri x=I0.0, y=I0.1, z=I0.2, u=I0.3, v=I0.4 w=I0.5 ve K=Q0.1 olmak üzere bu ifadeye karşı düşen LAD programı Şekil 9.63'te, doğrudan parantezli VE komutları ile gerçeklenen STL programı ve I0.0=1, I0.1=1 ve I0.2=1 için durum sözcüğü Şekil 9.64'te verilmiştir.



Şekil 9.63 K=x•(y•z+u•(v+w)) mantık fonksiyonuna ilişkin LAD programı

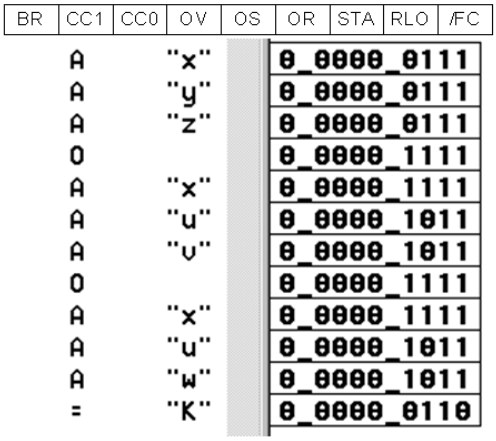
			BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC
A	I	0.0						0_0000_0111			
A(								0_0000_0110			
A	I	0.1						0_0000_0111			
A	I	0.2						0_0000_0111			
O								0_0000_1111			
A	I	0.3						0_0000_1011			
A(								0_0000_0110			
O	I	0.4						0_0000_0001			
O	I	0.5						0_0000_0001			
)								0_0000_1111			
)								0_0000_0111			
=	Q	0.1						0_0000_0110			

Şekil 9.64 K=x•(y•z+u•(v+w)) mantık fonksiyonuna ilişkin STL programı ve durum sözcüğü

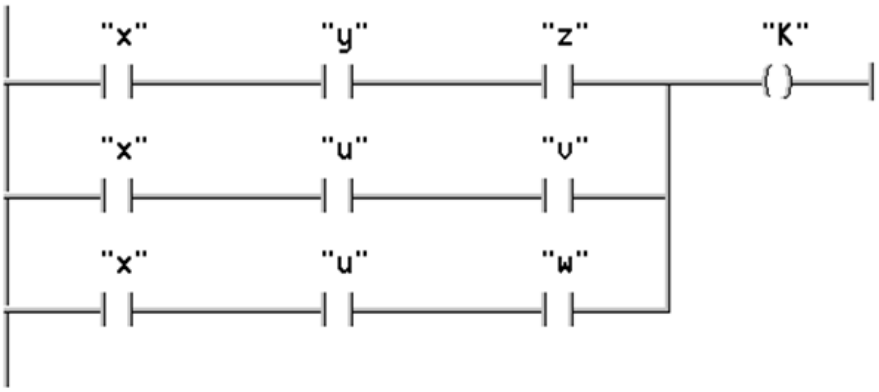
Aynı mantıksal ilişkiyi sağlayan mantık fonksiyonu Boole cebri işlemleri ile çarpımların toplamı biçimine getirilerek Şekil 9.65'teki gibi de programlanabilir.

K=x•(y•z+u•(v+w))=x•y•z+x•u•(v+w)=x•y•z+x•u•v+x•u•w

Bu fonksiyona ilişkin programın LAD biçimi Şekil 9.66'da verilmiştir.

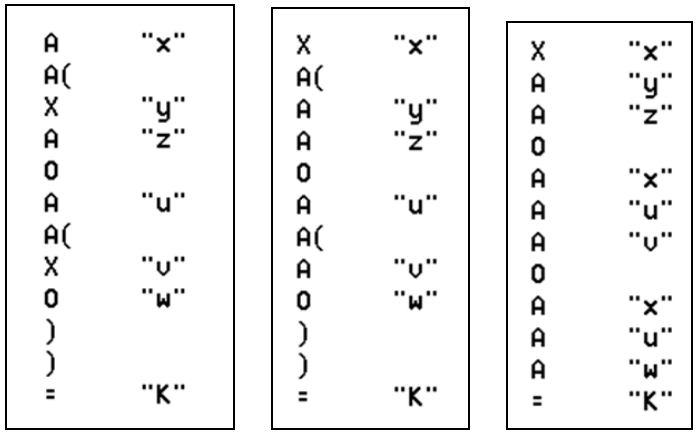


Şekil 9.65 K= x•y•z+x•u•v+x•u•w mantık fonksiyonuna ilişkin STL programı ve durum sözcüğü



Şekil 9.66 K= x•y•z+x•u•v+x•u•w mantık fonksiyonuna ilişkin LAD programı

Şekil 9.65 ve Şekil 9.66'da verilen programlar, aynı mantıksal ilişkiyi gerçeklemek üzere değişik biçimlerde düzenlenebilir. Örneğin, durum sözcüğünde /FC=0 olan bazı satırları izleyen komutların yerine X komutu yazılarak elde edilen Şekil 9.67'deki her üç program da aynı işlevi sağlar; ancak, bu biçimde yazılan STL programları, X komutu içerdiğinden, LAD veya FBD biçimine dönüşmez.



Şekil 9.67 LAD veya FBD biçimine dönüştürülemeyen aynı işlevli çeşitli program örnekleri

9.5.2 ATLAMA İŞLEMLERİ

STL ile programlamada, durum sözcüğündeki bitlerin durumuna bağlı olarak atlama işlevini yerine getiren birçok komut kullanılır. Çoğu LAD ve FBD ile programlamada geçerli olmayan bu komutların kullanımı ile RLO, BR, OV, OS, CC1 ve CC0 bitlerinin durumuna göre atlama işlemi gerçeklenir. Atlama komutları ile ilgili koşullar ve işlevleri aşağıda açıklanmıştır.

JU xxxx : xxxx etiketi ile belirtilen noktaya atla (koşulsuz)

JC xxxx : RLO=1 ise xxxx etiketi ile belirtilen noktaya atla,

JCN xxxx : RLO = 0 ise xxxx etiketi ile belirtilen noktaya atla,

JCB xxxx : RLO = 1 ise xxxx etiketi ile belirtilen noktaya atla ve BR=1 yaz,

JNB xxxx : RLO = 0 ise xxxx etiketi ile belirtilen noktaya atla ve BR=0 yaz,

JBI xxxx : BR = 1 ise xxxx etiketi ile belirtilen noktaya atla,

JNBI xxxx : BR = 0 ise xxxx etiketi ile belirtilen noktaya atla,

JO xxxx : OV = 1 ise xxxx etiketi ile belirtilen noktaya atla,

JOS xxxx : OS = 1 ise xxxx etiketi ile belirtilen noktaya atla,

JZ xxxx : CC1=0 ve CC0=0 ise xxxx etiketi ile belirtilen noktaya atla,

JN xxxx : CC1 <> CC0 ise xxxx etiketi ile belirtilen noktaya atla,

JP xxxx : CC1=1 ve CC0=0 ise xxxx etiketi ile belirtilen noktaya atla,

JM xxxx : CC1=0 ve CC0=1 ise xxxx etiketi ile belirtilen noktaya atla,

JPZ xxxx : CC1=0 ve CC0=0 veya CC1=1 ve CC0=0 ise xxxx etiketi ile belirtilen noktaya atla,

JMZ xxxx : CC1=0 ve CC0=0 veya CC1=0 ve CC0=1 ise xxxx etiketi ile belirtilen noktaya atla,

JUO xxxx: Geçersiz reel sayı veya sıfıra bölme varsa xxxx etiketi ile belirtilen noktaya atla.

Yukarıda verilen tüm komutlar, koşul gerçeklendiğinde işletilir ve etiket ile belirtilen satıra atlanır. Belirli koşullarda bir noktadan, çok noktaya atlama işlevini sağlamak için JL ve JU komutlarından oluşan bir yapı kullanılır. Bu kullanım biçiminde, ACCU 1'in en düşük anlamlı 8 bitlik bölümüne 0 ile 255 arasında bir tamsayının yüklenmesini sağlayan bir komut yazılır, Bu komutu izleyen satıra JL ve JL satırını izleyen satırlara atlanacak nokta sayısı kadar JU komutu yazılır ve bir etiket alanı ile sonlandırılır. Bu biçimde yazılan programda atlama işlemi şu şekilde gerçeklenir.

Eğer ACCU1 içeriği, JU sayısına eşit ve büyükse JL komutu ile belirtilen satıra atlanır.

Eğer ACCU1=0 ise JL komutunu izleyen ilk JU komutunda belirtilen etiket alanına,

Eğer ACCU1=1 ise JL komutunu izleyen ikinci JU komutunda belirtilen etiket alanına,

Eğer ACCU1=2 ise JL komutunu izleyen üçüncü JU komutunda belirtilen etiket alanına,

Eğer ACCU1=n ise JL komutunu izleyen (n+1). JU komutunda belirtilen etiket alanına atlanır.

Şekil 9.68'de bu yapının kullanımına ilişkin bir program örneği verilmiştir.

Bu programa göre,

Eğer, MB80 adresinin içeriği 0 ise JL komutunu izleyen JU _000 komutu işletilir ve _000 etiketi ile belirtilen satıra atlanır. Burada MW2 adresinin içeriğine 10 sayısı yüklenir ve ESON etiketine atlanır.

Eğer, MB80 adresinin içeriği 3 ise JL komutunu izleyen JU _003 komutu işletilir ve _003 etiketi ile belirtilen satıra atlanır. Burada MW2 adresinin içeriğine 16 sayısı yüklenir ve ESON etiketine atlanır.

Eğer, MB80 adresinin içeriği 6 ise JL komutunu izleyen yedinci satırdaki JU _002 komutu işletilir ve _002 etiketi ile belirtilen satıra atlanır. Burada MW2 adresinin içeriğine 12 sayısı yüklenir ve ESON etiketine atlanır.

Eğer, MB80 adresinin içeriği 6 tamsayısından büyükse ($n \geq 7$) ise JL komutu işletilir ve LSON etiketi ile belirtilen satıra, bu satırdaki JU komutu nedeniyle SON etiketi ile belirtilen satıra atlanır ve burada MW2 adresinin içeriğine 0 sayısı yüklenir.

```
Network 1 : Title:
      L     MB    80
      JL    LSON
      JU    _000
      JU    _001
      JU    _002
      JU    _003
      JU    _004
      JU    _005
      JU    _002
LSON: JU    SON

_000: L     10
      T     MW     2
      JU    ESON

_002: L     12
      T     MW     2
      JU    ESON

_001: L     14
      T     MW     2
      JU    ESON

_003: L     16
      T     MW     2
      JU    ESON

_004: L     18
      T     MW     2
      JU    ESON

_005: L     20
      T     MW     2

SON:  L     0
      T     MW     2

ESON: NOP   0
```

Şekil 9.68 Bir noktadan çok noktaya atlama işlemine ilişkin program örneği

9.5.3 MATEMATİKSEL İŞLEMLER

Tüm matematiksel işlemler akümülatörler üzerinde yapılır. S7-300 ailesi PLC'lerde 2 akümülatör (ACCU1, ACCU2), S7-400 ailesi PLC'lerde 4 akümülatör (ACCU1, ACCU2, ACCU3, ACCU4) bulunur. Her iki PLC ailesinde de, bir verinin akümülatöre yüklenmesi için yükleme (L) komutu, ACCU1 akümülatörü içeriğinin belirli bir adrese yazılması için aktarma (T) komutu kullanılır.

L komutunun işletilmesi ile ACCU1'in içeriği ACCU2'ye yazılır, ACCU1'in içeriği sıfırlanır ve işlenen adresin içeriği ACCU1'e yüklenir, ACCU1 akümülatörünün içeriğini belirli bir adrese yazmak için aktarma (T) komutu kullanılır.

İki sayısal veri üzerinde yapılan toplama (+I, +D, +R), çıkarma (-I, -D, -R), çarpma (*I, *D, *R) ve adrese bölme (/I, /D, /R) gibi işlemler, ■ simgesi işlem olmak üzere

ACCU1:=ACCU2 ■ ACCU1 biçiminde gerçeklenir.

Şekil 9.69'da verilen programda 16 bitlik işaretli tamsayılar üzerinde $MW20 = (115 - 5)/10$ işlemine ilişkin komutlar ve her komut yürütüldükten sonraki ACCU1 (STANDARD) ve ACCU2 akümülatörlerinin içerikleri gösterilmiştir.

Network 2 :			STANDARD	ACCU 2
L	115		115	0
L	5		5	115
-I			110	0
L	10		10	110
/I			11	0
T	MW	20	11	0

Şekil 9.69 MW20=(115-5)/10 işlemine ilişkin program örneği

Reel sayılar üzerinde yapılan mutlak değer (ABS), kare (SQR), karekök (SQRT), üstel fonksiyon (EXP), doğal logaritma (LN), sinüs (SIN), kosinüs (COS), tanjant (TAN), ters sinüs (ASIN), ters kosinüs (ACOS) ve ters tanjant (ATAN) gibi bir sayısal veriler üzerinde yapılan işlemler ACCU1 üzerinde gerçeklenir. Bu komutlar yürütüldüğünde, işlem ACCU1'deki veriye uygulanır, sonuç ACCU1'e yazılır.

Şekil 9.70'te, $y = a^P$ biçiminde verilen bir fonksiyonun hesaplanmasına ilişkin bir program verilmiş, a=2.0, P=2.5 için, her komut yürütüldükten sonraki ACCU1 (STANDARD) ve ACCU2 içerikleri gösterilmiştir. Bu fonksiyonu doğrudan hesaplayan bir komut olmadığından, fonksiyon önce $y = e^{P\ln(a)}$ biçimine getirilir ve hesaplanır.

Network 2 : Title:			STANDARD	ACCU 2
L	2.000000e+000		+2.000e+000	0
LN			+6.931e-001	0
L	2.500000e+000		+2.500e+000	1060205080
*R			+1.733e+000	0
EXP			+5.657e+000	0
T	MD	60	+5.657e+000	0

Şekil 9.70 $y = e^{P\ln(a)}$ işlemine ilişkin program örneği

Karşılaştırma komutları (==I, <>I, >I, <I, >=I, <=I ; ==D, <>D, >D, <D, >=D, <=D; ==R, <>R, >R, <R, >=R, <=R) iki sayısal veri üzerinde yapıldığından ACCU2'deki veri, ACCU1'deki veriye göre karşılaştırılır, koşul sağlanıyorsa RLO=1 olur. Şekil 9.71'de karşılaştırma işlemlerine ilişkin bir program örneği verilmiş, MW80=11, MD84=125000 için her komut yürütüldükten sonraki RLO, ACCU1 (STANDARD), ACCU2 ve durum sözcüğünün içerikleri gösterilmiştir.

Network 5 :	RLO	STANDARD	ACCU 2	STATUS WORD
L MW 80	0	11	11	0 1000 0100
L 12	0	12	11	0 1000 0100
==I	0	12	11	0 0100 0001
= M 0.0	0	12	11	0 0100 0000
L MW 80	0	11	12	0 0100 0000
L 12	0	12	11	0 0100 0000
<=I	1	12	11	0 0100 0111
= M 0.1	1	12	11	0 0100 0110
L MD 84	1	125000	12	0 0100 0110
L L#120000	1	120000	125000	0 0100 0110
<=D	0	120000	125000	0 1000 0001
= M 0.2	0	120000	125000	0 1000 0000

Şekil 9.71 Karşılaştırma işlemlerine ilişkin program örneği

Dönüştürme komutları (BTI, ITB, BTD, ITD, DTB, DTR, INVI, INVD, NEGI, NRGD, NEGR, CAW, CAD, RND, TRUNC, RND+, RND-) bir sayısal veriye uygulandığından ACCU1 üzerinde gerçeklenir ve sonuç ACCU1'e yazılır.

Şekil 9.72'de, MD190 adresindeki reel bir x sayısına, $y = \cos(x)\tan(x)\exp(-x)$ fonksiyonu uygulandıktan sonra, reel sayı olarak MD200 adresine yazılmasını sağlayan bir program örneği verilmiş, MD190=0.5 için, her komut yürütüldükten sonraki ACCU1 (STANDARD) ve ACCU2 içerikleri gösterilmiştir.

Network 6 :	STANDARD	ACCU 2
L MD 190	+5.000e-001	+1.682e-040
COS	+8.776e-001	+1.682e-040
L MD 190	+5.000e-001	+8.776e-001
TAN	+5.463e-001	+8.776e-001
*R	+4.794e-001	+0.000e+000
L MD 190	+5.000e-001	+4.794e-001
NEGR	-5.000e-001	+4.794e-001
EXP	+6.065e-001	+4.794e-001
*R	+2.908e-001	+0.000e+000
T MD 200	+2.908e-001	+0.000e+000

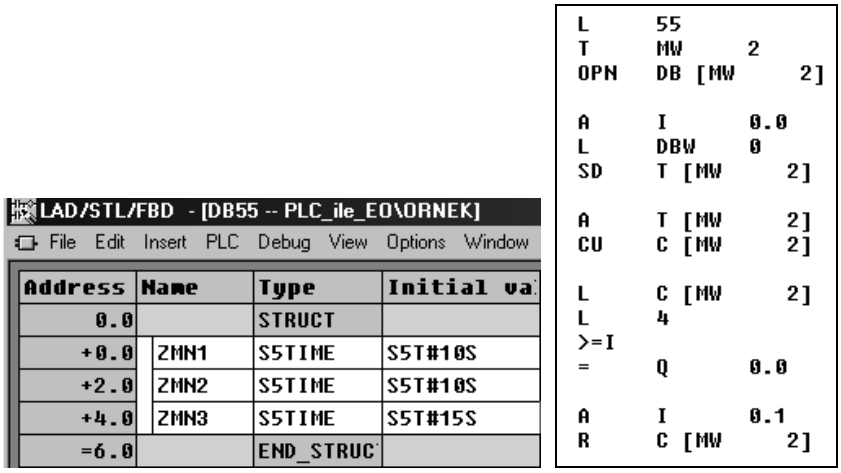
Şekil 9.72 $y = \cos(x)\tan(x)\exp(-x)$ fonksiyonunun hesaplanmasına ilişkin program örneği

9.5.4 DOLAYLI ADRESLEME

Dolaylı adresleme, belirli bir adresteki veriye, işaretçi olarak tanımlanan başka bir adresten erişme yöntemidir. İşaretçi olarak M, L, ve DB bellek alanları ya da AR1 ve AR2 adres kaydedicileri kullanılabilir. İşaretçi köşeli parantez içine yazılır.

Zamanlayıcı (T), sayıcı (C), veri bloğu (DB, DI), fonksiyon (FC), fonksiyon blok (FB) için 16 bitlik işaretçiler kullanılır.

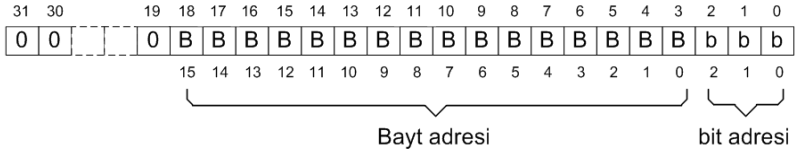
Şekil 9.73'te zamanlayıcı, sayıcı ve veri bloğunun dolaylı adreslendiği bir program örneği verilmiştir.



Şekil 9.73 16 bitlik işaretçi ile dolaylı adreslemeye ilişkin bir program örneği

Bu programda, işaretçi olarak seçilen MW2 adresine 55 tamsayısı yazılarak, veri bloğunun DB55, zamanlayıcının T55 ve sayıcının C55 olduğu belirtilmiştir.

32 bitlik işaretçiler ile 1, 8, 16 ve 32 bitlik bellek alanlarına erişilebilir. 32 bitlik bir işaretçinin yapısı Şekil 9.74'te gösterilmiştir. En düşük anlamlı 3 bit (0-2), bit adresini; (3-18) bitleri bayt adresini gösterir.



Şekil 9.74 32 bitlik işaretçinin yapısı

En düşük anlamlı 3 bit, bit boyutundaki adreslere erişmek için kullanılır. 8, 16 ve 32 bitlik bellek alanlarına erişilirken bu bitler sıfır yapılır. Şekil 9.75'te bit boyutundaki verilerin dolaylı adreslenmesine ilişkin bir program örneği verilmiştir.

```
L     P#0.6
T     MD   100

L     P#2.1
T     MD   104

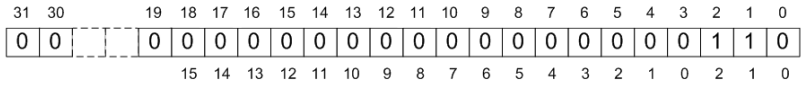
L     MD   100
L     P#0.4
+D
T     MD   108

A     I [MD   100]
=     Q [MD   104]

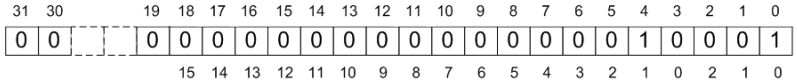
A     I [MD   104]
=     Q [MD   108]
```

Şekil 9.75 32 bitlik işaretçi ile bit boyutundaki verilerin dolaylı adreslenmesine ilişkin bir program örneği

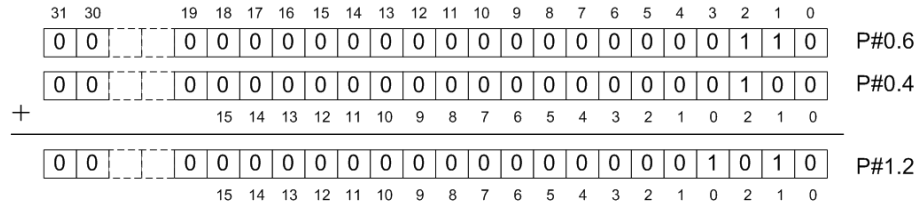
Bu programın yürütüldüğünde, MD100 adresinin içeriği



MD104 adresinin içeriği



MD108 adresinin içeriği



olur. Buna göre Q2.1=I0.6 ve Q1.2=I2.1 mantıksal işlemleri gerçeklenir.

Şekil 9.76'da 8, 16 ve 32 bitlik verilerin dolaylı adreslenmesine ilişkin bir örnek program verilmiştir. Bu programa göre, IB10 adresindeki veri MW12 adresine, PIW288 adresindeki veri MD288 adresine yazılır.

```
L     P#10.0
T     MD   120

L     P#12.0
T     MD   124

L     IB [MD   120]
T     MW [MD   124]

L     MD   120
L     P#278.0
+D
T     MD   128

L     PIW [MD   128]
T     MD [MD   128]
```

Şekil 9.76 32 bitlik işaretçi ile 8,16 ve 32 bit boyutundaki verilerin dolaylı adreslenmesine ilişkin bir program örneği

Adres kaydedicileri (AR1, AR2) kullanılarak da 1, 8, 16 ve 32 bit boyutundaki verilere erişilebilir. Bu tür erişimde adres kaydedicilerinin yapısı Şekil 9.74'te gösterildiği gibidir. Adres kaydedicileri ile adreslemede, bir toplama sabiti (offset) kullanılır. Adres kaydedicisi ile bu sabitin toplamı, İşaret edilen adresi belirtilir.

Adres kaydedicilerine işaretçi formatında bir sayıyı ACCU1'den yüklemek için LAR1 ve LAR2 komutları kullanılır.

LAR1 komutu yürütüldüğünde ACCU1'in içeriği AR1'e,

LAR2 komutu yürütüldüğünde ACCU1'in içeriği AR2'ye yazılır.

Adres kaydedicilerine işaretçi formatında bir sayıyı herhangi bir adresten veya sabit bir değer yüklemek için LAR1 <adres>, LAR2 <adres>, LAR1 P#<adres>, LAR2 P#<adres> komutları kullanılır.

Adres kaydedicilerinin içeriğini ACCU1'e yazmak için TAR1 ve TAR2, herhangi bir adrese yazmak için TAR1 <adres> ve TAR2 <adres> komutları kullanılır.

Adres kaydedicilerinin karşılıklı içeriklerini değiştirmek için TAR komutu kullanılır ve bu komutla AR1'in içeriği AR2'ye AR2'nin içeriği AR1'e yazılır.

Adres kaydedicilerinin içeriklerini arttırmak için AR1+ ve AR2+ veya +AR1 P#byte.bit, +AR2 P#byte.bit komutları kullanılır. +AR1 komutu ile ACCU1'in düşük anlamlı 16 bitlik bölümü AR1'e, +AR2 komutu ile ACCU1'in düşük anlamlı 16 bitlik bölümü AR2'ye eklenir. +AR1 P#10.0 komutu ile AR1'e P#10.0 değeri eklenir.

Şekil 9.77'de adres kaydedicileri ile 1, 8, 16 ve 32 bit boyutundaki verilerin dolaylı adreslenmesine ilişkin bir program örneği verilmiştir.

```
L     P#0.0
LAR1

LAR2  P#2.0

A     I [AR1,P#0.1]
=     Q [AR2,P#1.1]

L     P#10.0
+AR1

+AR2  P#4.0

L     MB [AR1,P#0.0]
T     MW [AR2,P#0.0]
T     MD [AR2,P#6.0]
```

Şekil 9.77 Adres kaydedicileri ile 1, 8, 16 ve 32 bit boyutundaki verilerin dolaylı adreslenmesine ilişkin bir program örneği

Bu programda,

AR1=P#0.0, AR2=P#2.0 ve I[AR1+P#0.1]=I[P#0.0+P#0.1]=I0.1,

Q[AR2+P#1.1]= Q[P#2.0+P#1.1]=Q3.1,

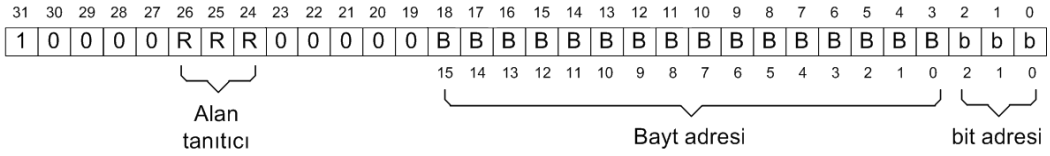
MB[AR1, P#0.0]= MB[P#0.0+P#10.0+ P#0.0]=MB10,

MW[AR2, P#0.0]= MW[P#2.0+P#4.0+ P#0.0]=MW6,

MD[AR2, P#6.0]= [P#2.0+P#4.0+P#6.0]=MD12,

gerçek adresleri üzerinde işlem yapılır.

Adres kaydedicileri ile alan (I, Q, M, L, ..) bilgisi de adreslenebilir. Bu tür adresleme, alan geçişli dolaylı adresleme olarak tanımlanır. Alan geçişli dolaylı adreslemede işaretçi yapısı Şekil 9.78'de verilmiştir.



Şekil 9.78 Alan geçişli işaretçi yapısı

Alan geçişli işaretçide alan bilgisi, işaretçinin RRR (24-26) bitlerinde tutulur. Eğer,

RRR=000 ise (I/O), RRR=001 ise (PII), RRR=010 ise (PIQ) RRR=011 ise M bellek,

RRR=100 ise DB'deki veri, RRR=101 ise DI'deki veri, RRR=110 ise yerel bellek (L)' RRR=111 ise çağrılan bloğun yerel bellek alanı olduğu tanıtılmış olur. Şekil 9.79'da alan geçişli dolaylı adreslemeye ilişkin bir program örneği verilmiştir.

```
LAR1  P#I 0.0

LAR2  P#Q 2.0

A      [AR1,P#0.1]
=      [AR2,P#1.1]

L     P#M 20.0
LAR1

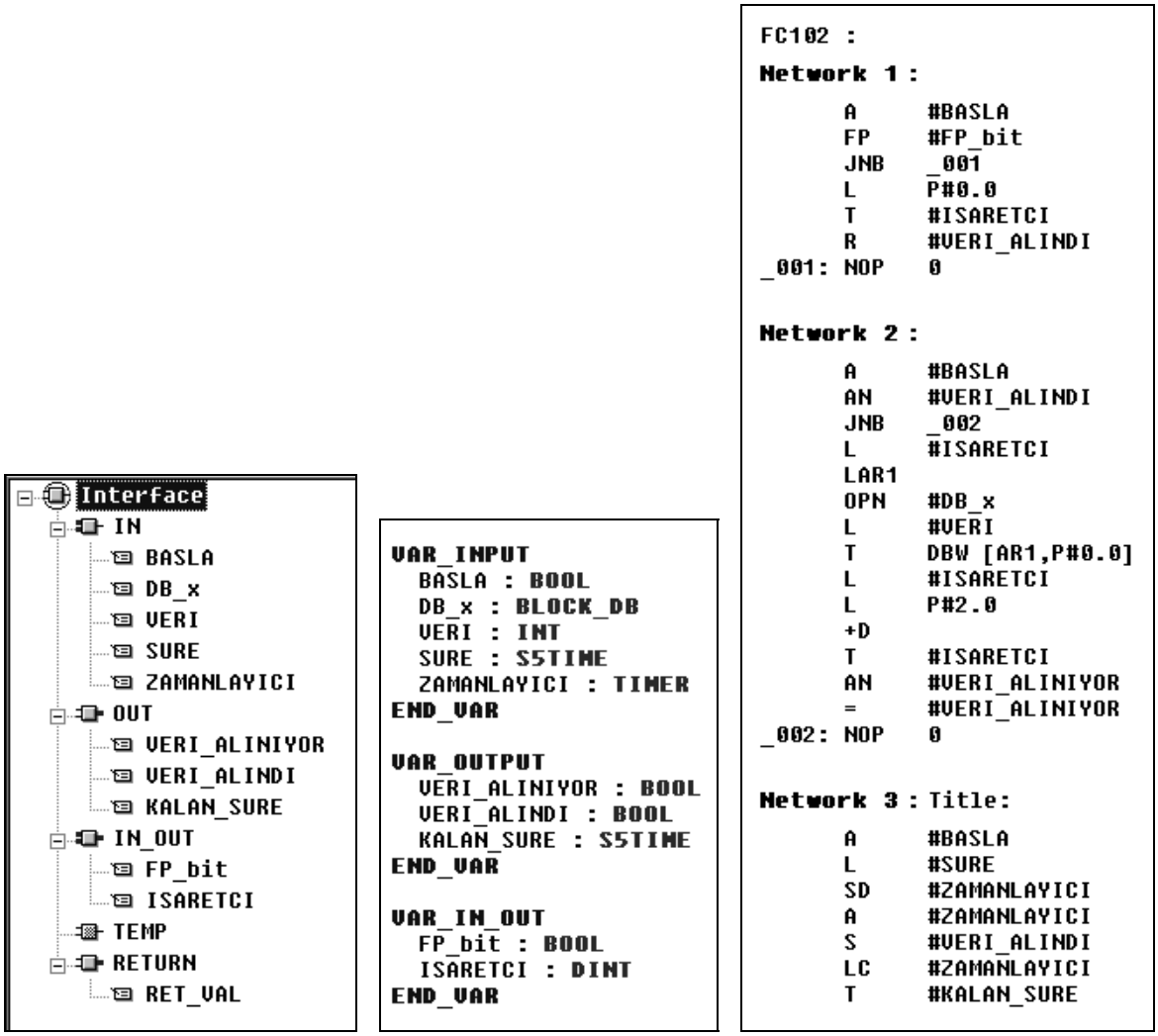
L     B [AR1,P#2.0]
T     W [AR1,P#4.0]
T     D [AR1,P#6.0]
```

Şekil 9.79 Alan geçişli dolaylı adreslemeye ilişkin bir program örneği

Bu programda, Q3.1=I0.1 mantıksal işlemi gerçeklenir ve MB22 adresindeki veri MW24 ve MD26 adreslerine aktarılır.

Örnek 9.7 Bir sürecin, basamak girişe yanıtına ilişkin verilerin toplanmasını sağlayan bir fonksiyon (FC) yazınız.

Bu amaçla kullanılacak bir fonksiyona ilişkin giriş (VAR_INPUT), çıkış (VAR_OUTPUT), giriş/çıkış (VAR_IN_OUT) değişkenleri ve program Şekil 9.80'deki gibi yazılabilir.

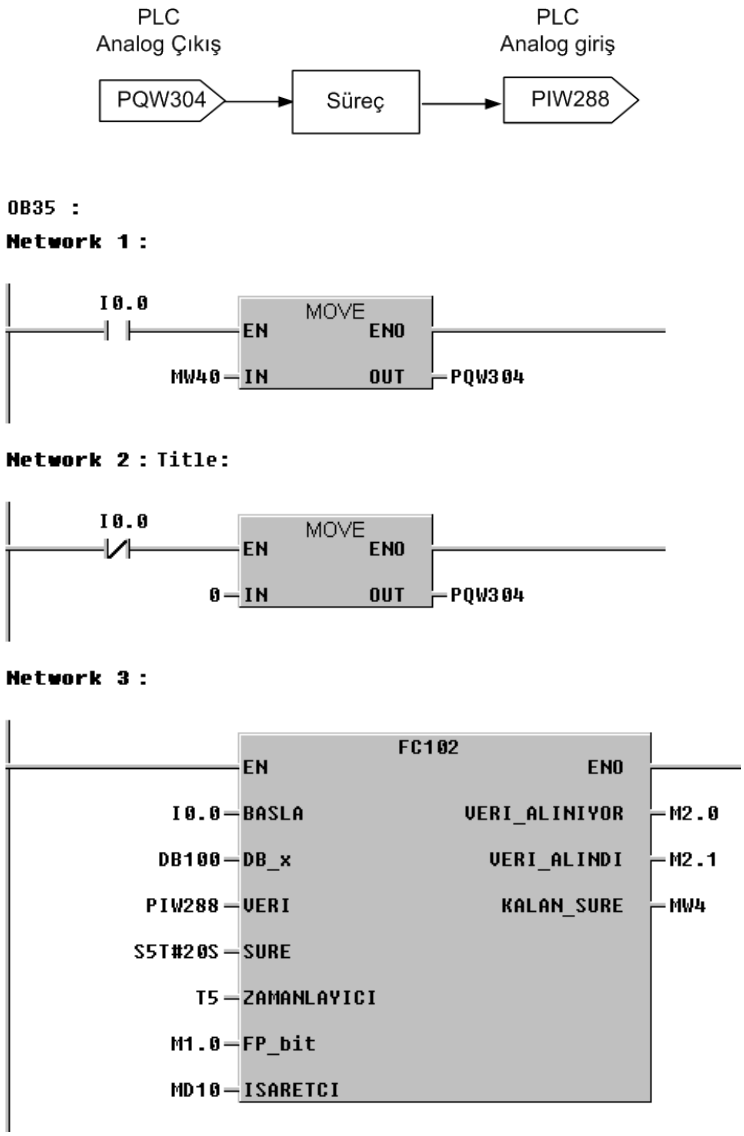


Şekil 9.80 Veri toplamaya ilişkin bir fonksiyon programı

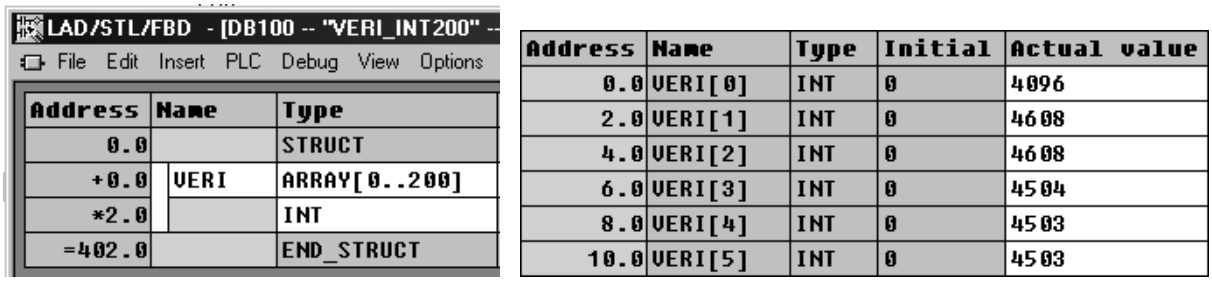
Bu fonksiyon kullanılarak veri toplamaya ilişkin bir program örneği Şekil 9.81'de verilmiştir. Burada PLC PQW304 analog çıkış adresi üzerinden sürece MW40 adresindeki sayısal veri uygulanmakta ve süreç yanıtı PIW288 analog giriş adresinden okunarak DB100 veri bloğuna kaydedilmektedir. Kaydetme süreci I0.0=1 yapıldığında başlamakta ve süre dolduğunda sona ermektedir. Verilerin belirli aralıklarla kaydedilmesini sağlamak için fonksiyon, zamana bağlı kesmeli çalışan bir organizasyon bloğundan (OB35) işletilmektedir. Verilerin 100ms aralıklarla kaydedilmesi için CPU donanım

ayarları bölümünde, OB35'in kesmeli çalışma sıklığının tanımlandığı "Cyclic Interrupts" başlığı altındaki alana 100 değeri yazılır.

İşaretli tamsayı türündeki verilerin (16 bit), 100ms aralıklarla ve 20s süre ile kaydedileceği göz önüne alınarak DB100 veri bloğu Şekil 9.82'de gösterildiği gibi tanımlanabilir.



Şekil 9.81 Veri toplamaya ilişkin bir program örneği



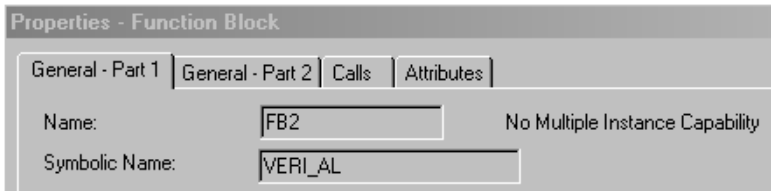
Şekil 9.82 DB100 veri bloğuna ilişkin tanımlamalar ve verilerin görünümü

Örnek 9.8 Bir sürecin, basamak girişe yanıtına ilişkin verilerin toplanmasını sağlayan bir fonksiyon blok (FB) yazınız.

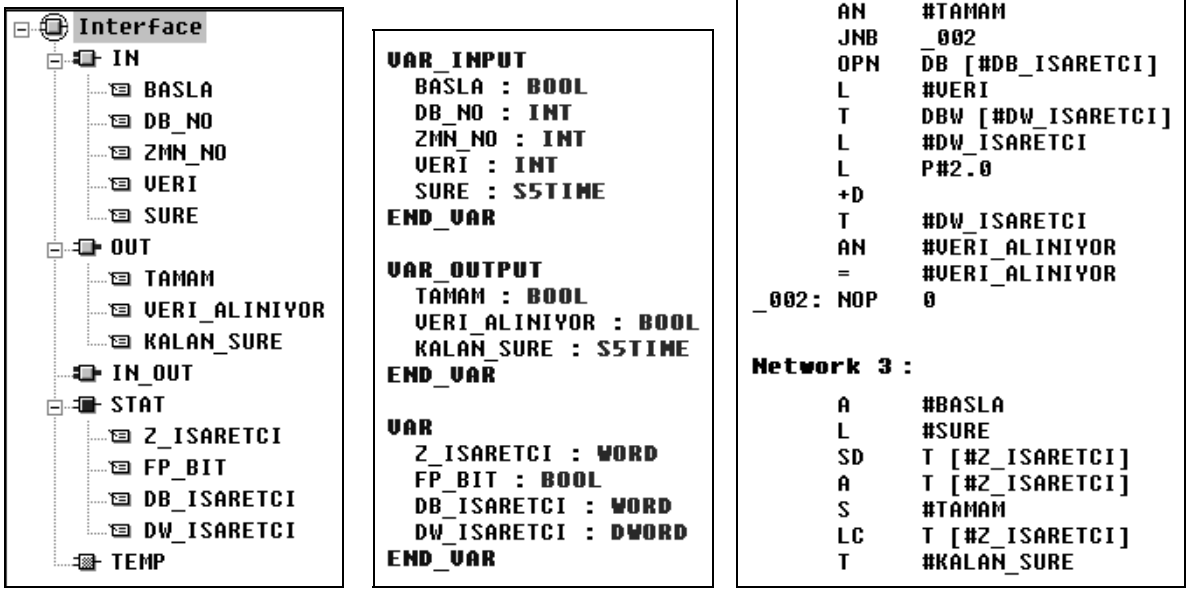
Bu amaçla kullanılacak bir fonksiyon bloğuna ilişkin giriş (VAR_INPUT), çıkış (VAR_OUTPUT) değişkenleri, kalıcı (STAT) değişkenler ve program Şekil 9.83'deki gibi yazılabilir. Fonksiyon bloklarında geçerli olan kalıcı değişkenler (STAT) işaretçi olarak kullanılabilir. Ancak, bu değişkenlerin işaretçi olarak kullanılabilmesi için FB özellikleri bölümündeki "Mul. Ins. Cap." özelliği kaldırılmalıdır. Bu işlem FB açılırken yapılmalıdır. Bir veri bloğunun birden çok FB ile kullanılmasına olanak sağlayan bu özelliğin kaldırılmaması durumunda "STAT" türünde tanımlanmış değişkenler işaretçi olarak geçerli değildir.



Bu özellik kaldırıldıktan sonra, fonksiyon blok özellikleri bölümü

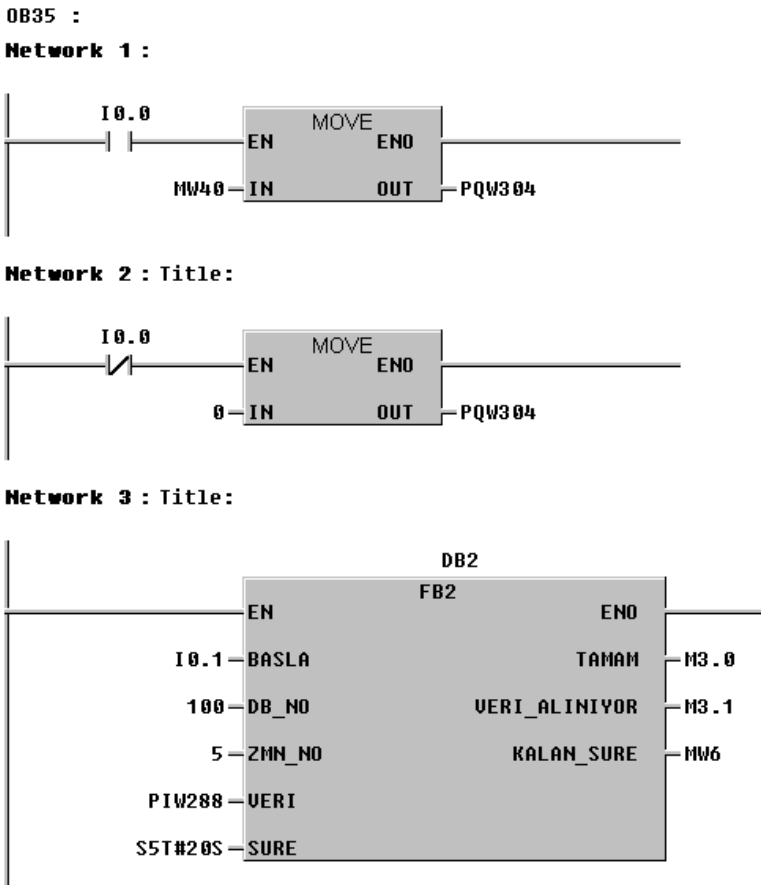


biçiminde görülür.



Şekil 9.83Veri toplamaya ilişkin bir fonksiyon blok programı

Bu fonksiyon blok kullanılarak veri toplamaya ilişkin bir program örneği Şekil 9.84'te verilmiştir.

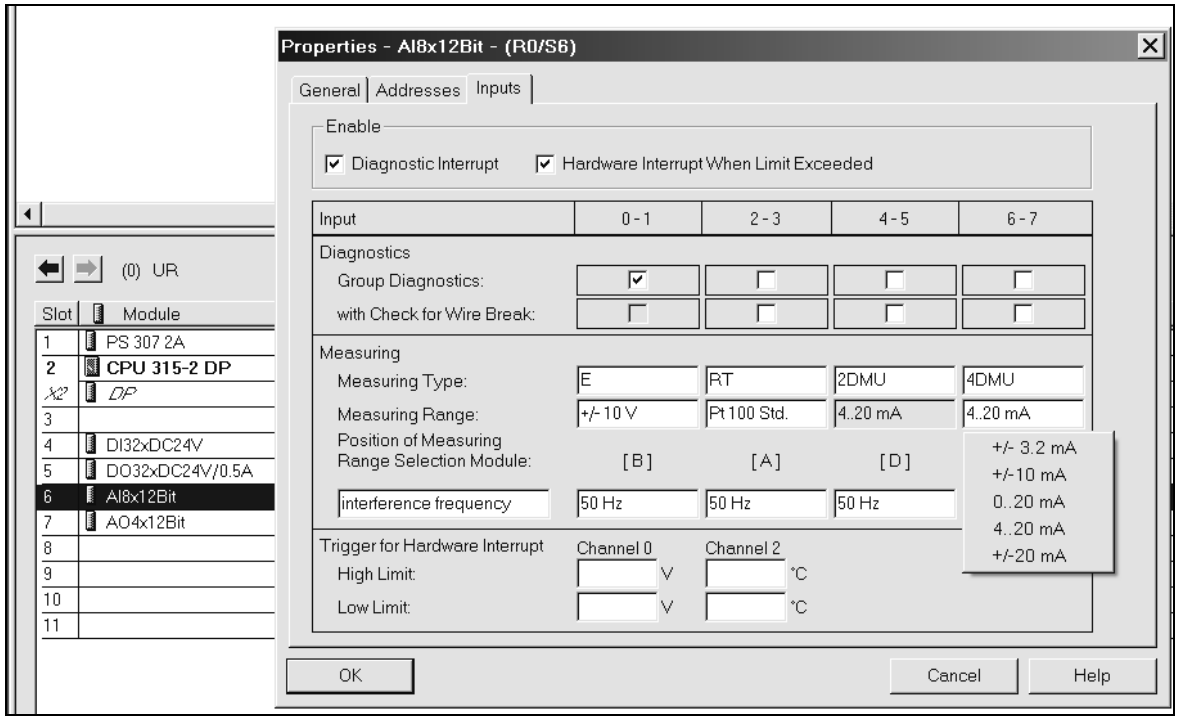


Şekil 9.84 FB ile veri toplamaya ilişkin bir program örneği

Şekil 9.82'de tanımlanan DB100, DB2, FB2 ve OB35 blokları PLC'ye yüklenerek, Örnek 9.7 ile aynı işlevi sağlayan program elde edilir.

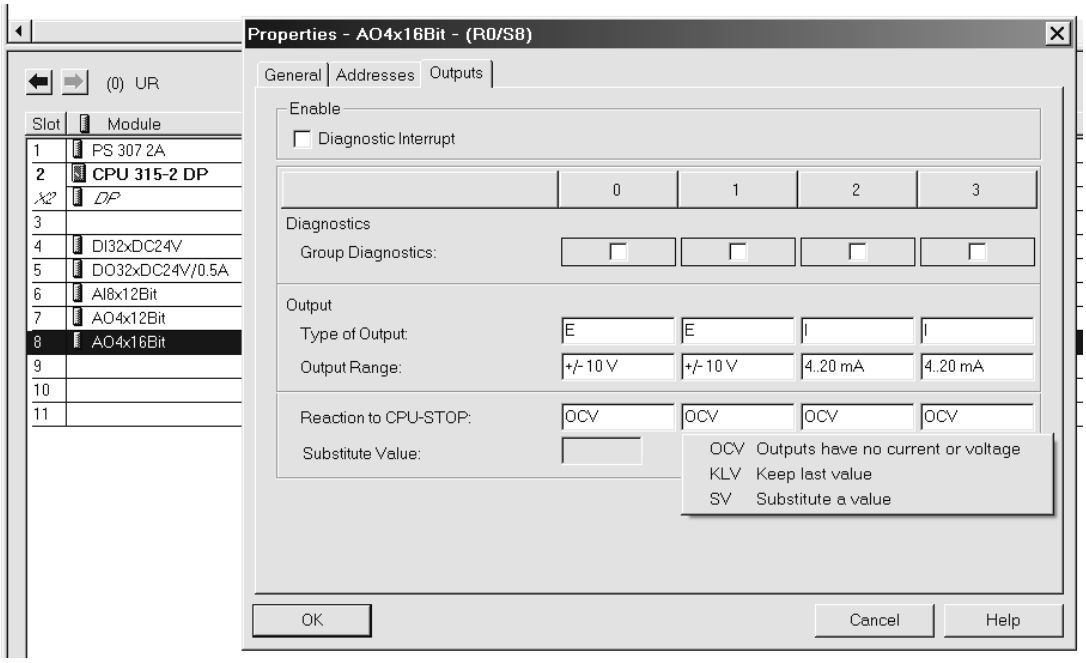
9.6 ANALOG VERİLERE ERİŞİM VE KONTROL UYGULAMALARI

Kontrol edilen sistem ile PLC arasındaki iletimi sağlayan analog giriş-çıkış birimlerinin özelliklerinin tanımlanması, "SIMATIC Manager" yazılımındaki donanım (hardware) ayarları bölümünde yapılır. Şekil 9.85'te bir analog giriş birimi, Şekil 9.86'da bir analog çıkış biriminin ayarlarına ilişkin bölümler gösterilmiştir.



Şekil 9.85 Analog giriş birimi donanım ayarlarına ilişkin seçenekler

Analog birimlerin çözünürlük değerlerine bağlı olarak ölçülen değerler belirli bir hata payı ile sayısal değerlere veya analog işaretlere dönüştürülür. Şekil 9.87'de çözünürlüğü bağlı olarak elde edilen sayısal değer aralıkları gösterilmiştir.



Şekil 9.86 Analog çıkış birimi donanım ayarlarına ilişkin seçenekler

Bit no.	min. birim		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit değeri	Dec.	Hex.	VZ	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^{9}	2^{8}	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}
8	128	80	*	*	*	*	*	*	*	*	*	0	0	0	0	0	0	0
9	64	40	*	*	*	*	*	*	*	*	*	*	0	0	0	0	0	0
10	32	20	*	*	*	*	*	*	*	*	*	*	*	0	0	0	0	0
11	16	10	*	*	*	*	*	*	*	*	*	*	*	*	0	0	0	0
12	8	8	*	*	*	*	*	*	*	*	*	*	*	*	*	0	0	0
13	4	4	*	*	*	*	*	*	*	*	*	*	*	*	*	*	0	0
14	2	2	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	0
15	1	1	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*

Şekil 9.87 Çözünürlüğe bağlı olarak analog değerlere karşı düşen sayısal değer aralıkları

Analog giriş biriminden okunan işaretlerin genliğine bağlı olarak elde edilen sayısal değerler, Şekil 9.88'de, analog çıkış birimi adresine yazılan sayısal verilere karşı düşen işaretlerin genlikleri Şekil 9.89'da gösterilmiştir.

	Gerilim		Akım		Sıcaklı ölçümü Pt100	
	± 10V	Değer	4 - 20mA	Değer	-200...+850°C	Değer
Yukarı taşma	>= 11.759	32767	>= 22.815	32767	>= 1000.1	32767
Anma değer üstü	11.7589 : 10.0004	32511 : 27649	22.810 : 20.0005	32511 : 27649	1000.0 : 850.1	10000 : 8501
Anma değeri	10.00 7.50 : -7.5 -10.00	27648 20736 : -20736 -27648	20.000 16.000 : 4.000	27648 20736 : 0	850.0 : : -200.0	8500 : : -2000
Anma değer altı	- 10.0004 : - 11.759	- 27649 : - 32512	3.9995 : 1.1852	- 1 : - 4864	- 200.1 : - 243.0	- 2001 : - 2430
Aşağı taşma	<= - 11.76	- 32768	<= 1.1845	- 32768	<= - 243.1	- 32768

Şekil 9.88 Analog işaret büyüklükleri ve karşı düşen sayısal değerler

Alan	Sayısal değer	Gerilim			Akım		
		Çıkış ayar değerleri			Çıkış ayar değerleri		
		0 - 10V	1 - 5V	± 10V	0 - 20mA	4 - 20mA	± 20mA
Yukarı taşma	>=32767	0	0	0	0	0	0
Anma değer üstü	32511 : 27649	11.7589 : 10.0004	5.8794 : 5.0002	11.7589 : 10.0004	23.515 : 20.0007	22.81 : 20.005	23.515 : 20.0007
Anma değeri	27648 : 0 : - 6912 - 6913 : : : - 27648	10.0000 : 0 0	5.0000 : 1.0000 0.9999 0 0	10.0000 : 0 : : : : : : : -10.0000	20.000 : 0 0	20.000 : 4.000 3.9995 0 0	20.000 : 0 : : : : : : : -20.000
Anma değer altı	- 27649 : - 32512			- 10.0004 : - 11.7589			- 20.007 : - 23.515
Aşağı taşma	<=- 32513			0			0

Şekil 9.89 Sayısal değerlere karşı düşen analog işaret büyüklükleri

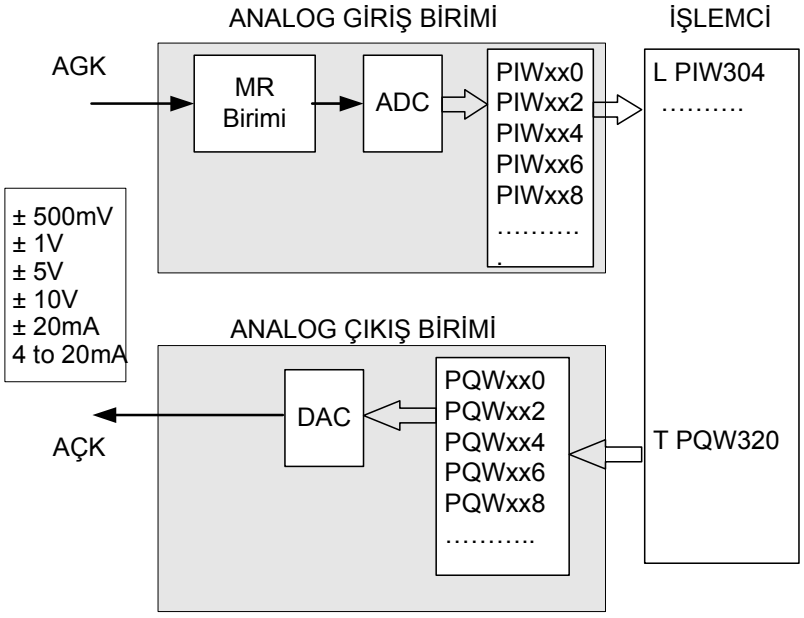
9.6.1 ANALOG VERİLERE ERİŞİM

Analog giriş biriminden alınan analog işaretler bir analog sayısal dönüştürücü (ADC) birimi üzerinden, sayısal değerlere dönüştürülerek analog giriş birimi belleğine yazılır (Şekil 9.90). Analog giriş birimi belleğine yazılan veriler bir sonraki işleme kadar değişmez. Analog işaretlere ilişkin verilere "L" komutu ve ilgili kanala (AGK) ilişkin adres belirtilerek erişilir. Örneğin,

```
L     PIW  304
ITD
DTR
T     MD    10
```

I

komutları ile analog işarete ilişkin PIW304 adresindeki veri alınır ve reel sayı biçimine dönüştürülerek MD10 bellek adresine yazılır.



Şekil 9.90 Analog verilerin okunması ve yazılması

Analog çıkış birimi, analog çıkış birimi belleğine yazılan sayısal verileri sayısal analog dönüştürücü (DAC) birimi üzerinden, analog işaretlere dönüştürür. Analog çıkış birimine sayısal verilerin aktarılması "T" komutu ve ilgili adres belirtilerek yapılır. Örneğin,

```
L     MD    40
RND
T     PQW   320
```

komutları ile MD40 adresindeki reel sayı, işaretli tamsayıya dönüştürülerek analog çıkış biriminin ilgili kanalına (AÇK) ilişkin PQW320 adresine yazılır.

9.6.2 KONTROL UYGULAMALARI

9.6.2.1 İki konumlu kontrol

Bölüm 8.3.1'de incelenen bu kontrol yöntemine ilişkin kontrol kuralına göre istenen değer (SP) ile gerçek değer (PV) karşılaştırılır, SP>PV ise sisteme enerji verilir, değilse sisteme verilen enerji kesilir. Bu koşulları sağlayan en basit çözümü sağlamak için Şekil 9.91'de verilen fonksiyon kullanılabilir. Bu fonksiyona göre, DUR=1 ve BASLA=1 yapıldığında sistem devreye girer (SD=1), sistem devrede iken SP>PV olduğunda CALIS=1 ve buna bağlı olarak sisteme enerji verilmesini sağlayan düzeneği çalıştıran K1=1 olur. DUR=0 yapıldığında sistem devreden çıkar ve K1=0 olur. Fonksiyonda çıkış olarak gösterilen HATA işareti, SP ile PV değerleri arasındaki farkın işaretli tamsayı olarak karşılığını verir.

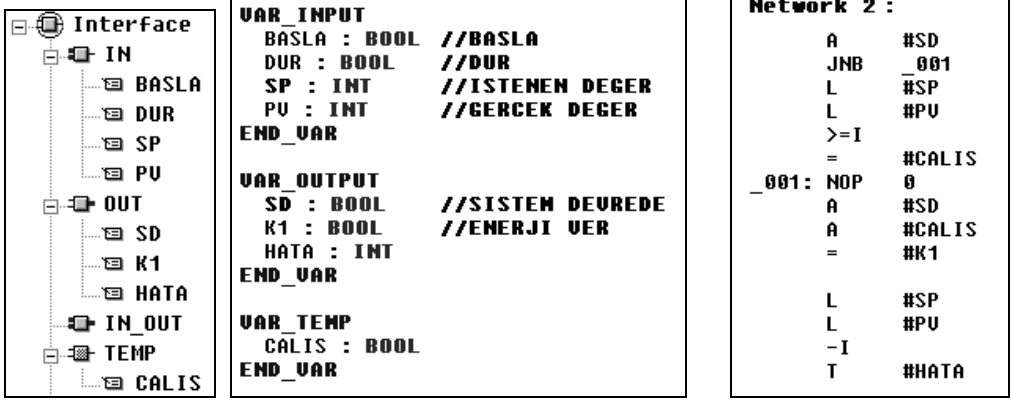
İki konumlu kontrol işleminin istenen bir sıklıkta yapılmasını sağlamak için, zaman bağlı kesmeli çalışan bir organizasyon bloğu veya bir zamanlayıcı kullanılabilir. Zamana bağlı kesmeli çalışan OB35 organizasyon bloğu ile gerçeklemek için Şekil 9.92'deki program yazılır.

Kontrol sisteminin anahtarlama sıklığını azaltmak için kontrol kuralı, belirli bir hata payı büyüklüğüne bağlı olarak ifade edilebilir. Eğer, kontrol kuralı

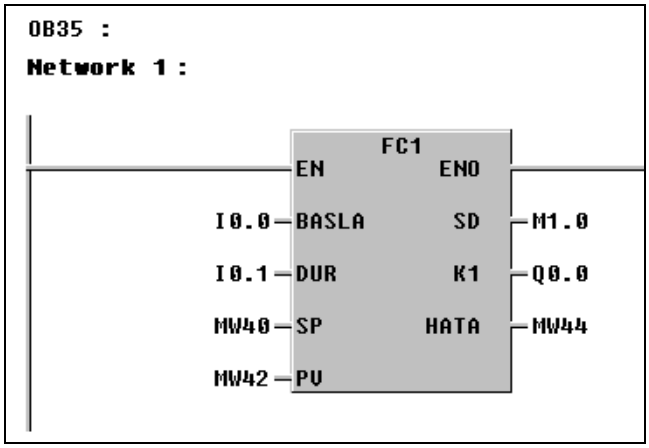
PV>SP+HATA_REF ise K1=0

PV<SP-HATA_REF ise K1=1

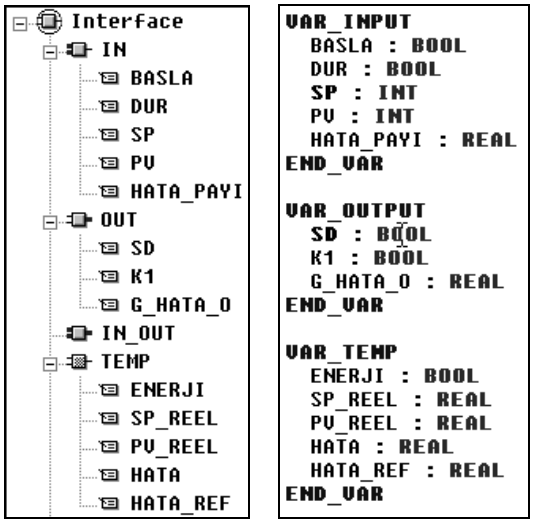
olarak alınırsa sisteme enerji verme sıklığı azalır. Burada HATA_REF büyüklüğü, yüzde olarak verilen HATA_PAYI ile SP çarpımından oluşur. Bu kuralı gerçeklemek üzere Şekil 9.93'te verilen fonksiyon yazılabilir.



Şekil 9.91 İki konumlu kontrol kuralını gerçekleyen FC1 fonksiyonuna ilişkin program



Şekil 9.92 FC1 fonksiyonunun OB35 organizasyon bloğundan işletilmesine ilişkin program




```
VAR_INPUT
  BASLA : BOOL
  DUR : BOOL
  SP : INT
  PV : INT
  HATA_PAYI : REAL
END_VAR

VAR_OUTPUT
  SD : BOOL
  K1 : BOOL
  G_HATA_O : REAL
END_VAR

VAR_TEMP
  ENERJI : BOOL
  SP_REEL : REAL
  PV_REEL : REAL
  HATA : REAL
  HATA_REF : REAL
END_VAR
```

```
FC2 :
Network 1 :
      A     #BASLA
      S     #SD
      AN    #DUR
      R     #SD

Network 2 :
      A     #SD
      JNB   _001
      L     #SP
      ITD
      DTR
      T     #SP_REEL

      L     #PV
      ITD
      DTR
      T     #PV_REEL

      L     #SP_REEL
      L     #PV_REEL
      -R
      T     #HATA

      L     #HATA_PAYI
      L     #SP_REEL
      *R
      L     1.000000e+002
      /R
      T     #HATA_REF
```

```
Network 3 :

      L     #HATA
      L     #HATA_REF
      >R
      S     #ENERJI

      L     #HATA
      L     #HATA_REF
      NEGR
      <R
      R     #ENERJI
_001: NOP   0


Network 4 :

      L     #HATA
      L     #SP_REEL
      /R
      L     1.000000e+002
      *R
      T     #G_HATA_O

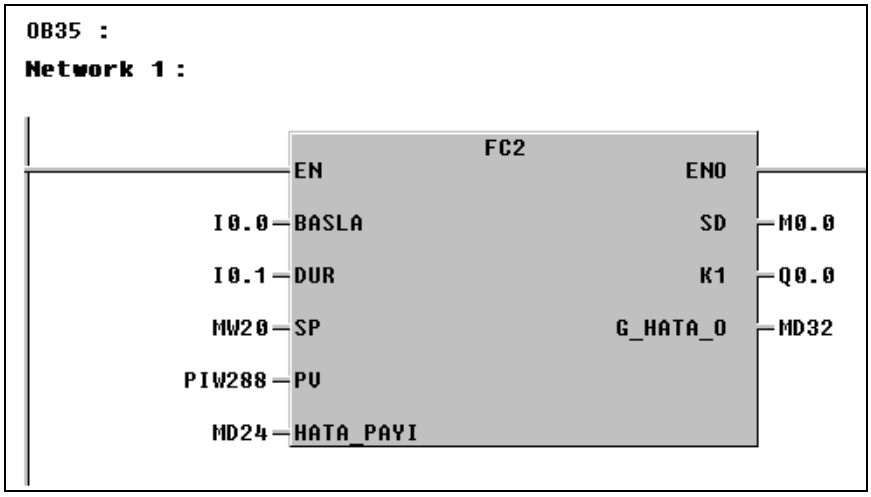
Network 5 : Title:

      A     #SD
      A     #ENERJI
      =     #K1
```

Şekil 9.93 İki konumlu kontrol kuralına ilişkin bir fonksiyon programı

Şekil 9.93'teki FC2 fonksiyonu, OB35 bloğuna Şekil 9.94'te görüldüğü gibi yazılarak yürütülebilir. Bu durumda, DUR=1 ve BASLA=1 yapıldığında sistem devreye girer (SD=1), sistem devrede iken PV<SP-HATA_REF ise K1=1, PV>SP+HATA_REF ise K1=0 işareti üretilir. Eğer Örneğin, HATA_PAYI=5.0 için $\text{HATA_REF}=\text{HATA_PAYI}\times\text{SP}=5/100\times\text{SP}=0.05\times\text{SP}$ olur ve PV değeri SP değerinin 5% üstüne çıktığı anda K1=0, 5% altında iken K1=1 olur.

Hatanın yüzde değeri $\text{G_HATA_O}=100\times(\text{SP}-\text{PV})/\text{SP}$ ilişkisine göre hesaplanır ve G_HATA_O işareti olarak yazılır.



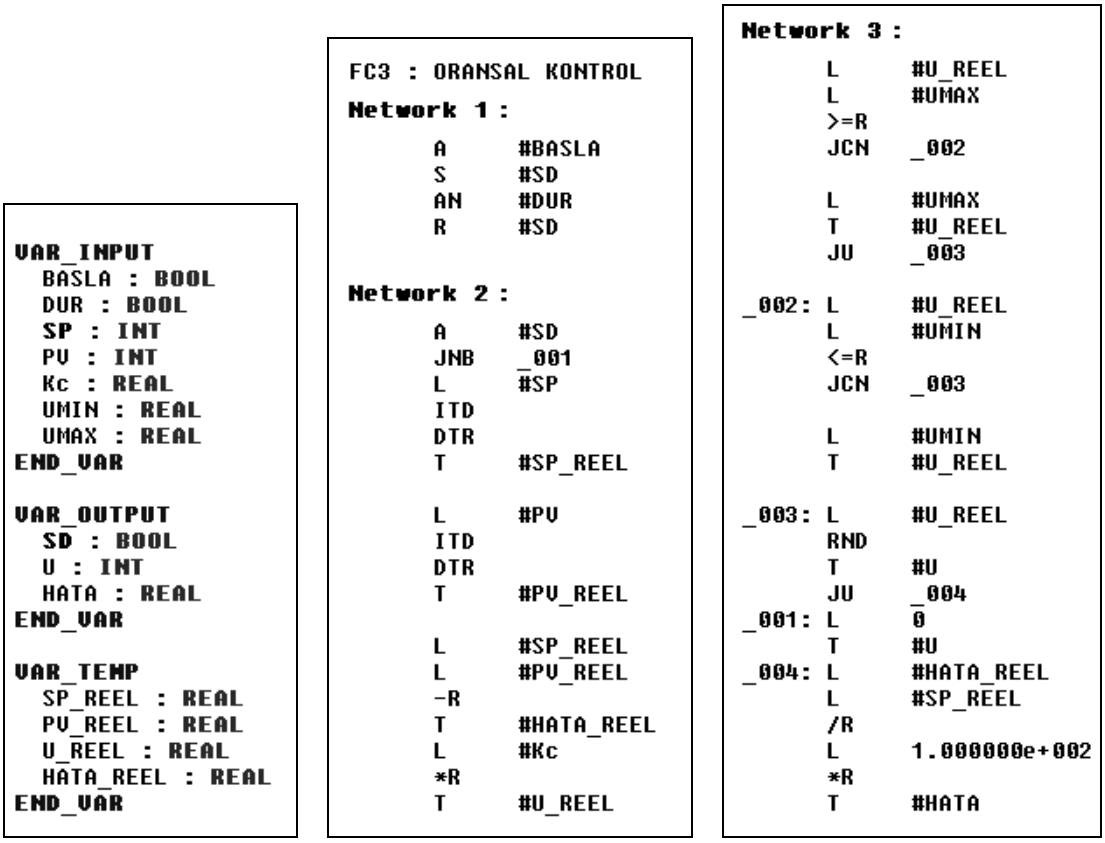
Şekil 9.94 FC2 fonksiyonun OB35 bloğundan işletilmesine ilişkin program

9.6.2.2 Oransal Kontrol

Kontrol kuralı,

$$u = K_C(SP - PV) \tag{9.1}$$

biçiminde ifade edilen bir kontrol yöntemi olarak tanımlanır. Burada, SP kontrol edilen büyüklüğe ilişkin istenen değer, PV kontrol edilen büyüklüğün gerçek değeri ve K_C oransal katsayıdır. Oransal kontrol kuralının gerçeklenmesine ilişkin bir fonksiyon (FC3) Şekil 9.95'te verilmiştir. Bu fonksiyonda UMAX ve UMIN girişleri kontrol işaretinin uygulanabilir sınır değerlerini belirtmek için kullanılır.



```
VAR_INPUT
  BASLA : BOOL
  DUR : BOOL
  SP : INT
  PV : INT
  Kc : REAL
  UMIN : REAL
  UMAX : REAL
END_VAR

VAR_OUTPUT
  SD : BOOL
  U : INT
  HATA : REAL
END_VAR

VAR_TEMP
  SP_REEL : REAL
  PV_REEL : REAL
  U_REEL : REAL
  HATA_REEL : REAL
END_VAR
```

```
FC3 : ORANSAL KONTROL
Network 1 :
      A     #BASLA
      S     #SD
      AN    #DUR
      R     #SD

Network 2 :
      A     #SD
      JNB   _001
      L     #SP
      ITD
      DTR
      T     #SP_REEL

      L     #PV
      ITD
      DTR
      T     #PV_REEL

      L     #SP_REEL
      L     #PV_REEL
      -R
      T     #HATA_REEL
      L     #Kc
      *R
      T     #U_REEL
```

```
Network 3 :
      L     #U_REEL
      L     #UMAX
      >=R
      JCN   _002

      L     #UMAX
      T     #U_REEL
      JU    _003

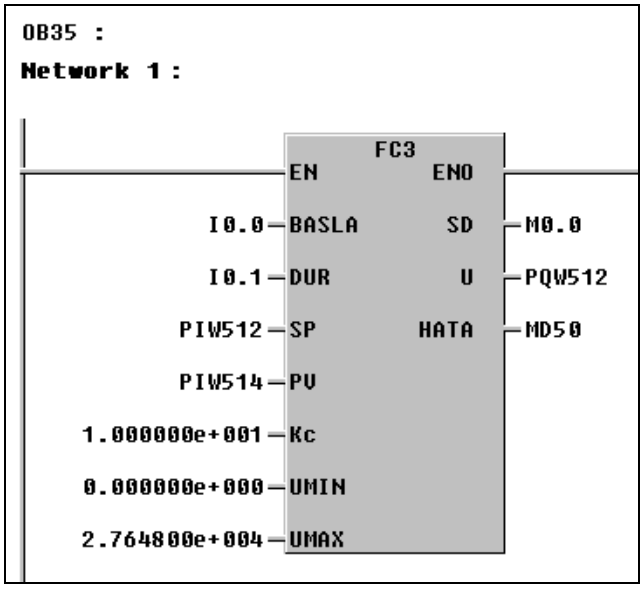
_002: L     #U_REEL
      L     #UMIN
      <=R
      JCN   _003

      L     #UMIN
      T     #U_REEL

_003: L     #U_REEL
      RND
      T     #U
      JU    _004
_001: L     0
      T     #U
_004: L     #HATA_REEL
      L     #SP_REEL
      /R
      L     1.000000e+002
      *R
      T     #HATA
```

Şekil 9.95 Oransal kontrol kuralına ilişkin bir fonksiyon programı

Şekil 9.95'te verilen fonksiyon, Şekil 9.96'daki gibi yazılarak bir organizasyon bloğundan yürütülür. Kontrol kuralı örnekleme zamanına bağlı olmamasına karşın, oransal kontrol fonksiyonun zamana bağlı kesmeli çalışan bir OB içinde yürütülmesi ile, hem belirli bir zaman diliminde kontrol işaretinin uygulanması güvence altına alınır, hem de analog-sayısal ve sayısal-analog dönüştürme sürelerinin altında fonksiyonun işletilmesi engellenmiş olur.



Şekil 9.96 FC3 fonksiyonun OB35 bloğundan işletilmesine ilişkin program

9.6.2.3 PI Kontrol

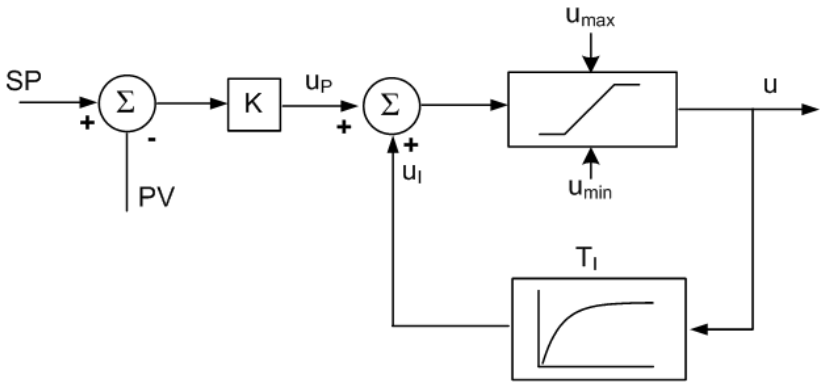
PI kontrol kuralı; u kontrol işareti, e hata işareti, K_C oransal katsayı ve T_I integral zaman sabiti olmak üzere

$$u(t) = K_C[e(t) + \frac{1}{T_I}\int e(t)dt] \tag{9.2}$$

biçiminde ifade edilir. Bu ifadenin sayısal gerçeklenmesi ile PI kontrol işlevini sağlayan bir fonksiyon veya fonksiyon blok elde edilebilir. Bölüm 8.3.3'te incelenen gerçekleme şeması Şekil 9.97'deki gibi düzenlenirse, gerçeklemeye ilişkin fark denklemler

$$\begin{aligned} u_p(k) &= K_C[SP(k) - PV(k)] \\ u_I(k) &= Au_I(k-1) + (1-A)u(k-1) \\ u(k) &= u_p(k) + u_I(k) \end{aligned} \tag{9.3}$$

biçiminde yazılabilir. Bu ifadelerin gerçeklenmesine ilişkin bir fonksiyon blok (FB44) Şekil 9.98'deki gibi yazılabilir. Burada $A=\exp(-T/T_I)$ olarak tanımlanan bir katsayıdır. İntegral katsayısı $T_I=0$ yapıldığında bu değer hesaplanmaz ve fonksiyon blok yalnız oransal kontrol işlevini sağlar. I0.0=1 yapıldığında integral içeriği ve kontrol işareti sıfırlanır ve PI kontrol işlevi yerine getirilmez.



Şekil 9.97 PI kontrol kuralını gerçeklemeye ilişkin blok diyagramı

Şekil 9.98'de verilen fonksiyon blok, Şekil 9.99'da gösterildiği gibi bir organizasyon bloğundan yürütülerek PI kontrol işlevi sağlanır. Bu programa göre, istenen değer (SP) analog PIW304 adresinden, kontrol edilen büyüklüğe ilişkin ölçülen değer analog PIW306 adresinden alınır ve hesaplanan kontrol işaretinin sayısal değeri PQW320 analog çıkış adresine, hatanın yüzde değeri MD20 adresine yazılır.

```
FUNCTION_BLOCK FB 44

VAR_INPUT
  SP : INT
  PV : INT
  Kc : REAL
  TI : REAL
  T : REAL
  UMAX : REAL
  UMIN : REAL
  RST : BOOL
END_VAR

VAR_OUTPUT
  U : INT
  E : REAL
END_VAR

VAR
  UI_1 : REAL
  U_1 : REAL
END_VAR

VAR_TEMP
  PV_REEL : REAL
  SP_REEL : REAL
  UP_REEL : REAL
  UI_REEL : REAL
  U_REEL : REAL
  E_REEL : REAL
  A : REAL
END_VAR
```

```
FB44 : PI_AWR
Network 1 :
      A     #RST
      JCN   _001
      L     0.000000e+000
      T     #UI_1
      T     #U_1
      BE

Network 2 :
_001: L     #SP
      ITD
      DTR
      T     #SP_REEL

      L     #PV
      ITD
      DTR
      T     #PV_REEL

      L     #SP_REEL
      L     #PV_REEL
      -R
      T     #E_REEL
      L     #Kc
      *R
      T     #UP_REEL

      L     #TI
      L     0.000000e+000
      >R
      JCN   _002
      L     #T
      L     #TI
      /R
      NEGR
      EXP
      T     #A
      L     #UI_1
      *R
      T     #UI_REEL

      L     1.000000e+000
      L     #A
      -R
      L     #U_1
      *R
      L     #UI_REEL
      +R
      T     #UI_REEL
```

```
Network 3 :
      L     #UP_REEL
      L     #UI_REEL
      +R
      T     #U_REEL
      JU    _003

Network 4 :
_002: L     #UP_REEL
      T     #U_REEL

Network 5 :

_003: L     #U_REEL
      L     #UMAX
      >=R
      JCN   _004
      L     #UMAX
      T     #U_REEL
      JU    _005

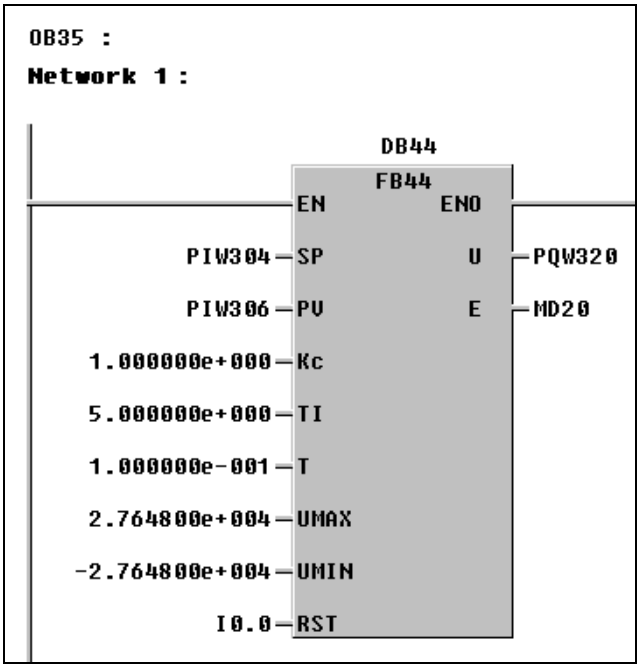
_004: L     #U_REEL
      L     #UMIN
      <=R
      JCN   _005
      L     #UMIN
      T     #U_REEL
_005: NOP   0

Network 6 :
      L     #U_REEL
      T     #U_1
      L     #UI_REEL
      T     #UI_1

Network 7 :
      L     #U_REEL
      RND
      T     #U

Network 8 :
      L     #E_REEL
      L     #SP_REEL
      /R
      L     1.000000e+002
      *R
      T     #E
```

Şekil 9.98 PI kontrol kuralına ilişkin bir fonksiyon programı



Şekil 9.99 FB44 PI fonksiyonun OB35 bloğundan işletilmesine ilişkin program

9.6.2.4 Birinci Mertebeden Alçak Geçiren Filtre Gerçekleme

Köşe frekansı ω_c olan birinci mertebeden alçak geçiren bir fitreye ilişkin fark denklemi

$$y(k) = A_{WC}y(k-1) + (1-A_{WC})x(k-1) \qquad (9.4)$$

biçiminde yazılabilir (Bölüm 8.3.4). Bu fark denkleminin gerçeklenmesine ilişkin bir fonksiyon Şekil 9.100'deki gibi programlanabilir. Bu fonksiyon, Şekil 9.101'de gösterildiği gibi bir organizasyon bloğundan yürütülerek alçak geçiren filtre işlevi görür. Bu programa göre, analog PIW304 adresinden okunan sayısal işaret, köşe frekansı $\omega_c = 0.1\text{rad/s}$ olan bir filtre ile süzülerek MW200 adresine yazılır. Burada, örnekleme zamanı, Bölüm 8.3.4'te verilen seçim ölçütüne göre

$$\omega_s = 2\times 100\times \omega_c = 2\times 100\times 0.1 = 20\text{rad/s}$$

$$T \le \frac{2\pi}{\omega_s} = \frac{2\pi}{20} = 0.314\text{s}$$

koşulunu sağlayan bir değer, örneğin T=0.3s alınabilir.

```
FUNCTION FC 4 :

VAR_INPUT
  X : INT
  Wc : REAL
  T : REAL
  XMIN : REAL
  XMAX : REAL
  YMIN : REAL
  YMAX : REAL
END_VAR

VAR_OUTPUT
  Y : INT
END_VAR

VAR_IN_OUT
  X_1 : REAL
  Y_1 : REAL
END_VAR

VAR_TEMP
  Awc : REAL
END_VAR
```

```
FC4 :
Network 1 :
      L     #T
      L     #Wc
      *R
      NEGR
      EXP
      T     #Awc
      L     #Y_1
      *R
      T     #Y_1

      L     1.000000e+000
      L     #Awc
      -R
      L     #X_1
      *R
      L     #Y_1
      +R
      T     #Y_1

Network 2 :
      L     #Y_1
      L     #YMAX
      >=R
      JCN   _001
      L     #YMAX
      T     #Y_1
_001: NOP   0

Network 3 :
      L     #Y_1
      L     #YMIN
      <=R
      JCN   _002
      L     #YMIN
      T     #Y_1
_002: NOP   0
```

```
Network 4 :
      L     #X
      ITD
      DTR
      T     #X_1

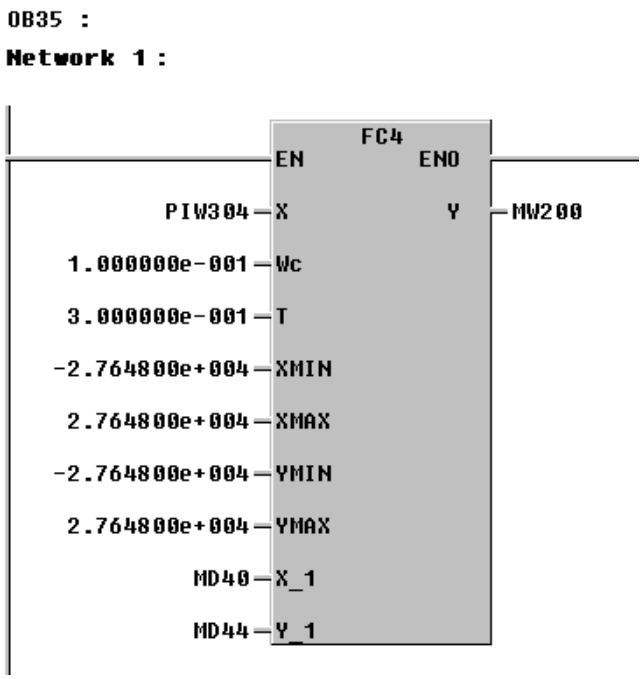
Network 5 :
      L     #X_1
      L     #XMAX
      >=R
      JCN   _003
      L     #XMAX
      T     #X_1
_003: NOP   0

Network 6 :
      L     #X_1
      L     #XMIN
      <=R
      JCN   _004
      L     #XMIN
      T     #X_1
_004: NOP   0

Network 7 :
      L     #Y_1
      RND
      T     #Y


Network 8 :
      AN    OS
      AN    UO
      SAVE
```

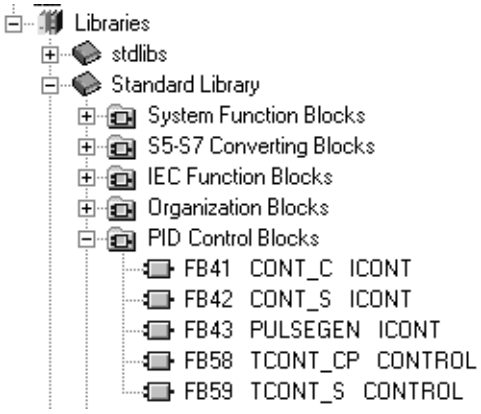
Şekil 9.100 Alçak geçiren filtreye ilişkin bir fonksiyon programı



Şekil 9.101 FC4 alçak geçiren filtre fonksiyonun OB35 bloğundan işletilmesine ilişkin program

9.6.2.5 PID Kontrol

PID kontrol uygulamaları için hazır kütüphane fonksiyonlarından (Şekil 9.102) yararlanılabilir.



Şekil 9.102 Kontrol uygulamalarına ilişkin kütüphane fonksiyonları

Bölüm 8.3.2'de verilen PID kontrol kuralına ilişkin

$$u(t) = K_C\left(e(t) + \frac{1}{T_I}\int e(t)dt + T_D\frac{de(t)}{dt} \right)$$

ifadeyi gerçekleyen bir program yazılarak, aynı işlevi sağlayan bir fonksiyon (FC) ya da fonksiyon blok (FB) elde edilebilir. Şekil 9.103'te PID kontrol kuralını gerçeklemeye uygun bir blok diyagramı verilmiştir. Bu blok diyagramı, SP değerinin ani değişikliği nedeniyle oluşan türev vuruşu ve integral sarması olarak bilinen olayları önleyen bir uygulama şemasıdır. Bu şemaya göre, kontrol kuralı ifadesi

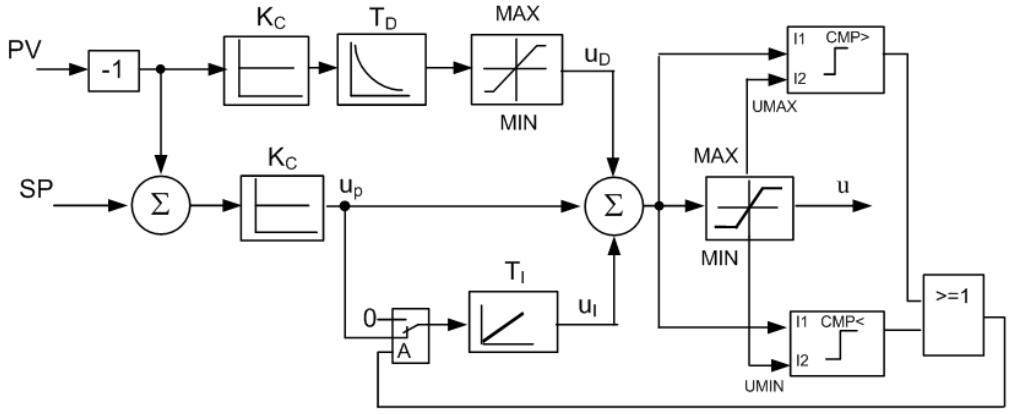
$$e(t) = SP(t) - PV(t)$$

$$u(t) = K_C\left(SP(t) - PV(t) + \frac{1}{T_I}\int [SP(t) - PV(t)]dt + T_D\frac{d[SP(t) - PV(t)]}{dt} \right)$$

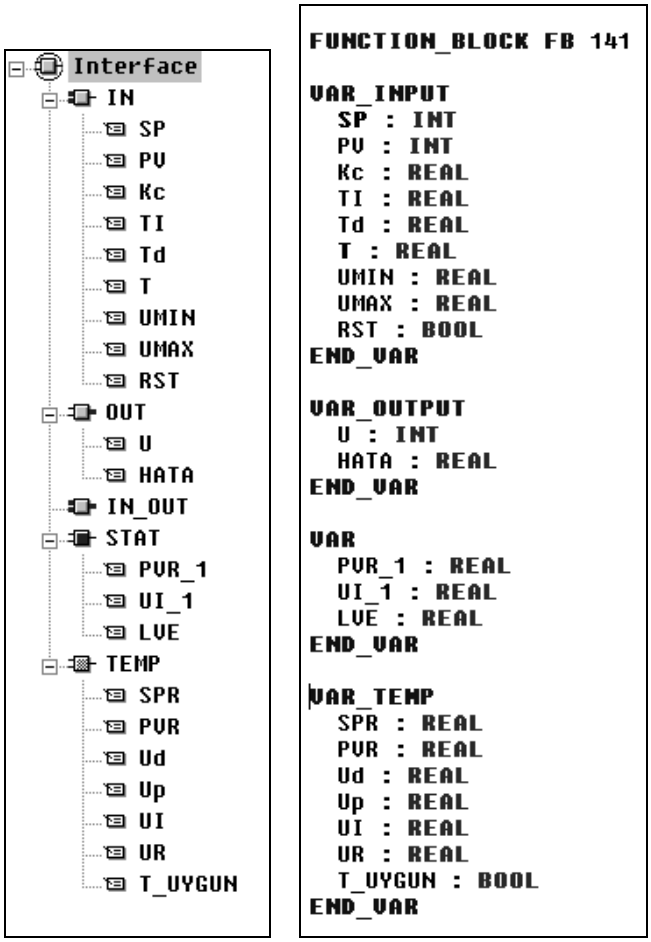
ve SP(t)=SP sabit değeri için

$$u(t) = K_C\left(SP(t) - PV(t) + \frac{1}{T_I}\int [SP(t) - PV(t)]dt - T_D\frac{dPV(t)]}{dt} \right) \qquad (9.5)$$

olur. Bu ifadenin gerçeklenmesine ilişkin bir fonksiyon bloğu (FB141) giriş-çıkış tanımlamaları Şekil 9.104'te ve program Şekil 9.105'te verilmiştir.



Şekil 9.103 Türev vuruşu ve integral sarması olaylarını önleyen bir PID uygulama şeması



Şekil 9.104 PID kontrol kuralına ilişkin FB141 fonksiyon bloğu giriş-çıkış tanımlamaları

```
FB141 : PID_AWR
Network 1 :
      L     #T
      L     0.000000e+000
      >R
      =     #T_UYGUN

      O     #RST
      ON    #T_UYGUN
      JCN   _001
      L     0.000000e+000
      T     #UI_1
      L     0
      T     #U
      BE

Network 2 :
_001: L     #SP
      ITD
      DTR
      T     #SPR

      L     #PV
      ITD
      DTR
      T     #PVR


      L     #SPR
      L     #PVR
      -R
      L     #Kc
      *R
      T     #Up

      L     #TI
      L     0.000000e+000
      >R
      JCN   _002
      L     #T
      L     #TI
      /R
      L     #Up
      *R
      L     #LVE
      *R
      L     #UI_1
      +R
      T     #UI
      JU    _003

Network 3 :
_002: L     0.000000e+000
      T     #UI_1
      T     #UI

Network 4 :

_003: L     #PVR_1
      L     #PVR
      -R
      L     #Kc
      *R
      L     #Td
      *R
      L     #T
      /R
      T     #Ud

Network 5 : Title:
      L     #Ud
      L     #UMAX
      >=R
      JCN   _004
      L     #UMAX
      T     #Ud
      JU    _005

_004: L     #Ud
      L     #UMIN
      <=R
      JCN   _005
      L     #UMIN
      T     #Ud

Network 6 :
_005: L     #Up
      L     #UI
      +R
      L     #Ud
      +R
      T     #UR

Network 7 :
      L     1.000000e+000
      T     #LVE
      L     #UI
      T     #UI_1
      L     #PVR
      T     #PVR_1

Network 8 : Title:
      L     #UR
      L     #UMAX
      >=R
      JCN   _006
      L     #UMAX
      T     #UR
      L     0.000000e+000
      T     #LVE
      JU    _007

_006: L     #UR
      L     #UMIN
      <=R
      JCN   _007
      L     #UMIN
      T     #UR
      L     0.000000e+000
      T     #LVE

Network 9 :
_007: L     #UR
      RND
      T     #U

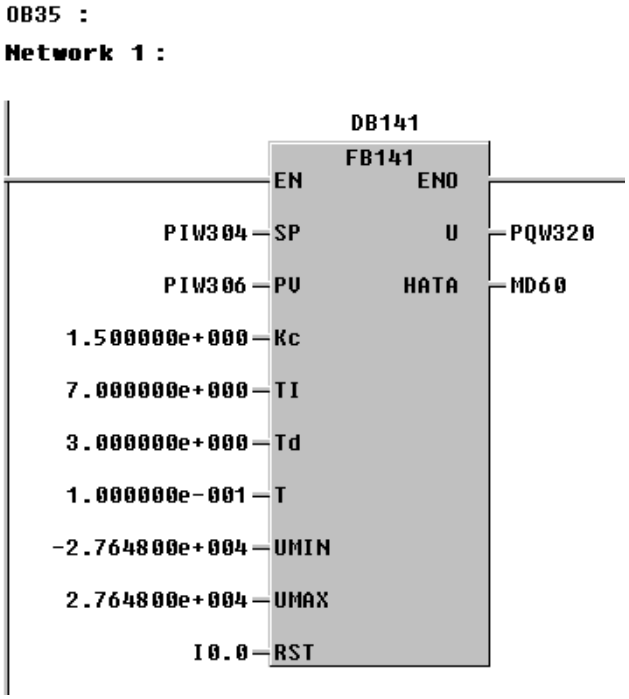
Network 10 :
      L     #SPR
      L     #PVR
      -R
      L     #SPR
      /R
      L     1.000000e+002
      *R
      T     #HATA

Network 11 :

      AN    OS
      AN    UO
      SAVE
```

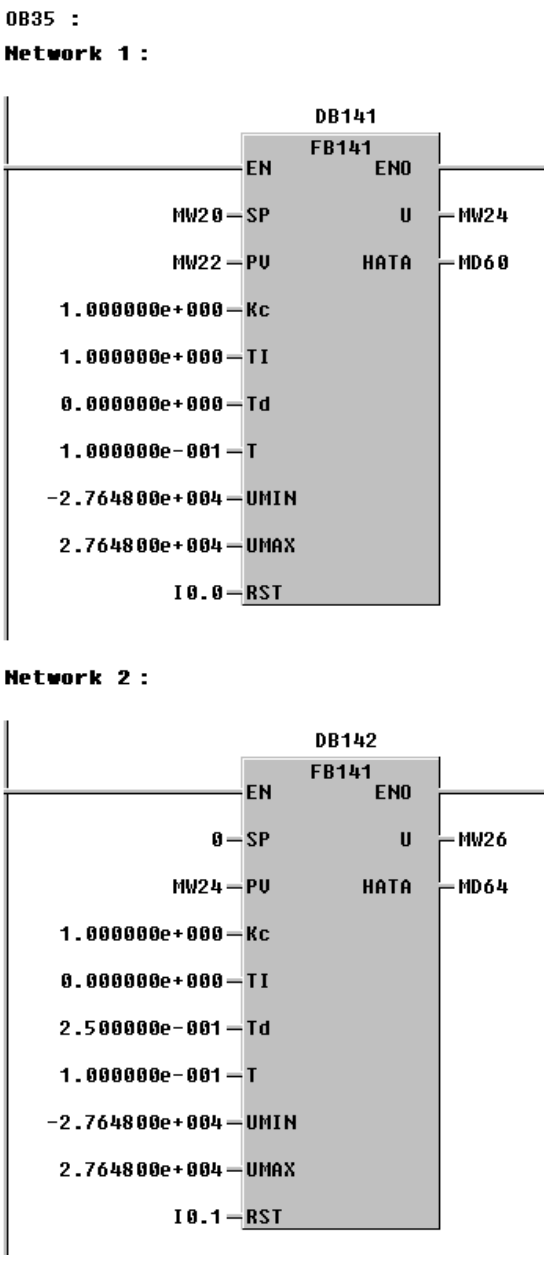
Şekil 9.105 PID kontrol kuralına ilişkin FB141 fonksiyon blok programı

Şekil 9.105'te verilen fonksiyon blok, Şekil 9.106'da gösterildiği gibi bir organizasyon bloğundan yürütülerek PID kontrol işlevi sağlanır. Bu programa göre, istenen değer (SP) analog PIW304 adresinden, kontrol edilen büyüklüğe ilişkin ölçülen değer analog PIW306 adresinden alınır ve hesaplanan kontrol işaretinin sayısal değeri PQW320 analog çıkış adresine, hatanın yüzde değeri MD60 adresine yazılır.

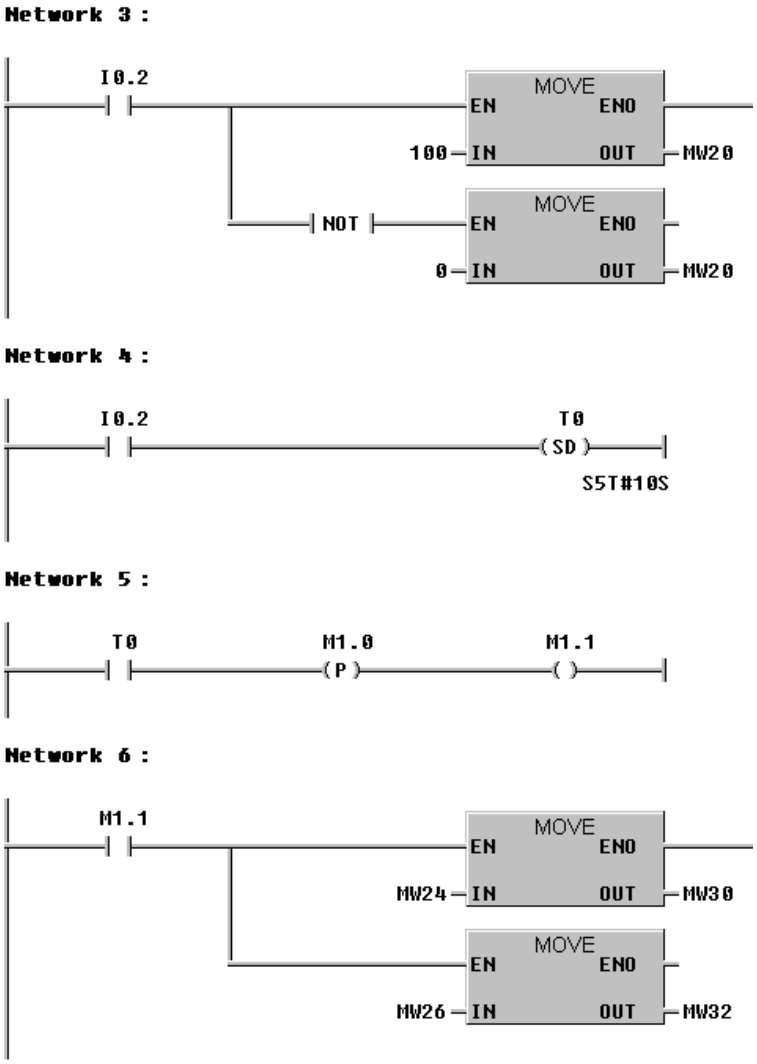


Şekil 9.106 FB141 PID fonksiyon bloğunun OB35 bloğundan işletilmesine ilişkin program

Şekil 9.107'de PID fonksiyon bloğunun işlevini doğrulamak amacıyla düzenlenen bir program örneği verilmiştir. Burada, birinci PID fonksiyon bloğu PI, ikinci PID bloğu PD işlevini sağlayacak şekilde düzenlenmiştir. Bu programa göre I0.2=1 yapıldığında birinci PID bloğu SP girişine genliği 100 tamsayı değerinde bir basamak giriş uygulanır ve 10s sonra PID bloklarının ürettiği kontrol işaretleri MW30 ve MW32 adreslerine yazılır. Burada elde edilen sayısal veriler, PI ve PD işlevlerinin doğruluğunun değerlendirilmesinde kullanılabilir.



Şekil 9.107 FB141 PID fonksiyon bloğunun işlevini doğrulamaya ilişkin program (1/2)



Şekil 9.107 FB141 PID fonksiyon bloğunun işlevini doğrulamaya ilişkin program (2/2)

Birinci PID bloğu; SP=100, PV=0, T_I=1, K_C=1 ve T_D=0 olduğundan kontrol kuralına ilişkin (9.5) ifadesi

$$u(t) = K_C\left(SP(t) - PV(t) + \frac{1}{T_I}\int[SP(t) - PV(t)]dt - T_D\frac{dPV(t)]}{dt}\right)$$

$$u(t) = 1\left(100 - 0 + \frac{1}{1}\int [100 - 0]dt - 0 \right) = 100 + 100t$$

olur. Buna göre, 10s sonra MW30 adresindeki sayı 1100 olmalıdır.

İkinci PID bloğu; SP=0, PV=u(t), T_I=0 (integral devre dışı), K_C=1 ve T_D=0.25s olduğundan kontrol kuralına ilişkin (9.5) ifadesi

$$u(t) = K_C\left(0 - (100 + 100t) - T_D \frac{d(100 + 100t)}{dt} \right) = -100 - 100t - 0.25 \times 100 = -125 - 100t$$

olur. Buna göre 10s sonra MW32 adresindeki sayı -1125 olmalıdır.

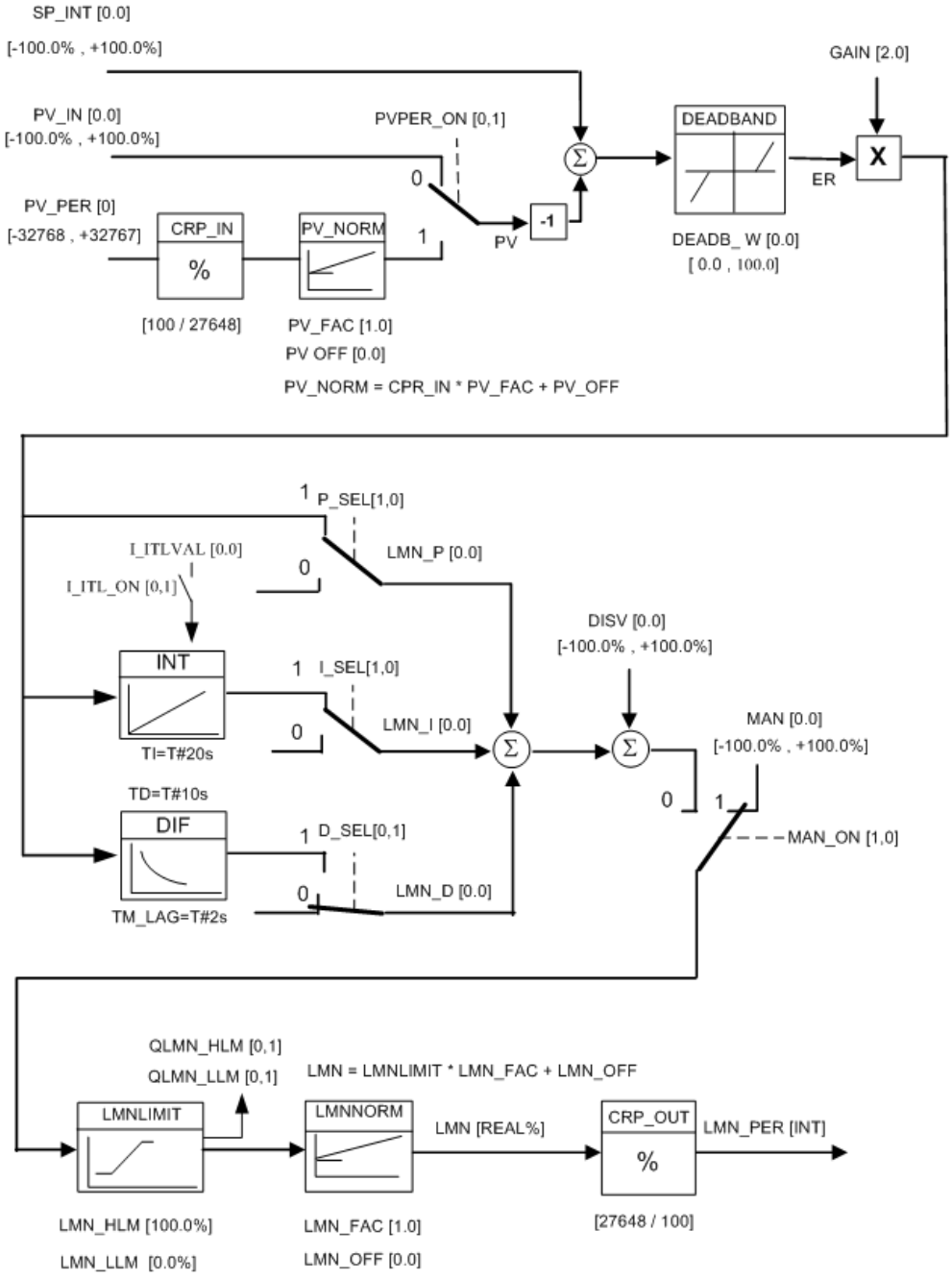
9.6.2.6 FB41 PID fonksiyon bloğu

PID kontrol işlevini sağlamak üzere, yaygın olarak "SIMATIC Manager" kütüphanesinde bulunan FB41 PID fonksiyon bloğu kullanılır. Bu fonksiyon bloğuna ilişkin gerçekleme şeması Şekil 9.108'de, OB35 organizasyon bloğundan işletilmesini sağlayan program Şekil 9.109'da verilmiştir. Bu fonksiyonun girişlerinden I0.0=1 yapıldığında PID fonksiyonuna ilk değerler yüklenir ve LMN_PER=0 olur. I0.1=1 yapıldığında MD104 (MAN) adresinin içeriği, doğrudan yüzde olarak LMN adresine ve tamsayı değere dönüştürülerek LMN_PER adresine yazılır. Örneğin, MAN=50.0 için LMN=50.0,

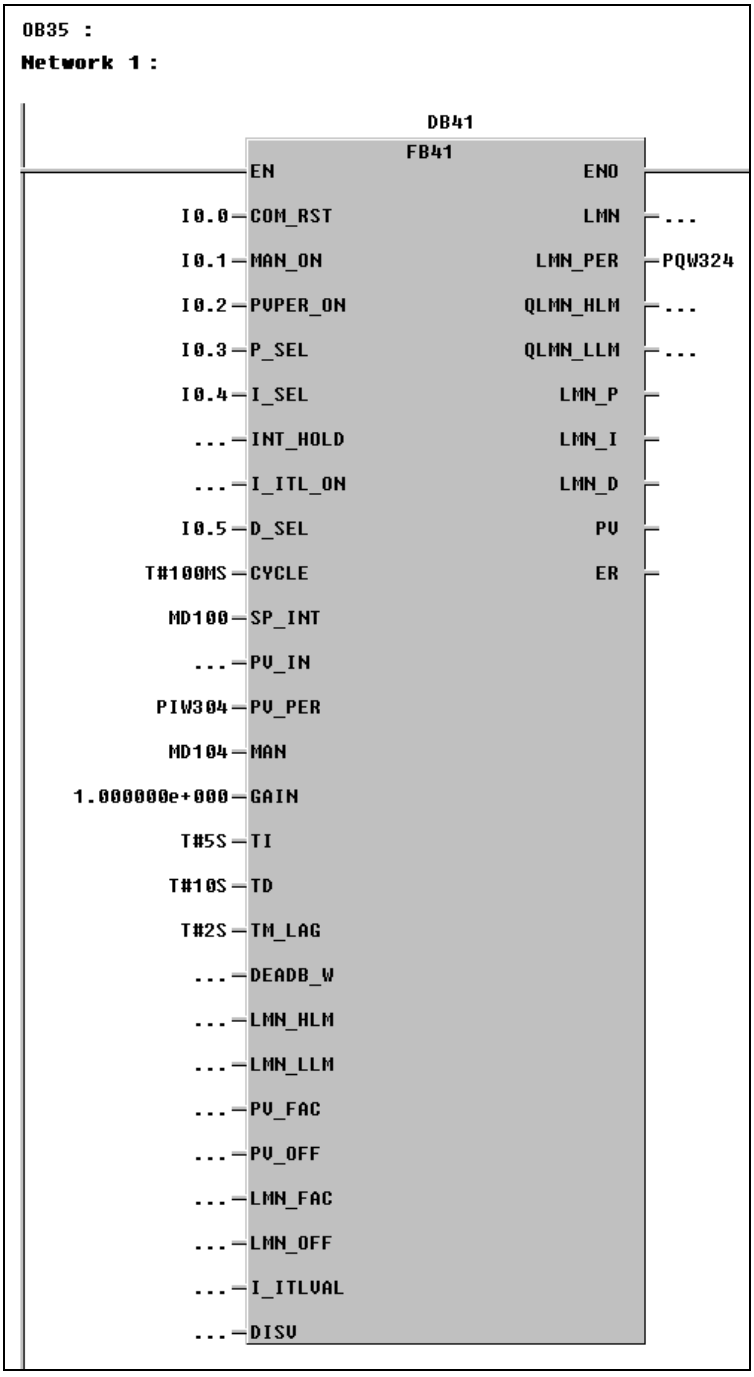
LMN_PER=27648 ×50/100)=13824 olur.

Girişlerden I0.2=1 yapıldığında kontrol edilen büyüklüğe ilişkin veri, işaretli tamsayı olarak PIW304 (PV_PER) adresinden okunur; I0.3=1 yapıldığında oransal, I0.4=1 yapıldığında integral ve I0.5=1 yapıldığında türev işlevi gerçeklenir. İstenen değer yüzde olarak MD100 (SP_INT) adresine yazılır. CYCLE girişine OB35 bloğunun kesmeli çalışma süresi, GAIN girişine oransal kazanç (K_C), TI girişine integral zaman sabiti, TD girişine türev zaman sabiti ve TM_LAG girişine türev zaman sabitinin yaklaşık 1/5 katı filtre zaman sabiti değerleri yazılır. Üretilen kontrol işareti yüzde olarak LMN çıkışına, işaretli tamsayı olarak LMN_PER çıkışına yazılır. Boş bırakılan diğer girişler ve çıkışlar FB41 fonksiyon bloğuna ilişkin DB41 veri bloğundan okunur ve yazılır. FB41 bloğuna ilişkin tüm giriş ve çıkışların özellikleri ve ilk değerleri Şekil 9.110'daki veri bloğunda gösterilmiştir. Boş bırakılan tüm girişlere bu bloktaki ilk değerler atanır.

FB41 PID fonksiyonun diğer özellikleri Şekil 9.108'de verilen gerçekleme şeması incelenerek görülebilir.



Şekil 9.108 FB41 fonksiyon bloğuna ilişkin gerçekleme şeması



Şekil 9.109 FB41 PID fonksiyon bloğunun OB35 bloğundan yürütülmesine ilişkin program

	Addres	Decl	Name	Type	Initial value	Actual value	Comment
1	0.0	in	COM_RST	BOOL	FALSE	FALSE	complete restart
2	0.1	in	MAN_ON	BOOL	TRUE	TRUE	manual value on
3	0.2	in	PVPER_ON	BOOL	FALSE	FALSE	process variable peripherie on
4	0.3	in	P_SEL	BOOL	TRUE	TRUE	proportional action on
5	0.4	in	I_SEL	BOOL	TRUE	TRUE	integral action on
6	0.5	in	INT_HOLD	BOOL	FALSE	FALSE	integral action hold
7	0.6	in	I_ITL_ON	BOOL	FALSE	FALSE	initialization of the integral action
8	0.7	in	D_SEL	BOOL	FALSE	FALSE	derivative action on
9	2.0	in	CYCLE	TIME	T#1S	T#1S	sample time
10	6.0	in	SP_INT	REAL	0.000000e+000	0.000000e+000	internal setpoint
11	10.0	in	PV_IN	REAL	0.000000e+000	0.000000e+000	process variable in
12	14.0	in	PV_PER	WORD	W#16#0	W#16#0	process variable peripherie
13	16.0	in	MAN	REAL	0.000000e+000	0.000000e+000	manual value
14	20.0	in	GAIN	REAL	2.000000e+000	2.000000e+000	proportional gain
15	24.0	in	TI	TIME	T#20S	T#20S	reset time
16	28.0	in	TD	TIME	T#10S	T#10S	derivative time
17	32.0	in	TM_LAG	TIME	T#2S	T#2S	time lag of the derivative action
18	36.0	in	DEADB_W	REAL	0.000000e+000	0.000000e+000	dead band width
19	40.0	in	LMN_HLM	REAL	1.000000e+002	1.000000e+002	manipulated value high limit
20	44.0	in	LMN_LLM	REAL	0.000000e+000	0.000000e+000	manipulated value low limit
21	48.0	in	PV_FAC	REAL	1.000000e+000	1.000000e+000	process variable factor
22	52.0	in	PV_OFF	REAL	0.000000e+000	0.000000e+000	process variable offset
23	56.0	in	LMN_FAC	REAL	1.000000e+000	1.000000e+000	manipulated value factor
24	60.0	in	LMN_OFF	REAL	0.000000e+000	0.000000e+000	manipulated value offset
25	64.0	in	I_ITLVAL	REAL	0.000000e+000	0.000000e+000	initialization value of the integral action
26	68.0	in	DISV	REAL	0.000000e+000	0.000000e+000	disturbance variable
27	72.0	out	LMN	REAL	0.000000e+000	0.000000e+000	manipulated value
28	76.0	out	LMN_PER	WORD	W#16#0	W#16#0	manipulated value peripherie
29	78.0	out	QLMN_HLM	BOOL	FALSE	FALSE	high limit of manipulated value reached
30	78.1	out	QLMN_LLM	BOOL	FALSE	FALSE	low limit of manipulated value reached
31	80.0	out	LMN_P	REAL	0.000000e+000	0.000000e+000	proportionality component
32	84.0	out	LMN_I	REAL	0.000000e+000	0.000000e+000	integral component
33	88.0	out	LMN_D	REAL	0.000000e+000	0.000000e+000	derivative component
34	92.0	out	PV	REAL	0.000000e+000	0.000000e+000	process variable
35	96.0	out	ER	REAL	0.000000e+000	0.000000e+000	error signal

Şekil 9.110 FB41 PID fonksiyon bloğuna ilişkin veri bloğu

9.6.3 SAYISAL BENZETİM UYGULAMALARI

Bölüm 8.4'te, S7-200 PLC ailesi ile gerçeklenen sayısal benzetim uygulamaları, benzer işlemler yapılarak S7-300/S7-400 PLC ailesi ile de gerçeklenebilir.

9.6.3.1 İkinci Mertebeden Sistemler

İkinci mertebeden sistemlere ilişkin z-tanım bölgesi transfer fonksiyonu

$$G(z) = \frac{Y(z)}{X(z)} = \frac{b_1 z + b_0}{z^2 + a_1 z + a_0} \tag{9.6}$$

biçiminde verilebilir. Bu ifadeden elde edilen

$$y(k) = -a_1 y(k-1) - a_0 y(k-2) + b_1 x(k-1) + b_0 x(k-2) \tag{9.7}$$

fark denklemi programlanarak ikinci mertebeden sistemler için bir benzetim fonksiyonu veya fonksiyon bloğu gerçeklenebilir. Şekil 9.111'de bir fonksiyon (FC5), Şekil 9.112'de bir fonksiyon blok (FB50) örneği verilmiştir.

Ölü zamanlı ikinci mertebeden sistemler için z-tanım bölgesi transfer fonksiyonu L ölü zaman, T örnekleme zamanı ve d=L/T olmak üzere

$$G(z) = \frac{Y(z)}{X(z)} = \frac{b_1 z + b_0}{z^d (z^2 + a_1 z + a_0)} \tag{9.8}$$

fark denklemi

$$y(k) = -a_1 y(k-1) - a_0 y(k-2) + b_1 x(k-d-1) + b_0 x(k-d-2) \tag{9.9}$$

biçiminde verilebilir (Bölüm 8.4.2). Bu fark denklemi programlanarak ölü zamanlı sistemler için bir benzetim fonksiyonu ya da fonksiyon bloğu elde edilebilir.

```
FUNCTION FC 5

VAR_INPUT
  X : INT
  a1 : REAL
  a0 : REAL
  b1 : REAL
  b0 : REAL
  XMIN : REAL
  XMAX : REAL
  YMIN : REAL
  YMAX : REAL
END_VAR

VAR_OUTPUT
  Y : INT
END_VAR

VAR_IN_OUT
  X_1 : REAL
  X_2 : REAL
  Y_1 : REAL
  Y_2 : REAL
END_VAR

VAR_TEMP
  X_TEMP : REAL
  Y_TEMP : REAL
END_VAR
```

```
FC5 :
Network 1 :
      L     #Y_2
      L     #a0
      NEGR
      *R
      T     #Y_TEMP
      L     #Y_1
      L     #a1
      NEGR
      *R
      L     #Y_TEMP
      +R
      T     #Y_TEMP

Network 2 :
      L     #X_2
      L     #b0
      *R
      T     #X_TEMP
      L     #X_1
      L     #b1
      *R
      L     #X_TEMP
      +R
      T     #X_TEMP
      L     #Y_TEMP
      +R
      T     #Y_TEMP

Network 3 :
      L     #Y_TEMP
      L     #YMAX
      >=R
      JCN   _001
      L     #YMAX
      T     #Y_TEMP
_001: NOP   0

Network 4 : Title:
      L     #Y_TEMP
      L     #YMIN
      <=R
      JCN   _002
      L     #YMIN
      T     #Y_TEMP
_002: NOP   0
```

```
Network 5 :
      L     #Y_1
      T     #Y_2
      L     #Y_TEMP
      T     #Y_1

Network 6 :
      L     #X_1
      T     #X_2
      L     #X
      ITD
      DTR
      T     #X_1

Network 7 :
      L     #X_1
      L     #XMAX
      >=R
      JCN   _003
      L     #XMAX
      T     #X_1
_003: NOP   0

Network 8 :
      L     #X_1
      L     #XMIN
      <=R
      JCN   _004
      L     #XMIN
      T     #X_1
_004: NOP   0

Network 9 :
      L     #Y_TEMP
      RND
      T     #Y

Network 10 :
      AN    OS
      AN    UO
      SAVE
```

Şekil 9.111 İkinci mertebeden sistemlere ilişkin bir fonksiyon programı

```
FUNCTION_BLOCK FB 50

VAR_INPUT
  X : INT
  a1 : REAL
  a0 : REAL
  b1 : REAL
  b0 : REAL
  XMIN : REAL
  XMAX : REAL
  YMIN : REAL
  YMAX : REAL
END_VAR

VAR_OUTPUT
  Y : INT
END_VAR

VAR
  X_1 : REAL
  X_2 : REAL
  Y_1 : REAL
  Y_2 : REAL
END_VAR

VAR_TEMP
  X_TEMP : REAL
  Y_TEMP : REAL
END_VAR
```

```
FB50 :
Network 1 :
      L     #Y_2
      L     #a0
      NEGR
      *R
      T     #Y_TEMP
      L     #Y_1
      L     #a1
      NEGR
      *R
      L     #Y_TEMP
      +R
      T     #Y_TEMP

Network 2 :
      L     #X_2
      L     #b0
      *R
      T     #X_TEMP
      L     #X_1
      L     #b1
      *R
      L     #X_TEMP
      +R
      T     #X_TEMP
      L     #Y_TEMP
      +R
      T     #Y_TEMP

Network 3 :
      L     #Y_TEMP
      L     #YMAX
      >=R
      JCN   _001
      L     #YMAX
      T     #Y_TEMP
_001: NOP   0

Network 4 :
      L     #Y_TEMP
      L     #YMIN
      <=R
      JCN   _002
      L     #YMIN
      T     #Y_TEMP
_002: NOP   0
```

```
Network 5 :
      L     #Y_1
      T     #Y_2
      L     #Y_TEMP
      T     #Y_1

Network 6 :
      L     #X_1
      T     #X_2
      L     #X
      ITD
      DTR
      T     #X_1

Network 7 :
      L     #X_1
      L     #XMAX
      >=R
      JCN   _003
      L     #XMAX
      T     #X_1
_003: NOP   0

Network 8 :
      L     #X_1
      L     #XMIN
      <=R
      JCN   _004
      L     #XMIN
      T     #X_1
_004: NOP   0

Network 9 :
      L     #Y_TEMP
      RND
      T     #Y

Network 10 :
      AN    OS
      AN    UO
      SAVE
```

Şekil 9.112 İkinci mertebeden sistemlere ilişkin bir fonksiyon blok programı

Örnek 9.9 Sönüm oranı $\zeta = 0.5$, doğal frekansı $\omega_n = 1\text{rad/s}$ ve kazancı K=1 olan ikinci mertebe sisteme ilişkin benzetim programı yazınız.

Bu sisteme ilişkin s-tanım bölgesi transfer fonksiyonu

$$G(s) = \frac{Y(s)}{X(s)} = \frac{K\omega_n^2}{s^2 + 2\zeta\omega_n s + \omega_n^2} = \frac{1}{s^2 + s + 1}$$

Örnekleme frekansı ω_s, doğal frekansın 20 katı seçilirse, örnekleme zamanı için

$$T \le \frac{2\pi}{\omega_s} = \frac{2\pi}{20\omega_n} = \frac{\pi}{10\omega_n} = 0.314\text{s}$$

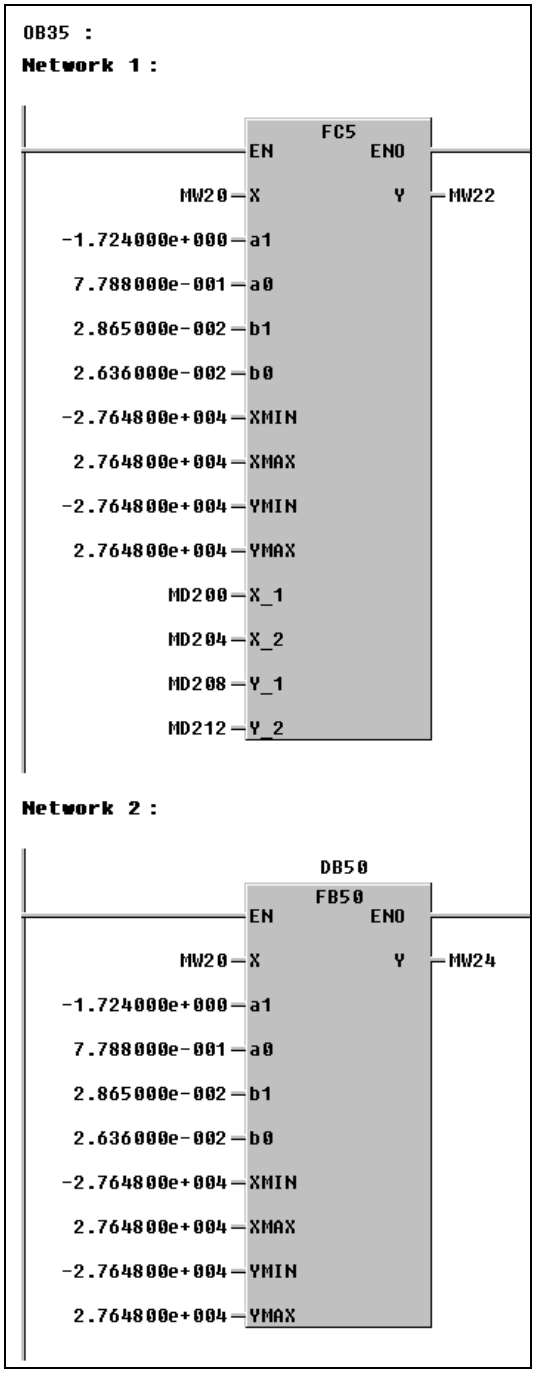
koşulu elde edilir.

Örnekleme zamanı T=0.25s için sıfırıncı mertebeden tutucu ile gerçeklenen ayrık sisteme ilişkin z-tanım bölgesi transfer fonksiyonu

$$G(z) = \frac{Y(z)}{X(z)} = \frac{b_1 z + b_0}{z^2 + a_1 z + a_0} = \frac{0.02865z + 0.02636}{z^2 - 1.724z + 0.7788}$$

elde edilir.

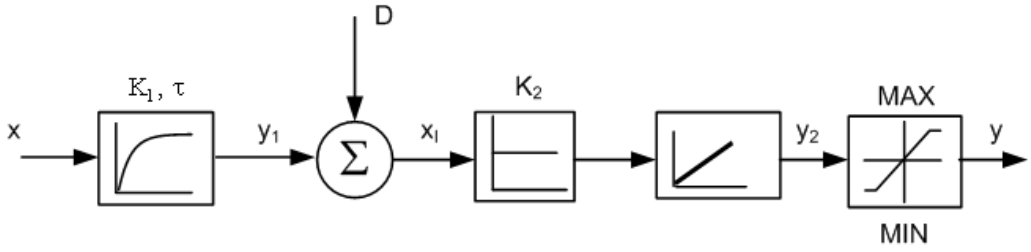
Bu sisteme ilişkin benzetim uygulaması, Şekil 9.111'de verilen FC5 fonksiyonu veya Şekil 9.112'de verilen FB50 fonksiyonu bloğu, kesmeli çalışma süresi 250ms değerine ayarlanan bir kesmeli çalışan organizasyon bloğundan işletilerek gerçeklenebilir. Şekil 9.113'te hem FC5 hem de FB50 bloklarının OB35 bloğundan yürütülmesini sağlayan bir program örneği gösterilmiştir.



Şekil 9.113 FC5 fonksiyonu ve FB50 fonksiyon bloğunun OB35 bloğundan yürütülmesine ilişkin program

9.6.3.2 Birinci Mertebeden İntegral Etkili Sistemler

Birinci mertebeden integral etkili sistemlere ilişkin bir blok diyagramı Şekil 9.114'te verilmiştir. Burada x sistem girişi, D bozucu işareti ve y sistem çıkışıdır.



Şekil 9.114 Birinci mertebeden integral etkili sisteme ilişkin blok diyagramı

Bu sistem için sıfırıncı mertebeden tutucu ile gerçeklenen ayrık zaman modeline ilişkin z-tanım bölgesi transfer fonksiyonları

$$G_1(z)=\frac{Y_1(z)}{X(z)}=\frac{K_1[1-\exp(-T/\tau)]}{z-\exp(-T/\tau)}=\frac{K_1[1-A)]}{z-A},\quad A=\exp(-T/\tau)$$

$$G_1(z)=\frac{Y_2(z)}{X_1(z)}=K_2\frac{Tz}{z-1},\qquad X_1(z)=Y_1(z)+D(z)$$

biçiminde yazılır ve bu ifadelerden

$$y_1(k)=Ay_1(k-1)+K_1(1-A)x(k-1)$$

$$y_2(k)=y_2(k-1)+K_2Tx_1(k)=y_2(k-1)+K_2T[y_1(k)+D(k)]$$

fark denklemleri elde edilir. Bu fark denklemleri programlanarak birinci mertebeden integral etkili sistemlere ilişkin bir benzetim fonksiyonu ya da fonksiyon bloğu elde edilir. Şekil 9.115'te bir fonksiyon bloğuna ilişkin program verilmiştir. Bu fonksiyon blok Şekil 9.116'da gösterildiği gibi programlanarak K_1=1.0, $\tau = 5.0s$ ve K_2=0.25 değerlerinde bir sistem için benzetim programı gerçeklenir. Bu uygulamada, örnekleme zamanı T=0.1s seçildiğinden OB35 organizasyon bloğunun kesmeli çalışma süresi 100ms değerine ayarlanır.

```
FUNCTION_BLOCK FB 120

VAR_INPUT
  X : INT
  D : INT
  K1 : REAL
  TO : REAL
  K2 : REAL
  T : REAL
  XMIN : REAL
  XMAX : REAL
  YMIN : REAL
  YMAX : REAL
  RST : BOOL
END_VAR

VAR_OUTPUT
  Y : INT
END_VAR

VAR
  X_1 : REAL
  Y1_1 : REAL
  Y2_1 : REAL
END_VAR

VAR_TEMP
  A : REAL
  DR : REAL
END_VAR
```

```
FB120 :
Network 1 :
      A     #RST
      JNB   _001
      L     0.000000e+000
      T     #Y1_1
      T     #Y2_1
      BE

Network 2 :
_001: L     #T
      L     #TO
      NEGR
      /R
      EXP
      T     #A

Network 3 :
      L     #D
      ITD
      DTR
      T     #DR

Network 4 :
      L     #DR
      L     #XMAX
      >=R
      JCN   _002
      L     #XMAX
      T     #DR
_002: NOP   0

Network 5 :
      L     #DR
      L     #XMIN
      <=R
      JCN   _003
      L     #XMIN
      T     #DR
_003: NOP   0
```

Şekil 9.115 Birinci mertebeden integral etkili sistemlere ilişkin bir fonksiyon blok programı(1/2)

```
Network 6 :
      L     #Y1_1
      L     #A
      *R
      T     #Y1_1

      L     1.000000e+000
      L     #A
      -R
      L     #K1
      *R
      L     #X_1
      *R
      L     #Y1_1
      +R
      T     #Y1_1

Network 7 :
      L     #Y1_1
      L     #DR
      +R
      L     #T
      *R
      L     #K2
      *R
      L     #Y2_1
      +R
      T     #Y2_1

Network 8 :
      L     #Y2_1
      L     #YMAX
      >=R
      JCN   _004
      L     #YMAX
      T     #Y2_1
_004: NOP   0

Network 9 :
      L     #Y2_1
      L     #YMIN
      <=R
      JCN   _005
      L     #YMIN
      T     #Y2_1
_005: NOP   0
```

```
Network 10 :
      L     #X
      ITD
      DTR
      T     #X_1

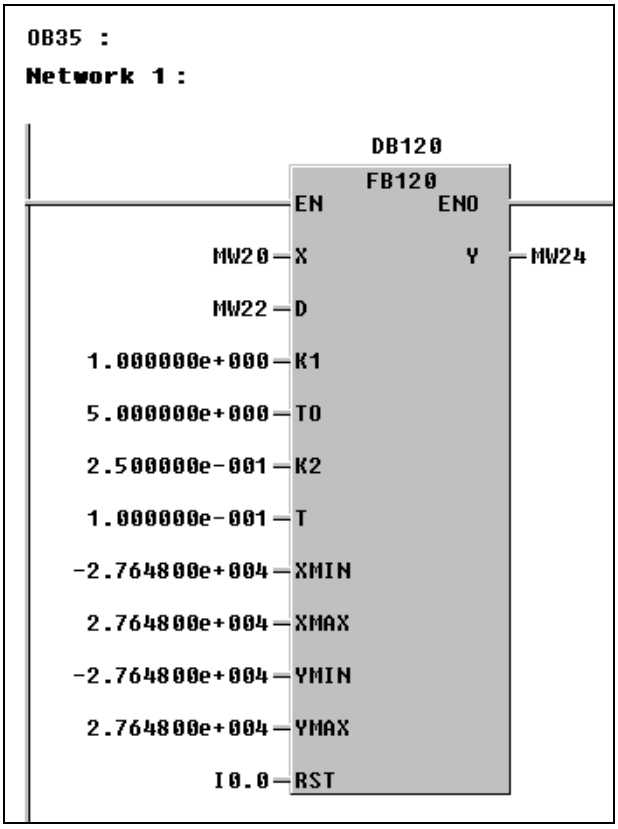
Network 11 :
      L     #X_1
      L     #XMAX
      >=R
      JCN   _006
      L     #XMAX
      T     #X_1
_006: NOP   0

Network 12 :
      L     #X_1
      L     #XMIN
      <=R
      JCN   _007
      L     #XMIN
      T     #X_1
_007: NOP   0

Network 13 :
      L     #Y2_1
      RND
      T     #Y

Network 14 :
      AN    OS
      AN    UO
      SAVE
```

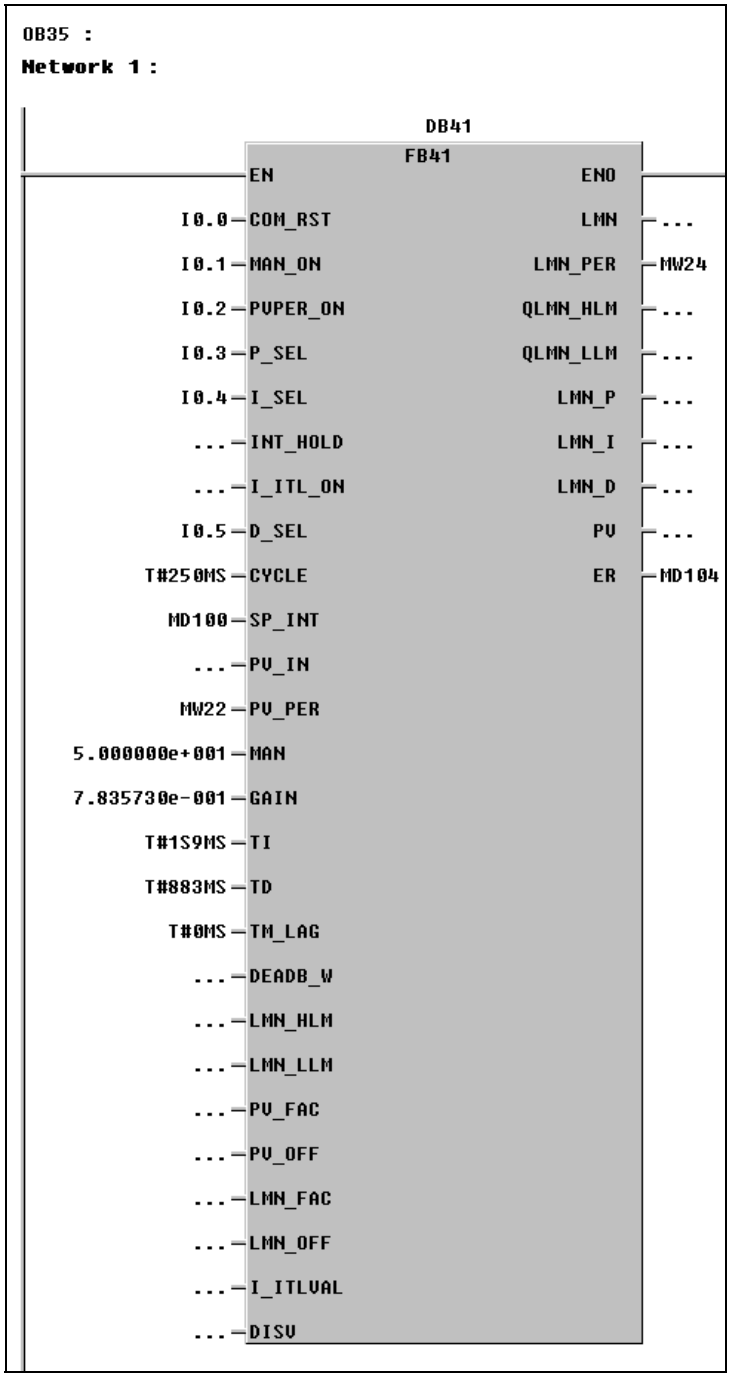
Şekil 9.115 Birinci mertebeden integral etkili sistemlere ilişkin bir fonksiyon blok programı(2/2)



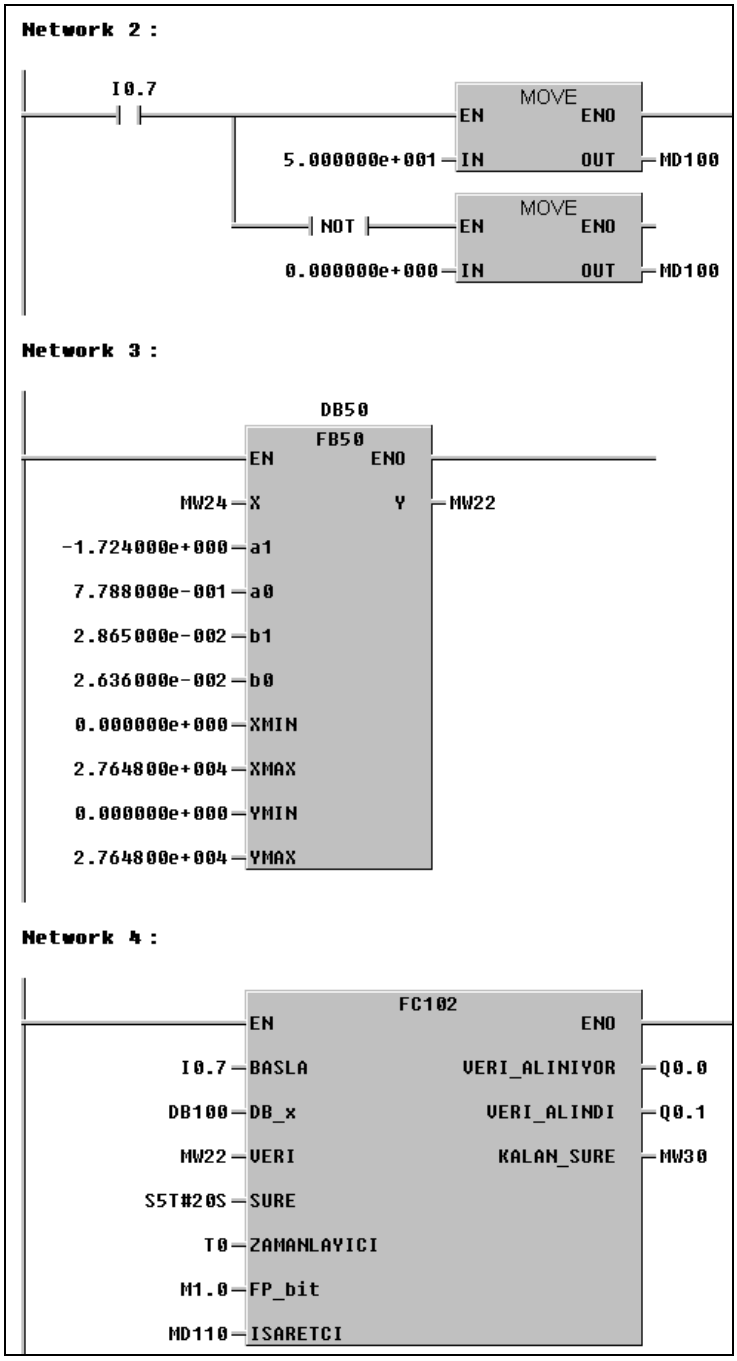
Şekil 9.116 FB120 fonksiyon bloğunun OB35 bloğundan yürütülmesine ilişkin program

9.6.3.3 Kontrol Sisteminin Tasarımında Benzetim Uygulamaları

Bir süreç için tasarlanan kontrol sisteminin özellikleri, PLC'de gerçeklenen gerçek zamanlı benzetim yazılımı kullanılarak değerlendirilebilir. Şekil 9.117'de Örnek 9.9'a ilişkin Şekil 9.112'de verilen ikinci mertebeden bir sistemin FB41 PID fonksiyon bloğu ile kontrol edilmesi durumunun incelendiği bir program örneği verilmiştir. Burada oransal kazanç (GAIN), integral zaman sabiti (TI) ve türev zaman sabiti (TD) değerleri değiştirilerek sistem yanıtı gözlemlenebilir. Bu programa göre, I0.7=1 yapıldığında kontrol edilen sisteme 50% değerinde basamak giriş uygulanır ve Şekil 9.80'de verilen FC102 veri toplama fonksiyonu kullanılarak sistem çıkış değeri kaydedilir. Bu programda PID işlevi; I0.2, I0.3, I0.4 ve I0.5 girişleri 1 yapılarak sağlanır. Eğer, I0.1=1 yapılırsa sistem açık çevrim çalışma durumuna geçer ve MAN girişine yazılan değer sisteme uygulanır.



Şekil 9.117 Kontrol sistemi için bir benzetim uygulamasına ilişkin program (1/2)



Şekil 9.117 Kontrol sistemi için bir benzetim uygulamasına ilişkin program (1/2)

KAYNAKÇA

Bateson, N. R. (1996): Control System Technology, Prentice Hall.

Batten, G. L. (1994): Programmable Controllers: Hardware, Software, and Applications, McGraw-Hill.

Berger, H. (2001): Automating with STEP 7 in LAD and FBD: SIMATIC S7-300/400 Programmable Controllers, Publicis MCD Corporate Publishing.

Berger, H. (2003): Automating with SIMATIC, Publicis Corporate Publishing.

Bolton, W. (1997): Programmable Logic Controllers: An Introduction, Butterworth-Heinemann.

Chesmond, C. J. (1982): Control System Technology, Edward Arnold.

Crispin, A. J. (1996): Programmable Logic Controllers and Their Engineering Applications, Books Britain.

Dunning, G. (1998): Introduction to Programmable Logic Controllers, Delmar.

Filer, R. , Leinonen, G. (1992): Programmable Controllers and Designing Sequential Logic, Saunders College Publishing.

Kurtulan, S. (2007): Endüstriyel Kumanda Sistemleri, NobelYayın, ANKARA.

Kurtulan, S. (1996): Programlanabilir Lojik Kontrolörler ve Uygulamaları, Bileşim Yayın, IST.

Levine, W. S. (1999): Control System Applications, CRC Pres.

Levine, W. S. (1999): Control System Fundamentals, CRC Press.

Lewis, R. W. (1998): Programming Industrial Control Systems Using IEC 1131-3, IEE.

Mano, M. M. (2003): Digital Design, Prentice Hall.

Olsson, G. , Piani, G. (1992): Computer Systems for Automation and Control, Prentice-Hall.

Parr, E. A. , Parr, A. (1993): Programmable Controllers: An Engineer's Guide, Butterworth-Heinemann.

Pessen, D. W. (1989): Industrial Automation: Circuit Design and Components, Wiley.

Petruzella, F. (1998): Programmable Logic Controllers, McGraw-Hill.

Phipps, C. A. (1998): Fundamentals of Electrical Control, The Fairmont Press.

Rexford, K. B. (1992): Electrical Control for Machines, Delmar Pub.

Ridley, J. E. (1997): Introduction to Programmable Logic Controllers: The Mitsubishi Fx", John Wiley & Sons.

Roginskii, V. N. (1963): The Synthesis of Relay Switching Circuits, D.V.Nostrand C. London.

Rohner, P. (1996): PLC: Automation With Programmable Logic Controllers, International Specialized Book Service.

Rosandich, R. G. (1997): Fundamentals of Programmable Logic Controllers, EC&M Books.

Sasao, T. (1999):Switching Theory for Logic Synthesis, Kluwer Academic Publisher.

Siemens, (2004): SIMATIC S7-200 Programmable Controller System Manuel.

Siemens, (2000), Statement List (STL) for S7-300 and S7-400 Programming Reference Manual.

Silva, C. W.(1989): Control Sensors and Actuators, Prentice-Hall.

Simpson, C. D. (1994): Programmable Logic Controllers, Prentice Hall, 1994.

Stenerson, J. (1998): Fundamentals of Programmable Logic Controllers, Sensors and Communications, Prentice Hall.

Warnock, I. G. (1989): Programmable Controllers, Prentice Hall.

Webb, J. W. , Reis, R. A. (2003): Programmable Logic Controllers: Principles and Applications, Prentice Hall.

EKLER

EK 1

SIMATIC S7-200 CPU 22x giriş-çıkış konfigürasyon seçenekleri

Module	5V ma	Digital Inputs	Digital Outputs	Analog Inputs	Analog Outputs
CPU 221	No expansion possible				
CPU 222					
Max Digital In/Out					
CPU	+340	8	6		
2 x EM 223 DI16/DO16 x DC24V	-320	32	32		
or	or				
2 x EM 223 DI16/DO16 x DC24V/Rly	-300				
Total =	>0	40	38		
Max Analog In					
CPU	+340	8	6		
2 x EM 235 AI4/AQ1	-60			8	2
Total =	>0	8	6	8	2
Max Analog Out					
CPU	+340	8	6		
2 x EM 232 AQ2	-40			0	4
Total =	>0	8	6	0	4
CPU 224					
Max Digital In/Rly Out					
CPU	+660	14	10		
4 x EM 223 DI16/DO16 x DC24V/Rly	-600	64	64		
2 x EM 221 DI8 x DC24V	-60	16			
Total =	=0	94	74		
Max Digital In/DC Out					
CPU	+660	14	10		
4 x EM 223 DI16/DO16 x DC24V	-640	64	64		
Total =	>0	78	74		
Digital In/Max Rly Out					
CPU	+660	14	10		
4 x EM 223 DI16/DO16 x DC24V/Rly	-600	64	64		
1x EM 222 DO8x Rly	-40		8		
Total =	>0	78	82		
CPU 226					
Max Digital In/Rly Out					
CPU	+1000	24	16		
6 x EM 223 DI16/DO16 x DC24V/Rly	-900	96	96		
1 x EM 223 DI8/DO8 x DC24V/Rly	-80	8	8		
Total =	>0	128	120		
Max Digital In/DC Out					
CPU	+1000	24	16		
6 x EM 223 DI16/DO16 x DC24V	-960	96	96		
1 x EM 221 DI8 x DC24V	-30	8			
Total =	>0	128	112		
CPU 224 or CPU 226					
Max Analog In					
CPU	>660	14 (24)	10 (16)		
7 x EM 235 AI4/AQ1	-210			28	7
Total =	>0	14 (24)	10 (16)	28	7
Max Analog Out					
CPU	>660	14 (24)	10 (16)		
7 x EM 232 AQ2	-140			0	14
Total =	>0	14 (24)	10 (16)	0	14

EK 2-1

S7-200 CPU 22x Bellek alanları ve erişim adresleri

Description		CPU 221	CPU 222	CPU 224	CPU 224XP	CPU 226
User program size with run mode edit without run mode edit		 4096 bytes 4096 bytes	 4096 bytes 4096 bytes	 8192 bytes 12288 bytes	 12288 bytes 16384 bytes	 16384 bytes 24576 bytes
User data size		2048 bytes	2048 bytes	8192 bytes	10240 bytes	10240 bytes
Process-image input register		I0.0 to I15.7	I0.0 to I15.7	I0.0 to I15.7	I0.0 to I15.7	I0.0 to I15.7
Process-image output register		Q0.0 to Q15.7	Q0.0 to Q15.7	Q0.0 to Q15.7	Q0.0 to Q15.7	Q0.0 to Q15.7
Analog inputs (read only)		AIW0 to AIW30	AIW0 to AIW30	AIW0 to AIW62	AIW0 to AIW62	AIW0 to AIW62
Analog outputs (write only)		AQW0 to AQW30	AQW0 to AQW30	AQW0 to AQW62	AQW0 to AQW62	AQW0 to AQW62
Variable memory (V)		VB0 to VB2047	VB0 to VB2047	VB0 to VB8191	VB0 to VB10239	VB0 to VB10239
Local memory (L)[1]		LB0 to LB63	LB0 to LB63	LB0 to LB63	LB0 to LB63	LB0 to LB63
Bit memory (M)		M0.0 to M31.7	M0.0 to M31.7	M0.0 to M31.7	M0.0 to M31.7	M0.0 to M31.7
Special Memory (SM)		SM0.0 to SM179.7	SM0.0 to SM299.7	SM0.0 to SM549.7	SM0.0 to SM549.7	SM0.0 to SM549.7
Read only		SM0.0 to SM29.7	SM0.0 to SM29.7	SM0.0 to SM29.7	SM0.0 to SM29.7	SM0.0 to SM29.7
Timers		256 (T0 to T255)	256 (T0 to T255)	256 (T0 to T255)	256 (T0 to T255)	256 (T0 to T255)
Retentive on-delay	1 ms	T0, T64	T0, T64	T0, T64	T0, T64	T0, T64
	10 ms	T1 to T4, and T65 to T68	T1 to T4, and T65 to T68	T1 to T4, and T65 to T68	T1 to T4, and T65 to T68	T1 to T4, and T65 to T68
	100 ms	T5 to T31, and T69 to T95	T5 to T31, and T69 to T95	T5 to T31, and T69 to T95	T5 to T31, and T69 to T95	T5 to T31, and T69 to T95
On/Off delay	1 ms	T32, T96	T32, T96	T32, T96	T32, T96	T32, T96
	10 ms	T33 to T36, and T97 to T100	T33 to T36, and T97 to T100	T33 to T36, and T97 to T100	T33 to T36, and T97 to T100	T33 to T36, and T97 to T100
	100 ms	T37 to T63, and T101 to T255	T37 to T63, and T101 to T255	T37 to T63, and T101 to T255	T37 to T63, and T101 to T255	T37 to T63, and T101 to T255
Counters		C0 to C255	C0 to C255	C0 to C255	C0 to C255	C0 to C255
High-speed counters		HC0 to HC5	HC0 to HC5	HC0 to HC5	HC0 to HC5	HC0 to HC5
Sequential control relays (S)		S0.0 to S31.7	S0.0 to S31.7	S0.0 to S31.7	S0.0 to S31.7	S0.0 to S31.7
Accumulator registers		AC0 to AC3	AC0 to AC3	AC0 to AC3	AC0 to AC3	AC0 to AC3
Jumps/Labels		0 to 255	0 to 255	0 to 255	0 to 255	0 to 255
Call/Subroutine		0 to 63	0 to 63	0 to 63	0 to 63	0 to 127
Interrupt routines		0 to 127	0 to 127	0 to 127	0 to 127	0 to 127
Positive/negative transitions		256	256	256	256	256
PID loops		0 to 7	0 to 7	0 to 7	0 to 7	0 to 7
Ports		Port 0	Port 0	Port 0	Port 0, Port 1	Port 0, Port 1

[1] LB60 to LB63 are reserved by STEP 7–Micro/WIN, version 3.0 or later.

EK 2-2

S7-200 işlemcilerinin karşılaştırılması

Feature	CPU 221	CPU 222	CPU 224	CPU 224XP	CPU 226
Physical size (mm)	90 x 80 x 62	90 x 80 x 62	120.5 x 80 x 62	140 x 80 x 62	190 x 80 x 62
Program memory: with run mode edit without run mode edit	 4096 bytes 4096 bytes	 4096 bytes 4096 bytes	 8192 bytes 12288 bytes	 12288 bytes 16384 bytes	 16384 bytes 24576 bytes
Data memory	2048 bytes	2048 bytes	8192 bytes	10240 bytes	10240 bytes
Memory backup	50 hours typical	50 hours typical	100 hours typical	100 hours typical	100 hours typical
Local on-board I/O Digital Analog	 6 In/4 Out –	 8 In/6 Out –	 14 In/10 Out –	 14 In/10 Out 2 In/1 Out	 24 In/16 Out –
Expansion modules	0 modules	2 modules[1]	7 modules[1]	7 modules[1]	7 modules[1]
High-speed counters Single phase Two phase	 4 at 30 kHz 2 at 20 kHz	 4 at 30 kHz 2 at 20 kHz	 6 at 30 kHz 4 at 20 kHz	 4 at 30 kHz 2 at 200 kHz 3 at 20 kHz 1 at 100 kHz	 6 at 30 kHz 4 at 20 kHz
Pulse outputs (DC)	2 at 20 kHz	2 at 20 kHz	2 at 20 kHz	2 at 100 kHz	2 at 20 kHz
Analog adjustments	1	1	2	2	2
Real-time clock	Cartridge	Cartridge	Built-in	Built-in	Built-in
Communications ports	1 RS-485	1 RS-485	1 RS-485	2 RS-485	2 RS-485
Floating-point math	Yes				
Digital I/O image size	256 (128 in, 128 out)				
Boolean execution speed	0.22 microseconds/instruction				

EK 3-1

SIMATIC S7-200 CPU 22x komut kümesi I

Boolean Instructions		
LD	Bit	Load
LDI	Bit	Load Immediate
LDN	Bit	Load Not
LDNI	Bit	Load Not Immediate
A	Bit	AND
AI	Bit	AND Immediate
AN	Bit	AND Not
ANI	Bit	AND Not Immediate
O	Bit	OR
OI	Bit	OR Immediate
ON	Bit	OR Not
ONI	Bit	OR Not Immediate
LDBx	IN1, IN2	Load result of Byte Compare IN1 (x:<, <=,=, >=, >, <>I) IN2
ABx	IN1, IN2	AND result of Byte Compare IN1 (x:<, <=,=, >=, >, <>) IN2
OBx	IN1, IN2	OR result of Byte Compare IN1 (x:<, <=,=, >=, >, <>) IN2
LDWx	IN1, IN2	Load result of Word Compare IN1 (x:<, <=,=, >=, >, <>) IN2
AWx	IN1, IN2	AND result of Word Compare IN1 (x:<, <=,=, >=, >, <>)I N2
OWx	IN1, IN2	OR result of Word Compare IN1 (x:<, <=,=, >=, >, <>) IN2
LDDx	IN1, IN2	Load result of DWord Compare IN1 (x:<, <=,=, >=, >, <>) IN2
ADx	IN1, IN2	AND result of DWord Compare IN1 (x:<, <=,=, >=, >, <>)IN2
ODx	IN1, IN2	OR result of DWord Compare IN1 (x:<, <=,=, >=, >, <>) IN2
LDRx	IN1, IN2	Load result of Real Compare IN1 (x:<, <=,=, >=, >, <>) IN2
ARx	IN1, IN2	AND result of Real Compare IN1 (x:<, <=,=, >=, >, <>) IN2
ORx	IN1, IN2	OR result of Real Compare IN1 (x:<, <=,=, >=, >, <>) IN2
NOT		Stack Negation
EU		Detection of Rising Edge
ED		Detection of Falling Edge
=	Bit	Assign Value
=I	Bit	Assign Value Immediate
S	Bit, N	Set bit Range
R	Bit, N	Reset bit Range
SI	Bit, N	Set bit Range Immediate
RI	Bit, N	Reset bit Range Immediate
LDSx	IN1, IN2	Load result of String Compare IN1 (x: =, <>) IN2
ASx	IN1, IN2	AND result of String Compare IN1 (x: =, <>) IN2
OSx	IN1, IN2	OR result of String Compare IN1 (x: =, <>) IN2
ALD		And Load
OLD		Or Load
LPS		Logic Push (stack control)
LRD		Logic Read (stack control)
LPP		Logic Pop (stack control)
LDS	N	Load Stack (stack control)
AENO		And ENO

Math, Increment, and Decrement Instructions		
+I	IN1, OUT	Add Integer, Double Integer or Real IN1+OUT=OUT
+D	IN1, OUT	
+R	IN1, OUT	
-I	IN1, OUT	Subtract Integer, Double Integer, or Real OUT-IN1=OUT
-D	IN1, OUT	
-R	IN1, OUT	
MUL	IN1, OUT	Multiply Integer (16*16->32)
*I	IN1, OUT	Multiply Integer, Double Integer, or Real IN1 * OUT = OUT
*D	IN1, OUT	
*R	IN1, IN2	
DIV	IN1, OUT	Divide Integer (16/16->32)
/I	IN1, OUT	Divide Integer, Double Integer, or Real OUT / IN1 = OUT
/D,	IN1, OUT	
/R	IN1, OUT	
SQRT	IN, OUT	Square Root
LN	IN, OUT	Natural Logarithm
EXP	IN, OUT	Natural Exponential
SIN	IN, OUT	Sine
COS	IN, OUT	Cosine
TAN	IN, OUT	Tangent
INCB	OUT	Increment Byte, Word or DWord
INCW	OUT	
INCD	OUT	
DECB	OUT	Decrement Byte, Word, or DWord
DECW	OUT	
DECD	OUT	
PID	TBL, LOOP	PID Loop
Timer and Counter Instructions		
TON	Txxx, PT	On-Delay Timer
TOF	Txxx, PT	Off-Delay Timer
TONR	Txxx, PT	Retentive On-Delay Timer
BITIM	OUT	Beginning Interval Timer
CITIM	IN, OUT	Calculate Interval Timer
CTU	Cxxx, PV	Count Up
CTD	Cxxx, PV	Count Down
CTUD	Cxxx, PV	Count Up/Down
Real Time Clock Instructions		
TODR	T	Read Time of Day clock
TODW	T	Write Time of Day clock
TODRX	T	Read Real Time Clock Extended
TODWX	T	Set Real Time Clock Extended
Program Control Instructions		
END		Conditional End of Program
STOP		Transition to STOP Mode
WDR		WatchDog Reset (300 ms)
JMP	N	Jump to defined Label
LBL	N	Define a Label to Jump to
CALL	N [N1,...]	Call a Subroutine [N1, ... up to 16 optional parameters]
CRET		Conditional Return from SBR
FOR	INDX,INIT,FINAL	For/Next Loop
NEXT		
LSCR	N	Load, Transition, Conditional End, and End Sequence Control Relay
SCRT	N	
CSCRE		
SCRE		
DLED	IN	Diagnostic LED

EK 3-2

SIMATIC S7-200 CPU 22x komut kümesi II

Move, Shift, and Rotate Instructions		
MOVB MOVW MOVD MOVR	IN, OUT IN, OUT IN, OUT IN, OUT	Move Byte, Word, DWord, Real
BIR BIW	IN, OUT IN, OUT	Move Byte Immediate Read Move Byte Immediate Write
BMB BMW BMD	IN, OUT, N IN, OUT, N IN, OUT, N	Block Move Byte, Word, DWord
SWAP	IN	Swap Bytes
SHRB	DATA, S_BIT, N	Shift Register Bit
SRB SRW SRD	OUT, N OUT, N OUT, N	Shift Right Byte, Word, DWord
SLB SLW SLD	OUT, N OUT, N OUT, N	Shift Left Byte, Word, DWord
RRB RRW RRD	OUT, N OUT, N OUT, N	Rotate Right Byte, Word, DWord
RLB RLW RLD	OUT, N OUT, N OUT, N	Rotate Left Byte, Word, DWord
Logical Instructions		
ANDB ANDW ANDD	IN1, OUT IN1, OUT IN1, OUT	Logical AND of Byte, Word, and DWord
ORB ORW ORD	IN1, OUT IN1, OUT IN1, OUT	Logical OR of Byte, Word, and DWord
XORB XORW XORD	IN1, OUT IN1, OUT IN1, OUT	Logical XOR of Byte, Word, and DWord
INVB INVW INVD	OUT OUT OUT	Invert Byte, Word and DWord (1's complement)
String Instructions		
SLEN	IN, OUT	String Length
SCAT	IN, OUT	Concatenate String
SCPY	IN, OUT	Copy String
SSCPY	IN, INDX, N, OUT	Copy Substring from String
CFND	IN1, IN2, OUT	Find First Character within String
SFND	IN1, IN2, OUT	Find String within String

Table, Find, and Conversion Instructions		
ATT	DATA, TBL	Add data to table
LIFO FIFO	TBL, DATA TBL, DATA	Get data from table
FND= FND<> FND< FND>	TBL, PTN, INDX TBL, PTN, INDX TBL, PTN, INDX TBL, PTN, INDX	Find data value in table that matches comparison
FILL	IN, OUT, N	Fill memory space with pattern
BCDI IBCD	OUT OUT	Convert BCD to Integer Convert Integer to BCD
BTI ITB ITD DTI	IN, OUT IN, OUT IN, OUT IN, OUT	Convert Byte to Integer Convert Integer to Byte Convert Integer to Double Integer Convert Double Integer to Integer
DTR TRUNC ROUND	IN, OUT IN, OUT IN, OUT	Convert DWord to Real Convert Real to Double Integer Convert Real to Double Integer
ATH HTA ITA DTA RTA	IN, OUT, LEN IN, OUT, LEN IN, OUT, FMT IN, OUT, FM IN, OUT, FM	Convert ASCII to Hex Convert Hex to ASCII Convert Integer to ASCII Convert Double Integer to ASCII Convert Real to ASCII
DECO ENCO	IN, OUT IN, OUT	Decode Encode
SEG	IN, OUT	Generate 7-segment pattern
ITS DTS RTS	IN, FMT, OUT IN, FMT, OUT IN, FMT, OUT	Convert Integer to String Convert Double Integer to String Convert Real to String
STI STD STR	STR, INDX, OUT STR, INDX, OUT STR, INDX, OUT	Convert Substring to Integer Convert Substring to Double Integer Convert Substring to Real
Interrupt Instructions		
CRETI		Conditional Return from Interrupt
ENI DISI		Enable Interrupts Disable Interrupts
ATCH DTCH	INT, EVNT EVNT	Attach Interrupt routine to event Detach event
Communications Instructions		
XMT RCV	TBL, PORT TBL, PORT	Freeport transmission Freeport receive message
NETR NETW	TBL, PORT TBL, PORT	Network Read Network Write
GPA SPA	ADDR, PORT ADDR, PORT	Get Port Address Set Port Address
High-Speed Instructions		
HDEF	HSC, MODE	Define High-Speed Counter mode
HSC	N	Activate High-Speed Counter
PLS	Q	Pulse Output

EK 4-1

S7-200 CPU 22x işlemcilerinde komut yürütme süreleri

Instruction			µs
=	Using:	I SM, T, C, V, S, Q, M L	0.24 1.3 10.5
+D			29
-D			29
*D			47
/D			250
+I			25
-I			25
*I			37
/I			64
=I	Using:	Local outputs Expansion outputs	16 24
+R			71 Typ 99 Max
-R			72 Typ 100 Max
*R			56 Typ 166 Max
/R			177 Typ 230 Max
A	Using:	I SM, T, C, V, S, Q, M L	0.22 0.72 6.1
AB <=, =, >=, >, <, <>			18
AD <=, =, >=, >, <, <>			27
AENO			0.4
AI	Using:	Local inputs Expansion inputs	15 21
ALD			0.22
AN	Using:	I SM, T, C, V, S, Q, M L	0.22 0.72 6.1
ANDB			19
ANDD			30
ANDW			25
ANI	Using:	Local inputs Expansion inputs	15 21
AR <=, =, >=, >, <, <>			29
AS=, <>	Time = Base + (LM * N) Base Length multiplier (LM) N is the number of characters compared		 33 6.3
ATCH			12
ATH	Time = Base + (length * LM) Base (constant length) Base (variable length) Length multiplier (LM)		 23 31 10.2
ATT			36
AW <=, =, >=, >, <, <>			23
BCDI			35

Instruction			µs
BITIM			16
BIR	Using:	Local inputs Expansion inputs	23 30
BIW	Using:	Local outputs Expansion outputs	24 32
BMB	Time = Base + (length * LM) Base (constant length) Base (variable length) Length multiplier (LM)		 10 28 5.7
BMD	Time= Base + (length * LM) Base (constant length) Base (variable length) Length multiplier (LM)		 11 29 10.6
BMW	Time= Base + (length * LM) Base (constant length) Base (variable length) Length multiplier (LM)		 10 28 8.6
BTI			16
CALL	Using no parameters: Using parameters: Time = Base + Σ (operand time) Base Operand time bit (input, output) byte (input, output) word (input, output) Dword (input, output) Note: processing of output operands occurs during the return from the subroutine		9 14 10, 11 8, 7 10, 9 12, 10
CEVNT			24
CFND	Maximum Time = Base + N1 * ((LM1 * N2) + LM2) Base Length multiplier 1 (LM1) Length multiplier 2 (LM2) N1 is length of the source string N2 is the length of the character set string		 35 8.6 9.5
CITIM			23
COS			900 Typ 1070 Max
CRET	Power flow present Power flow not present		16 0.8
CRETI	Power flow not present		0.2
CSCRE			3.1
CTD	On transition of count input Otherwise		27 19
CTU	On transition of count input Otherwise		31 19
CTUD	On transition of count input Otherwise		37 24
DECB			16
DECD			22
DECO			19
DECW			20
DISI			9
DIV			67

EK 4-2

S7-200 CPU 22x işlemcilerinde komut yürütme süreleri

Instruction		µs
DLED		14
DTA		302
DTI		21
DTCH		12
DTR		35 Typ 40 Max
DTS		305
ED		8
ENCO		24 Max
END	Power flow not present	0.2
ENI		11
EU		8
EXP		720 Typ 860 Max
FIFO	Time = Base + (length * LM) Base Length multiplier (LM)	 30 7
FILL	Time= Base + (length * LM) Base (constant length) Base (variable length) Length multiplier (LM)	 15 29 3.2
FND <, =, >, <>	Time = Base+(length * LM) Base Length multiplier (LM)	 39 6.5
FOR	Time = Base+(Number of loops * LM) Base Loop multiplier (LM)	 35 28
GPA		16
HDEF		18
HSC		30
HTA	Time= Base+ (length * LM) Base (constant length) Base (variable length) Length multiplier (LM)	 20 28 5.2
IBCD		52
INCB		15
INCD		22
INCW		20
INT	Typical with 1 interrupt	24
INVB		16
INVD		22
INVW		20
ITA		136
ITB		17
ITD		20
ITS		139
JMP		1.8
LBL		0.22

Instruction			µs
LD	Using:	I SM, T, C, V, S, Q, M L	0.22 0.8 6
LDB <=, =, >=, >, <, <>			18
LDD <=, =, >=, >, <, <>			27
LDI	Using:	Local inputs Expansion inputs	15 21
LDN	Using:	I SM, T, C, V, S, Q, M L	0.3 0.9 6.1
LDNI	Using:	Local inputs Expansion inputs	15 21
LDR<=, =, >=, >, <, <>			29
LDS			0.22
LDS=, <>	Time = Base + (LM * N) Base Length multiplier (LM) N is the number of characters compared		 33 6.3
LDW <=, =, >=, >, <, <>			24
LIFO			37
LN			680 Typ 820 Max
LPP			0.22
LPS			0.24
LRD			0.22
LSCR			7.3
MOVB			15
MOVD			20
MOVR			20
MOVW			18
MUL			37
NETR			99
NETW	Time = Base + (LM * N) Base Length multiplier (LM) N is the number of bytes to send		 95 4
NEXT			0
NOP			0.22
NOT			0.22
O	Using:	I SM, T, C, V, S, Q, M L	0.22 0.72 6.4
OB <=, =, >=, >, <, <>			18
OD <=, =, >=, >, <, <>			26
OI	Using:	Local inputs Expansion inputs	15 21
OLD			0.22
ON	Using:	I SM, T, C, V, S, Q, M L	0.22 0.72 6.4

EK 4-3

S7-200 CPU 22x işlemcilerinde komut yürütme süreleri

Instruction		µs
ONI	Using: Local inputs Expansion inputs	15 21
OR<=, =, >=, >, <, <>		29
ORB		19
ORD		29
ORW		25
OS=, < >	Time = Base + (LM * N) Base Length multiplier (LM) N is the number of characters compared	 33 6.3
OW <=, =, >=, >, <, <>		24
PID	Typical Manual-to-auto transition Coefficient recalculation Auto-tune	400 800 Max 770 Max 650 Max
PLS:	Using: PWM PTO single segment PTO multiple segment	31 36 50
R	Length=1 and specified as a constant Base for Counters (C) Base for Timers (T) Base for all others Otherwise: Time = Base + (length * LM) Base for Counters Base for Timers (T) Base for all others Length multiplier (LM) for operand C Length multiplier (LM) for operand T Length multiplier (LM) for all others If length stored as variable, add to base	 9.3 16 2.9 8.6 8.3 14 5.1 9.9 0.5 17
RCV		51
RET		16
RI	Time = Base + (length * LM) Base Length multiplier (LM) using local outputs Length multiplier (LM) using expansion outputs If length stored as a variable, add to Base	 8.9 13 21 17
RLB	Time = Base + (LM * N) Base Length multiplier (LM) N is the shift count	 23 0.2
RLD	Time = Base + (LM * N) Base Length multiplier (LM) N is the shift count	 28 1.4
RLW	Time = Base + (LM * N) Base Length multiplier (LM) N is the shift count	 27 0.9
ROUND		56 Typ 110 Max
RRB	Time = Base + (LM * N) Base Length multiplier (LM) N is the shift count	 22 0.5
RRD	Time = Base + (LM * N) Base Length multiplier (LM)) N is the shift count	 28 1.7

Instruction		µs
RRW	Time = Base + (LM * N) Base Length multiplier (LM) N is the shift count	 26 1.2
RTA	Time = Base + (LM * N) Base (for first digit in result) Length multiplier (LM) N is the number of additional digits in result	 149 96
RTS	Time = Base + (LM * N) Base (for first digit in result x) Length multiplier (LM) N is the number of additional digits in result	 154 96
S	For length = 1 and specified as a constant Otherwise: Time = Base + (length * LM) Base Length multiplier (LM) If length is stored as a variable, add to Base	2.9 14 0.5 17
SCAT	Time = Base + (LM * N) Base Length multiplier (LM) N is the number of appended characters	 30 5.3
SCPY	Time = Base + (LM * N) Base Length multiplier (LM) N is the number of copied characters	 27 4.6
SCRE		0.24
SCRT		10
SEG		15
SFND	Maximum Time = Base +((N1–N2) * LM2) +(N2 * LM1) Base Length multiplier 1 (LM1) Length multiplier 2 (LM2) N1 is the length of the source string N2 is the length of the search string	 39 7.6 6.8
SHRB	Time = Base + (length * LM1) + ((length /8) * LM2) Base (constant length) Base (variable length) Length multiplier 1 (LM1) Length multiplier 2 (LM2)	 48 52 1.0 1.5
SI	Time = Base + (length * LM) Base LM using local output LM using expansion output If length is stored as a variable, add to Base	 8.9 13 21 17
SIN		900 Typ 1070 Max
SLB	Time = Base + (LM * N) Base Length multiplier (LM) N is the shift count	 23 0.2
SLD	Time = Base + (LM * N) Base Length multiplier (LM) N is the shift count	 29 1.1

EK 4-4

S7-200 CPU 22x işlemcilerinde komut yürütme süreleri

Instruction		µs
SLEN		21
SLW	Time = Base + (LM + N) Base Length multiplier (LM)) N is the shift count	 27 0.6
SPA		371
SQRT		460 Typ 550 Max
SRB	Time = Base + (LM + N) Base Length multiplier (LM)) N is the shift count	 22 0.6
SRD	Time = Base + (LM + N) Base Length multiplier (LM)) N is the shift count	 28 1.5
SRW	Time = Base + (LM + N) Base Length multiplier (LM)) N is the shift count	 27 1
SSCPY	Time = Base + (LM * N) Base Length multiplier (LM) N is the number of copied characters	 42 5.3
STD	Time = Base + (LM * N) Base (for 1st source character) Length multiplier (LM) N is the number of additional source characters	 69 27
STI	Time = Base + (LM * N) Base (for 1st source character) Length multiplier (LM) N is the number of additional source characters	 59 27

Instruction		µs
STOP	Power flow not present	4
STR	Time = Base + (LM * N) Base (for 1st source character) Length multiplier (LM) N is the number of additional source characters	 51 81
SWAP		17
TAN		1080 Typ 1300 Max
TODR		331
TODRX	Daylight Saving Time correction	391 Typ 783 Typ
TODW		436
TODWX		554
TOF		36
TON		33
TONR		32
TRUNC		53 Typ 106 Max
WDR		7
XMT		42
XORB		19
XORD		29
XORW		25

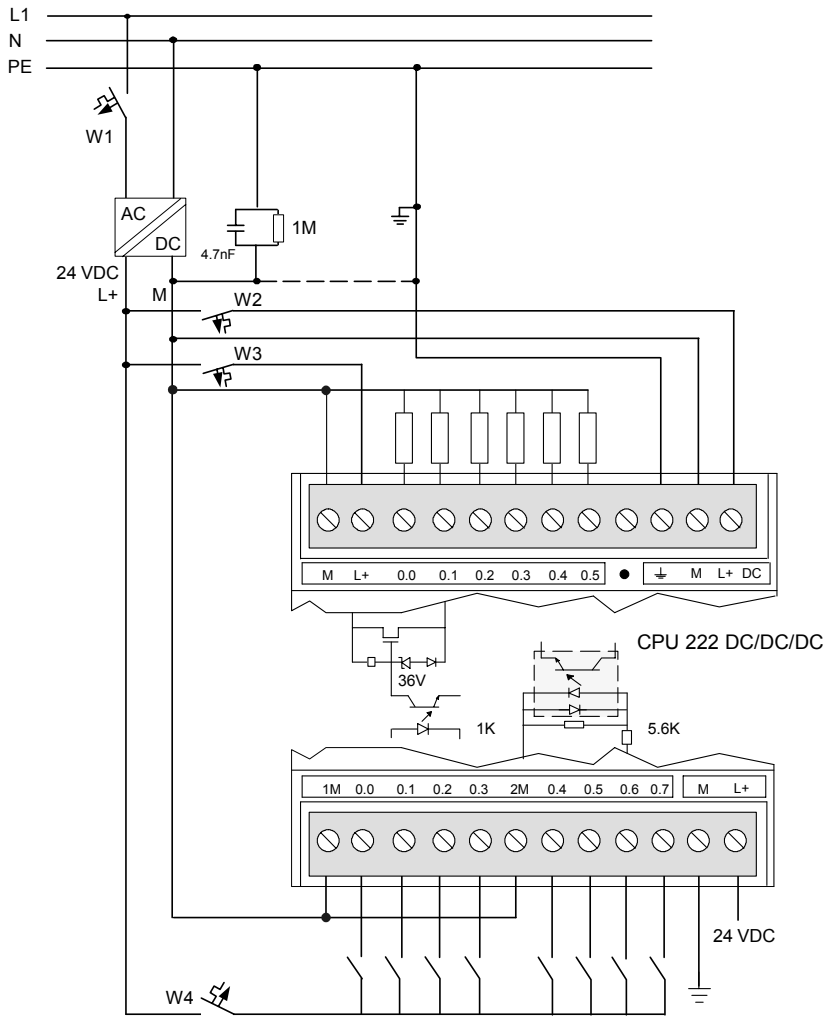
EK 5

S7-200 CPU 22x işlemcilerinde bazı komutların program belleğinde kapladığı alan

Komut	Alan	Komut	Alan
LD I0.1	2 Byte	AW= AC0, AC1	2 byte
A I0.1	2 byte	LDR<= VD200, VD204	7 byte
O I0.1	2 byte	LDR<= AC0, VD204	5 byte
LDN I0.1	2 byte	LDR<= AC0, AC1	3 byte
AN I0.1	2 byte	CTU C0, VW0	6 byte
ON I0.1	2 byte	CTU C0, AC0	4 byte
LDI I0.0	3 byte	CTD C0, VW0	7 byte
AI I0.0	3 byte	CTUD C0, VW0	6 byte
OI I0.0	3 byte	TON T33, VW100	6 byte
LDNI I0.0	3 byte	TONR T0, VW102	6 byte
ANI I0.0	3 byte	TOF T37, VW104	7 byte
ONI I0.0	3 byte	TODR VB100	4 byte
NOT	1 byte	+R VD100, VD104	7 byte
EU	1 byte	-R VD100, VD104	7 byte
ED	1 byte	*R VD100, VD104	7 byte
ALD	1 byte	+I VW0, VW2	6 byte
OLD	1 byte	+I VW0, AC0	4 byte
LPS	1 byte	+I AC0, AC1	2 byte
LDS 1	2 byte	+D VD0, VD4	6 byte
LRD	1 byte	MOVB VB0, VB1	6 byte
LPP	1 byte	MOVW VW0, VW2	6 byte
= I0.1	2 byte	MOVD VD0, VD4	6 byte
=I Q0.0	3 byte	MOVR VD10, VD14	7 byte
S Q0.2, VB0	8 byte	MOVB VB0, AC0	4 byte
R Q0.2, VB0	8 byte	MOVW VW0, AC1	4 byte
SI Q0.2, VB0	8 byte	MOVD VD0, AC2	4 byte
RI Q0.2, VB0	8 byte	MOVR VD10, AC3	5 byte
AENO	1 byte	MOVD AC0, AC2	2 byte
NOP 1	2 byte	MOVR AC1, AC3	3 byte
LDW>= VW100, VW102	6 byte	SQRT VD0, VD4	7 byte
AW>= VW108, VW110	6 byte	SQRT AC0, AC0	3 byte
AW= VW120, VW122	6 byte	SIN AC0, AC0	3 byte
AW= AC0, VW122	4 byte	PID VB0, 0	5 byte

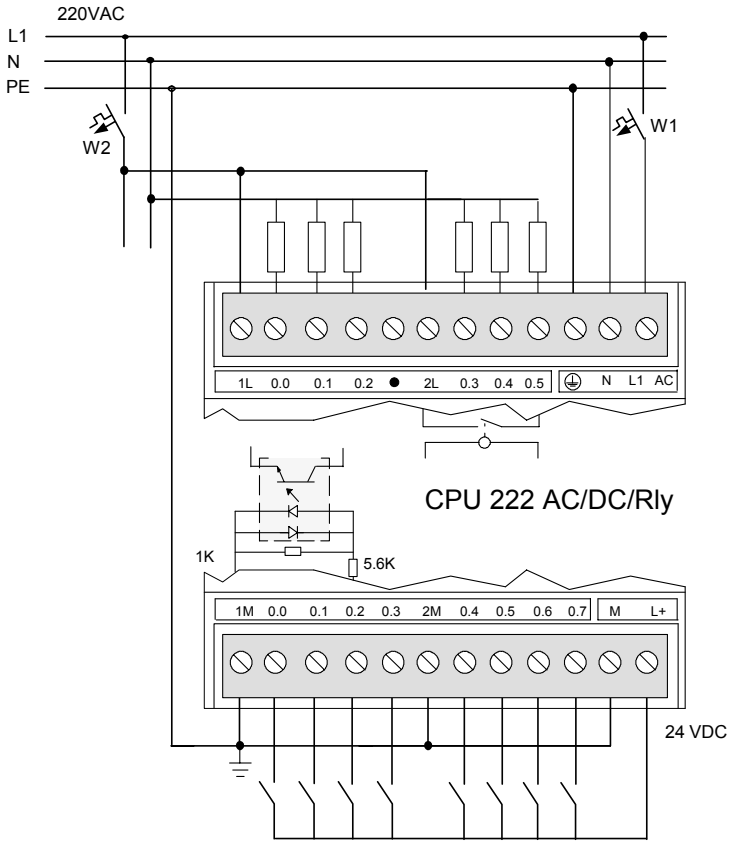
EK 6

Doğru akım beslemeli SIMATIC S7-200 CPU 22x PLC için dış devre ve besleme bağlantı şeması



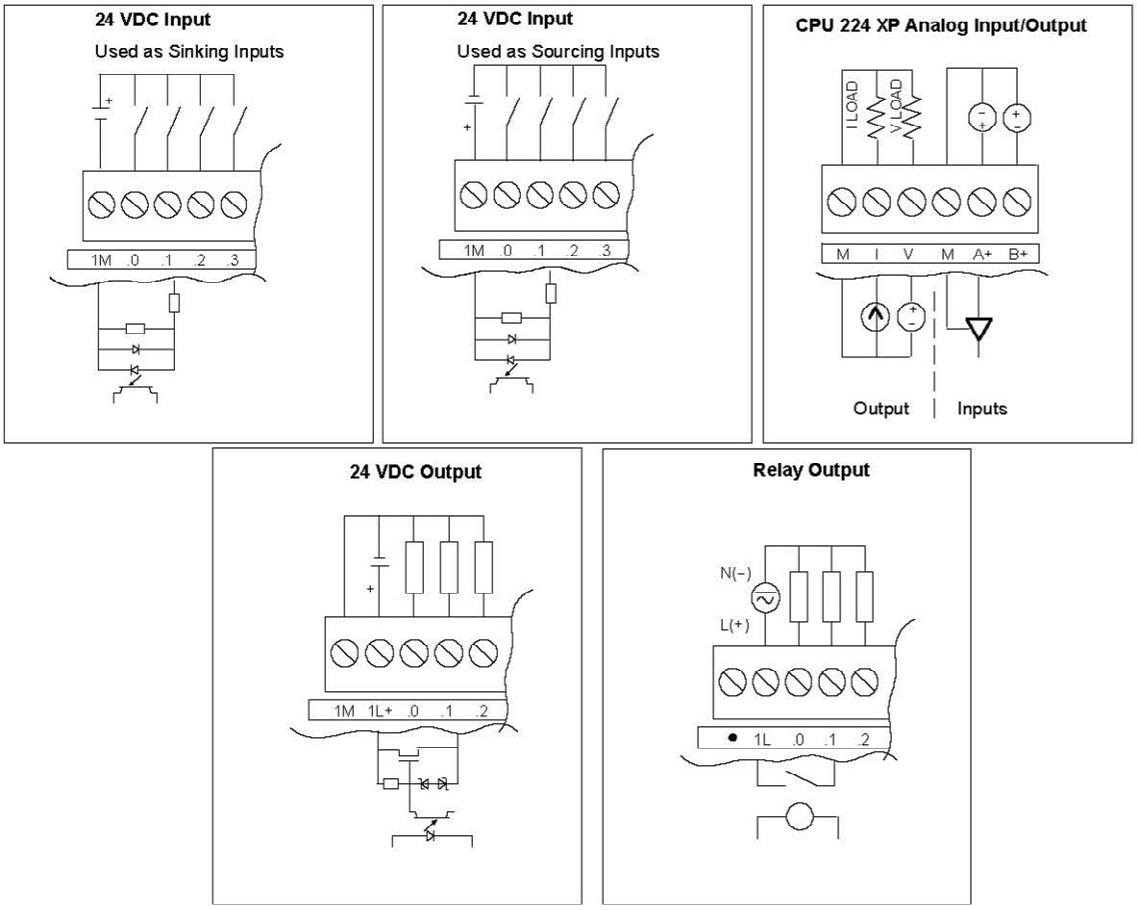
EK 7

Alternatif akım beslemeli SIMATIC S7-200 CPU 22x PLC için dış devre ve besleme bağlantı şeması



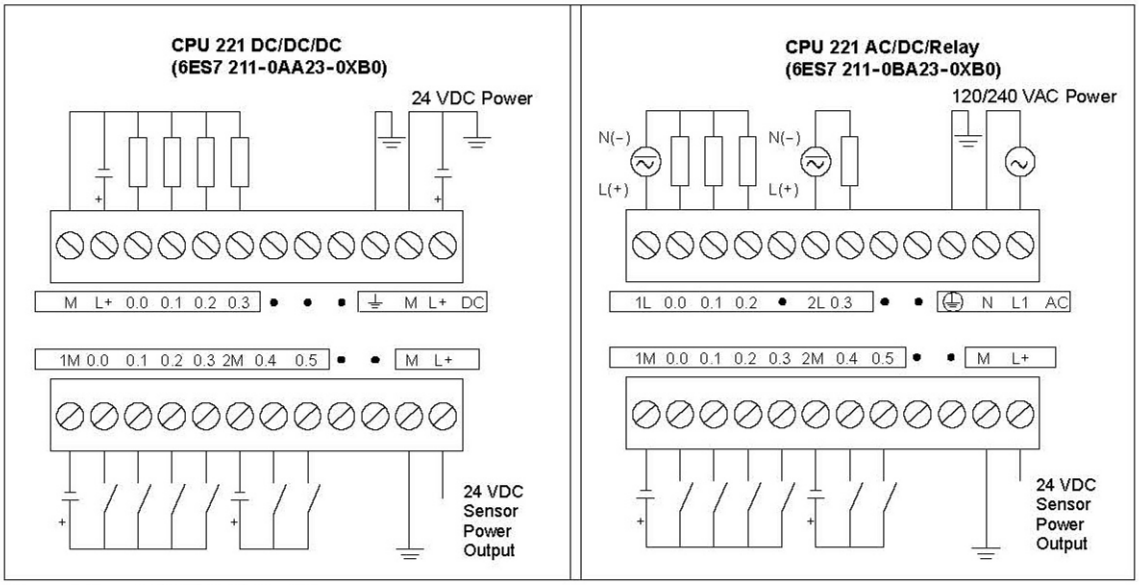
EK 8

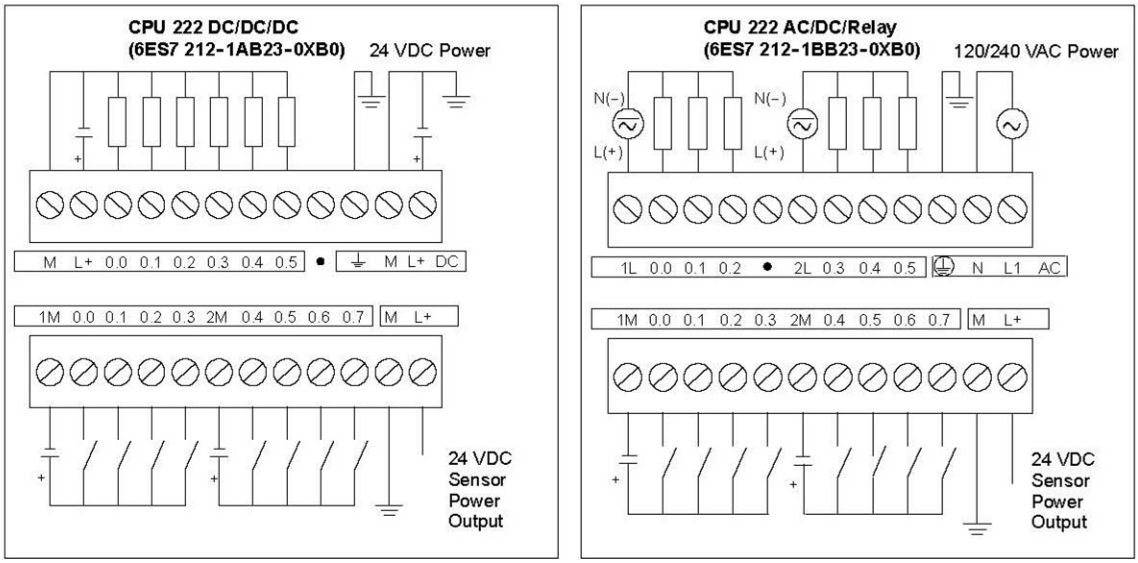
S7-200 CPU 22x giriş-çıkış genel bağlantı şeması



EK 9

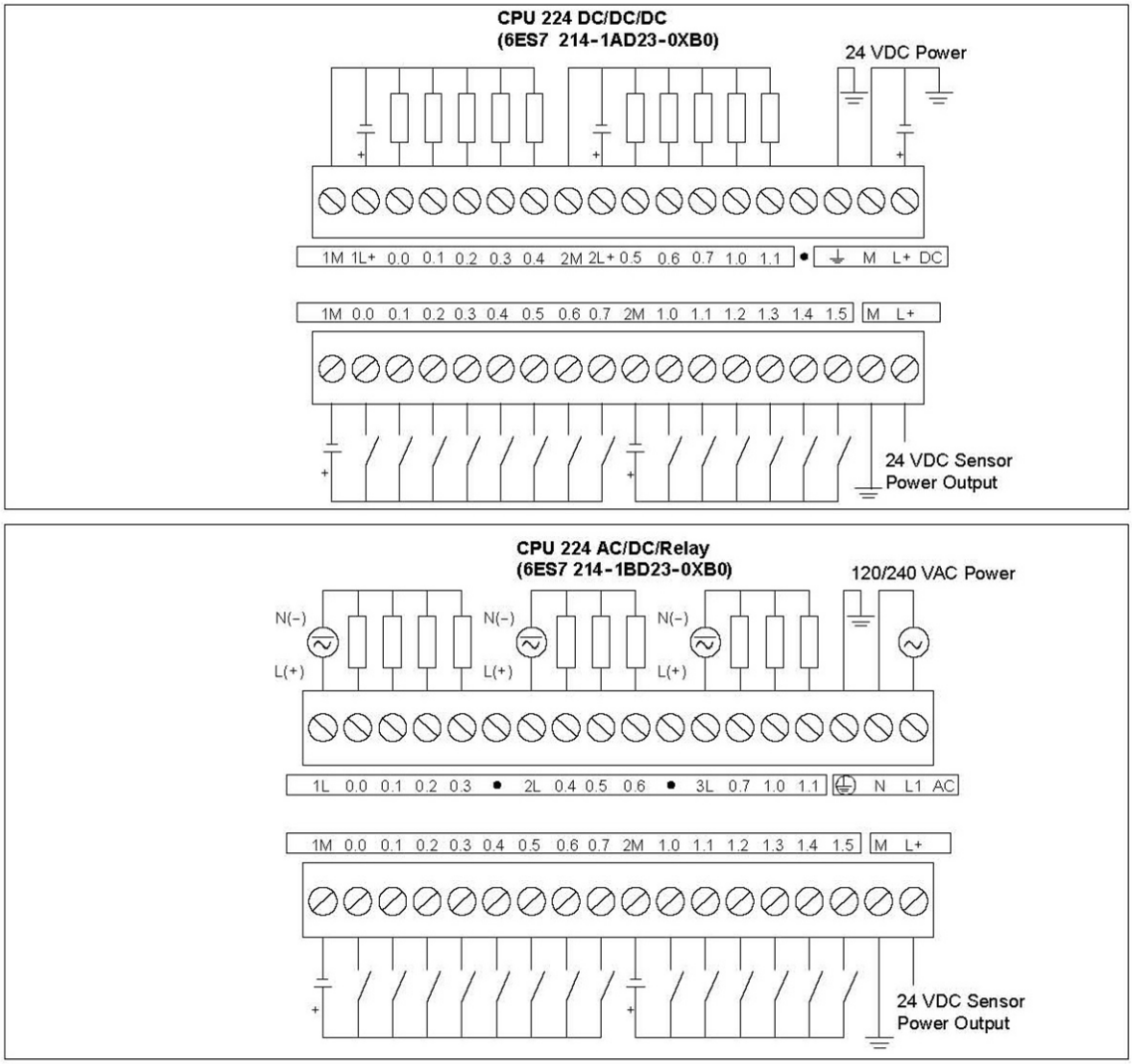
S7-200 CPU 221- CPU 222 giriş-çıkış bağlantı şemaları





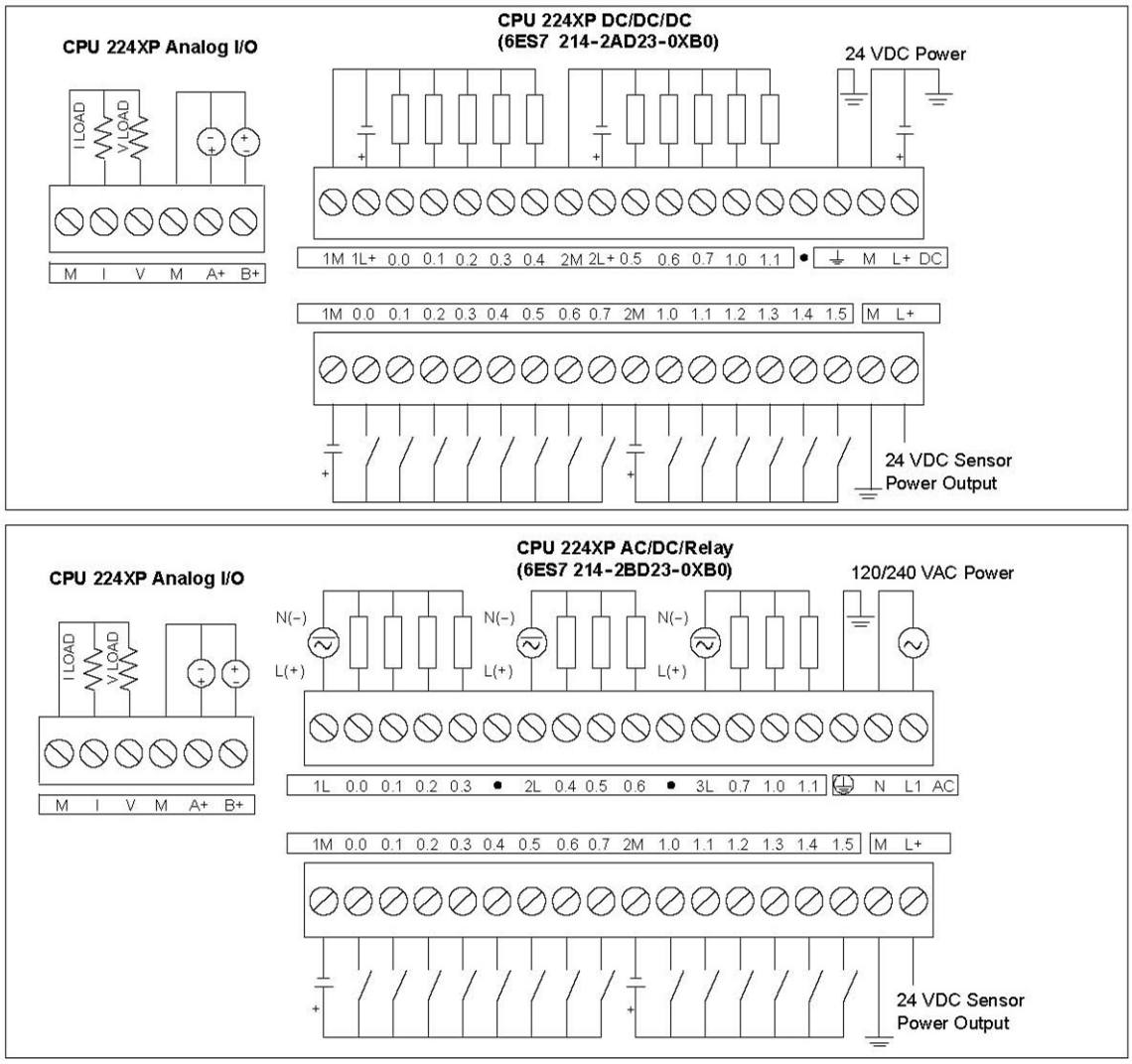
EK 10-1

S7-200 CPU 224 giriş-çıkış bağlantı şeması



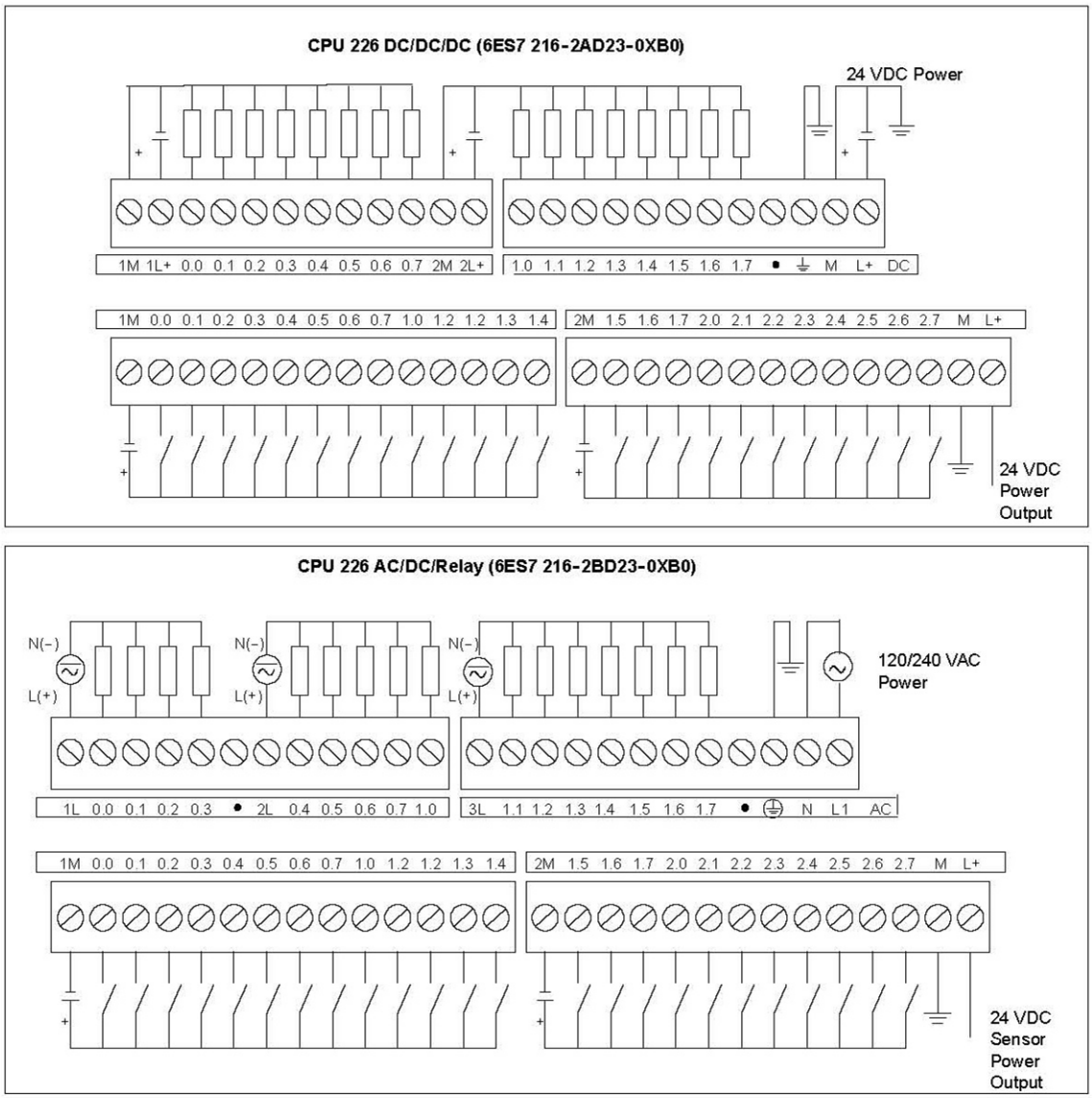
EK 10-2

S7-200 CPU 224XP giriş-çıkış bağlantı şeması



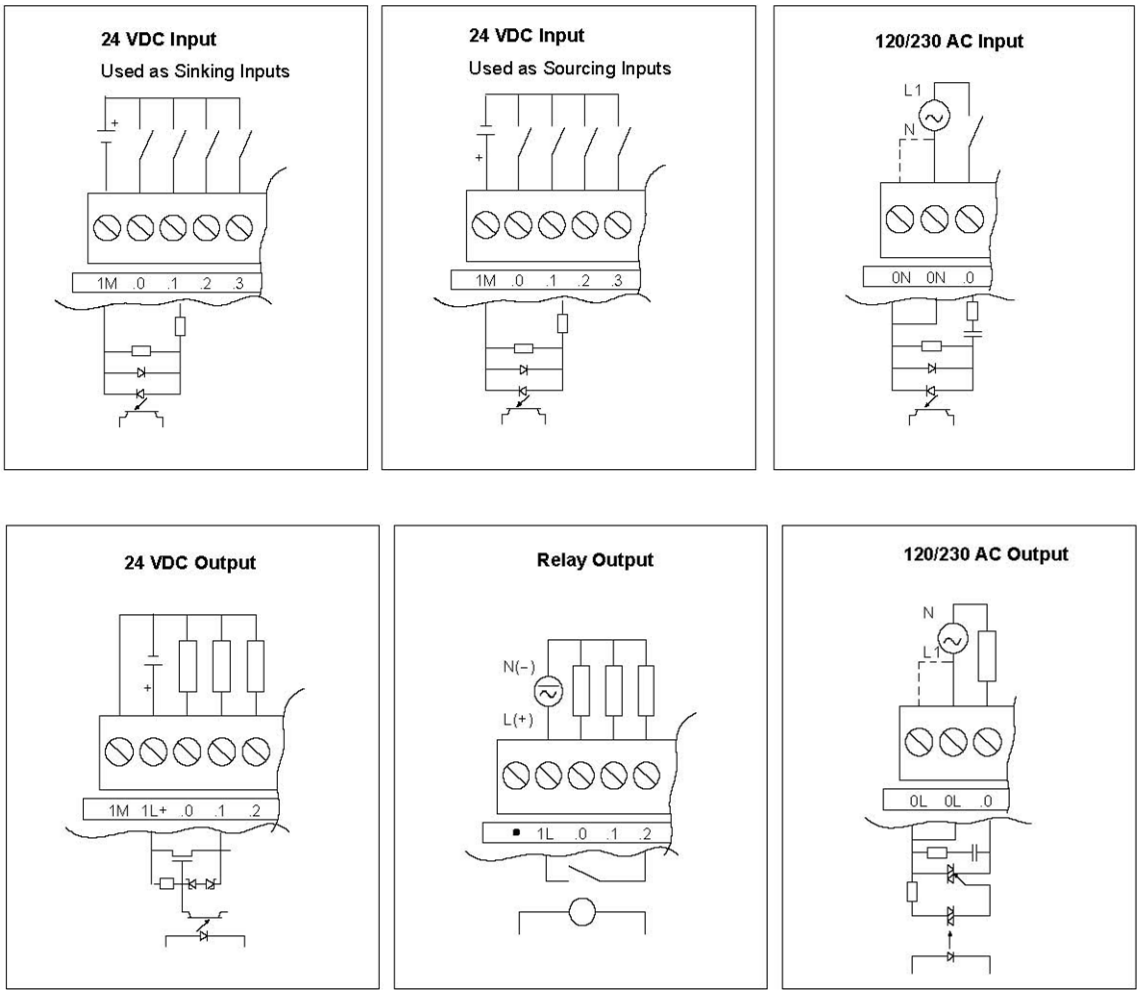
EK 11

S7-200 CPU 226 giriş-çıkış bağlantı şeması



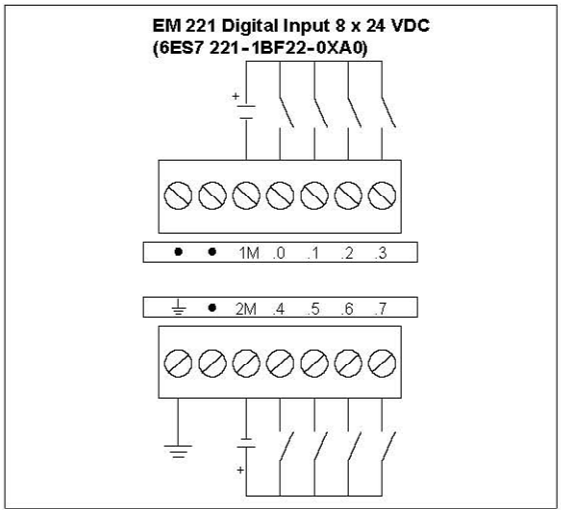
EK 12

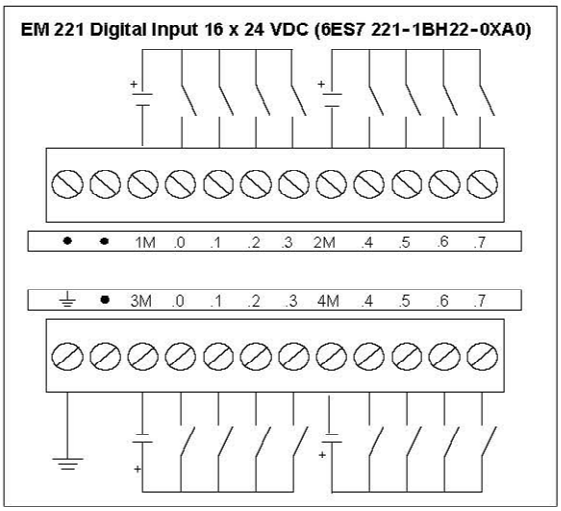
S7-200 CPU 22x genişleme birimleri genel giriş-çıkış bağlantı şeması

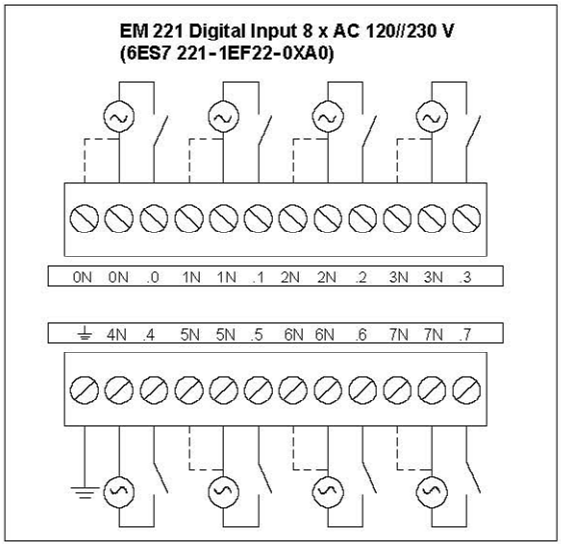


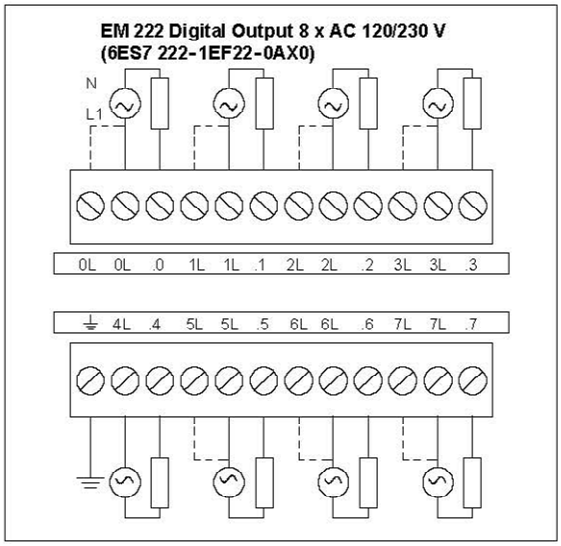
EK 13

S7-200 EM 221 giriş, EM 222 çıkış birimleri bağlantı şemaları



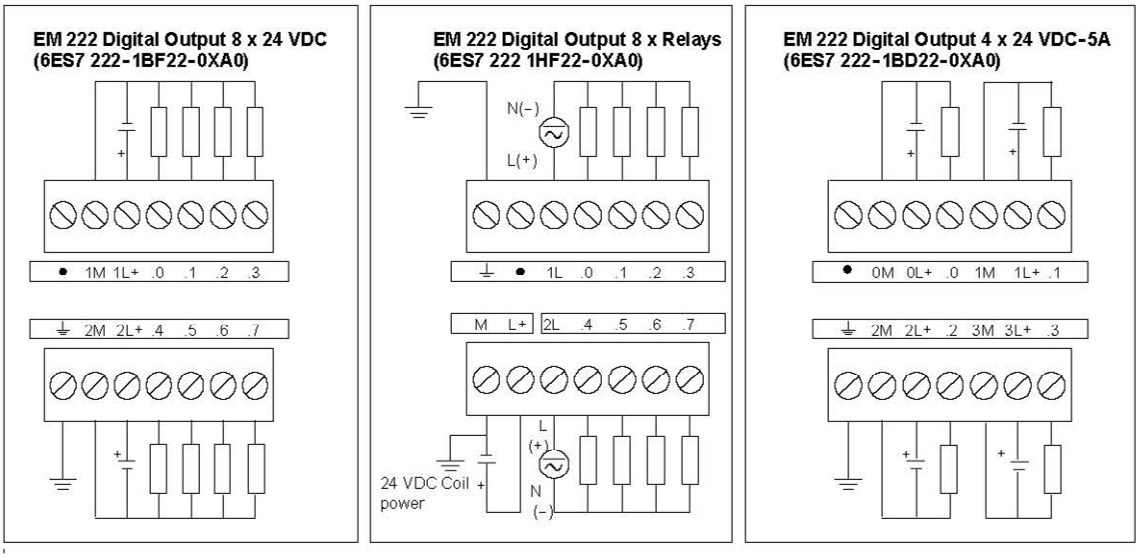




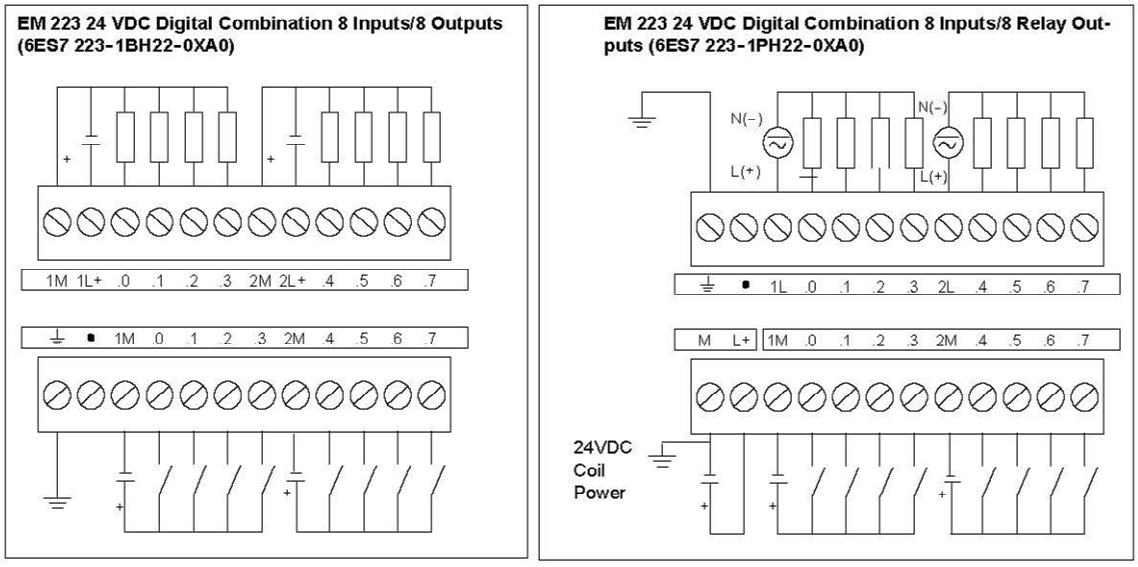


EK 14

S7-200 EM 222 çıkış birimleri bağlantı şemaları

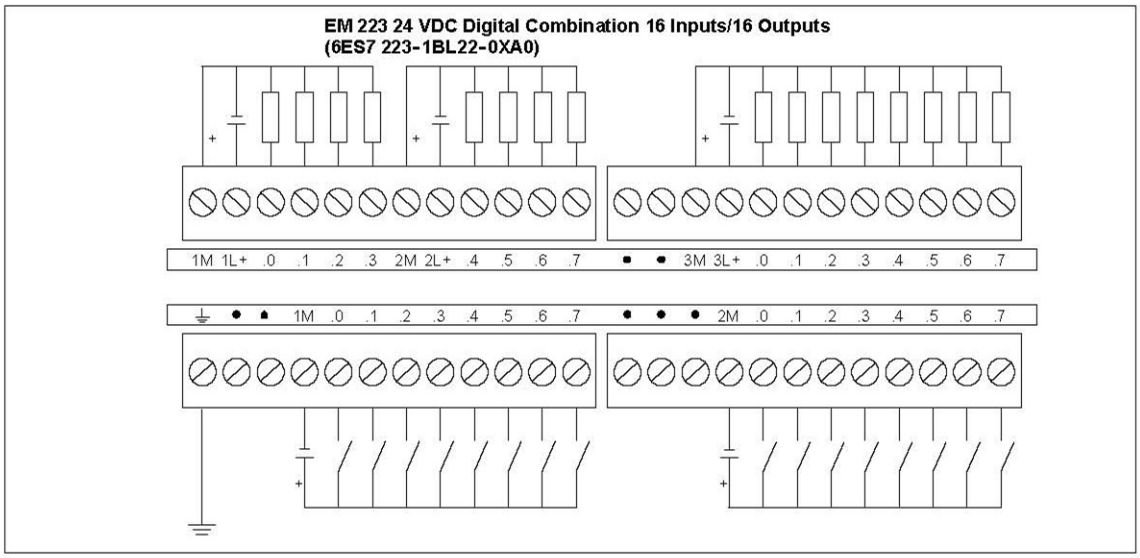


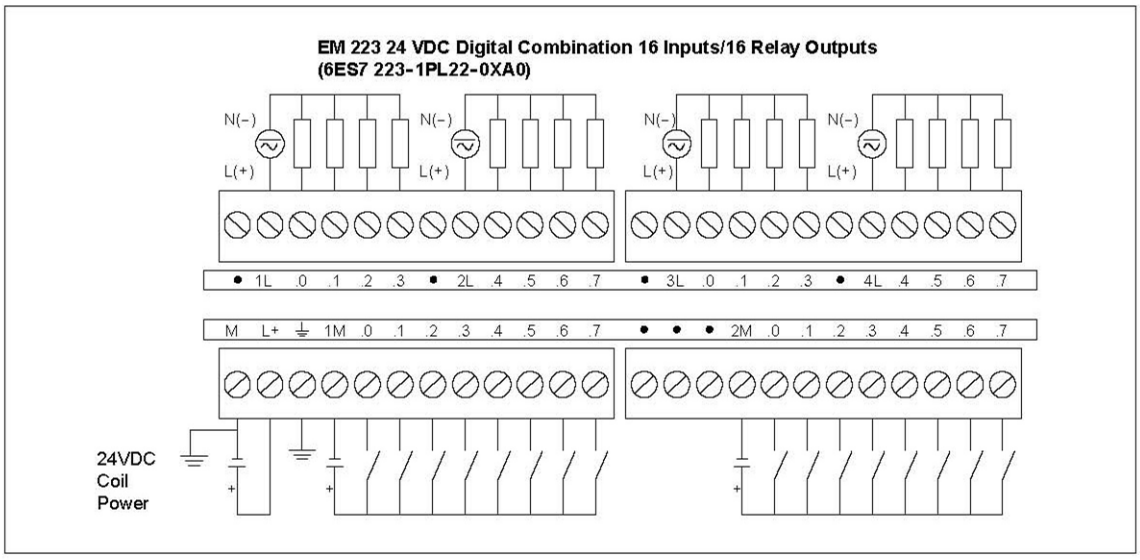
S7-200 EM 223 giriş-çıkış birimleri bağlantı şemaları



EK 15

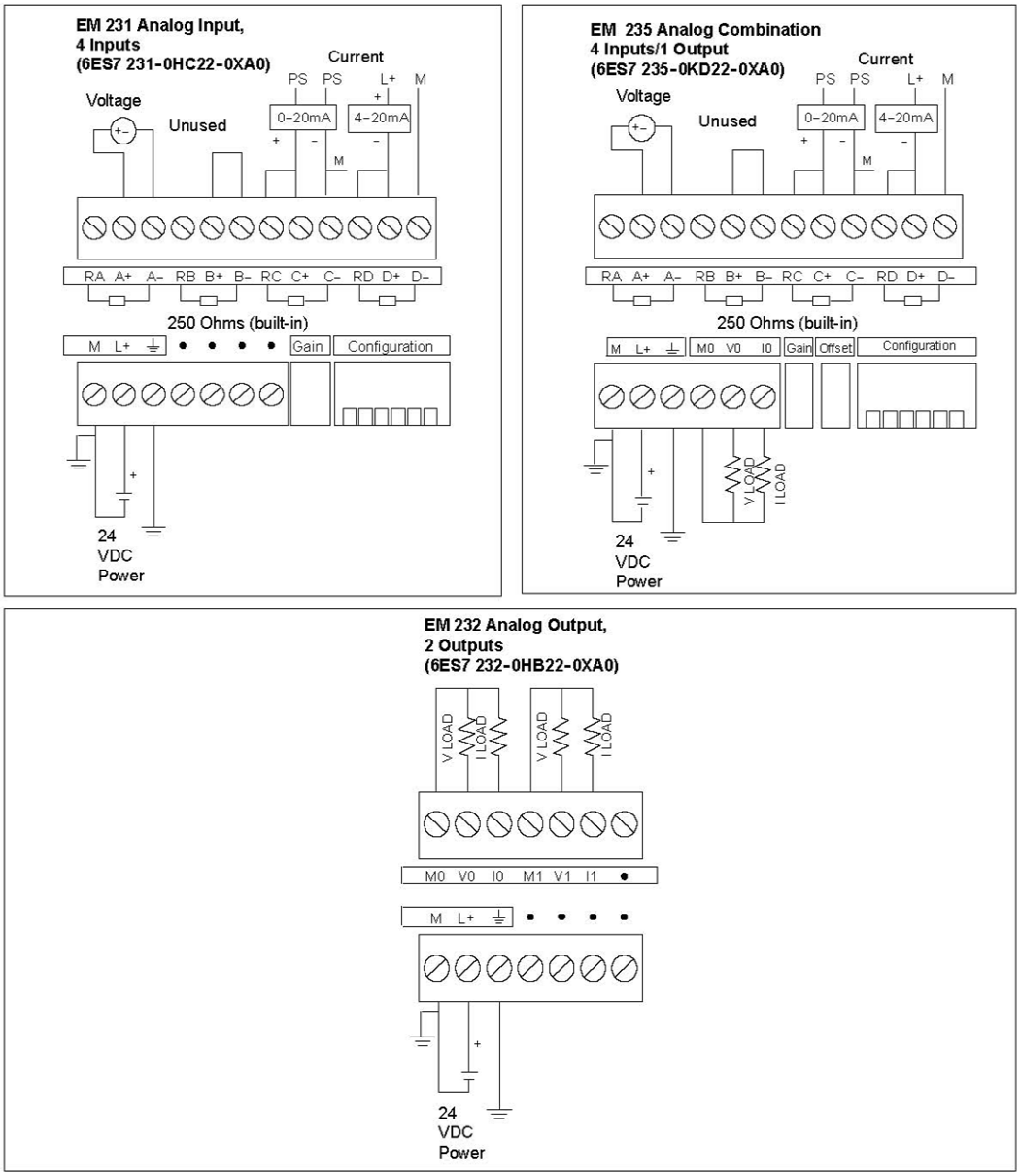
S7-200 EM 223 giriş-çıkış birimleri bağlantı şemaları





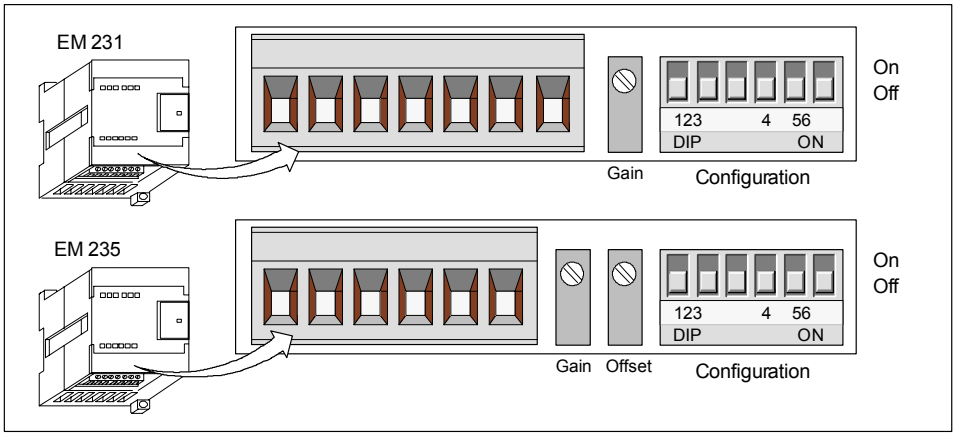
EK 16

S7-200 EM 231, EM 235, EM232 analog giriş-çıkış birimleri bağlantı şemaları



EK 17-1

S7 –200 EM 231ve EM 235 analog giriş-çıkış birimleri ölçeklendirme ve çözünürlük ayarı anahtarları



EM 231 analog biriminin ölçeklendirilmesine ilişkin anahtarlama konumları

Tek kutuplu (Unipolar)				
SW1	SW2	SW3	Tam ölçek giriş	Çözünürlük
ON	OFF	ON	0-10V	2.5 mV
	ON	OFF	0-5V	1.25 mV
			0-20mA	5 μA
Çift kutuplu (Bipolar)				
SW1	SW2	SW3	Tam ölçek giriş	Çözünürlük
OFF	OFF	ON	5V	2.5mV
	ON	OFF	2.5V	1.25mV

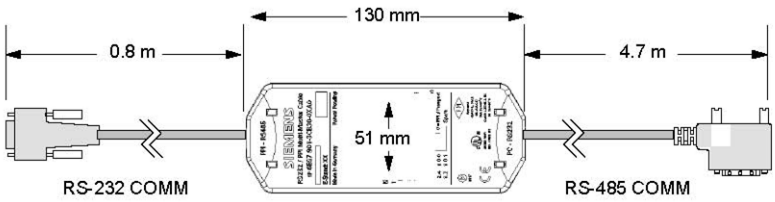
EK 17-2

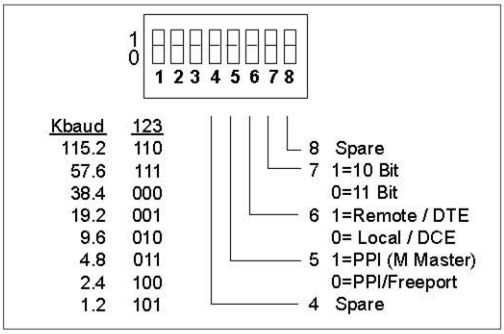
EM 235 Analog biriminin ölçeklendirilmesine ilişkin anahtarlama konumları

Tek kutuplu (Unipolar)							
SW1	SW2	SW3	SW4	SW5	SW6	Tam ölçek giriş	Çözünürlük
ON	OFF	OFF	ON	OFF	ON	0 - 50 mV	12.5 μV
OFF	ON	OFF	ON	OFF	ON	0 - 100 mV	25 μV
ON	OFF	OFF	OFF	ON	ON	0 - 500 mV	125 μV
OFF	ON	OFF	OFF	ON	ON	0 - 1V	250 μV
ON	OFF	OFF	OFF	OFF	ON	0 - 5V	1.25 mV
ON	OFF	OFF	OFF	OFF	ON	0 - 20mA	5 μA
OFF	ON	OFF	OFF	OFF	ON	0 - 10V	2.5 mV
Çift kutuplu (Bipolar)							
SW1	SW2	SW3	SW4	SW5	SW6	Tam ölçek giriş	Çözünürlük
ON	OFF	OFF	ON	OFF	OFF	± 25 mV	12.5 μV
OFF	ON	OFF	ON	OFF	OFF	±50 mV	25 μV
OFF	OFF	ON	ON	OFF	OFF	±100 mV	50 μV
ON	OFF	OFF	OFF	ON	OFF	±250 mV	125 μV
OFF	ON	OFF	OFF	ON	OFF	±500 mV	250 μV
OFF	OFF	ON	OFF	ON	OFF	±1V	500 μV
ON	OFF	OFF	OFF	OFF	OFF	±2.5 V	1.25 mV
OFF	ON	OFF	OFF	OFF	OFF	±5V	2.5 mV
OFF	OFF	ON	OFF	OFF	OFF	±10 V	5m V

EK 18-1

PLC / PC iletişim donanımı (RS-232/RS-485)





LED	Color	Description
Tx	Green	RS-232 transmit indicator
Rx	Green	RS-232 receive indicator
PPI	Green	RS–485 transmit indicator

S7-200 iletişim birimi (RS 485) bağlantı noktaları

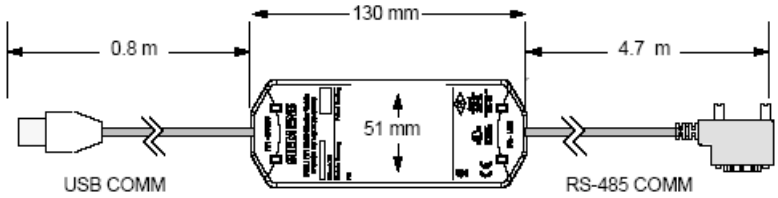
pin	Açıklama
1	Toprak
2	lojik ortak uç
3	RS 485 B işareti
4	RTS(TTL)
5	lojik ortak uç
6	+5V
7	+24V
8	RS 485 A işareti
9	10 bit pro. seçme

(RS 232) birimi bağlantı noktaları

pin	
1	DCD
2	RD
3	TD
4	DTR
5	GRN
6	DSR
7	RTS
8	CTS
9	RI

EK 18-2

PLC / PC iletişim donanımı (USB/RS-485)



LED	Color	Description
Tx	Green	USB transmit indicator
Rx	Green	USB receive indicator
PPI	Green	RS-485 transmit indicator

RS-485 Connector Pin-out		USB Connector Pin-out	
Pin Number	**Signal Description**	**Pin Number**	**Signal Description**
1	No connect	1	USB – DataP
2	24 V Return (RS-485 logic ground)	2	USB – DataM
3	Signal B (RxD/TxD+)	3	USB 5V
4	RTS (TTL level)	4	USB logic ground
5	No connect		
6	No connect		
7	24 V Supply		
8	Signal A (RxD/TxD–)		
9	Protocol select (low = 10 bit)		

EK 19

S7-200 CPU 22x elektriksel özellikler

General	24 VDC Input (CPU 221, CPU 222, CPU 224, CPU 226)		24 VDC Input (CPU 224XP)	
Type	Sink/Source (IEC Type 1 Sink)		Sink/Source (IEC Type 1 Sink, except I0.3 to I0.5)	
Rated voltage	24 VDC at 4 mA typical		24 VDC at 4 mA typical	
Max. continuous permissible voltage	30 VDC			
Surge voltage	35 VDC for 0.5 s			
Logic 1 (min.)	15 VDC at 2.5 mA		15 VDC at 2.5 mA (I0.0 to I0.2 and I0.6 to I1.5) 4 VDC at 8 mA (I0.3 to I0.5)	
Logic 0 (max.)	5 VDC at 1 mA		5 VDC at 1 mA (I0.0 to I0.2 and I0.6 to I1.5) 1 VDC at 1 mA (I0.3 to I0.5)	
Input delay	Selectable (0.2 to 12.8 ms)			
Connection of 2 wire proximity sensor (Bero) Permissible leakage current (max.)	1 mA			
Isolation (field to logic) Optical (galvanic) Isolation groups	Yes 500 VAC for 1 minute See wiring diagram			
High Speed Counter (HSC) input rate HSC Inputs All HSC All HSC HC4, HC5 on CPU 224XP only	Logic 1 Level 15 to 30 VDC 15 to 26 VDC > 4 VDC	Single phase 20 kHz 30 kHz 200 kHz	Two phase 10 kHz 20 kHz 100 kHz	
Inputs on simultaneously	All		All CPU 224XP AC/DC/RELAY only: All at 55° C with DC inputs at 26 VDC max. All at 50° C with DC inputs at 30 VDC max.	
Cable length (max.) Shielded Unshielded	500 m normal inputs, 50 m HSC inputs[1] 300 m normal inputs			

Order Number	Module Name and Description	Dimensions (mm) (W x H x D)	Weight	Dissipation	VDC Available +5 VDC	VDC Available +24 VDC[1]
6ES7 211-0AA23-0XB0	CPU 221 DC/DC/DC 6 Inputs/ 4 Outputs	90 x 80 x 62	270 g	3 W	0 mA	180 mA
6ES7 211-0BA23-0XB0	CPU 221 AC/DC/Relay 6 Inputs/ 4 Relays	90 x 80 x 62	310 g	6 W	0 mA	180 mA
6ES7 212-1AB23-0XB0	CPU 222 DC/DC/DC 8 Inputs/ 6 Outputs	90 x 80 x 62	270 g	5 W	340 mA	180 mA
6ES7 212-1BB23-0XB0	CPU 222 AC/DC/Relay 8 Inputs/ 6 Relays	90 x 80 x 62	310 g	7 W	340 mA	180 mA
6ES7 214-1AD23-0XB0	CPU 224 DC/DC/DC 14 Inputs/ 10 Outputs	120.5 x 80 x 62	360 g	7 W	660 mA	280 mA
6ES7 214-1BD23-0XB0	CPU 224 AC/DC/Relay14 Inputs/ 10 Relays	120.5 x 80 x 62	410 g	10 W	660 mA	280 mA
6ES7 214-2AD23-0XB0	CPU 224XP DC/DC/DC 14 Inputs/10 Outputs	140 x 80 x 62	390 g	8 W	660 mA	280 mA
6ES7 214-2BD23-0XB0	CPU 224XP AC/DC/Relay 14 Inputs/10 Relays	140 x 80 x 62	440 g	11 W	660 mA	280 mA
6ES7 216-2AD23-0XB0	CPU 226 DC/DC/DC 24 Inputs/16 Outputs	196 x 80 x 62	550 g	11 W	1000 mA	400 mA
6ES7 216-2BD23-0XB0	CPU 226 AC/DC/Relay 24 Inputs/16 Relays	196 x 80 x 62	660 g	17 W	1000 mA	400 mA

EK 20

Kesme olayları ve öncelik sıralaması

Event Number	Interrupt Description	Priority Group	Priority in Group
8	Port 0: Receive character	Communications (highest)	0
9	Port 0: Transmit complete		0
23	Port 0: Receive message complete		0
24	Port 1: Receive message complete		1
25	Port 1: Receive character		1
26	Port 1: Transmit complete		1
19	PTO 0 complete interrupt	Discrete (middle)	0
20	PTO 1 complete interrupt		1
0	I0.0, Rising edge		2
2	I0.1, Rising edge		3
4	I0.2, Rising edge		4
6	I0.3, Rising edge		5
1	I0.0, Falling edge		6
3	I0.1, Falling edge		7
5	I0.2, Falling edge		8
7	I0.3, Falling edge		9
12	HSC0 CV=PV (current value = preset value)		10
27	HSC0 direction changed		11
28	HSC0 external reset		12
13	HSC1 CV=PV (current value = preset value)		13
14	HSC1 direction input changed		14
15	HSC1 external reset		15
16	HSC2 CV=PV		16
17	HSC2 direction changed		17
18	HSC2 external reset		18
32	HSC3 CV=PV (current value = preset value)		19
29	HSC4 CV=PV (current value = preset value)		20
30	HSC4 direction changed		21
31	HSC4 external reset		22
33	HSC5 CV=PV (current value = preset value)		23
10	Timed interrupt 0	Timed (lowest)	0
11	Timed interrupt 1		1
21	Timer T32 CT=PT interrupt		2
22	Timer T96 CT=PT interrupt		3

EK 21-1

S7-300 /S7-400 PLC Ailesi Komut Kümesi

English Mnemonics	German Mnemonics	Program Elements Catalog	Description
+	+	Integer math Instruction	Add Integer Constant (16, 32-Bit)
=	=	Bit logic Instruction	Assign
)	)	Bit logic Instruction	Nesting Closed
+AR1	+AR1	Accumulator	AR1 Add ACCU 1 to Address Register 1
+AR2	+AR2	Accumulator	AR2 Add ACCU 1 to Address Register 2
+D	+D	Integer math Instruction	Add ACCU 1 and ACCU 2 as Double Integer (32-Bit)
–D	–D	Integer math Instruction	Subtract ACCU 1 from ACCU 2 as Double Integer (32-Bit)
*D	*D	Integer math Instruction	Multiply ACCU 1 and ACCU 2 as Double Integer (32-Bit)
/D	/D	Integer math Instruction	Divide ACCU 2 by ACCU 1 as Double Integer (32-Bit)
? D	? D	Compare	Compare Double Integer (32-Bit) ==, <>, >, <, >=, <=
+I	+I	Integer math Instruction	Add ACCU 1 and ACCU 2 as Integer (16-Bit)
–I	–I	Integer math Instruction	Subtract ACCU 1 from ACCU 2 as Integer (16-Bit)
*I	*I	Integer math Instruction	Multiply ACCU 1 and ACCU 2 as Integer (16-Bit)
/I	/I	Integer math Instruction	Divide ACCU 2 by ACCU 1 as Integer (16-Bit)
? I	? I	Compare	Compare Integer (16-Bit) ==, <>, >, <, >=, <=
+R	+R	Floating point Instruction	Add ACCU 1 and ACCU 2 as a Floating-Point Number (32-Bit IEEE-FP)
–R	–R	Floating point Instruction	Subtract ACCU 1 from ACCU 2 as a Floating-Point Number (32-Bit IEEE-FP)
*R	*R	Floating point Instruction	Multiply ACCU 1 and ACCU 2 as Floating-Point Numbers (32-Bit IEEE-FP)
/R	/R	Floating point Instruction	Divide ACCU 2 by ACCU 1 as a Floating-Point Number (32-Bit IEEE-FP)
? R	? R	Compare	Compare Floating-Point Number (32-Bit) ==, <>, >, <, >=, <=
A	U	Bit logic Instruction	And
A(	U(	Bit logic Instruction	And with Nesting Open
ABS	ABS	Floating point Instruction	Absolute Value of a Floating-Point Number (32-Bit IEEE-FP)
ACOS	ACOS	Floating point Instruction	Generate the Arc Cosine of a Floating-Point Number (32-Bit)
AD	UD	Word logic Instruction	AND Double Word (32-Bit)
AN	UN	Bit logic Instruction	And Not
AN(	UN(	Bit logic Instruction	And Not with Nesting Open

EK 21-2

English Mnemonics	German Mnemonics	Program Elements Catalog	Description
ASIN	ASIN	Floating point Instruction	Generate the Arc Sine of a Floating-Point Number (32-Bit)
ATAN	ATAN	Floating point Instruction	Generate the Arc Tangent of a Floating-Point Number (32-Bit)
AW	UW	Word logic Instruction	AND Word (16-Bit)
BE	BE	Program control	Block End
BEC	BEB	Program control	Block End Conditional
BEU	BEA	Program control	Block End Unconditional
BLD	BLD	Program control	Program Display Instruction (Null)
BTD	BTD	Convert	BCD to Integer (32-Bit)
BTI	BTI	Convert	BCD to Integer (16-Bit)
CAD	TAD	Convert	Change Byte Sequence in ACCU 1 (32-Bit)
CALL	CALL	Program control	Block Call
CALL	CALL	Program control	Call Multiple Instance
CALL	CALL	Program control	Call Block from a Library
CAR	TAR	Load/Transfer	Exchange Address Register 1 with Address Register 2
CAW	TAW	Convert	Change Byte Sequence in ACCU 1-L (16-Bit)
CC	CC	Program control	Conditional Call
CD	ZR	Counters	Counter Down
CDB	TDB	Convert	Exchange Shared DB and Instance DB
CLR	CLR	Bit logic Instruction	Clear RLO (=0)
COS	COS	Floating point Instruction	Generate the Cosine of Angles as Floating-Point Numbers (32-Bit)
CU	ZV	Counters	Counter Up
DEC	DEC	Accumulator	Decrement ACCU 1-L-L
DTB	DTB	Convert	Double Integer (32-Bit) to BCD
DTR	DTR	Convert	Double Integer (32-Bit) to Floating-Point (32-Bit IEEE-FP)
ENT	ENT	Accumulator	Enter ACCU Stack
EXP	EXP	Floating point Instruction	Generate the Exponential Value of a Floating-Point Number (32-Bit)
FN	FN	Bit logic Instruction	Edge Negative
FP	FP	Bit logic Instruction	Edge Positive
FR	FR	Counters	Enable Counter (Free) (free, FR C 0 to C 255)
FR	FR	Timers	Enable Timer (Free)
INC	INC	Accumulator	Increment ACCU 1-L-L
INVD	INVD	Convert	Ones Complement Double Integer (32-Bit)
INVI	INVI	Convert	Ones Complement Integer (16-Bit)
ITB	ITB	Convert	Integer (16-Bit) to BCD
ITD	ITD	Convert	Integer (16-Bit) to Double Integer (32-Bit)
JBI	SPBI	Jumps	Jump if BR = 1
JC	SPB	Jumps	Jump if RLO = 1

EK 21-3

English Mnemonics	German Mnemonics	Program Elements Catalog	Description
JCB	SPBB	Jumps	Jump if RLO = 1 with BR
JCN	SPBN	Jumps	Jump if RLO = 0
JL	SPL	Jumps	Jump to Labels
JM	SPM	Jumps	Jump if Minus
JMZ	SPMZ	Jumps	Jump if Minus or Zero
JN	SPN	Jumps	Jump if Not Zero
JNB	SPBNB	Jumps	Jump if RLO = 0 with BR
JNBI	SPBIN	Jumps	Jump if BR = 0
JO	SPO	Jumps	Jump if OV = 1
JOS	SPS	Jumps	Jump if OS = 1
JP	SPP	Jumps	Jump if Plus
JPZ	SPPZ	Jumps	Jump if Plus or Zero
JU	SPA	Jumps	Jump Unconditional
JUO	SPU	Jumps	Jump if Unordered
JZ	SPZ	Jumps	Jump if Zero
L	L	Load/Transfer	Load
L DBLG	L DBLG	Load/Transfer	Load Length of Shared DB in ACCU 1
L DBNO	L DBNO	Load/Transfer	Load Number of Shared DB in ACCU 1
L DILG	L DILG	Load/Transfer	Load Length of Instance DB in ACCU 1
L DINO	L DINO	Load/Transfer	Load Number of Instance DB in ACCU 1
L STW	L STW	Load/Transfer	Load Status Word into ACCU 1
L	L	Timers	Load Current Timer Value into ACCU 1 as Integer (the current timer value can be a number from 0 to 255, for example, L T 32)
L	L	Counters	Load Current Counter Value into ACCU 1 (the current counter value can be a number from 0 to 255, for example, L C 15)
LAR1	LAR1	Load/Transfer	Load Address Register 1 from ACCU 1
LAR1 <D>	LAR1<D>	Load/Transfer	Load Address Register 1 with Double Integer (32-Bit Pointer)
LAR1 AR2	LAR1 AR2	Load/Transfer	Load Address Register 1 from Address Register 2
LAR2	LAR2	Load/Transfer	Load Address Register 2 from ACCU 1
LAR2 <D>	LAR2 <D>	Load/Transfer	Load Address Register 2 with Double Integer (32-Bit Pointer)
LC	LC	Counters	Load Current Counter Value into ACCU 1 as BCD (the current timer value can be a number from 0 to 255, for example, LC C 15)
LC	LC	Timers	Load Current Timer Value into ACCU 1 as BCD (the current counter value can be a number from 0 to 255, for example, LC T 32)
LEAVE	LEAVE	Accumulator	Leave ACCU Stack

EK 21-4

English Mnemonics	German Mnemonics	Program Elements Catalog	Description
LN	LN	Floating point Instruction	Generate the Natural Logarithm of a Floating-Point Number (32-Bit)
LOOP	LOOP	Jumps	Loop
MCR(	MCR(	Program control	Save RLO in MCR Stack, Begin MCR
)MCR	)MCR	Program control	End MCR
MCRA	MCRA	Program control	Activate MCR Area
MCRD	MCRD	Program control	Deactivate MCR Area
MOD	MOD	Integer math Instruction	Division Remainder Double Integer (32-Bit)
NEGD	NEGD	Convert	Twos Complement Double Integer (32-Bit)
NEGI	NEGI	Convert	Twos Complement Integer (16-Bit)
NEGR	NEGR	Convert	Negate Floating-Point Number (32-Bit, IEEE-FP)
NOP 0	NOP 0	Accumulator	Null Instruction
NOP 1	NOP 1	Accumulator	Null Instruction
NOT	NOT	Bit logic Instruction	Negate RLO
O	O	Bit logic Instruction	Or
O(	O(	Bit logic Instruction	Or with Nesting Open
OD	OD	Word logic Instruction	OR Double Word (32-Bit)
ON	ON	Bit logic Instruction	Or Not
ON(	ON(	Bit logic Instruction	Or Not with Nesting Open
OPN	AUF	DB call	Open a Data Block
OW	OW	Word logic Instruction	OR Word (16-Bit)
POP	POP	Accumulator	CPU with Two ACCUs
POP	POP	Accumulator	CPU with Four ACCUs
PUSH	PUSH	Accumulator	CPU with Two ACCUs
PUSH	PUSH	Accumulator	CPU with Four ACCUs
R	R	Bit logic Instruction	Reset
R	R	Counters	Reset Counter (the current counter can be a number from 0 to 255, for example, R C 15)
R	R	Timers	Reset Timer (the current timer can be a number from 0 to 255, for example, R T 32)
RLD	RLD	Shift/Rotate	Rotate Left Double Word (32-Bit)
RLDA	RLDA	Shift/Rotate	Rotate ACCU 1 Left via CC 1 (32-Bit)
RND	RND	Convert	Round
RND–	RND–	Convert	Round to Lower Double Integer
RND+	RND+	Convert	Round to Upper Double Integer
RRD	RRD	Shift/Rotate	Rotate Right Double Word (32-Bit)
RRDA	RRDA	Shift/Rotate	Rotate ACCU 1 Right via CC 1 (32-Bit)
S	S	Bit logic Instruction	Set
S	S	Counters	Set Counter Preset Value (the current counter can be a number from 0 to 255, for example, S C 15)
SAVE	SAVE	Bit logic Instruction	Save RLO in BR Register

EK 21-5

English Mnemonics	German Mnemonics	Program Elements Catalog	Description
SD	SE	Timers	On-Delay Timer
SE	SV	Timers	Extended Pulse Timer
SET	SET	Bit logic Instruction	Set
SF	SA	Timers	Off-Delay Timer
SIN	SIN	Floating point Instruction	Generate the Sine of Angles as Floating-Point Numbers (32-Bit)
SLD	SLD	Shift/Rotate	Shift Left Double Word (32-Bit)
SLW	SLW	Shift/Rotate	Shift Left Word (16-Bit)
SP	SI	Timers	Pulse Timer
SQR	SQR	Floating point Instruction	Generate the Square of a Floating-Point Number (32-Bit)
SQRT	SQRT	Floating point Instruction	Generate the Square Root of a Floating-Point Number (32-Bit)
SRD	SRD	Shift/Rotate	Shift Right Double Word (32-Bit)
SRW	SRW	Shift/Rotate	Shift Right Word (16-Bit)
SS	SS	Timers	Retentive On-Delay Timer
SSD	SSD	Shift/Rotate	Shift Sign Double Integer (32-Bit)
SSI	SSI	Shift/Rotate	Shift Sign Integer (16-Bit)
T	T	Load/Transfer	Transfer
T	T	Load/Transfer	Transfer ACCU 1 into Status Word
TAK	TAK	Accumulator	Toggle ACCU 1 with ACCU 2
TAN	TAN	Floating point Instruction	Generate the Tangent of Angles as Floating-Point Numbers (32-Bit)
TAR1	TAR1	Load/Transfer	Transfer Address Register 1 to ACCU 1
TAR1	TAR1	Load/Transfer	Transfer Address Register 1 to Destination (32-Bit Pointer)
TAR1	TAR1	Load/Transfer	Transfer Address Register 1 to Address Register 2
TAR2	TAR2	Load/Transfer	Transfer Address Register 2 to ACCU 1
TAR2	TAR2	Load/Transfer	Transfer Address Register 2 to Destination (32-Bit Pointer)
TRUNC	TRUNC	Convert	Truncate
UC	UC	Program control	Unconditional Call
X	X	Bit logic Instruction	Exclusive Or
X(	X(	Bit logic Instruction	Exclusive Or with Nesting Open
XN	XN	Bit logic Instruction	Exclusive Or Not
XN(	XN(	Bit logic Instruction	Exclusive Or Not with Nesting Open
XOD	XOD	Word logic Instruction	Exclusive OR Double Word (32-Bit)
XOW	XOW	Word logic Instruction	Exclusive OR Word (16-Bit)

DİZİN

Alçak geçiren filtre, 261,383
Ana denetim komutları, 326
Analog giriş-çıkış birimi, 225
Analog veriler üzerinde işlemler, 230
Analog verilere erişim, 226, 374
Anında aktarma işlemi, 132
Anında okuma işlemi, 131
Ardışıl denetim, 117
Atlama komutları, 111, 325, 356
Az sönümlü sistemler, 275

BCD sayılardan dönüşüm, 338
Bellek alanları, 32
Birinci mertebeden integral etkili sistemler, 291,401
Birinci mertebeden sistemler, 265

Çıkan ve düşen kenarı algılama, 54, 305
Çıkış birimi, 5

Darbe katarı üretilmesi (tek parçalı), 209
çok parçalı, 212
Dış devre bağlantıları, 36
Doğrudan okuma, 55
Doğrudan yazma, 55
Dolaylı adresleme, 164, 361
Dönüştürme işlemleri, 338, 339

Enerji kesilmesi durumu, 36

Fonksiyon blok diyagramı, 11

Giriş birimi, 4

IEC 1131-3 programlama modu, 72
İki konumlu kontrol, 234, 375
İkinci mertebeden sistemler, 274, 396
İşaretli tamsayılar üzerinde işlemler, 135, 140, 330
İşaretli tamsayıya dönüştürme, 153, 360

Kesmeli çalışma
-Hızlı sayıcı kaynaklı, 192
-Olaya bağlı, 185
-Zamana bağlı, 187
-Zamanlayıcı kaynaklı, 190

Kare dalga üretilmesi, 208
Karekök alma, 146, 359
Karşılaştırma, 69, 324
Kaydırma ve döndürme, 169
Kısa devre asenkron Motor
-İki yönde kumanda, 80
-Yıldız-üçgen yol verme, 85, 316
Komut durum sözcüğü, 346
Kritik Sönümlü Sistem, 286
Kurma ve silme (SET-RESET) işlemleri, 52, 303
Kurma, silme baskın iki kararlı işlemler, 52, 305

Mantıksal işlemler, 167
Mantıksal yığın işlemleri, 56
Matematiksel işlemler, 358
Merdiven mantığı ile programlama, 10

Oransal kontrol, 378
Özel bellek alanı (SMx), 34

PI kontrol, 241,380
PI kontrolör gerçekleme, 254
PID fonksiyon bloğu, 392
PID kontrol, 385
Program denetim komutları, 113

Reel sayılar üzerinde işlemler, 143, 333
Reel sayılara dönüştürme, 151
Rotoru sargılı asenkron motorun kumandası, 92

Sayısal benzetim, 265, 396
Sayıcılar
-Geri sayıcı, 67, 323
-İleri sayıcı, 66, 322
-İleri-geri sayıcı, 68, 321
STL ile programlama, 346

Tanılama (Diagnostic) işareti üretme, 122

Uzun süreli 1ms artımlı zamanlama, 180
Uzun süreli gecikme işareti, 102

Veri aktarma, 130, 341
Veri bloğu aktarma işlemi, 133
Veri blokları, 343
Verilerin korunması, 36

Yer değiştirme, 132
Yüksek hızlı çıkış işareti, 207, 215

Zaman bilgisinin okunması, 175
Zamanlayıcı
Darbe (S_PULSE, SP), 307
-Gecikmeli açan (S_OFFDT, SF), 314
-Gecikmeli açan (TOF), 64
-Gecikmeli kapatan kalıcı (TONR), 62
-Gecikmeli kapatan (S_ODT, SD), 311
-Gecikmeli kapatan (TON), 61
-Gecikmeli kapatan kalıcı (S_ODTS, SS), 312
-Uzatılmış darbe (S_PEXT, SE), 309