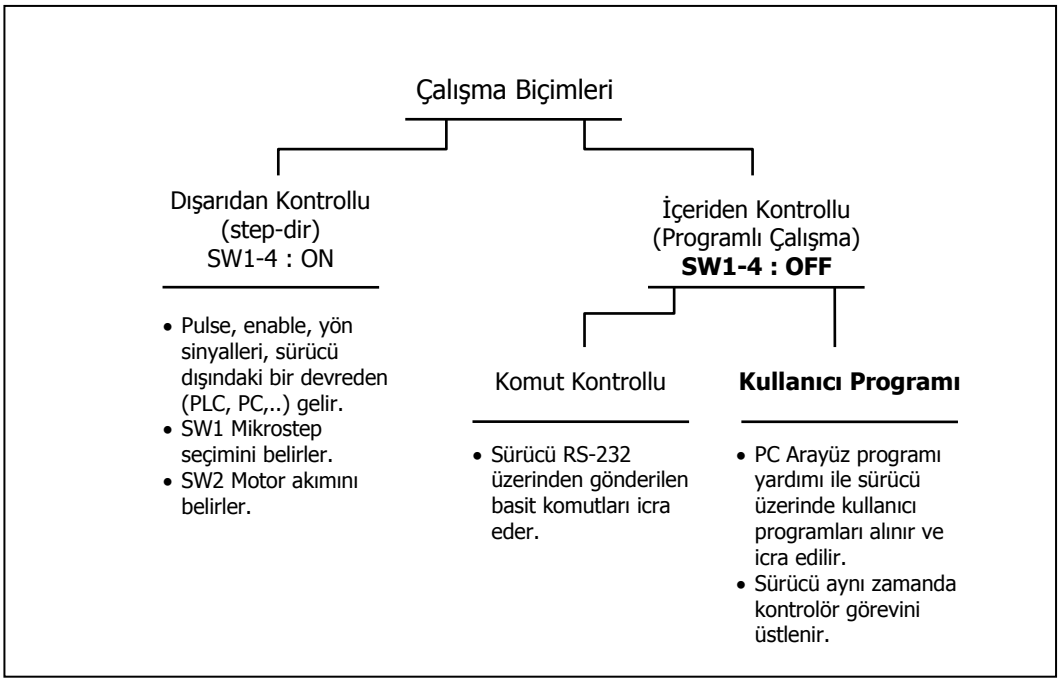
STEMO-3 PROGRAMLAMA KILAVUZU

1 Giriş

STEMO3 programlama dili, Stemo3 iki fazlı step motor sürücüsü için geliştirilmiş olup kullanıcıların sürücüyü kendi uygulamalarına göre programlayarak çalıştırılmasını sağlar. Programlama dili, sürücü için tanımlanmış motor hızı, dönüş yönü, motor akımı, zaman gecikmeleri, pozisyonlama, lojik giriş ve çıkışlarla ilişkilendirme gibi temel fonksiyonları bir program akışı içinde düzenlenleyerek çalıştırılmasını sağlar. Program oluşturma ve sürücüye yüklemek için PC üzerinde çalışan bir arayüz programı STEMO-PS mevcuttur. Bu arayüz program ile komutlar liste halinde yazılır, hatalı yazımlara karşı sentaks kontrolu yaptırılır, düzeltmeler yapılır, yazılan program farklı dosya isimleri ile bilgisayar hafızasına kayıt edilebilir, daha önceden kayıt edilmiş programlar çağırılıp üzerinde çalışmalar/değişiklikler yapılabilir, yazılan program sürücüye yüklenip çalıştırılabilir.

Kullanıcı programının çalışması için sürücü üzerindeki ayar anahtarı “SW1-4:OFF” konumunda olmalıdır.



2 Sürücü Arayüz Programı STEMO-PS’nin Kullanılması

2.1 Genel

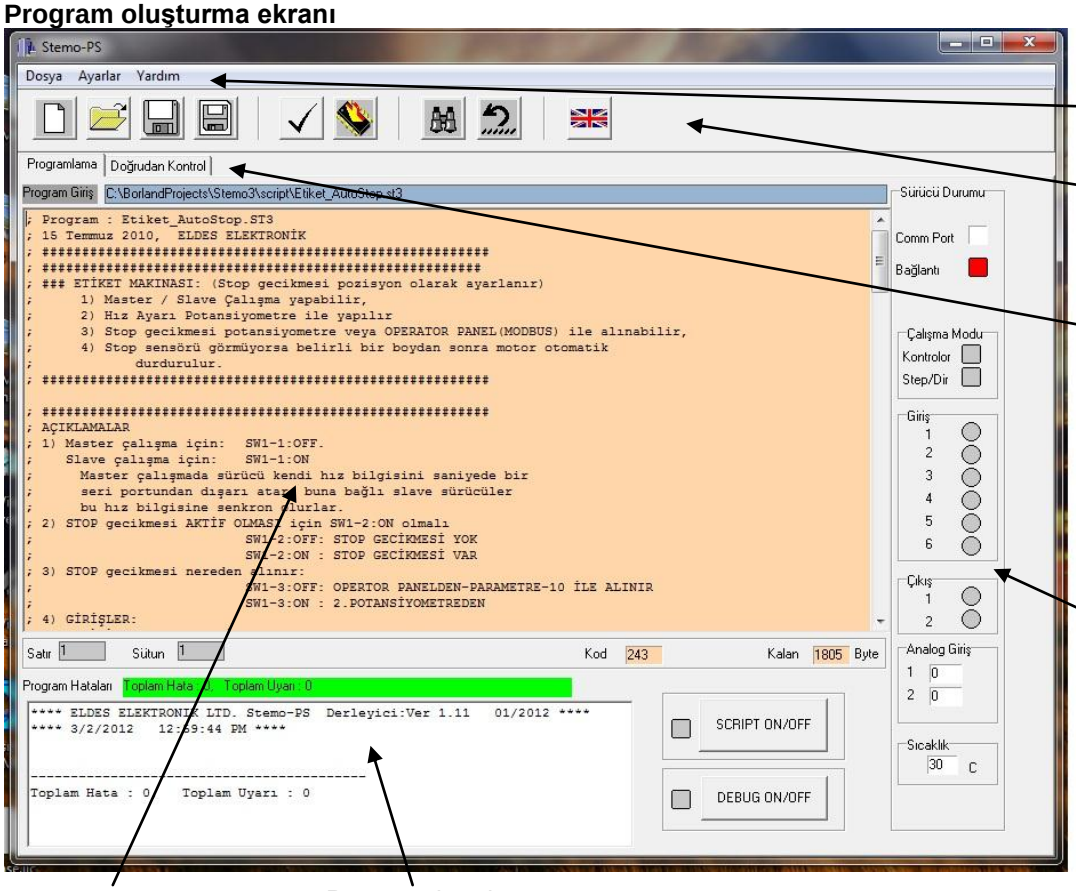
STEMO-PS, sürücüye program yüklemek veya sürücüyü gözlemek üzere geliştirilmiş Windows-XP veya Windows-Vista işletim sisteminde çalışabilen bilgisayar programıdır. Bu bilgisayar programı ile kullanıcı, uygulamasının gereğini yerine getiren programları geliştirir, hataları ayıklar ve sürücüye yükleyerek sürücünün kendi başına bir kontrolör olarak çalışmasını sağlar. STEMO-PS ile ayrıca sürücünün durumunu gözleyebilir, bazı parametrelerini değiştirebilir, motoru çalıştırır, durdurur, belirli step kadar döndürmesini yapabilir.

Arayüz programı sürücüye RS-232 bağlantısı veya USB ile bağlanır.
STEMO-PS menüleri Türkçe ve İngilizce olarak değiştirilebilir.

2.2 Ekranlar

STEMO-PS iki ana ekrana sahiptir: “Program Oluşturma” ekranı ve “Doğrudan kontrol” ekranı.

Program oluşturma ekranı

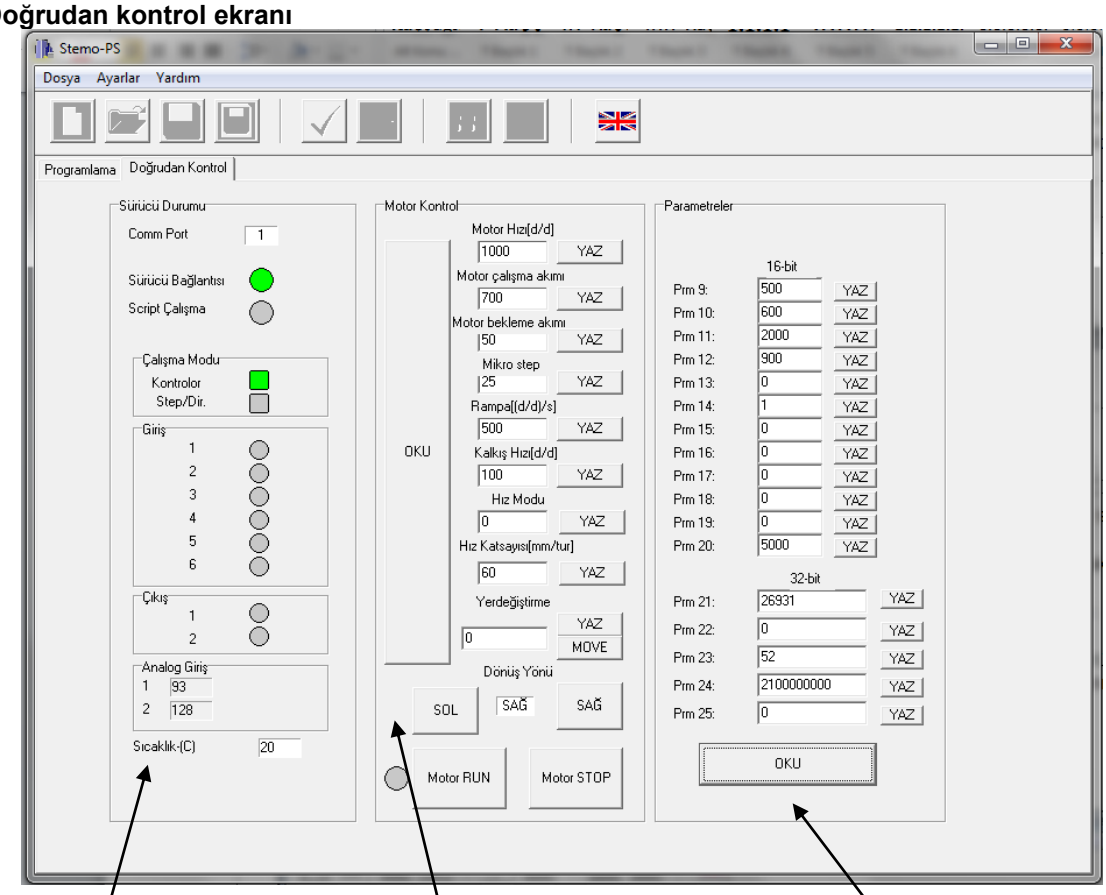


- Program yazma penceresi: Program dosyasının oluşturulduğu, komutların listesinin yazıldığı editor penceresidir. Komut satırları, program oluşturma kuralları içinde yazılır. Silme, kopyalama, bulma gibi eylemler yapılabilir. Enter tuşu ile yeni satıra inilir, ok tuşları veya fare imleci ile istenilen yere gelinebilir. Tab tuşu geçerlidir. Uzun program listelerinde kaydırma çubuğu (scroll bar) belirir. F1 tuşuna basıldığında imleçin bulunduğu yerdeki komut ile ilgili yardım penceresi açılır.
- Program hataları penceresi: Yazılan program kontrol edildiğinde yazım hataları, programlama hataları, uyarılar bu pencerede gösterilir. Hatanın kısa tanımı ve yazma penceresindeki satır numarası verilir. Bu pencerede hata satır numarasının olduğu yer üzerinde fare tek tıklanırsa program yazma penceresinde hatanın bulunduğu satır işaretlenecektir. Program listesinde hatalar var ise uyarı kırmızı renkli olacaktır. Hata yok ise uyarı yeşil renkli olacaktır ve sürücüye gönderme yapılabilir.
- Sürücü durumu : Sürücü ile bağlantının olup olmadığı, hangi comm port kullanıldığı, girişlerin ve çıkışların aktif olup olmadığı, analog çıkış değerleri ve sürücü sıcaklığı bu pencerede gösterilir. Ayrıca yazılmış ve sürücüye yüklenmiş geçerli bir program (script) “SCRIPT ON/OFF” butonu ile koşturulur veya durdurulabilir. Script çalışıyor ise buton yanındaki lamba yeşil olur. Sürücü üzerindeki SW1-4 anahtarının konumu Çalışma Modu penceresinde görülür.
- DEBUG butonu: Program geliştirme sırasında, program yüklenip çalıştırılırken istenirse DEBUG özelliği açılır. Bu sayede programın nerelerden geçtiği ve beklediği

gözlenmiş olur. Çalışan komutun hangisi olduğu program yazma penceresinde ilgili satırın renklendirilmesi ile belirtilir.

- Kısayol Butonlar : Programın kullanılması ile ilgili komutlar butonlar şeklinde düzenlenmiştir. Menü ile yapılacak işlemler ayrıca butonlar ile de yapılabilir.
- Ekran Seçimi : Ekran seçim sekmeleri tıklanarak iki ana ekran arasında geçiş yapılır.
- Komut kodu penceresi : Komut kodu penceresi doğru oluşturulmuş programın sürücüye gönderilecek komut kodunu gösterir. Bu ekran sadece gösterme amaçlıdır, kullanıcının yapması gereken bir ayarı yoktur.

Doğrudan kontrol ekranı



Sürücü durum penceresi

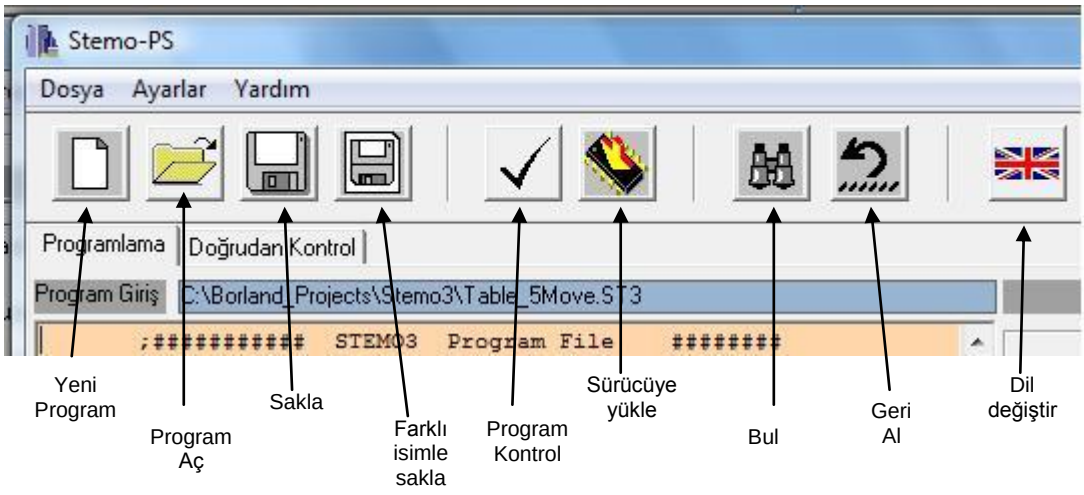
Motor kontrol penceresi

Parametreler penceresi

- Sürücü durumu : Sürücü ile bağlantının olup olmadığı, hangi comm port kullanıldığı, girişlerin ve çıkışların aktif olup olmadığı, analog çıkış değerleri ve sürücü sıcaklığı bu pencerede gösterilir. Ayrıca yazılmış ve sürücüye yüklenmiş geçerli bir program (script) "SCRIPT ON/OFF" butonu ile koşturulur veya durdurulabilir. Script çalışıyor ise buton yanındaki lamba yeşil olur. Sürücü üzerindeki SW1-4 anahtarının konumu Çalışma Modu penceresinde görülür.
- Motor kontrol penceresi : Motora ait ayarların göründüğü, motoru doğrudan kontrol etmeye yarayan butonların bulunduğu penceredir. "OKU" butonu ile motora ait parametrelerin tamamı okunup gösterilir. Herbir parametrenin yanındaki "YAZ" butonu ilgili parametreye yeni değer vermek için kullanılır.
  Motor hızı : devir/dakika[d/dak] cinsinden motorun dönüş hızını ayarlar.
  Motor çalışma akımı : 0-1000 aralığında motor faz akımını oransal değer olarak ayarlar. 1000 ⇒7 Amper.

Motor bekleme akımı : 0-100 aralığında motor faz akımının yüzdesi olarak motorun dönmediği ancak frende beklediği akım değeri.

Kısayol butonlar



3 Program Oluşturma

3.1 Programlama Genel İlkeleri : Program Listesi Biçimi

Program listesi, programlama diline ait program komutlarını içeren satırlardan oluşur. Program satırları ile ilgili genel kurallar aşağıda sıralanmıştır.

- Her bir komut ifadesi ayrı bir satırda bulunmalıdır.
- Ayrı satırda bulunan anlamlı komut ifadesi (komut kelimesi, işlemler ve parametreleri) sonunda mutlaka ";" (noktalı virgül) karakteri bulunmalıdır. Noktalı virgül bulunmayan satır, eğer etiket değil ise, hatalı satır olarak değerlendirilir.
- Program satırlarının en sonunda mutlaka "END;" komutu bulunmalıdır.
- Boş (hiç bir şey yazılmamış) satıra izin verilir.
- Satırda ";" karakterinden sonra yazılan karakterler değerlendirilmez. Dolayısı ile ";" 'den sonra açıklama ifadeleri yazılabilir.
- Hem etiket hem de komut ifadesi aynı satırda bulunabilir.
- Satır en sola dayalı olabilir.

3.2 Programlama Genel İlkeleri : Program Akış Kuralları

Programlama dili, Stemo3 için tanımlanmış komutlar kümesinin bir liste halinde yazılarak sıra ile icra edilmesi ilkesine dayanır. Burada dikkat edilecek en önemli şey, herbir komut satırındaki komutun yerine getirilmesinden sonra bir sonraki komut satırına geçileceğidir. Bir tür yorumlamalı program tekniği olarak nitelendirilebilir. Ancak komutun sürücüde başlatacağı işlemlerin süreçleri kısa veya uzun olabilir. Duruma göre, bir önceki komutun başlattığı süreç devam ederken yeni komutun başlatacağı yeni bir süreç olabilir. Bu anlamda paralel çalışan durumlar oluşabilir. Yeni komut bir önceki komutun süreçlerini bozabilir, değiştirebilir veya bitirebilir. Bunlar programlama sırasında dikkat edilmesi gereken durumlardır.

Program akışı ayrıntılarına bakılırsa; program listesinde sırası gelmiş komut çözümlenir, komuta göre sürücüde bazı işlemler başlatılır. Sürücüde bazı işlemlerin başlatılması ile ilgili satırdaki komutun görevi tamamlanmıştır ve yeni komuta geçilir. Komutların tiplerine göre komut satırındaki işlem sürücü tarafından anında icra edilir ve yeni komut satırına geçilirken bu komutun süreci tamamlanmıştır. Başka bir komut türünde ise sürücüde bir işlem başlatılmıştır ve devam etmektedir. Bu tür komutlarda sürücüde işlemlerin devam etmesi komutun tamamlanmadığı, yani program akışının bekleyeceği anlamı taşımaz. Komutun görevi o işlemi başlatmaktır ve başlatıldığı anda komut yerine getirilmiştir ve bir sonraki komut satırına geçilir.

Liste-1

```
...
SPEED = 1000;
CURON = 500;
MICROS = 16;
DISP = 1000000;
MOVE;
DELAY = 500;
WAIT
STOP;
...
```

Liste-2

```
...
SPEED = 1000;
CURON = 500;
MICROS = 16;
DISP = 1000000;
MOVE;
WAIS;
DELAY = 500;
WAIT
STOP;
...
```

Yukarıda anlatılan kavramlar, örnek iki program listesi üzerinde açıklansın. Liste-1 ve -2' de SPEED komutu sürücüde motor hızını 100.0 devir/dakika'ya ayarlayan komuttur. Bu komut icra edilirken hız değeri sürücüde ayarlanır ve ayarlama işlemi bittikten sonra bir sonraki satır olan CURON komutuna geçilir. Süreçlerin iç içe geçmesi durumu yoktur. Benzer şekilde sıradaki CURON, MICROS, DISP komutları için de durum aynıdır. Komutlar icra edildiklerinde ilgili parametre değerleri sürücüde oluşturulmuştur. MOVE komutu, DISP komutu tarafından verilmiş yerdeğiştirme değerine konumlama

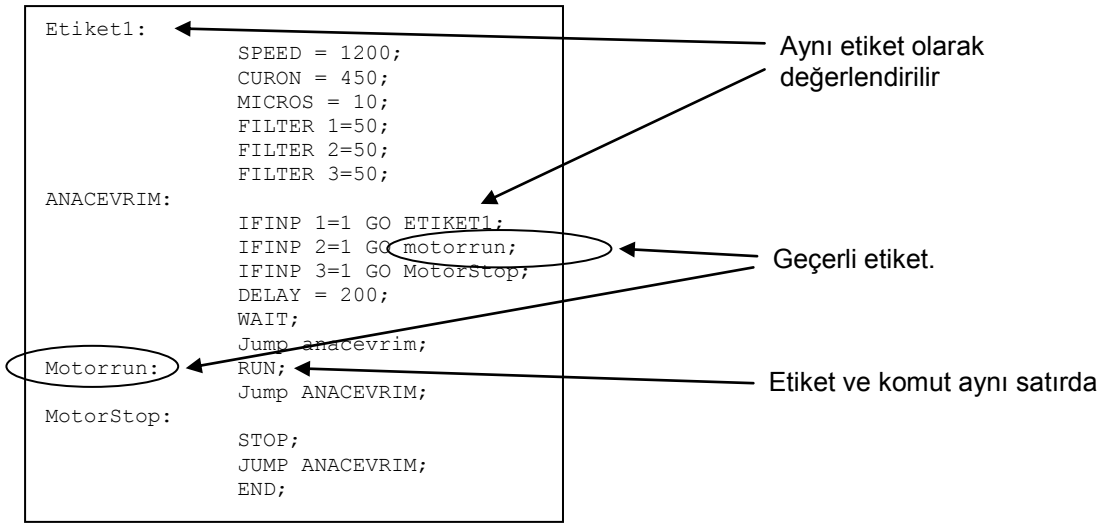
yapmak üzere motoru harekete geçiren komuttur. Komutun icra edilmesi ile motor harekete başlar ve bulunduğu yerden itibaren bir milyonuncu adıma gitmeye başlar. Doğaldır ki motor hızına bağlı olarak bu bir süre alacaktır. Ancak, motor haraketine başlamasıyla MOVE komutunun işi tamamlanmıştır ve bir sonraki satırın icrasına geçilir. Bu aşamada liste-1 ve liste-2 farklı davranış sergileyecektir.
Liste-1'de DELAY komutu ile 500 milisaniye gecikme bildirimi yapılır ve WAIT komutu ile bu sürenin geçmesi beklenir. WAIT komutu program akışını DELAY süresi kadar bekletecektir. Ancak hatırlanırsa motor MOVE komutu ile hareketine devam etmekte ve belirlenen konuma yaklaşmaktadır. Burada, MOVE komutu ile hareket eden motor ve WAIT komutu ile sürenin beklenmesi iç içe geçen süreçlerdir. WAIT komutu beklemesi tamamlanınca STOP komutu icra edilecek ve motor koşulsuz durdurulacaktır. Bu aşamada, motor MOVE komutu ile belirlenmiş konuma ulaşmamış olabilir ve konumlama yapılamaz.
Liste-2'de, MOVE komutundan sonra eklenen WAIS komutu iç içe geçen süreci engeller ve motorun konumlama yapmasını kesin olarak bekler, program akışını ilerletmez. Verilen konuma ulaşınca DELAY komutu ve WAIT komutu icra edilerek 500 milisaniye beklenir ve motor durdurulur.

3.3 Programlama Genel İlkeleri : Dallanma (Etiket)

Program listesi oluşturulurken uygulama gereği program akışının farklı satırlara yönlendirilmesi gereklidir. Bunun için program dallanma yeri belirlenmelidir ve etiket (label) kullanılır. Etiketler program akışını kontrol eden ve program satırları içine konan karakter dizileridir. Etiket olabilmesi için karakter dizilerinin sonunda mutlaka ":" (iki nokta üst üste) karakteri olmalıdır. Program içinde etiketlere dallanmak için komutun dallanma bilgisine etiket dizisi ":" kullanılmadan yazılır.
Etiketler ile ilgili kurallar:

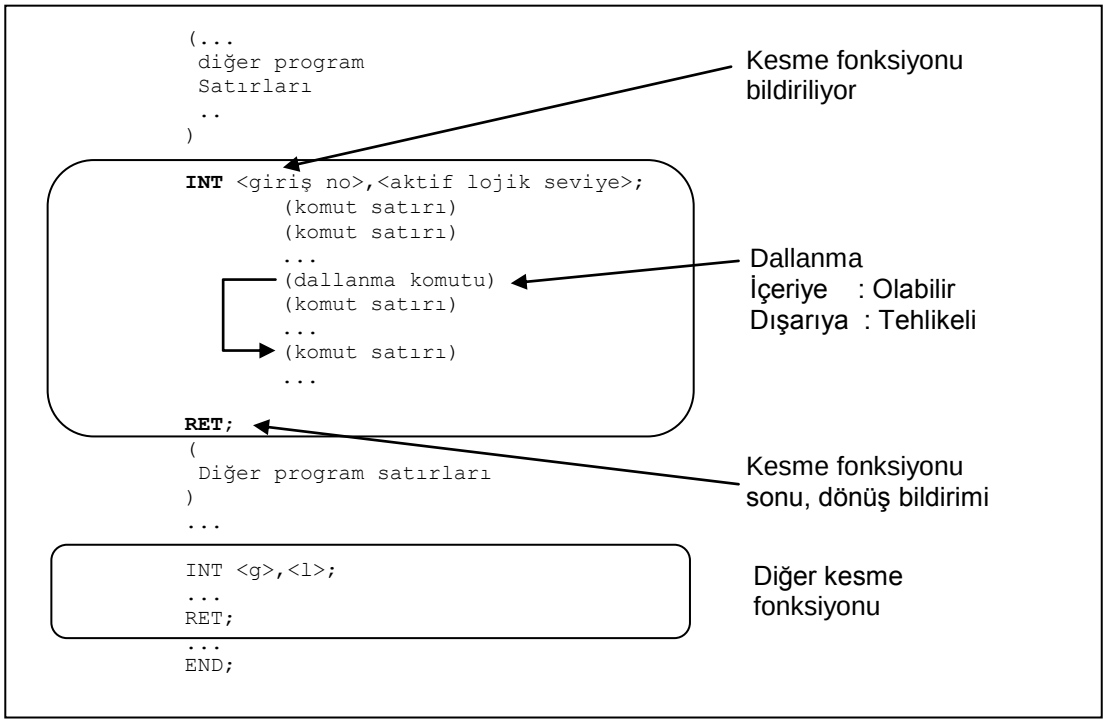
- Etiket karakter dizisinin en sonunda mutlaka ":" karakteri bulunmalıdır.
- Etiketteki karakter sayısı sınırsızdır.
- Harf veya rakam kullanılabilir, veya sadece rakam kullanılabilir.
- Boşluk karakteri ve '=' karaktari kullanılamaz.
- Büyük küçük harf farkı yoktur. Yazılan bütün etiketler sentaks kontrolünde büyük harfe çevrilerek değerlendirilir. Bu nedenle küçük ve büyük harf ile yazılmış aynı karakterli etiketler aynı etiket olarak değerlendirilir.
- Etiket dizisi ile komut ifadesi aynı satırda bulunabilir.

Etiket kullanım örnekleri :

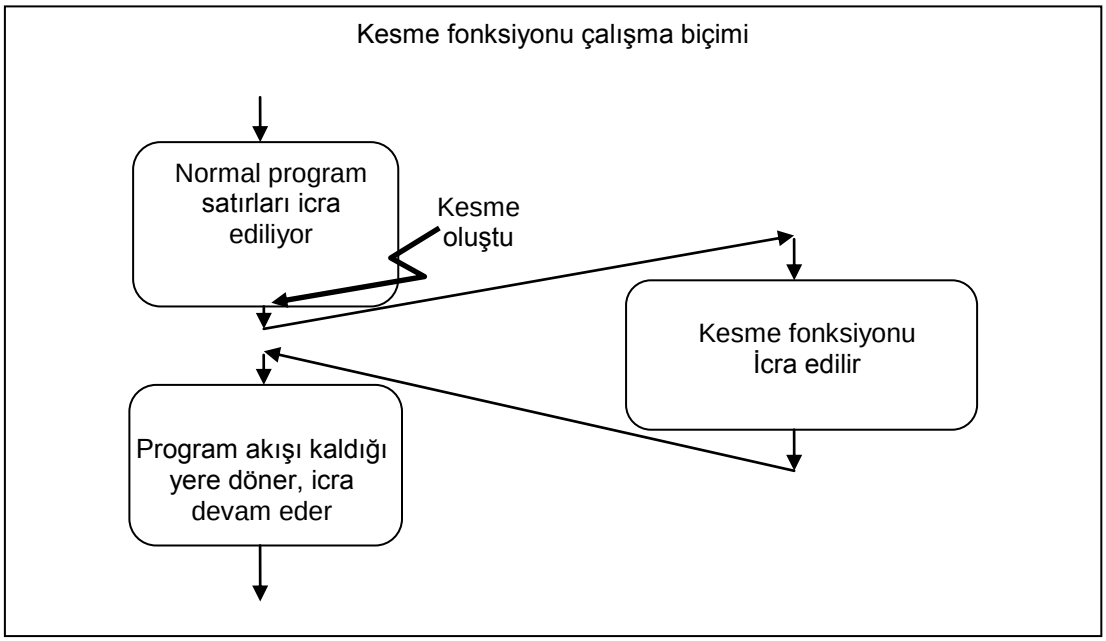


3.4 Programlama Genel İlkeleri : Kesme Fonksiyonları (Interrupt)

Stemo3 programlama dilinde lojik girişler ile ilişkilendirilmiş olarak kesme fonksiyonları tanımlanabilmektedir. Kesme için tanımlanmış giriş aktif olduğunda program normal akışını bırakır, kesme fonksiyonu adresine dallanır. Kesme fonksiyonu içindeki komut satırları icra edilir. RET komutu ile karşılaşınca kesme oluştuğu andaki normal program akışına geri döner. Kesme fonksiyonu program listesi içinde aşağıdaki biçimde tanımlanır;



Kesme fonksiyonu program listesi içinde herhangi bir yerde oluşturulabilir, ancak END satırından önceki yerlere tanımlanması daha uygun olur.



RET komutu iki şekilde oluşturulabilir:

1) RET;
2) RET <etiket>;

Birinci durumda, kesme fonksiyonu tamamlanınca program akışı dönüş yeri, kesmenin geldiği anda icra edilmekte olan komutun bir altındaki komuta dönülmektedir.
İkinci durumda, program akışı program listesinde <etiket> ile belirlenmiş noktaya yönlendirilir.

"INT" Fonksiyonunda dikkat edilecekler

- Kesme için tanımlanmış GİRİŞ, ikinci kez kesme işlemi için kullanılamaz.
- Kesme fonksiyonu için yazılan program satırlarında JUMP, IFINP, IFVRB gibi dallanma komutları ile programın herhangi bir yerine dallanılabilir. Bu nedenle programcı RET komutunun icra edilmesi konusunda dikkatli olmak durumundadır. Kesme programı kendi RET komutu ile sonlanmadığı takdirde yeni kesmeler oluşmaz.
- Kesme fonksiyonu icra edilirken baska kesme fonksiyonu tarafından kesilebilir. Fakat kendisi tarafından yeniden kesilemez.
- Farklı GİRİŞler için kesme programları ayrı ayrı tanımlanıp yazılmalıdır. INT bildiriminden sonra RET komutu görmeden yeniden INT bildirimi yapılamaz. INT bildirimi olmadan RET komutu olamaz.
- Kesme oluştuğu anda DELAY komutu ile devam etmekte olan gecikme var ise gecikme değerinin kalan süresi kayıt edilir, kesme fonksiyonu sonunda süre kaldığı yerden devam eder.

3.5 Aritmetik işlemler

Değişken kullanımında, çarpma (*), bölme (/), toplama (+) ve çıkarma (-) olmak üzere temel aritmetik işlemler yapılabilmektedir. Temel işlemler sadece VRB komutu ile atama yapılmasında kullanılır. VRB komutu ile işlem sonucu alındıktan sonra değişken içindeki değer parametre ve tablo değerlerine aktarılabilir.

Kullanım biçimi aşağıdaki gibidir.

VRB <z> = <operand1> <işlem> <operand2>;

z={1,...,20,21,..,25}. 1 ile 20 arası değişken 16-bit sayı alır, 21 ile 25 arası değişken 32-bit sayı alır.

<operand1> ve **<operand2>** için aşağıdakiler geçerlidir:

Nümerik sayı : 32-bit (Long) işaretsiz sayı. 0 ile 4294967295 arası.
V<n> : Değişken içeriği. n={1,..,20,21,..,25}. 1 ile 20 arası değişken 16-bit sayıdır, 21 ile 25 arası değiken 32-bit sayıdır.
P<n> : Parametre içeriği. n={1,..,20,21,..,25}. 1 ile 20 arası parametre 16-bit sayıdır, 21 ile 25 arası parametre 32-bit sayıdır.
T<m> : Tablo içeriği. m={1,..,32}. Tablo değeri 32-bit sayıdır.
A<k> : Analog kanal değeri. k={1,2}. Analog değer 8-bit sayıdır.

<işlem> için aşağıdakiler kullanılır:

\+ : operand1 ve operand2 32-bit sayı olarak toplanırlar. Sonuç 32-bit sayıdır.
\- : operand2, operand1'den çıkartılır. Sonuç 32-bit sayıdır.
* : operand1 ve operand2 çarpılır. Sonuç 32-bit sayıdır.
/ : operand1, operand2 ile bölünür. Sonuç 32-bit sayıdır.

Aritmetik işlem 32-bit işaretsiz sayı olarak yapılır ve taşma kontrolu yapılmaz. İşlem sonunda oluşan değer yine 32-bit sayı olarak tutulur. **Sayıların sonucu 32-biti aşması durumunda hatalı değerler elde edilir. İşleme giren sayıların sınırları üzerinde programcı dikkatli olmalıdır**.

Aritmetik işlemleri kullanma yöntemleri

Bir değişken ile nümerik sayının işlemleri:

VRB 3 = V12 * 3000; veya
VRB 3 = 3000 * V12; Değişken-12 içeriği ile 3000 çarpılır. Sonuç değişken-3'e aktarılır.
VRB 3 = V12 + 3000; veya
VRB 3 = 3000 + V12; Değişken-12 içeriği ile 3000 toplanır. Sonuç değişken-3'e aktarılır.
VRB 3 = V3-10; Değişken-3 içeriğindenden 10 çıkartılır. Sonuç yine değişken-3'e aktarılır.

Bir değiken ile bir parametrenin işlemi:

VRB 23 = V5 * P11; Parametre-11 ile değişken-5 çarpılır, sonuç değişken-23'e atanır.
VRB 3 = P10-V25; Parametre-10'dan değişken-25 çıkartılır, sonuç değişken-3'e atanır.

Analog kanal değeri ile işlemler:

VRB 2 = A1*100; Analog kanal-1 değeri ile 100 sayısı çarpılır, sonuç değişken-2'ye atanır.

Ardışıl işlemler:

A = (B*C)/D + E şeklinde bir işlem yapılacak ise birkaç satır arka arkaya kullanılarak yapılmalıdır. Örnek: Parametre-21 değeri, değişken-3 ile tablo-5 değerinin çarpılıp 100'e bölündükten sonra analog kanal-2 değeri ile toplanarak elde edilecektir. Burada A=Parametre-21, B=Değişken-3, C=Tablo-5, D=100 ve E=Analog kanal-2 olmaktadır. Program satırları:

```
VRB 1 = V3*T5;      //değişken-1 ara işlem elemanı olarak kullanılır. Değişken-1 = B*C.
VRB 1 = V1 / 100;   //değişken-1 = (B*C)/D.
VRB 1 = V1 + A2;    //değişken-1 = (B*C)/D+E.
PRM 21 = V1;        //parametre-21=(B*C)/D+E.
```

Komut Grupları Özet Tablo

Komut	Sentaks	Parametre	Kod	Açıklama
Motor Parametreleri İle İlgili Komutlar				
CURON	CURON = <n>;	1 ≤ n ≤1000	0x02	Motor çalışma akımı.
CUROFF	CUROFF= <n>;	1 ≤ n ≤ 100	0x0a	Motor frende bekleme akımı (%CURON).
MICROS	MICROS = <n>;	n={ 1,2,4,5,8,10,16, 25 }	0x03	Motor mikrostep seçimi
Hareket İle İlgili Komutlar				
SPEED	SPEED = <değer>;	0 ≤ değer ≤15000	0x04	Motor hızı: Doğrudan değer
	SPEED = V<m>;	m = {1,2,...,24,25}	0x0d	Motor hızı: Değişkenden
ACCEL	ACCEL = <n>;	1 ≤n ≤ 9999 m = {1,2,...,24,25}	0x05	Rampa katsayısı. Saniyedeki dev/dakika değişimi [(dev/dak)/sn]
	ACCEL = V<m>;		0x38	Rampa katsayısı değişkenden alınır.
INITV	INITV = <n>;	0 ≤n ≤15000	0x06	Motor kalkış hızı- [dev/dak]
DIR	DIR = <n>;	n = {0,1}	0x08	Motor dönüş yönü seçme.
RUN	RUN;		0x16	Motor sürekli çalıştırma.
STOP	STOP;		0x17	Motor koşulsuz durdurma.
MOVE	MOVE;		0x15	Motor belirli adım ilerletme.
MOVT	MOVT <n>;	n = {1,2,...,31,32}	0x1b	Motor tablo indisine göre belirli adım ilerletme : indis doğrudan verilir.
	MOVT V<m>;	m = {1,2,...,24,25}	0x1c	Motor tablo indisine göre belirli adım ilerletme : indis değişken ile verilir.
WAIS	WAIS;		0x14	Belirli stepe gidişte bekleme.
DISP	DISP = <n>;	0 ≤ n ≤ 2000000000	0x07	Pozisyon değiştirme miktarı .
	DISP = V<m>;	m = {1,2,...,24,25}	0x29	Pozisyon değiştirme değişkenden.
REFPOS	REFPOS;		0x1a	Referans (sıfır) konumu tanımlama
LOCATE	LOCATE <değer>;	0 ≤ değer ≤ 4,294,967,295 n = {1,2,...,31,32}	0x34	Mutlak pozisyonlama : Pozisyon değeri doğrudan verilir.
	LOCATE V<n>;		0x35	Mutlak pozisyonlama : Pozisyon değeri değişkenden alınır.
Program Akışı Kontrol Komutları				
JUMP	JUMP <label>;		0x09	Programda dallanma.
IFINP	IFINP <i> = <lojik değer> GO <label>;	i = {1,2,3,4,5,6} lojik değer = {0,1}	0x0b	Giriş durumuna göre dallanma. IFINP 2=1 GO label1;
IFVRB	IFVRB <n> <koşul> <değer> GO <label>;	n = {1,2,...,24,25} koşul = {=,<,>} 0 ≤ değer ≤ 65535	0x0c	Değişken ile sabit karşılaştırılması ve dallanma. IFVRB 3<100 GO label3;
	IFVRB <v> <koşul> V<n> GO <label>;		0x32	Değişken ile değişken karşılaştırlması ve dallanma IFVRB 12 > V21 GO LABEL10;
Lojik Giriş/Çıkış Komutları				
OUT	OUT <n>=<lojik değer>;	n = {1,2} lojik değer = {0,1}	0x23	Lojik çıkış; OUT 1;
FILTER	FILTER <n>=<değer>;	n = {1,2,3,4,5,6} 0 ≤ değer ≤ 250	0x26	Girişler için filtreleme FILTER 1 = 50;
INT	INT <m>, <lojik durum>;	m = {1,2,3,4,5,6} lojik durum = {0,1}	0x1d	Interrup(Kesme) fonksiyonu tanımlama.
RET	RET;		0x1e	Interrupt dönüş komutu(program kaldığı yere)
	RET <lsbel>;			Interrupt dönüş komutu(programda etiket yerine)
Atama Komutları				
VRB	VRB <n> = <değer>;	n = {1,2,...,24,25} m = {1,2,3,4,5,6} c = {1,2} s = {1,2,3,4,5,6,7,8} k = {1,2,...,31,32}	0x0e	Değişkene doğrudan değer atama.
	VRB <n> = I<m>;		0x18	Değişkene girişten değer atama.
	VRB <n> = A<c>;		0x20	Değişkene analog değer atama – 8 bit.
	VRB <n> = S<s>;		0x2a	Değişkene ayar anahtarı değeri atama.
	VRB <n> = P<n>;		0x2b	Değişkene parametre değeri atama.
	VRB <n> = Q;		0x2d	Değişkene motor hareket durumu atanır.
	VRB <n> = T<k>;		0x31	Değişkene tablodan değer atanır.
	VRB <n> = X;		0x33	Değişkene o andaki mutlak pozisyon atanır.
	VRB <n> = B<c>;		0x36	Değişkene analog değer atama – 10 bit
	VRB <n> = V<n>;		0x39	Değişkene değişkenden atama
	VRB <n> = W;		0x3a	Değişkene Delay/wait status atanır. 0:bitti, 1:bitmedi
VRBINC	VRBINC <n>, <değer>;	n = {1,2,...,24,25} 0 ≤ değer ≤ 65535	0x0f	Değişken değer artırma.
VRBDEC	VRBDEC <n>, <değer>;	n = {1,2,...,24,25} 0 ≤ değer ≤ 65535	0x10	Değişken değer azaltma.
PRM	PRM <n> = <değer>;	n = {1,2,...,24,25} 0 ≤ değer ≤ 65535	0x11	Parametreye değer atama.
	PRM <n> = V<n>;		0x2c	Parametreye değişkenden değer atama.
LIM	LIM <n>=<değer>;	n = {1,2,...,24,25} 0 ≤ değer ≤ 65535	0x1f	Değişken limit değer tanımlama.
SPAN	SPAN <n> = <değer>;	n = {1,2,...,19,20} 0 ≤ değer ≤ 65535	0x21	Değişken tam skala değer tanımlama.
TABLE	TABLE <n> = <değer>;	n = {1,2,...,19,20} m = {1,2,...24,25} 0 ≤ değer ≤ 4,000,000,000	0x19	Tablo oluşturma, tabloya doğrudan değer atama
	TABLE <n> = P<m>;		0x2e	Tabloya parametreden değer atama
	TABLE <n> = V<m>;		0x30	Tabloya degiskenden deger atama
ENBINT	ENBINT <m>;	m={1,2,3,4,5,6}	0x24	Iligili giriş için interrupt fonksiyonuna izin verilir.
DISINT	DISINT <m>;	m={1,2,3,4,5,6}	0x25	İlgili giriş için interrupt fonksiyonu yasaklanır.

Komut Grupları Özet Tablo (Devam)

Komut	Sentaks	Parametre	Kod	Açıklama
Zamanlama – Gecikme Komutları				
DELAY	DELAY = <değer>;	0 ≤ değer ≤ 65535 n = {1,2,...,19,20}	0x12	Gecikme zamanı doğrudan.
	DELAY = P<n>;		0x22	Gecikme zamanı parametreden.
	DELAY = V<n>;		0x27	Gecikme zamanı değişkenden.
WAIT	WAIT;		0x13	Gecikme zamanı bekletme.
Aritmetik İşlemler				
VRB	VRB <n> = <op1><islem><op2>;	n = {1,2,...,24,25} op1,op2= {V<n>,P<n>,T<n>,A<n>, 32-bit sayı} islem = {+, -, *, /}	0x2f	Operand1 ve operand2 arasında aritmetik işlem yapılır, sonuç ilgili değişkene atanır.

Not : <n>, <değer> gibi ifadeler komuta verilecek değişik parametreleri, sayıları anlatmak için kullanılır. Program listesine yazılırken < , > karakterleri yazılmaz.

4 Komut Açıklamaları

4.1 CURON : Motor Çalışma Akımı - [0x02].

Sentaks : CURON = <n>;
<n> : 1 ile 1000 arası bir sayı.
Örnek : CURON = 750;

Açıklama :
CURON komutu, motorun hareketli durumlarında faz sargısından geçirilecek sinüs biçimli akımın tepe değerini tanımlar. Oransal büyüklüktür. 1000 en yüksek akımı belirler. Sargılardan geçen akım, sürücü donanımındaki referans akım sinyali ve bu komutun belirlediği oransal değer çarpılarak oluşturulur. Sargıdan geçen gerçek akımın tepe değeri aşağıdaki bağıntı ile hesaplanır :

Gerçek sargı akımı = (0.34 * n)/(1000 * R)

R : Sürücü kartında kullanılan akım ölçme şönt direnci. (Genellikle 0.05 ohm)

4.2 CUROFF : Motor Frende Bekleme Akımı - [0x0a].

Sentaks : CUROFF = <n>;
<n> : 0 ile 100 arası bir sayı.
Örnek : CUROFF = 50;

Açıklama :
CUROFF komutu, motorun frende bekleme anlarında faz sargısından geçirilecek akımın yüzde (%) olarak ne oranda azalacağını belirtir. Oranlama, motor CURON komutu ile verilmiş akıma uygulanır. Motor bekleme durumundayken sargılardan daha az akım geçmesine rağmen tutma kuvveti (tork) yeterince büyük olur. Bu nedenle bekleme durumlarında gereksiz enerji harcamayı azaltmak için bekleme akımı daha küçük değerlere ayarlanabilir. Böylece motorun ve sürücünün gereksiz ısınması da engellenmiş olur. Oransal büyüklüktür. 100 değeri akım azaltılmayacağını, sıfır değeri ise akımın kesileceğini belirtir. Sargılardan geçen akım, sürücü donanımındaki referans akım sinyali ile bu komutun belirlediği oransal değer çarpılarak oluşturulur. Sargıdan geçen gerçek akımın tepe değeri aşağıdaki bağıntı ile hesaplanır :

Gerçek sargı akımı = [(0.34 * CURON)/(1000 * R)] * n / 100

R : Sürücü kartında kullanılan akım ölçme şönt direnci. (Genellikle 0.05 ohm)

4.3 MICROS : Mikrostep Seçme - [0x03].

Sentaks : MICROS = <n>;
<n> : 1, 2, 4, 5, 8, 10, 16, 25 değerlerini alabilen sayı.
Örnek : MICROS = 10;
Açıklama :
Mikrostep, step motorun hareketi sırasında motor tam adımlarını belirli kesirlere bölerek daha hassas pozisyonlamayı ifade eder. N değeri, motor tam adımının kaç parçaya bölüneceğini belirler. Örneğin n, 5 olarak verilsin; Motorun tam adımı 5 parçaya bölünmüştür ve motora tam adım attırmak için pulse sayısı (step sayısı) 5 olmalıdır. 1.8 derecelik step motor için motorun bir turunda 200 tam adım vardır. Motoru bir tam tur döndürmek için 5*200 = 1000 pulse (step sayısı) vermek gereklidir. Bu ise motorun tam turunun 1000 parçaya bölündüğünü ifade eder.

Uyarı: Step motorların yapım özellikleri en kararlı pozisyonlamanın tam adımlarda olmasını gerektirmektedir. Mikrostep çalışma sırasında seçilen bölme oranlarında kusursuz pozisyonlama mümkün olmayabilir. Bunun nedeni motorun mekanik yükünün motoru tam adımlara doğru yaklaştırmaya zorlamasıdır. Yük ne kadar büyük olur ise mikrostep pozisyonlarından sapmalar da o kadar fazla olacaktır.

4.4 SPEED : Motor Hızı – [0x04], [0x0d].

Sentaks : İki şekilde atama yapılabilir
1) SPEED = <değer>; veya
2) SPEED = V<n>;
<değer> : 1 ile 15000 arası sayı.
<n> : Değişken numarası. 1 ile 25 arası sayı.
Örnek : SPEED = 1450; Motor hızını 145.0 d/dak'ya ayarlar.
SPEED = V13; Motor hızı 13 numaralı değişkenin değerine ayarlanır.
Açıklama :
SPEED komutu, motor çalışma hızını ayarlar. Verilen değer, devir/dakika [d/dak] cinsinden ve 10 sayısı ile çarpılmış olarak girilmelidir. Örneğin motor hızı 150 devir/dakika olarak ayarlanmak istenirse girilecek sayı 1500 olmalıdır.
Program içinde motor hızı iki şekilde ayarlanabilir; birinci şekilde doğrudan hız değeri ataması yapılır, ikinci şekilde ise hız değeri "Vn" program değişkeninden alınır.
DİKKAT! 1 ile 20 arası değişken 16-bit sayıyı, 21 ile 25 arası değişkenler 32-bit sayıyı temsil eder.

4.5 ACCEL : Motor ivmelenme katsayısı (Rampa) – [0x05];

Sentaks : ACCEL = <n>;
<n> : İvmelenme katsayısı. 1 ile 5000 arası bir sayı.
Örnek : ACCEL = 300;
Açıklama :
İvmelenme katsayısı, motor hareketinin rampalı bir şekilde çalışmasında rampa ayarını değiştirir. İvmeli hareket, motor duruyorken ve sabit bir hızda duruyorken, ayarlanmış yeni hıza aniden yükselmeyip kademeli olarak yükselmesi, motor duruyorken de aniden durmayıp kademeli olarak yavaşlayıp durmasıdır. Verilen sayı motor tam adımları (full step) cinsinden ayarlanmış hıza ulaşma zamanıdır.

4.6 INITV : Motor Kalkış Hızı – [0x06].

Sentaks : INITV = <n>;
<n> : Motor kalkış hızı. Devir/dak cinsinden motorun duruyorken harekete başlayacağı ilk hızı belirler.
Örnek : INITV = 100;
Açıklama :
INITV komutu, motor hareketine belirli bir ilk hızdan başlamasını sağlar. Motor durmakta iken dönüş hareketine başlayacak ise rampalı hareket yapacaktır. Hareketine başlarken sıfır hızdan başlamak

yerine belirli bir hızdan başlaması istenebilir. Bu durmda INITV ile belirtilen hız kullanılacaktır. Verilen değer [devir/dakika] cinsinden sayının 10 ile çarpılmışı şeklinde verilir. 0 ile 15000 arası sayı girilebilir.

4.7 DISP : Pozisyon yerdeğiştirme miktarı – [0x07] , [0x29].

Sentaks : İki şekilde atama yapılabilir.
1) DISP = <n>;
2) DISP = V<m>;
<n> : Yer değiştirme miktarı. 0 ile 4000000000 arası sayı olabilir.
<m> : Değişken numarası. 1 ile 25 arası sayı.
Örnek : DISP = 16000;
Açıklama :
DISP komutu, konumlama yapmak amacı ile step sayısı olarak motor milinin yerdeğiştirme miktarını belirler. Verilen sayı motor dönüş yönünden bağımsızdır. Motor MOVE komutu ile hareket ettirildiğinde dönüş yönünde DISP ile belirtilen step sayısı kadar döner ve verilen sayıya ulaşınca durur. DISP değeri mikrostep ayarı dikkate alınarak kullanılır. Örneğin DISP değeri 200 olarak belirlensin. Mikrostep değeri 1 olarak ayarlanmış ise MOVE komutu ile motor tam bir tur atar(1.8 derecelik step motorlar için) , konuma ulaşır ve durur. DISP değeri aynı kalmak koşulu ile mikrostep ayarı 4 seçilir ve MOVE komutu çalıştırılır ise bu kez motor mili çeyrek tur dönecektir.
DİKKAT! 1 ile 20 arası değişken 16-bit sayıyı, 21 ile 25 arası değişkenler 32-bit sayıyı temsil eder.

4.8 DIR : Motor dönüş yönü – [0x08].

Sentaks : DIR = <n>;
<n> : Dönüş yönü. 0 veya 1 olabilir.
Örnek : DIR = 1;
Açıklama :
Motor dönüş yönünü değiştirmek için kullanılır. Değer 0 iken motor bir yöne döner, değerin 1 yapılması dönüş yönünü tersine çevirir.

4.9 JUMP : Program Koşulsuz Dallanma – [0x09].

Sentaks : JUMP <etiket>;
<etiket> :
Örnek : JMP L32;
Açıklama :
JUMP komutu ile program akışı, program listesinde <etiket> ile belirtilen yere yönlendirilir. <etiket> kuralları için Bölüm 3.3'e bakınız.

4.10 IFINP : Lojik Giriş Sinyali ile Program Koşullu Dallanma – [0x0b].

Sentaks : IFINP <i>= <logic durum> GO < etiket> ;
<i> : Giriş numarası. 1 ile 6 arası değer alır.
<lojik durum> : İlgili girişin o andaki durumu. Giriş aktif ise 1, aktif değil ise 0 ile tanımlanır.
< etiket > : Program listesi içinde tanımlanmış geçerli etiket kelimesi. Büyük-küçük harf duyarlılığı yoktur. <etiket> kuralları için Bölüm 3.3'e bakınız.

Örnek : IFINP 3 = 0 GO L99;

Açıklama :
IFINP komutu, program akışını girişlerden gelen bilgiye göre belirli bir satıra yönlendirilmesini sağlar. Giriş numarası ve ilgili girişin durumu koşulu tanımlar. Koşul sağlanıyor ise <etiket> ile belirtilen satıra atlanır, aksi halde program akışı bir alt satırdan devam eder.

4.11 IFVRB : Değişken değerine göre Program Dallanma – [0x0c], [0x32].

Sentaks :

1) IFVRB <n> < koşul operatörü ><değer> GO <etiket>;
2) IFVRB <n> < koşul operatörü > V<n>

Örnek : IFVRB 17>33500, L99;
<n> : Değişken numarası, 1 ile 25 arası değer alabilir.
<koşul operatörü> : (=), (<) veya (>) işlemlerinden biri koşul olarak verilir.
<değer> : 16 bit sayı. 0 ile 65535 arası sayı yazılabilir.
< etiket> : Program listesi içinde tanımlanmış geçerli etiket kelimesi. Büyük-küçük harf duyarlılığı yoktur. <etiket> kuralları için Bölüm 3.3'e bakınız.
Açıklama :
IFVRB komutu, program akışını kullanılan değişkenlerden birine göre belirli bir satıra yönlendirmesini sağlar. Karşılaştırma işlemi, i)bir değişken içeriği ile sabit sayı arasında ve ii)değişken içeriği ile başka bir değişken içeriği arasında yapılabilir. n ile karşılaştırma yapılacak değişken tanımlanır. Koşul sağlanıyor ise <etiket> ile belirtilen program satırına atlanır, aksi halde program akışı alt satırdan devam eder. <etiket> kuralları için Bölüm 3.3'e bakınız.
Değer 16-bit sayıdır. 1 ile 20 arası değişkenler 16-bit sayı, 21 ile 25 arası değişkenler 32-bit sayı temsil ederler.

Değişkenler ile ilgili daha fazla bilgi için VRB komutlarına bakınız.

4.12 VRB : Değişkene Değer Atama – [0x0e] , [0x18], [0x20], [0x2a], [0x2b] .

Sentaks : Aşağıdaki belirtilen biçimlerde atama yapılabilir.

1) VRB <n> = <değer>; [0x0e] : Doğrudan değer atanır.
2) VRB <n> = I<m>; [0x18] : Lojik girişlerin durumu değer olarak atanır.
3) VRB <n> = A<c>; [0x20] : Analog kanaldan değer alınır.
4) VRB <n> = S<s>; [0x2a] : Ayar anahtarları değeri alınır.
5) VRB <n> = P<n>; [0x2b] : Parametre değeri değişkene alınır.
6) VRB <n> = Q; [0x2d] : Motor durumunu değişkene alır.
7) VRB <n> = T<k>; [0x31] : Tablo değeri değişkene alınır.
8) VRB <n> = X; [0x31] : O andaki mutlak pozisyon değeri değişkene alınır.
9)

<n> : Değişken numarası. 1 ile 25 arası sayı olmalıdır.
<m> : Dikkate alınacak lojik girişlerin sayısı. 1 ile 6 arası olabilir.
<c> : Analog kanal numarası. 1 ile 2 olabilir.
<s> : Ayar anahtarı numarası. 1,2,3,4,5,6,7,8 değerlerini alabilir.
<k> : Tablo indis değeri. 1 ile 32 arasında değer alabilir.
1 : SW1-1 numaralı anahtarı,
2 : SW1-2 numaralı anahtarı,
3 : SW1-3 numaralı anahtarı,
4 : SW1-4 numaralı anahtarı,
5 : SW2-1 numaralı anahtarı,
6 : SW2-2 numaralı anahtarı,
7 : SW2-3 numaralı anahtarı,
8 : SW2-4 numaralı anahtarı temsil eder.
<değer> : 16-bit sayı veya 32-bit sayı.

Örnek :
VRB 12 = 12500; //12 numaralı değişkene 12500 atanır.
VRB 12 = I4; //12 numaralı değişkene girişlerden ilk 4 tanesinin durumu ikili sayı olarak atanır.
VRB 12=A1; //12 numaralı değişken 1 nolu analog girişten değer alır.

Açıklama :
VRB komutu, program içinde bazı değerleri saklayıp başka yerlerde kullanmayı sağlayan değişkenlere değer atamak için kullanılır. 25 farklı değişken kullanılabilir.
DİKKAT! 1 ile 20 arası değişken 16-bit sayıyı, 21 ile 25 arası değişkenler 32-bit sayıyı temsil eder.
Atama türleri:

- Birinci türde doğrudan sayı değeri değişkene atanır. 1 ile 20 arasındaki değişkene 16-bit sayı, 21 ile 25 arasındaki değişkene 32-bit sayı doğrudan atanabilir.
- I<m> : İkinci tür atamada ise lojik girişlerden yararlanılır. Lojik girişlerden, m ile belirtilen sayıda giriş dikkate alınarak temsil ettiği sayı karşılığı değişkene ikili sayı olarak atanır. Girişlerin ikili sayı olarak değerlendirilip işlem yapılması az giriş sayısı ile daha fazla durum yaratma imkanı sağlar. En düşük anlamlı bit, Giriş-1'e denk gelir. m, 1 ile 6 arasında sayı olabilir. Böylece girişler kullanılarak 1-bit ile 6-bit uzunluğunda sayı değeri değişkene atanabilir. Kullanım biçimine örnek olarak ikili sayı içeriği değişkene alındıktan sonra MOVT komutu kullanılarak tablo ile belirlenmiş yere konumlama yapılabilir.
- A<c> : Üçüncü tür atamada analog sinyal seviyesi kullanılır. Analog sinyal ile atama işlemi sürücünün bazıçalışma parametrelerinin/değişkenlerinin dışarıdan standart sinyal ile ayarlanmasını sağlar. Analog kanaldaki seviye analog dijital çeviricide sayıya dönüştürülür. Bu sayı her zaman 0 ile 255 arasında bir sayıdır. Tam skala, yani %100 değeri, 255 sayısı (8-bit) ile temsil edilmektedir. Seçilen analog kanalın o andaki (atama anındaki) sayısal değeri ilgili değişkene atanmadan önce, değişken için SPAN komutu ile belirlenmiş sayı ile oranlanır. Çıkan sonuç ilgili değişkene atanır. Bu oranlama işlemi, ham analog sayının sürücüde karşılığını bulacak fiziksel büyüklüğe dönüştürme işlemidir. Analog atama kullanılırken ilgili değişken için SPAN komutu ile anlamlı bir büyüklük verilmiş olmalıdır. SPAN komutu ile belirlenmemiş değişkenlerin tam skala değerleri 255 olarak ayarlıdır.
  Örnek: ikinci analog kanal motor hız ayarı için kullanılacak. Motor hızı için 1000 devir/dakika tam skala(span) olarak seçilecek. Bu işlemi yapan program satırları aşağıdaki gibi olabilir.

```
        ...<başka satırlar>
        SPAN 1=10000;          //1 nolu değişken tam skala değeri: 1000.0 d/dak, 10000 olarak girilir.
        ...<başka satırlar>
L 1;
        ...<başka satırlar>
        VRB  1=A1;             //2.analog giriş 1 nolu değişkene atanır. Oranlama otomatik yapılıyor.
        SPEED = V1;            //motor hızı 1 nolu değişkenden alınıyor.
        ...<başka satırlar>
        JUMP L1;               //program ana döngüsü içinde atama işlemi tazelenir.
```

- S<s> : Dördüncü tür atamada sürücünün 8 adet ayar anahtarının durumu değişkene alınabilir. Ayar anahtarı OFF konumda ise değişkene 1, ON konumda ise 0 değeri atanacaktır. Ayar anahtarları SW1-4 hariç tutulmak üzere sürücünün farklı çalışma biçimlerinde programlanmasını sağlayan bir bilgi girişi olarak kullanılabilir. Örneğin, SW1-2 girişi analog sinyallerin değerlendirilip değerlendirilmeyeceğini belirlemek üzere kullanılabilir. **Programlama konumunda çalışmak için SW1-4 anahtarının mutlaka OFF konumunda olması gerektiğini unutmayınız**.
- P<n> : Beşinci tür atamada değişkene parametrelerden birinin değeri atanır. Parametreler kalıcı hafızada saklandığı için sürücünün enerjisinin kesilip tekrar başlatılması durumunda parametrelerde tutulan bazı sistem ayarlarının program içinde uygun bir şekilde kullanılmasına olanak sağlanmış olur.
- Q : Altıncı tür atamada motor durumu değişkene atanır. Motor duruyor ise değişene sıfır, motor dönüyor ise 1 sayısı atanır.
- T<k> : Yedinci tür atamada 32 tablo değerinden birinin içeriği değişkene atanır. Tablo içeriği her zaman 32-bit olmaktadır.
- X : Mutlak pozisyon değeri sürücü içinde tutulmaktadır. Bu değer X ataması ile herhengi bir değişkene alınabilir. Mutlak pozisyon 32-bit işaretli sayıdır. Atama işlemi yapılırken 16-bit değerlikli değişken kullanılır ise pozisyon değerinde kayıp olabileceği göz önünde bulundurulmalıdır.

4.13 VRBINC : Değişken Değer Artırma – [0x0f].

Sentaks : VRBINC <n>, <değer>;
Örnek : VRBINC 12 , 3;
<n> : Değişken numarası. 1 ile 25 arası sayı olmalıdır.
<değer> : Artırma değeri. Değişkenin mevcut değerine toplanır. 0 ile 65535 arası değer alabilir.
Açıklama :

VRBINC komutu, program içinde kullanılan değişkenin mevcut değerini <değer> kadar artırır. Toplama işleminde taşma kontrolu yapılmaz. 1 ile 20 arasındaki 16-bit değişkenlerde yeni değer 65536 üzerine yuvarlanır. Örneğin 12 numaralı değişkenin mevcut değeri 65535 olsun, VRBINC 12, 3; komutu ile 12 numaralı değişkenin yeni değeri 2 olacaktır. 21 ile 25 arasındaki değişkenlerde yuvarlama 32-bit üzerinden olacaktır.

4.14 VRBDEC : Değişken Değer Eksiltme – [0x10].

Sentaks : VRBDEC <n>, <değer>;
Örnek : VRBDEC 12 , 4;
<n> : Değişken numarası. 1 ile 20 arası sayı olmalıdır.
<değer> : Azaltma değeri. Değişkenin mevcut değerinden çıkartılır. 0 ile 65535 arası değer alabilir.
Açıklama :
VRBINC komutu, program içinde kullanılan değişkenin mevcut değerini <değer> kadar azaltır. Azaltma işleminde taşma kontrolu yapılmaz. 1 ile 20 arasındaki 16-bit değişkenlerde yeni değer 65536 üzerine yuvarlanır. Örneğin 12 numaralı değişkenin mevcut değeri 2 olsun, VRBDEC 12, 4; komutu ile 12 numaralı değişkenin yeni değeri 65534 olacaktır. 21 ile 25 arasındaki değişkenlerde yuvarlama 32-bit üzerinden olacaktır.

4.15 PRM : Parametreye Değer Atama – [0x11].

Sentaks : İki şekilde kullanılabilir.
1) PRM <n> = <değer>; : Doğrudan değer atama.
2) PRM <n> = V<m>; : Bir değişkenin içeriği alınır.
Örnek : PRM 12 = 1200;
: PRM 10 = V19; //19 nolu değişken içeriği 10 numaralı parametreye atanır.
<n> : Parametre numarası. 1 ile 25 arası sayı olmalıdır.
<m> : Değişken numarası. 1 ile 25 arası sayı olmalıdır.
<değer> : 16-bit sayı.
Açıklama :
PRM komutu , sürücünün çalışmasını etkileyen ve kalıcı hafızaya kayıt edilen değişkenlerin atanmasını sağlar.
DİKKAT! 1 ile 20 arası parametreler 16-bit sayıyı, 21 ile 25 arası parametreler 32-bit sayıyı temsil eder.
DİKKAT! 1 ile 20 arası değişkenler 16-bit sayıyı, 21 ile 25 arası değişkenler 32-bit sayıyı temsil eder.

☼ *Dikkat! PRM komutu, değerleri kalıcı hafızaya aktardığı için kullanılması sırasında dikkatli olunmalıdır. Sürekli atamanın yapıldığı çevrimli yapılarda kullanılmamalıdır.*

4.16 DELAY : Bekleme Süresi Atama – [0x12], [0x22], [0x27].

Sentaks :1) DELAY = <değer>; : Doğrudan değer atama.
2) DELAY = P<n>; : Parametreden değer atama.
3) DELAY = V<n>; : Değişkenden değer atama

<değer> : 0 ile 15000000 arası sayı.
<n> : Paramtre/değişken no. 1 ile 25 arası sayı.
Örnek : DELAY = 1450; //1450 ms süre değeri bildirilir.
: DELAY = P15; //15 numaralı parametrenin değeri zamanlayıcıya atanır.
: DELAY = V1; // 1 numaralı değişkenden bekleme süresi alınır.
Açıklama :
DELAY komutu, bekleme komutu WAIT tarafından kullanılacak bekleme değerini atamak için kullanılır. Değer [ms] cinsinden zamana karşılık gelir. DELAY'e değer atandığı anda zamanlayıcı çalışır ve verilen sayıdan geri sayım başlar. Sayı sıfıra ulaştığı anda sayım biter.

DİKKAT! 1 ile 20 arası parametreler 16-bit sayıyı, 21 ile 25 arası parametreler 32-bit sayıyı temsil eder.
DİKKAT! 1 ile 20 arası değişkenler 16-bit sayıyı, 21 ile 25 arası değişkenler 32-bit sayıyı temsil eder.

4.17 WAIT : Zamanlayıcı Bekleme – [0x13].

Sentaks :WAIT;
Örnek :WAIT;
Açıklama :
WAIT komutu, DELAY komutu ile belirtilmiş zamanı bekleten komuttur. Program akışı bu komuta geldiğinde alt satırdaki komutlara geçmez ve DELAY değeri kontrol edilir. DELAY değeri sıfır olana kadar bekleme devam eder.

4.18 WAIS : Step Sayısı Bekleme – [0x14].

Sentaks :WAIS;
Örnek :WAIS;
Açıklama :
WAIS komutu, DISP komutu ile belirtilmiş step sayısını bekleten komuttur. Program akışı bu komuta geldiğinde alt satırdaki komutlara geçmez ve step sayısı kontrol edilir. Step sayısı belirtilen değere ulaşana kadar bekleme devam eder.
☼ *Dikkat! WAIS komutundan önce motorun harekete geçirilmesi gereklidir.*

4.19 MOVE : Motoru belirli adım sayısı kadar döndürme – [0x15].

Sentaks : MOVE;
Örnek : MOVE;
Açıklama :
MOVE komutu, motoru DISP komutu ile belirlenmiş adım sayısı kadar döndürür. Motor, INITV ile belirlenmiş kalkış hızı ile harekete başlar, ACCEL komutu ile belirlenmiş rampalı hareketi yaparak SPEED ile belirlenmiş hıza ulaşır, rampalı hareket yaparak yavaşlar ve hedef step sayısına ulaşınca durur. Motor pozisyona gelip durduktan sonra motor akımı kesilmez ve CUROFF komutunca belirlenmiş akım değeri uygulanır.

4.20 RUN : Motoru Koşulsuz çalıştırma – [0x16].

Sentaks : RUN;
Örnek : RUN;
Açıklama :
RUN komutu, motoru sürekli döndürme durumuna alır. Motor, INITV ile belirlenmiş kalkış hızı ile harekete başlar, ACCEL komutu ile belirlenmiş rampalı hareketi yaparak SPEED ile belirlenmiş hıza ulaşır ve sürekli döner durumda kalır.

4.21 STOP : Motoru Koşulsuz Durdurma – [0x17].

Sentaks : STOP;
Örnek : STOP;
Açıklama :
STOP komutu, motoru koşulsuz durdurur ve faz akımını keser.

4.22 TABLE : Değer Tablosu Oluşturma – [0x19], [0x2e],[0x30].

Sentaks : Aşağıdaki şekillerde atama yapılabilir:
1) TABLE <k> = <değer>; : Tabloya doğrudan değer atanır.

2) TABLE <k> = V<n>; [0x30]: Tabloya değişkenden değer atanır.
3) TABLE <k> = P<n>; [0x2e]: Tabloya parametreden değer atanır.
<k> : Tablo sıra numarası. 1 ile 32 arasında bir sayı olabilir.
<n> : Değişken/parametre no. 1 ile 25 arası değer alabilir.
<değer> : İlgili sıraya atanacak sayısal değer. 1 ile 4000000000 arasında sayı olabilir.

Örnek : TABLE 1 = 12000000;
TABLE 13 = 45;
Açıklama :
TABLE komutu ile 32 farklı sayısal değer dizisi oluşturulur. Tablo değerleri bazı komutlarla birlikte kullanılarak çeşitli işlevler yapılır.

DİKKAT! 1 ile 20 arası parametreler 16-bit sayıyı, 21 ile 25 arası parametreler 32-bit sayıyı temsil eder.
DİKKAT! 1 ile 20 arası değişkenler 16-bit sayıyı, 21 ile 25 arası değişkenler 32-bit sayıyı temsil eder.

☼ *Dikkat! TABLE komutu, değerleri kalıcı hafızaya aktardığı için kullanılması sırasında dikkatli olunmalıdır. Sürekli atamanın yapıldığı çevrimli yapılarda kullanılmamalıdır.*

1. Tablo ile pozisyonlama:
MOVT komutu ve tablo değerleri kullanılarak pozisyonlama yapılabilir. (MOVT komutuna bakınız.)

4.23 REFPOS : Motor Başlangıç Konumu (Referans Pozisyon) Atama – [0x1A].

Sentaks : REFPOS;
Örnek : REFPOS;
Açıklama :
REFPOS komutu, motorun o anda bulunduğu yeri sıfır konumu (referans pozisyon) olarak atar. Bundan sonraki tüm mutlak konumlamaları bu başlangıç noktasına göre yapar.

4.24 MOVT : Tablo Değerleri ile Konumlama – [0x1B] – [0x1C].

Sentaks : İki şekilde kullanılır.
1) MOVT <n>; : Doğrudan tablo indisi ile konumlama.
2) MOVT V<m>; : Değişken içeriği ile tablo indisi vererek konumlama.
<n> : Tablo indisi. 1 ile 32 arasında değer alır.
<m> : Değişken numarası. 1 ile 20 arası değer alır. *(Dikkat! Program içerisinde değişkenin içeriğinin 1 ile 32 arasında kalmasına dikkat edilmelidir.)*

Örnek : MOVT 3; //3. tablo indisinin belirttiği mutlak pozisyona gider.
: MOVT V12; //12 numaralı değişkenin içeriği indis olarak alınır ve indisin belirttiği tablo değerine konumlama yapılır.
Açıklama :
MOVT komutu, TABLE komutu tarafından belirlenmiş tablo değerlerine konumlama yapmayı sağlayan komuttur. Tablo değerleri mutlak konum ve pozitif değerler olarak kabul edilirler. REFPOS komutu ile belirlenmiş mutlak sıfır konumuna göre motor, indis ile belirtilen tablo değerinin belirlediği mutlak konuma gider.
İndis değeri iki şekilde verilir. Birincisi doğrudan indis numarası verilerek, ikincisi bir değişken içeriğinden indis numarası verilerek.

☼*Dikkat!* *Konumlanma yapılacak indis numarasına karşılık gelen tablo değerinin TABLE komutu ile belirlenmiş olduğuna dikkat ediniz.*

☼*Dikkat!* *Değişkenden indis verilirken indisin gösterdiği tablo değerinin TABLE komutu ile belirlenmiş olduğuna dikkat ediniz.*

4.25 INT / RET : Giriş sinyali ile Kesme Fonksiyonu Tanımlama – [0x1D].
: RET – [0x1E].

Sentaks :

INT <m>,<lojik durum>;
< program satırları>
RET; : Kesmenin geldiği yerdeki program satırına döner.

veya,

INT <m>,<lojik durum>;
< program satırları>
RET <etiket>; : etiket ile belirtilen yere döner.

<etiket> kuralları için Bölüm 3.3'e bakınız.

<m> : Giriş numarası. 1 ile 6 arası sayı.
<lojik durum> : Girişin hangi duruma geçerken kesme yaratacağını belirtir. 0 ve 1 olabilir.
<program satırları> : Kesme oluştuğunda icra edilecek program komutlarıdır.

Örnek :

```
INT 3, 0;              //3. giriş,  1 ==> 0 geçişi ile kesme yaratacak.
  MOVT 12;
  DELAY = 1000;
  WAIT;
  MOVT 1;
  DELAY = 2000;
  WAIT;
RET;                   //ilgili kesme geldiği andaki program satırına döner.
```

Açıklama :

Kesme fonksiyonu , bir giriş sinyalininin bir konumdan diğer konuma geçmesi ile normal program akışını keser, kendi program komutlarını icra eder. RET komutu ile tekrar normal akışına, yani kesme oluştuğu andaki program satırına dönülür.

"INT" Komutunda dikkat edilecek öğeler:

- Kesme için tanımlanmış GİRİŞ, programın başka bir yerinde başka kontrol için kullanılamaz.
- Kesme fonksiyonu için yazılan program satırlarında JUMP, IFINP, IFVRB gibi dallanma komutları ile programın herhangi bir yerine dallanılabilir. Bu nedenle programcı RET komutunun icra edilmesi konusunda dikkatli olmak durumundadır. Kesme programı kendi RET komutu ile sonlanmadığı takdirde yeni kesmeler oluşmaz.
- Kesme fonksiyonu icra edilirken baska kesme fonksiyonu tarafından kesilebilir. Fakat kendisi tarafından yeniden kesilemez.
- Farklı GİRİŞler için kesme programları ayrı ayrı tanımlanıp yazılmalıdır. INT bildiriminden sonra RET komutu görmeden yeniden INT bildirimi yapılamaz. INT bildirimi olmadan RET komutu olamaz.

RET komutu yalnız başına kullanıldığında kesme fonksiyonu tamamlandığında program akışı kesme gelmeden önceki noktaya döner. RET <etiket> şeklinde etiket belirtilerek kullanılırsa program akışı etiket ile belirtilen yere yönlendirilir.

4.26 ENBINT : Kesme İşlemine İzin Verme – [0x24]

Sentaks :ENBINT <n>;
<n> : Kesme numarası. Kesme işlemi girişlerle ilgili olduğundan 1 ile 6 arası sayıdır.
Örnek : ENBINT 3; //3 numaralı giriş için yazılmış kesme fonksiyonuna izin verilir.
Açıklama :
Girişler ile ilgili olarak tanımlanmış kesme fonksiyonları program içerisinde herhangi bir yerde kapatılabilir veya izin verilebilir. ENBINT komutu ilgili kesme fonksiyonunun icra edilmesine izin verir. Kesme fonksiyonu yazılmamış giriş için ENBINT komutu etkisizdir.

4.27 DISINT : Kesme İşlemini Kapatma – [0x25].

Sentaks : DISINT <n>;
<n> : Kesme numarası. Kesme işlemi girişlerle ilgili olduğundan 1 ile 6 arası sayıdır.
Örnek : DISINT 3; //3 numaralı giriş için yazılmış kesme fonksiyonu yasaklanır.
Açıklama :
Girişler ile ilgili olarak tanımlanmış kesme fonksiyonları program içerisinde herhangi bir yerde kapatılabilir veya izin verilebilir. DISINT komutu ilgili kesme fonksiyonunun icra edilmesini yasaklar. Kesme fonksiyonu yazılmamış giriş için DISINT komutu etkisizdir.

4.28 LIM : Değişkenler için Limit Kontrolu – [0x1F].

Sentaks : LIM <n>=<değer>;
<n> : Değişken numarası. 1 ile 25 arası sayı.
<değer> : Belirtilen değişken için değişkenin alabileceği en büyük sayıyı belirler. 16-bit işaretsiz sayıdır.
Örnek : LIM 12 = 43000; //12 numaralı değişkenin değeri 43000 sayısı ile sınırlanmıştır.

Açıklama :
LIM komutu, program çalışması sırasında değişkenlerin alabilecekleri en yüksek sayıları belirlemek için kullanılır. Değişkene yapılan bir atamadan sonra değişkenin limit değeri kontrol edilir, eğer atanan sayı limit değerinden büyük ise değişken içeriği limit değere eşitlenir. LIM komutu ile belirlenmemiş 1 ile 20 arasındaki değişkenlerin limit değerleri 65535 (16-bit), 21 ile 25 arasındaki değişkenlerin limit değerleri 4294967295(32-bit) olarak kabul edilir.
DİKKAT! 1 ile 20 arası değişkenler 16-bit sayıyı, 21 ile 25 arası değişkenler 32-bit sayıyı temsil eder.

4.29 SPAN : Değişken – Fiziksel Büyüklük Dönüşümü – [0x21].

Sentaks : SPAN <n> = <değer>;
<n> : Değişken numarası. 1 ile 20 arası sayı.
<değer> : Fiziksel büyüklük tam ölçek değerini ifade eden sayı. 16-bit.
Örnek : SPAN 2 = 5000; //
Açıklama :
SPAN komutu, analog giriş kanalları üzerinden değişkenlere atama yapılırken ilgili analog girişin, hız, zaman değeri, adım sayısı, v.s gibi büyüklüklerin tam ölçek değerini belirler.
Analog sinyal ile atama işlemi sürücünün bazı çalışma parametrelerinin/değişkenlerinin dışarıdan standart sinyal ile ayarlanmasını sağlar. Analog kanaldaki seviye analog dijital çeviricide sayıya dönüştürülür. Bu sayı her zaman 0 ile 250 arasında bir sayıdır. Tam skala, yani %100 değeri, 250 sayısı ile temsil edilmektedir. Seçilen analog kanalın o andaki (atama anındaki) sayısal değeri ilgili değişkene atanmadan önce, değişken için SPAN komutu ile belirlenmiş sayı ile oranlanır. Çıkan sonuç ilgili değişkene atanır. Bu oranlama işlemi, ham analog sayının sürücüde karşılığını bulacak fiziksel büyüklüğe dönüştürme işlemidir. Analog atama kullanılırken ilgili değişken için SPAN komutu ile anlamlı bir büyüklük verilmiş olmalıdır.
SPAN komutu ile belirlenmemiş değişkenlerin tam skala değerleri 250 olarak ayarlıdır.

4.30 OUT : Lojik Çıkış Kontrolu – [0x23].

Sentaks : OUT <n>=<lojik değer>;

<n> : Çıkış numarası. 1 veya 2 olabilir.
<lojik değer> : Çıkış etkin veya değil değeri. 0 veya 1 olabilir. 1:çıkış etkin, 0: etkin değil.
Örnek : OUT 2 = 0;
Açıklama :
Sürücü iki adet 24V çıkış sinyali verebilir. Çıkışlar optik transistor tipidir. PNP çıkış türündendir. OUT komutu ile ilgili çıkış konumu etkinleştirilir veya sıfırlanır.

4.31 FILTER : Giriş Kontrolu Filtreleme – [0x26]

Sentaks :FILTER <n>=<değer>;
<n> :Filtrelenecek giriş numarası. 1 ile 6 arasında bir sayı.
<değer> :Filtreleme süresi – milisaniye, [ms]. 0 ile 250 arası sayı.
Örnek : FILTER 3 = 40; //3.giriş, 40 ms ile filtrelenir.
Açıklama :
Program akışı içerisinde girişler ile ilgili kontroller yapılırken girişlerin kararlı duruma geçmeleri önemli olmaktadır. Özellikle butonlar, mekanik kontaklar, limit switchler ile ilişkilendirilmiş girişlerde elektriksel gürültü hatalı çalışmalara neden olabilmektedir. Bu nedenle girişlerin kararlı duruma gelmeleri beklenmelidir. Kararlı durum, girişlerin belirli süre boyunca kararlı oldukları gözlenerek anlaşılır. İlgili giriş için sıfır ile 100 milisaniye arasında filtreleme zamanı verilebilir.

- Filtre değeri sıfır olarak girilmiş ise giriş filtrelenmeden değerlendirilir.
- Program akışında belirtilmemiş girişlerin filtre değerleri sıfırdır.

4.32 LOCATE : Mutlak Konumlama – [0x34] – [0x35].

Sentaks : İki şekilde kullanılır.
1) LOCATE <değer>; : Doğrudan değer ile konumlama.
2) LOCATE V<m>; : Değişken içeriği ile konumlama.
<değer> : 32-bit sayıdır. 0 - 4,294,967,295 arasında değer alabilir.
<m> : Değişken numarası. 1 ile 32 arası değer alır.

Örnek : LOCATE 1500; //Motor Referans konuma göre 1500 step değerine konumlanır.
: LOCATE V12; //12 numaralı değişkenin içeriğine konumlama yapılır.
Açıklama :
LOCATE komutu, referans noktaya göre konumlama yapmayı sağlayan komuttur. Alınan değerler mutlak konum ve pozitif değerler olarak kabul edilirler. REFPOS komutu ile belirlenmiş mutlak sıfır konumuna göre motor, belirtilen mutlak konuma gider.

5 Doküman Baskı Tarihçesi

V3. 18.07.2009

1) Aritmetik işlemler bölümü eklendi.
2) Dokümanın tamamında, değişken ve parametre numaraları 1 ile 20 yerine 1 ile 25 arasında değiştirlidi.
3) Komut tablosuna eklemeler yapıldı.
4) VRB komutuna Q bilgi girişi eklendi.
5) VRB komutuna T (tablo) ataması eklendi.
6) JUMP komutunda hatalar düzeltildi.
7) Etiket kullanımına ait açıklamalar değiştirildi.
8) ACCEL komutu üst değeri 5000 olarak değiştirildi.

V2. 15.12.2008

1) Komut Gurupları Özet Tablosuna yeni komutlar eklendi.
2) VRB, DISP ve PRM komutlarına yeni komut eklemeleri yapıldı.
3) Giriş bölümüne ekleme yapıldı.
4) Baskı Tarihçesi Bölümü eklendi.

V1. 10.11.2008

İlk baskı.

DOKÜMAN SONU