DOKUZ EYLÜL ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

BİLGİSAYAR KONTROLLÜ DİK İŞLEM TEZGAHI TASARIMI VE UYGULAMASI

Meriç KAYALIK

Nisan, 2009
İZMİR

BİLGİSAYAR KONTROLLÜ DİK İŞLEM TEZGAHI TASARIMI VE UYGULAMASI

Dokuz Eylül Üniversitesi Fen Bilimleri Enstitüsü

Yüksek Lisans Tezi

Mekatronik Mühendisliği Bölümü, Mekatronik Mühendisliği Anabilim Dalı

Meriç KAYALIK

Nisan, 2009

İZMİR

YÜKSEK LİSANS TEZİ SINAV SONUÇ FORMU

MERİÇ KAYALIK, tarafından **YRD. DOÇ. DR. ZEKİ KIRAL** yönetiminde hazırlanan **"BİLGİSAYAR KONTROLLÜ DİK İŞLEM TEZGAHI TASARIMI VE UYGULAMASI"** başlıklı tez tarafımızdan okunmuş, kapsamı ve niteliği açısından bir Yüksek Lisans tezi olarak kabul edilmiştir.

….....

Yrd.Doç. Dr. Zeki KIRAL

Danışman

….....

Prof.Dr. Erol UYAR

Jüri Üyesi

…….....

Yrd. Doç. Dr. Mehmet ÇAKMAKÇI

Jüri Üyesi

Prof. Dr. Cahit HELVACI
Müdür
Fen Bilimleri Enstitüsü

TEŞEKKÜR

Bu çalışmam süresince yol göstericiliği ile proje yöneticim Sayın Yrd.Doç. Dr. Zeki KIRAL'a, anlayışlılığı ve bana olan güveni ile destek veren Sayın Prof. Dr. Erol UYAR'a, bilgilerini benimle paylaşan ve yardım için en ufak desteği esirgemeyen değerli arkadaşlarım; Nihat Engin TOKLU ve Onur KESKİN'e, imalat aşamasında teknik destek veren MEGATEK Mühendislik LTD. ŞTİ'ne ve destekleri, anlayışları hiçbir zaman tükenmeyen en büyük moral kaynağım olan değerli annem, babam ve ablama teşekkürlerimi bir borç bilirim.

Meriç KAYALIK

BİLGİSAYAR KONTROLLÜ DİK İŞLEM TEZGAHI TASARIMI VE UYGULAMASI

ÖZ

Bu tez çalışmasında, üniversitede bulunmakta olan eski bir CNC tezgahı, iki adet servo motorun eklenmesiyle modernize edilerek, iki eksenli bir dik işleme tezgahı haline getirilmiştir. Projede, Siemens'in Simotion D425 adlı servo motor kontrol ünitesi, bu üniteye bağlı iki adet CUA31 servo güç yükselticisi ve iki adet servo motor kullanılmıştır. Bu ürünün dik işlem tezgahı şeklinde çalışabilmesi için, motorları kontrol etmekte kullanılan Siemens Scout adlı programın içerisinde yer alan Structured Text adlı programlama dilinde, temel hareket fonksiyonları tanımlanmıştır. Visual Basic programlama dilinde geliştirilen bir yazılımla, G kodu tarzında verilen şekil komutları, temel hareket fonksiyonlarını çağıran Structured Text programlarına dönüştürülmüştür. Bu üretilen Structured Text programlarının Siemens Scout içerisinde çalıştırılmasıyla, istenilen şekillerin işletilmesi sağlanmıştır.

Anahtar Sözcükler: CNC, Simotion, ST, Structured Text, Servo Kontrol

DESIGN AND APPLICATION OF A COMPUTER CONTROLLED VERTICAL MILLING MACHINE

ABSTRACT

In this thesis study, an old CNC vertical milling machine have been modernized into a two axis milling machine, with the addition of two servo motors. A servo-motor control unit called Siemens Simotion D425, two CUA31 servo power amplifier units that connected to the control unit and two servo motors have been used in this project. To make this unit work as a vertical milling machine, basic movement functions have been defined in Structured Text language, which is one of the supported languages of the motor controller software called Siemens Scout. A Visual Basic software has been developed to convert G code styled shape commands to Structured Text programs which call the defined basic movement functions. When executed in Siemens Scout, these generated Structured Text programs operate the desired shapes.

Keywords: CNC, Simotion, ST, Structured Text, Servo Control

İÇİNDEKİLER

Sayfa

BÖLÜM BİR
GİRİŞ

1.1 Dik İşlem Tezgahı Tanımı ve Hakkında Kısa Tarihçe

İmalatı, hammadde halindeki bir materyalin bir veya birkaç işlem safhasından geçerek, istenen nitelikleri belirli toleranslar içerisinde üzerinden talaş kaldırarak veya kaldırmayarak hammadde üzerine aktarılması işlemine verilen ad, olarak açıklayabiliriz.

İmalatın tanımı, imalatın yapıldığı araçların tanımını yapabilmek açısından önem içermektedir. İmalat aletlerini belli ana başlıklar altında toplamak mümkün olsa da pratikte imalat tanımının esnekliğinden üretimin, şekil veya nitelik değişiminin yapıldığı her türlü alet, makine veya program imalat aracı tanımına girmektedir.

Dik işleme tezgahı; talaşlı imalatın yapıldığı, birçok hareket eden eksene sahip kesici takımın kartezyen koordinatlarda Z ekseni yönünde hareket ettiği, takım tezgahına verilen addır. Bilgisayar kontrollü dik işleme tezgahında ise program ile kesici takımın veya tezgaha bağlı eksenlerin hareketlerinin programlanabildiği, bilgisayar veya mikro işlemci kontrollü takım tezgahlarına verilen genel addır. CNC dik işleme tezgahları yapılarının kabul ettiği üzere birçok farklı tasarım şeklinde bulunabilirler. Fakat işlemlerin yapılabilmesi için minimum iki hareket eden eksen ve bunun yanında talaş kaldıran veya imalatı gerçekleştiren birimin bulunduğu eksenden oluşmaktadır.

1950 yıllarda nümerik programlamaya göre çalışan ve NC-Numerical Control denilen tezgahların ortaya çıkması ile uygulamalarda yerini bulmaya başlar. Takım tezgahları özellikler bakımında bu tarihlerde yapısal olarak iyi seviyelere ulaşmıştır. NC tezgahların bilgisayarla donatılması ile CNC (Computer Numerical Control) ve DNC (Direct Numerical Control) tezgahları oluşmuş, bilgisayarların kullanılması ile de bu tezgahlar işlemleri yüksek verimlilikle yapmaya başlamışlardır.

Şekil 1.1 Manüel dik işleme tezgahı (Grizzly, 2008)

1.2 Dik İşlem Tezgahını Oluşturan Başlıca Kısımlar

Dik işlem tezgahını oluşturan kısımları kısaca sayacak olursak; parçanın bağlandığı ve hareketin verildiği eksenleri içeren tabla, hareketi ileten hassas miller ve kızaklama tertibatı, millere tahrik veren servo motorlar, millere motorların bağlanabilmesine olanak sağlayan kaplinler, parçadan talaş kaldırmak sureti ile onu işleyen takımlar, takımların bağlandığı pensler ve fener mili, fener miline direk hareketi veren motorlar veya kayış kasnak mekanizması, redüktör v.b. hareket iletim elemanları ile hareket veren motorlar, bu sistemlerin tamamını yöneten ana kontrol bilgisayarı ve bunun yönettiği motor sürücüleri ve kumanda mikroişlemcileri olarak sıralayabiliriz.

Şekil 1.2 Modern bir CNC dik işleme tezgahı ve kontrol paneli (QSS,2008)

CNC tezgahların modernleşmesi ile kesici takım değiştirme gibi elle yapılması gereken işlemlerin de otomasyon haline getirilip magazin denen, işleme takımlarının dizildiği ve otomatik olarak fener miline takımı bağlayan servo kolların bulunduğu özel birimler de dik işleme tezgahına eklenerek arada beklemeleri minimuma indirecek hale getirilmiştir. Yeni nesil CNC dik işleme merkezlerinde parçanın tablandan sökülüp tekrar bağlanmasını gerektirecek hallerde, parçanın sökülmeden istenen konuma geçebilmesini sağlayan ekstra eksenler hareket ve işleme esnekliklerini, lazer torçlar ile parçaların yüzey sertlik özelliklerini değiştirecek olan ısıl işlemler gerçekleştirilebilmektedir. Bu gibi özellikler ile parçanın bir defa tezgaha bağlandıktan sonra son istenen durumuna olabildiğince yakın halde çıkabilmesi sağlanmıştır. Daha da farklı özellikler tezgahlara yenilik olarak getirilmektedir.

1.3 Problemin Tanımı

Tez'in ana mantığının anlaşılması için problem tanımı başlığı altında bilgiler verilmesi doğru olacaktır. Projede yapılması hedeflenen bilgisayar kontrollü bir dik işlem tezgahı tasarımı olmasıyla beraber, tezin yapımında üniversitenin mevcut imkanlarında hazır bulunan motor, motor sürücüsü, motor sürücü kontrol ekipmanı, bilgisayar ve eski tip bir tezgah kullanılmıştır. Bu bahsedilen elemanlar birleştirilerek projeye tezgah modernizasyonu işlemi gibi bir işlevsellik kazandırılmak istenmiş ve elde bulunan imkanların kullanılması yönünden bir mühendislik uygulaması olarak da tezin değer kazanması hedeflenmiştir. Tez içeriği problem tanımı şeklinde anlatılacak olursa: "Üniversitede kullanılmaz durumda bulunan eski bir CNC dik işlem tezgahını modernize etmek için, üniversiteye Siemens A.Ş. tarafından hediye edilmiş olan fakat daha çok robot kol ve akıllı servo motor uygulamalarının gerektiği yerler için üretilmiş bir yazılımı ve beraberinde bulunan servo motor, servo motor sürücü ve kontrol ekipmanı kullanılarak nasıl modernize edilmiş tezgaha sahip olabiliriz?" şeklinde yapmak, açıklamayı kolaylaştırmaktadır.

Başlangıç olarak bu bilgilerin verilmesi yeterli olacaktır. İlerleyen bölümlerde tezde yapılan çalışmalar anlatılmaya çalışılmıştır.

BÖLÜM İKİ
METODOLOJİ

Projenin teknik özellikleri bu bölüm içerisinde fazla detaya girilmeden anlatılmaya çalışılmıştır. Projenin içerisinde kullanılmış materyallerin çeşitli özellikleri v.b. tanımlamaları bu bölümde yapılmıştır. Bu bölüm okunduğu zaman içerisindeki bilgilerle sistem ile ilgili genel bilgi edinilmiş olunacaktır.

2.1 Servo Tanımı ve Servo Kontrol Mantığı

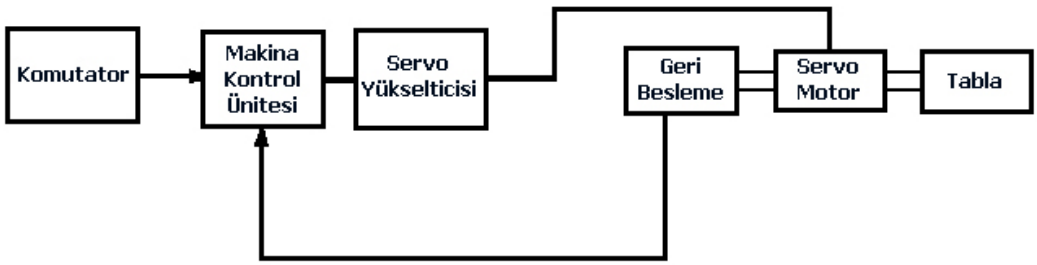
Servo: bir fonksiyonu veya görevi uygulayan, olarak terim anlamı içermektedir. Tezde kullanılan mantıkta olan servo kontrolünün açıklaması ise takip eden şekildeki gibi yapılabilir. Komut sinyali insan arabiriminden pozisyon kontrolörüne gelir. Pozisyon kontrolörü değişik görevler veya uygulamalar için bilgilerin depolandığı ve hareketin yapılması için gereken komutların aktarıldığı cihaza verilen addır. Pozisyon kontrolörü motoru aktif hale getirmek, hızı veya pozisyonu değiştirmek gibi şeyler için programlanır. Bu sinyal servo güç ünitesi veya “yükseltici” olarak adlandırılan kısma geçer. Bu kısım düşük güçteki bu sinyali alır, servo motorun hareket üretebileceği uygunlukta bir seviyeye gücü yükseltir veya güçlendirir.

Servo motor üzerine güç uygulanmaya başlandığında, motor dönmeye başlar. Motorun dönmesiyle birlikte hızı ve pozisyonu değişir. Bu değişimin algılanmasını ve ne oranda bir değişimin olduğunu kontrol organına geri bildiren bazı cihazlar vardır. Bu cihazlar takometre, enkoder veya resolver olabilir. Bu geri besleme, motorun işini doğru yapıp yapmadığını kontrol etmek için gerekmektedir.

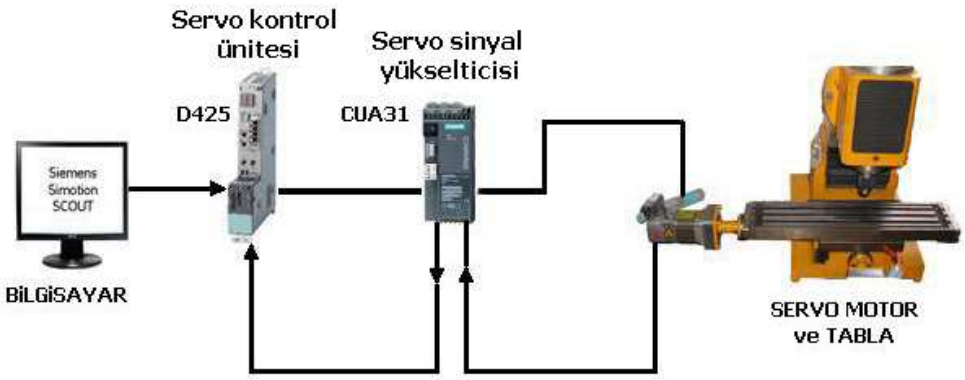
Pozisyon kontrolörü geri besleme sinyaline bakar, yükün servo motor tarafından doğru şekilde hareket ettirilip ettirilmediğini hesaplar ve eğer yapmıyorsa kontrolör gereken düzeltmeleri yapar. Örnek olarak kumanda sinyali yükü 1000dev/dak’da sürmek için üretilmiş olsun. Aslında bazı nedenlerle yükün hareketi istenildiği gibi olmamakta ve motor 900dev/dak’da dönmekte olsun. Geri besleme sinyali kontrolöre hızın 900dev/dak olduğu bilgisini verir.

Ardından kontrolör kumanda sinyalinin 1000dev/dak'sını ve geri besleme sinyalinin 900dev/dak'sı ile karşılaştırır ve bir hata sinyali oluşturur. Kontrolör geri besleme sinyali ile kumanda sinyali arasında hata kalmayacak şekilde, yani iki sinyal eşit oluncaya kadar hızın artması için motora voltaj uygulayarak çıkış sinyalini arttırır.

Servo kontrol mantığını açıklayan bu örnekte dikkat edileceği üzere motordan bilgiyi alıp düzenleyici organa bilgi veren geri besleme elemanları vardır. Servo sistemlerin genelde içerdiği bu mantık kapalı çevrim veya geri beslemeli sistemler olarak adlandırılır.



Şekil 2.1 Servo kontrol mantığını anlatan şematik çizim



Şekil 2.2 Servo kontrolün sistemimizin bir ekseni için çizilmiş diyagramı

2.2 Sistemde Kullanılan Servo Motorların Türünün Tanıtımı:

AC Senkron Motor: Senkron motorlar kısaca indüksiyon tipi motorlara benzerler fakat rotor yapılarında bazı ufak farklılıklar vardır. Rotor yapıları, bu tip motorlara aynı hızda (senkronizasyonda) dönebilme imkanı sağlar. Temel olarak iki tip senkron motor tipi vardır; kendinden uyarımlı (indüksiyon motorlarda olduğu gibi) ve direk uyarımlı (sabit mıknatıslılar) bu tiplerdir.

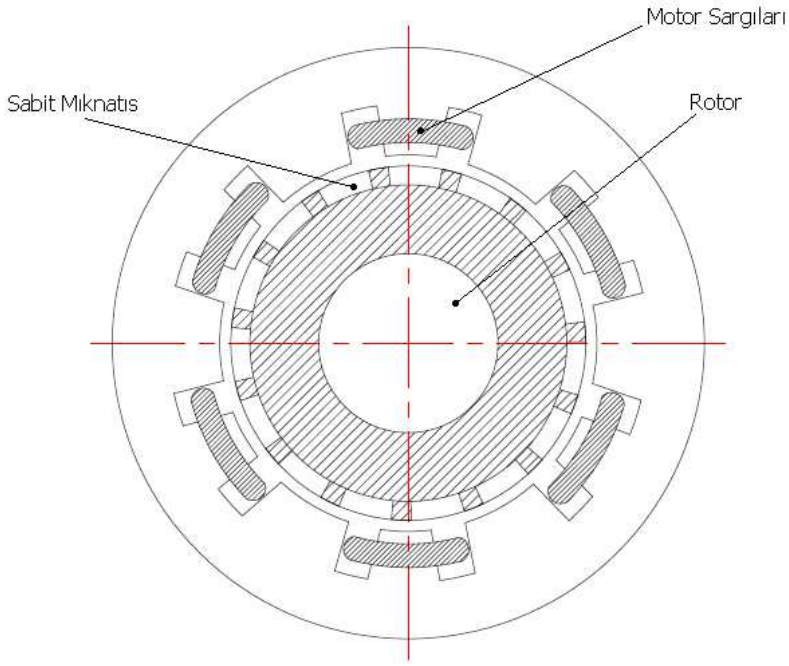
Kendinden uyarımlı motorlar (relüktans senkron, olarak anılabilinir) çentikli veya çevresinde dişleri olan rotorlara sahiptir. Çentik sayısı statordaki kutup sayısına bağlıdır. Çoğu zaman çentikler veya dişler belirgin kutuplu olarak adlandırılır. Bu belirgin kutuplar manyetik akının dolaşması için kolay bir yol oluşturur. Bu rotorun, dönen alana kilitlenip aynı hızda dönmesini sağlar.

Direk uyarımlı motor (histerisisli senkron olarak veya AC sabit mıknatıslı senkron motor olarak anılabilir) sabit mıknatıs alaşımından oluşan bir silindirle kaplı rotora sahiptir. Sabit mıknatısların kuzey ve güney kutupları etki olarak bu dizayndaki rotorun üzerinde bulunan belirgin yapıdaki dişlerdedir ve kaymayı sağlar.

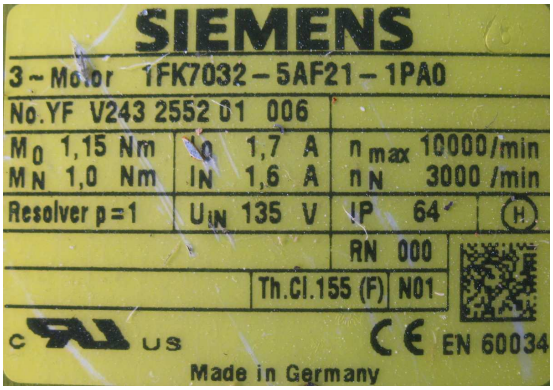
Direk uyarımlı veya kendinden uyarımlı motorların her ikisinde de bir bağlanma açısı vardır. Örnek olarak rotor, stator manyetik alanını küçük bir mesafe arkadan takip eder. Bu açı yükün artmasıyla büyür ve eğer yük motorun taşıyabileceğinden daha da büyürse rotor senkron hareketten çıkabilir.

Senkron motorlar genellikle açık çevrim konfigürasyonda birleşme açısı (veya koparma torku) limitlerinde çalışır. Bu değeri verilen yükleme için sabit hızın belirlenebilmesini sağlayan bir özelliktir. Bu kategorideki motorlar kendinden çalışan tipte değillerdir. Başlatma akımları (ayrık faz, kapasitif başlatma), frekans veya voltaj yavaşça rampalanarak dönmenin başlaması için kullanılır.

Senkron motorlar geri besleme cihazları eklenmek şartı ile hız kontrol sistemlerinde kullanılabilirler. Vektör kontrol yaklaşımı bu motora uygulandığı takdirde iyi sonuçlar alınacak şekilde kullanılabilir. (Baldor Electric(b.t))



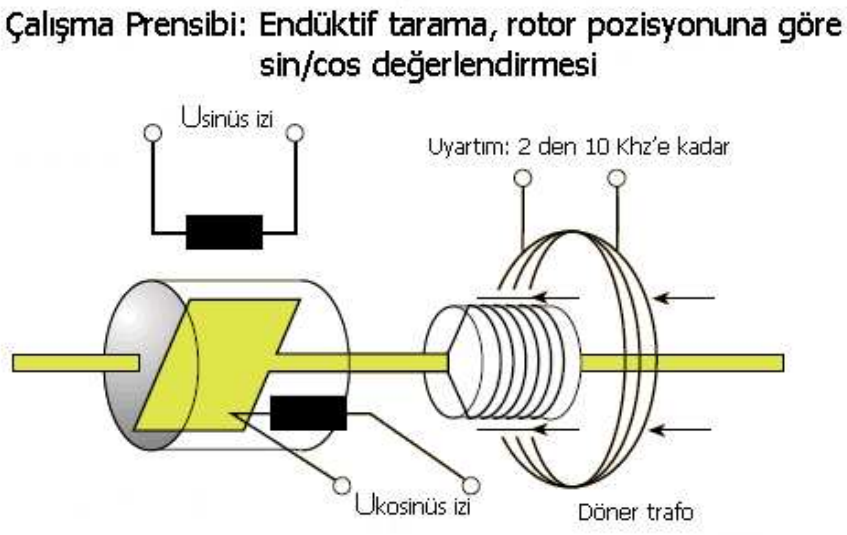
Şekil 2.3 fazlı 15 kutuplu senkron bir motorun kesiti (NEC Electronics, 2008)



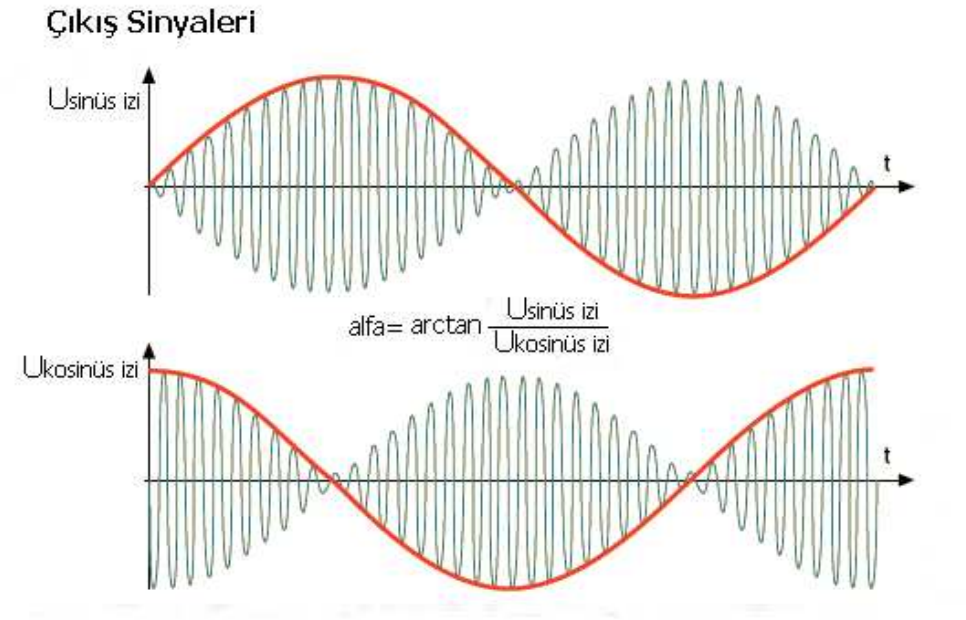
Şekil 2.4 Projede kullanılan servo motor özelliklerinin yazılı bulunduğu plaket

2.3 Resolver'ın Çalışma Mantığı

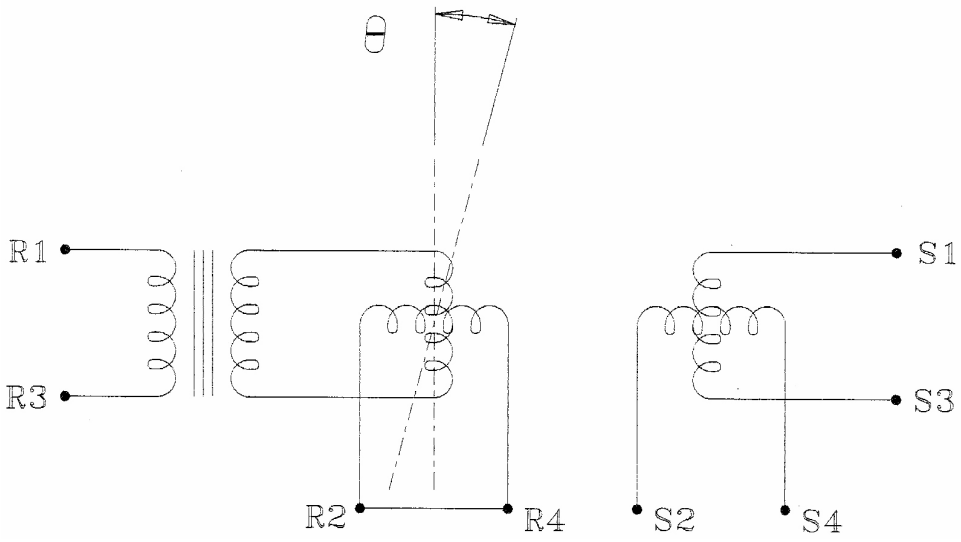
Resolver, rotor üzerinden akan AC uyarı sinyallerinden etkili bir şekilde faydalanarak resolverın bağlı bulunduğu mekanik aksamın açısal dönüşünden elde edilen sinüs ve kosinüs dalgaları ile orantılı olarak üretilen sinyallerin genliğini ayarlar. Bu sinüs ve kosinüs elektriksel bilgi çıktıları, stator akımları doğrultusunda ölçülerek pozisyon ve hız bilgisi için kullanılabilir. Bu doğrultuda, "resolver analog trigonometrik fonksiyon üreticisidir." şeklinde bir tanım doğru olacaktır. Resolverların çoğunun statoru içerisinde, birbirine dik açı yapan iki ana sargı ve rotor içerisinde birbirine dik açı yapan ikincil sargılardan vardır.



Şekil 2.5 Resolverların statorunda sinüs dalgasını oluşturacak şekilde bir birinden izoleli bobinler yer alır. (Siemens, 2008)



Şekil 2.6 Resolver üzerindeki bobinlerden elde edilen kosinüs ve sinüs sinyalleri. (Siemens, 2008)



Şekil 2.7 Resolverın şematik çizimi (Hyatt, Jr. Dayton ve Dayton 2008)

Eğer rotor bobini (R1-R3) belirli giriş voltajı ile uyarılırsa statordaki çıkış bobininin genliği rotorun θ açısının sinüsüyle orantılı ve ikincil stator bobininin çıktı genliği kosinüs θ ile orantılıdır. Bu genellikle “kontrol vericisi” modu olarak adlandırılır ve resolverdan dijitale çeviricilerde en son teknoloji olarak kullanılmaktadır.

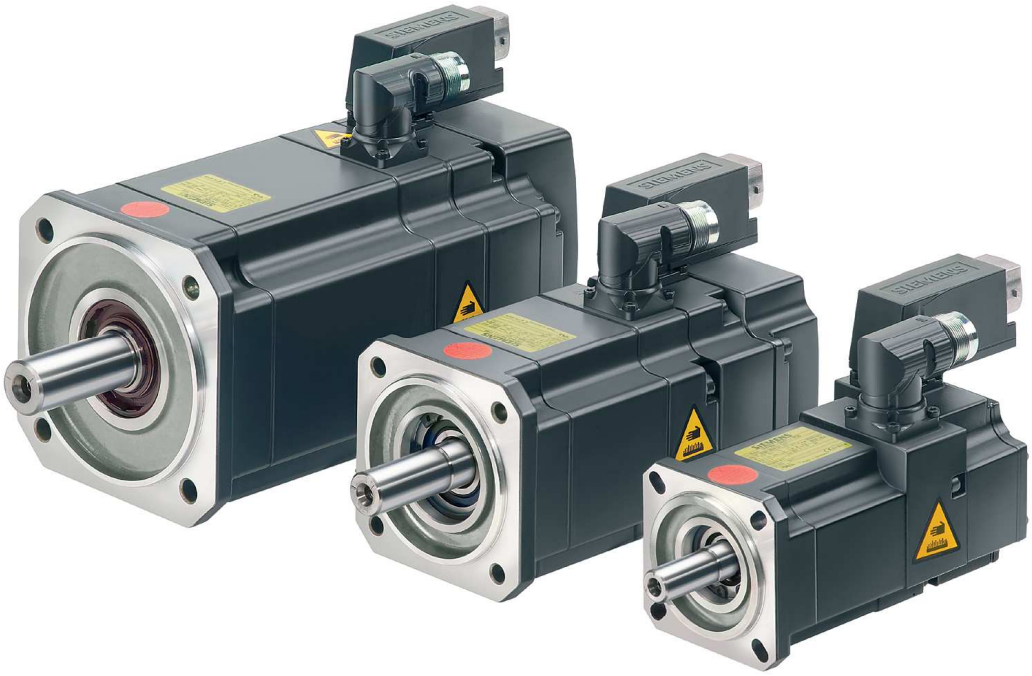
Kontrol vericisi modunda elektriksel sıfır (R1-R3) rotor bobini belirli bir voltaj değeri ile uyarıldığında stator üzerindeki S2-S4 arasında minimum voltaj olduğunda rotorun statora olan pozisyonu olarak tanımlanır. S2-S4 arasındaki sıfırlar 0º ve 180º pozisyonlarında ve S1-S3 arasındaki 90º ve 270º pozisyonlarında oluşur.

Eğer S1-S3 stator bobini belli bir girdi voltajı ile uyarılırsa ve S2- S4 stator bobini elektriksel girdi voltajı tam 90º kaymış olarak uyarılırsa ardından rotor bobininde çıktı olarak ölçülen R1-R3 rotorun dönüşüne göre genlik veya frekans olarak girdi referans sinyali tarafından değiştirilmez. Bu her iki çıktının toplamıdır. Referansı sıfır olan şaftın açısıyla belirli bir girdi tarafından zaman fazında değişir. Bu bir “analog faz” çıktısı ve “kontrol değiştiricisi” aracı olarak isimlendirilir. Sıfır noktasından sinyalin geçiş zamanının tespitiyle referans voltajı dalga formu ve çıkış voltajı dalga formunun arasındaki faz açısı hesaplanabilir. Bu fiziksel olarak çıkış şaftının açısal yer değişimidir.

Resolver analog bir cihazdır ve çıktıları 360º kesintisizdir. Bu nedenle resolverın teorik çözünürlüğü sonsuzdur. Bununla birlikte çıktı voltajında 360º dönüşte voltajın birincilden ikincile dönüşümünün yapısal varyasyonları sonucu oluşan belirsizlikler vardır. Bu belirsizlik doğru açısal pozisyonu hesaplamada hata olarak sonuçlanır.

Kural olarak, statorların katmanlarının çapı ne kadar büyükse cihazın doğruluğu artar ve çözünürlüğü yükselir. Bu cihaza yerleştirilebilen manyetik kutupların sayısının fonksiyonudur ki bu stator ve rotor katmanlarındaki slot sayılarının direk fonksiyonudur. (Hyatt, diğer., 2008)

Resolverların analog moda çalışmasını tanımlayan 7 adet fonksiyonel parametre mevcuttur. Bunlar: 1-Doğruluk, 2-Çalışma voltajı genliği, 3-Çalışma frekansı, 4-Girdi referans sinyaline göre çıktı voltajının faz kayması, 5-Maksimum izin verilebilir akım çekimi, 6-Çıkış voltajının giriş voltajına göre dönüşüm oranı, 7-boşta çalışma voltajıdır.



Şekil 2.8 Projede kullanılan tipte Siemens'in senkron motor ailesini oluşturan 1FK7*** tipi çeşitli büyüklüklerde motorlar, şaftlarına DriveCLIQ arabirimli resolverler bağlı durumda. (Siemens, 2008)

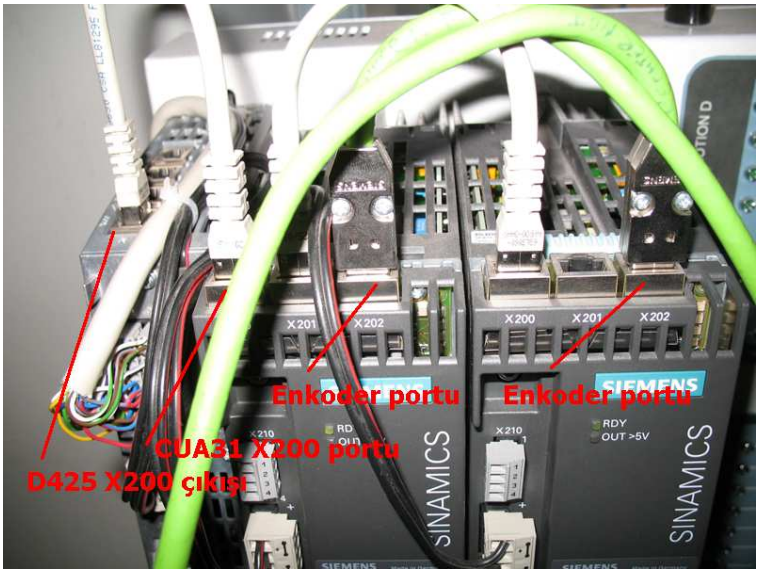
2.4 Siemens D425 Ünitesinin ve Sürücülerin Genel Tanıtımı

Simotion D sinamics S120 ailesinin sürücü tabanlı versiyonudur. Simotion D425 D4x5 kontrol birimi ailesindeki en temel performans değerlerine sahip model olup Max 16 eksen sürme kabiliyetindedir. İnterpolasyon çevrimi zamanı 2ms'dir. 4 adet de DriveCLIQ denen endüstriyel ethernet olarak tanımlanabilecek ara birime sahiptir. Bu ara birim sayesinde motorlar sisteme kolayca bağlanırlar.

D425 içerisindeki IEC 61131-3 standardını kapsayan bir PLC mevcuttur. Bu sayede yalnız hareketi kontrol etmekle kalmayıp bütün makineyi (hidrolik silindirlerin kontrolü, kamlarla yapılan hareketler tanımlanması, harici enkoderler bağlanabilmesi gibi benzer birçok özellik içerisinde uygulanabilir.) de kontrol edebilecek özelliklere sahiptir.

Siemens bu ürün ailesini paketleme makineleri, plastik ve kauçuk işleme makineleri, tekstil makineleri, baskı makineleri gibi bir çok hareket ekseninin bulunduğu kompleks işlemlerin yapıldığı uygulamaları, ürünün hitap ettiği hedef kitle olarak kabul etmiştir. (Siemens 2008)

Şekil 2.9 D425 16 eksen servo kontrol modülü (Siemens, 2008)



Şekil 2.10 D425 servo kontrol modülü ile CUA31 servo sinyal yükseltici modülünün bağlantısı

D425 servo kontrol ünitesinin üst kısmında bulunan X200 adlı portu CUA31servo sinyal yükseltici modülünün X200 adlı portuna bağlanır. CUA31'in üst kısmında bulunan X201 portu diğer bağlanan eksenin X200 portuna bağlanır. Her iki sinyal yükselticinin X202 portları, resolverların bağlanması için kullanılır.

D425 servo kontrol ünitesinin üzerinde bulunan ethernet portu ile bilgisayarın üzerindeki ağ kartında bulunan ethernet girişine direk bağlantı sağlanabilir.

Şekil 2.11 D425 Ethernet portu

2.5 Servo Bir Sistemin Matematiksel Modeli

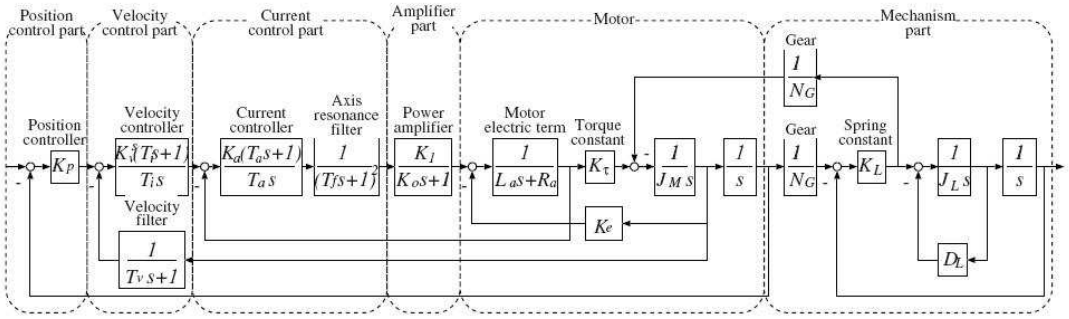
Bu bölümün verilmesindeki sebep, ilerideki bölümlerde görüleceği gibi motorlar otomatik olarak sistem tarafından tanınmakta ve kazanç katsayıları sistemin uyguladığı prosedürler ile hesaplanmaktadır. Bu sayede sistem kendisini bağlandığı mekaniğe göre optimize edebilmektedir. Sistemin hesapladığı kazanç katsayılarının neler olduğu, böyle bir sistemdeki kazanç katsayıları eğer elle hesap edilmesi gerekseydi nelerin göz önünde bulundurulması gerektiğine, bu bölümde kısaca değinilmeye çalışılmıştır. İlk olarak bir servo sistemin bütün ifadeleri kullanıldığı takdirde oluşan ve 13. mertebeden olan bir matematik model verilmiştir. Devamında ise bu matematik modelin gereken ihmalleri yapılarak 4. mertebeden bir sistem olarak ifade edilebildiği vurgulanmıştır. Mekatronik servo sistemle ilgili genel farklar ve ana noktalara aşağıda bahsedilmiştir.

Mekatronik servo sistemlerde iki tip kontrol vardır. Birincisi noktadan noktaya (“Point-to-Point”) pozisyon kontrolüdür. Bu kontrolde, gidiş yolu önemsenmez. Belirtilenler varış zamanı ve hedef noktanın konumudur. İkinci kontrol ise devamlı yolların (“Continuous Path”) kontur kontrolüdür. Bu kontrolde, son konum ile hedef noktanın konumu arasındaki yol belirtilir. İlkine örnek olarak eleman montajlamasında noktasal kaynak yapan bir robotun hareket kontrolü, veya delik delen bir mekanizmanın hareketli eksenlerinin kontrolü verilebilir. Diğer kontrol ise kaynak robotunun kolunda, boya yapan robotlarda, lazer kesim yapan robotlarda veya 3 boyut sürecinin oluşumunda görev alan mekanizmanın transfer eksenini kontrol etmede kullanılır.(Nakamura,Goto;Kyura, 1998)

Kontur kontrolünde servo sisteme pozisyon kontrolü için iyi bir hız kontrolü gerekir. Kaynak yapan bir robotu ele alırsak hızın ne kadar önemli olduğu öne çıkar. CNC olarak talaşlı imalat yapan bir makinede durum yine önemlidir.

2.5.1 Mekatronik Servo Sistem Kontrolü

Mekanizmaya bağlı bir servo sistem şekildeki gibi bir blok diyagramı ile ifade edilebilir. Şekilden de anlaşılabileceği gibi servo motor kontrol sisteminin içerisinde yer almaktadır.



Şekil 2.12 Servo kontrol sisteminin blok diyagramı (Nakamura ve diğer.,1998)

Endüstriyel servo kontrollü mekanik sistemlerde eksenlerin kontrolü birbirinden bağımsız olacak şekilde yapılır. Mekanik yapıların kendilerine has karakteristikleri (sürtünme, atalet v.b) olmasından dolayı kontrol organlarına yüklenecek veya sistemi tarif edecek denklemlerde kat sayıların hesaplanmasını sağlayacak değerler de bu sistemler gibi farklı olacaktır.

2.5.2 Endüstriyel Servo Mekatronik Bir Sistemin Kontrolü

Nakamura, Goto ve Kyura, (1998) yılında yaptıkları çalışmalarında her servo eksen için oluşturulacak blok diyagramı 13. Dereceden olacak demişlerdir. Bu modelde ihmal edilip sistemden çıkarılabilecek olan değerlerden aşağıda bahsedilmiştir.

“

1. Taşıyıcı frekans, büyük olarak tasarlanırsa güç amplifikatörü lineer olarak ele alınabilir.

2. Güç amplifikatöründeki ölü zaman ihmal edilebilinir.

3. Eksen torklarının rezonans frekansları mekanizmanın doğal frekansının 5~8 katı kadardır ve eksen rezonans filtresi kullanılarak ihmal edilebilinir.

4. Hız tanıma filtresinin kesilme frekansı ve eksen rezonans filtresi tüm mekatronik servo sistemin doğal frekansından büyük ise ihmal edilebilinir.

5. Akım kontrol kısmı, motorun elektriksel özelliklerinin dengesine göre dizayn edilmiş olarak düşünülebiliyor ise ihmal edilebilir

6. Pozisyon tanıma, yön ve saymanın artması/azalması iki puls sinyalinin mantıksal işlemlerinden elde edilir. Puls sayma bu sinyallerdeki gürültü yokmuş gibi düşünülür.

7. Cevaptaki gecikme eğer hız tanımanın hız cevabı mekanizmanın hız cevabından büyük ise ihmal edilebilinir.
8. Tork dağılımı (PI) kontrolörünün hız çevriminin I integral etkisi ile dengelenir ise.

Bu karakteristikler ışığında; Orijinal karmaşık endüstriyel servo sistem kontur kontrol içerisinde basit matematik model kullanarak ifade edilebilinir.”

2.5.3 Mekatronik Servo Sistemin 4. Dereceden Matematik Modelinin Türetilmesi

Nakamura, Goto ve Kyura, (1998) yılında, mekatronik servo sistem ile motorun mekaniksel parçasını tek bir mekanizma altında toplamak için oluşturdukları kütle modeli hakkında “İki kütle modeli, motorun atalet momenti ile yükün atalet momentini bir yay vasıtası ile birbirlerine bağlar. Hareket denklemi motor tarafı için veya mekanizma kısmı için transfer fonksiyonları aşağıdaki gibi ifade edilebilinir. Bu denklemlerde J_M: Motorun içsel atalet momenti, D_L: viskoz sürtünme katsayısı, K_L: bütün yay sabitesi N_G: motor ile mekanizmayı birbirlerine bağlayan dişli grubu veya

redüksiyon oranı TM: motorda elde edilen tork olarak ifade edilmiştir. TL(t) mekanizma tarafından motor tarafına eklenen reaksiyon kuvvetidir.

$$J_M \frac{d^2\theta_M(t)}{dt^2} = T_M(t) - T_L(t) \quad (2.1)$$

$$J_L \frac{d^2\theta_L(t)}{dt^2} = N_G T_M(t) - D_L \frac{d\theta_L(t)}{dt} \quad (2.2)$$

$$T_L(t) = \frac{K_L(\theta_M(t) - N_G\theta_L(t))}{N_G^2} \quad (2.3)$$

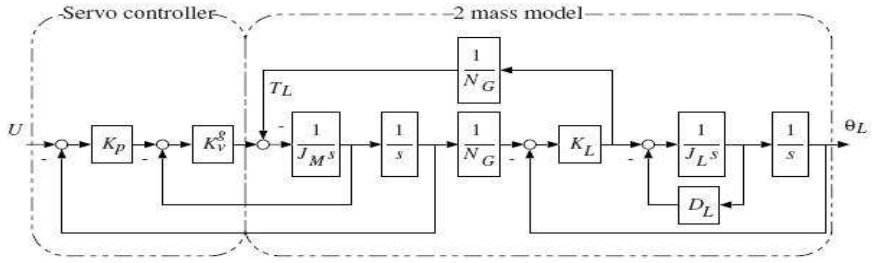
Motor tarafındaki sürtünme kayıpları çok küçük oldukları için ihmal edilmişlerdir. Denklem 1.1,1.2 Laplace transformasyonları kullanılarak çevrildiklerinde iki kütle modeli şekildeki gibi ifade edilirler.

$$\theta_M(s) = \frac{T_M(s) - T_L(s)}{J_M s^2} \quad (2.4)$$

$$\theta_L(s) = \frac{N_G}{J_L s^2 - D_L s} T_L(s) \quad (2.5)$$

$$T_L(s) = \frac{K_L}{N_G^2}(\theta_M(s) - N_G\theta_L(s)) \quad (2.6)$$

Bir önceki bölümde çeşitli kriterler ışığında göz ardı edilebilecek hususlar bu bölümde gereken yerlerin ifadesinde kullanılırlarsa sistemimiz şekildeki blok diyagramına indirgenir.” demişlerdir.



Şekil 2.13 İhmaller yapıldıktan sonra ifade edilmiş 4. mertebe sistem blok diyagramı (Nakamura ve diğer.,1998)

Motorun U(s) açı girdisinin sistemin açı çıktısı olarak ifade edebilecek transfer fonksiyonu aşağıdaki gibi ifade edilebilir.

$$G(s) = \frac{a_0}{N_G(s^4 + a_3 s^3 + a_2 s^2 + a_1 s + a_0)}$$

$$a_0 = \frac{K_L K_p K_\vartheta^g}{J_L J_M}$$

$$a_1 = \frac{K_L K_\vartheta^g}{J_L J_M} + \frac{D_L K_p K_\vartheta^g}{J_L J_M} + \frac{D_L K_L}{N_G^2 J_L J_M}$$

$$a_2 = \frac{K_L}{J_L} + \frac{D_L K_\vartheta^g}{J_L J_M} + \frac{K_p K_\vartheta^g}{J_M} + \frac{K_L}{N_G^2 J_M}$$

$$a_3 = \frac{D_L}{J_L} + \frac{K_\vartheta^g}{J_M}$$

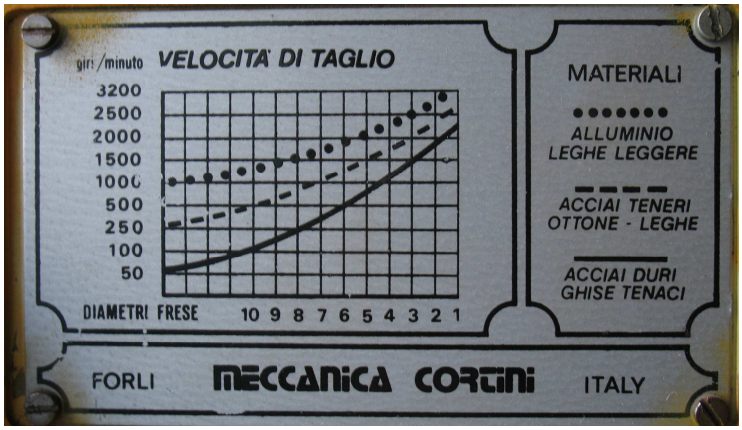
Bu 4. Dereceden transfer fonksiyonu servo parametresi tayini ve kontrol planlamasında kullanılabilir durumdadır.

Sistemin doğal frekansı ve sönüm faktörleri ifadedeki gibidir.

$$\omega_L = \sqrt{\frac{K_L}{J_L}} \qquad \varepsilon_L = \frac{D_L}{2\sqrt{J_L K_L}}$$

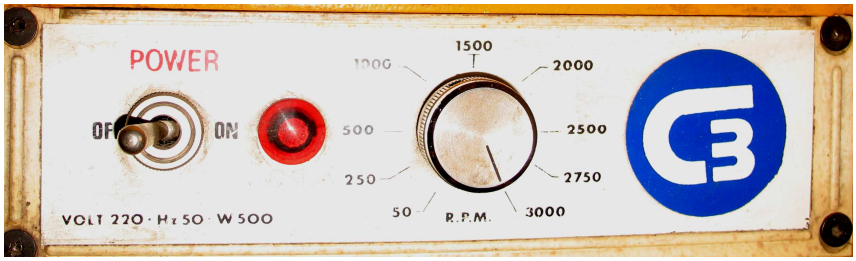
BÖLÜM ÜÇ
PROJENİN DONANIMSAL TASARIMI

Servo ünitelerin bağlanacak olduğu CNC freze tezgahı 1991 yapımı İtalyan üretimidir. Bağlanan çakının devrini iş milinin bağlı olduğu motordan, geri besleme alarak kontrol eden kontrol devresi, potansiyometre ile devri 3000dev/dak ile 50dev/dak ayar aralığında kontrol edebilmektedir. Daha önceden NC kodlarını üreten, şuan teknolojik olarak çok geri kalmış teyp kasetleriyle işleme programlarının yüklendiği ana bilgisayar ve üzerinde Berger Laher marka step motorların ve yine aynı marka sürücülerin bulunduğu CNC işlem tablası görevini yerine getiremez halde bulunmaktaydı. Tezgah üzerinden bu projede kullanılması hedeflenmeyen motorlar çıkarıldıktan sonra servo motorların bağlanabilmesi için gereken çalışma yapılmaya başlanmıştır. Önceden teknik özellikleri verilmiş olan servo motorların çıkış milleri ø14mm dış çapta ve bu çaptaki normlara uygun şekilde uygu kaması yuvasına sahiptir. sisteme eskiden bağlı bulunmakta olan step motorların çıkış milleri ø9mm dış çapta ve bu çaptaki normlara uygun olarak motor millerine kama yuvası açılmış durumdadır. Sistemde tahriği ileten ve XY kartezyen hareketlerin tablada yapılmasını sağlayan vidalı millerin, motora bağlantı girişleri step motorların çıkış millerinin içerisine girmelerine müsaade edecek şekilde delinmiş ve kama kanalı bulundurmaktadır. Yaptığımız bu çalışmada yeni servo motorların sisteme bağlanabilmesi için milde redüksiyon ve aynı zamanda kaplin vazifesi yapacak bir ara bağlantı mili ve sistem ile motorların birbirlerine bağlantısını sağlayacak olan kare flanşlar da tasarlanmıştır.



Şekil 3.1 İşlenecek malzemeye göre devrin karar verilebildiği grafik

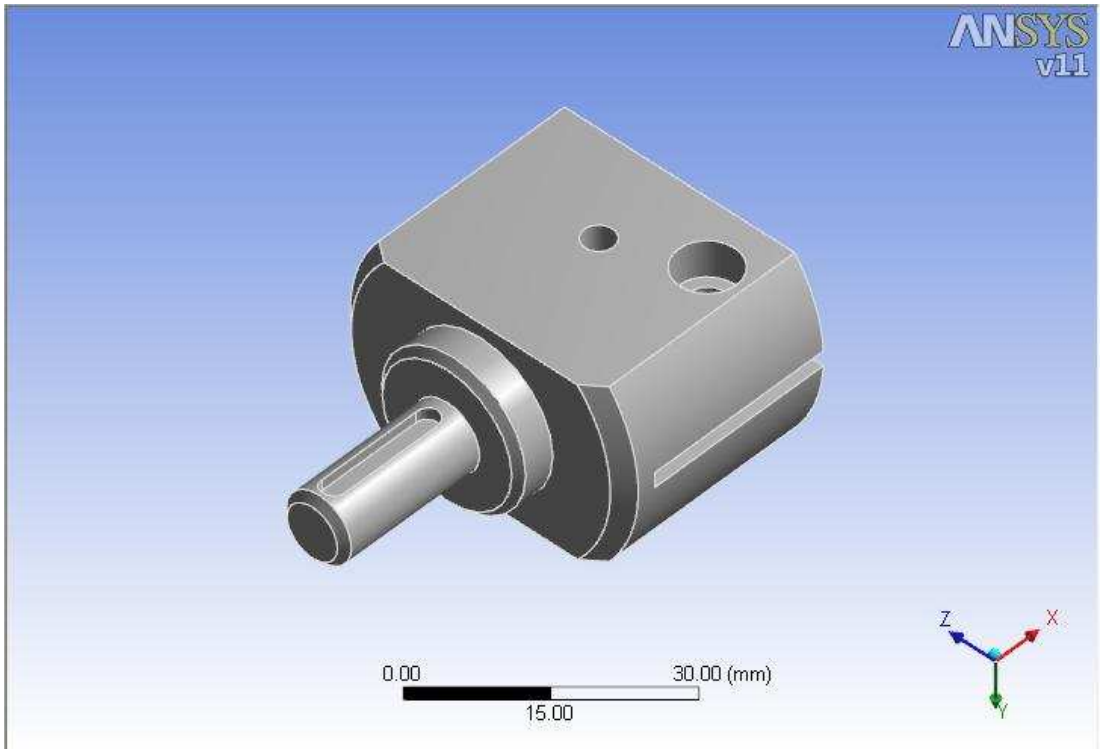
Şekil 3.2 Sistemi eskiden kontrol etmekte kullanılan bilgisayarlar



Şekil 3.3 İş milinin devir ayarını yapabilen kontrol devresi

3.1 Servo Redüksiyon Mili Tasarımı ve Analizi

Redüksiyon mili, sistemin üzerine yeni eklenecek mekanik elemanlardan en önem arz eden parçadır. Motordaki momenti ve pozisyonu motordan alarak tezgah eksenlerinin giriş millerine aktaracak olan mekanik parçalardır. Aynı zamanda kaplin vazifesi görecek ve eksenel kaçıklıkları tolere edecek olan ara bağlantı elemanı da bu olacaktır. Milin tasarımının mühendislik yaklaşımıyla yapılması için öncelikle motorun ve tezgahın fiziksel ölçüleri dikkate alınarak parçalar SolidWorks katı model programında tasarlanmıştır. Parçaların imalatı için gerekli olan mukavemet bilgilerine ise sonlu elemanlar analiz metodunu kullanan Ansys Workbench V11 programı ile yapılan analizlerden karar verilmiştir. Program üzerinde yapılan analizlerin kolay anlaşılabilmesi için bu bölümde anlatımlar program üzerinden alınan görüntülerle yapılmıştır.

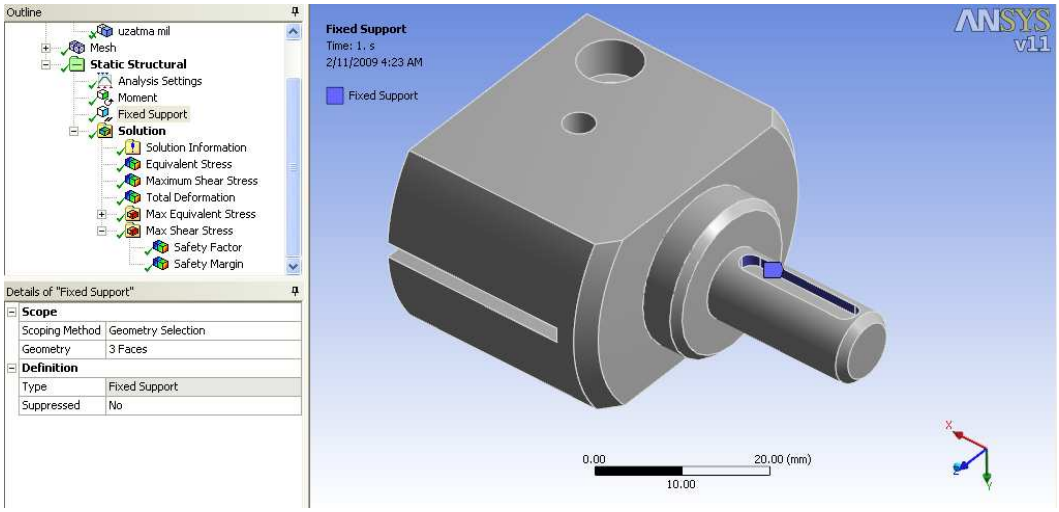


Şekil 3.4 Katı modelin Ansys Workbench V11 programında görüntülenmiş hali

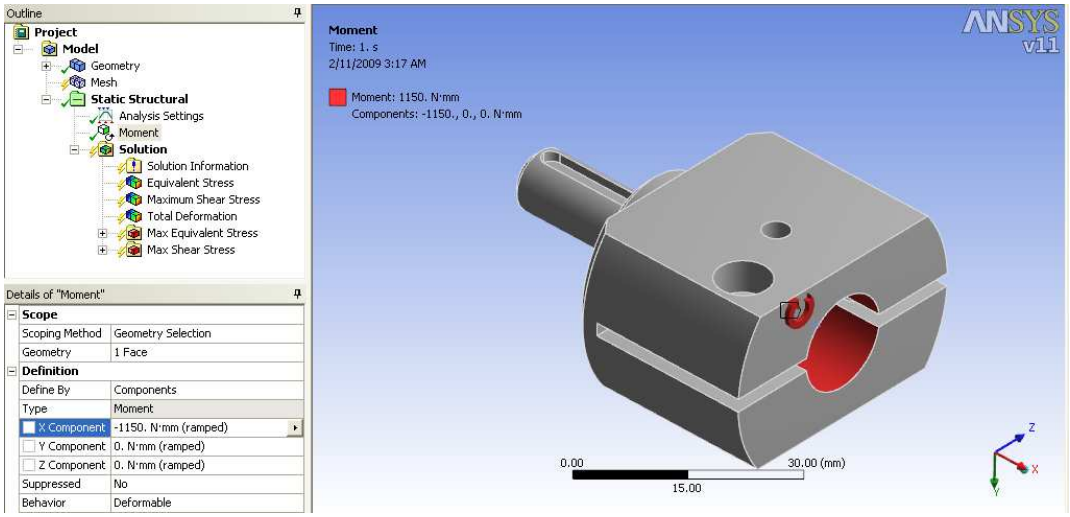
SolidWorks programında üç boyutlu çizimini yapmış olduğumuz modelin analizini yapmak üzere Ansys Workbench V11 programına aktarılmıştır. Ansys programına kolay aktarabilmek için SolidWorks menü listesinde yer alan Ansys V11 adlı menüden Workbench seçilmelidir. Program otomatik olarak modeli içeri aktararak yeni bir simülasyon dosyası oluşturmak için onay ister. Bu onay işlemi tamamlandıktan sonra parça üzerinde analizin yapılması için gereken ayarları yapan sihirbaz, analiz haritasından, istediğimiz analize ait olan “static structural analysis”, ardından sünek malzemeler için “ductile materials” radyo butonu seçilerek gereken işlem onaylanmış olur.

Analize başlamayı kolaylaştıran sihirbaz dışarıdan atanması gerekenler konusunda kullanıcıya görsel yönlendirmelerde bulunur. Bu yönlendirmelerden analiz için girilmesi gereken bilgiler girilir veya doğrulanır.

Parçaya sınır koşul olarak dışarıdan girilecek olan değerleri tanımlayacak olursak, motorun plaketi üzerinde yazan maksimum tork değeri olan 1,15Nm tezgahın bindirme durumunda kilitlenme hali düşünülerek 1150Nmm olarak etki ettirilmesi gerekir. Kilitlenmiş tezgahın vidalı mili dönemeyeceğinden, kama noktasına sabit mesnet konularak analiz için sınır koşullar belirlenir.

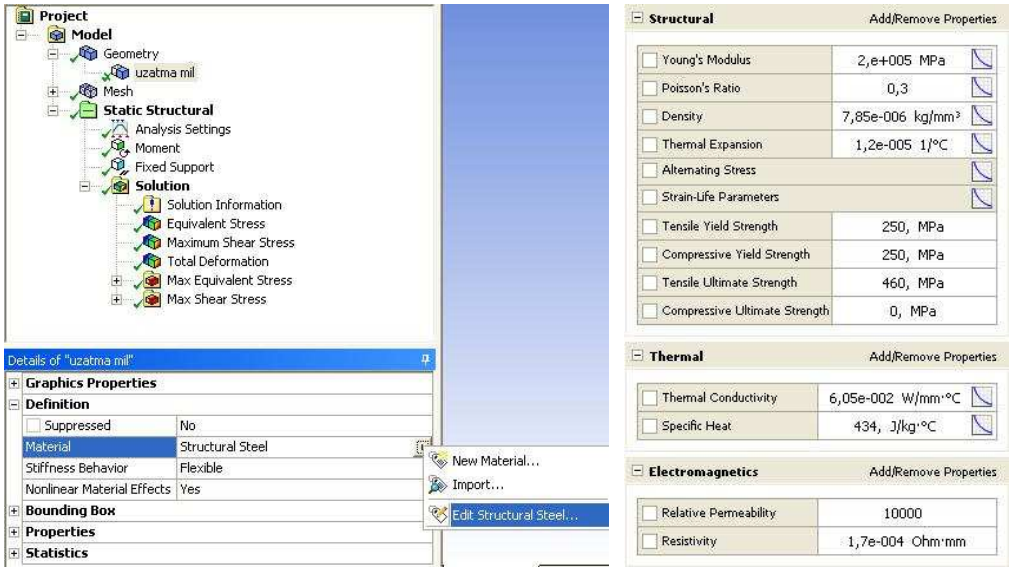


Şekil 3.5 Ankastre mesnet bölgesi



Şekil 3.6 Momentin uygulandığı bölge

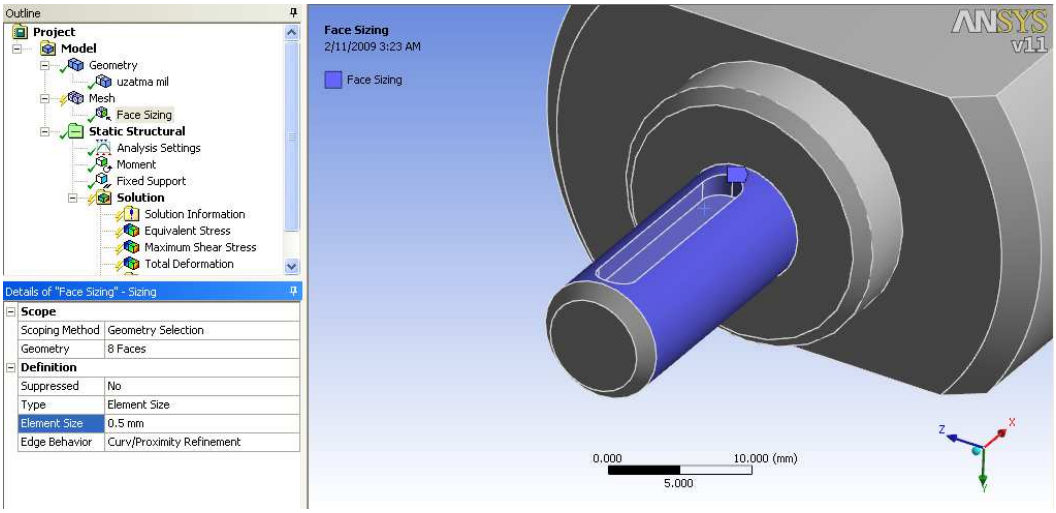
Daha sonra malzeme tanımlanması yapılacak olup, Ansys kütüphanesinde hazır durumda bulunan “structural steel” (St37) malzeme olarak atanmıştır. Bu yapısal çelik malzemenin analizden ilk fikir edinilmesi açısından kullanılmasında bir sakınca yoktur.



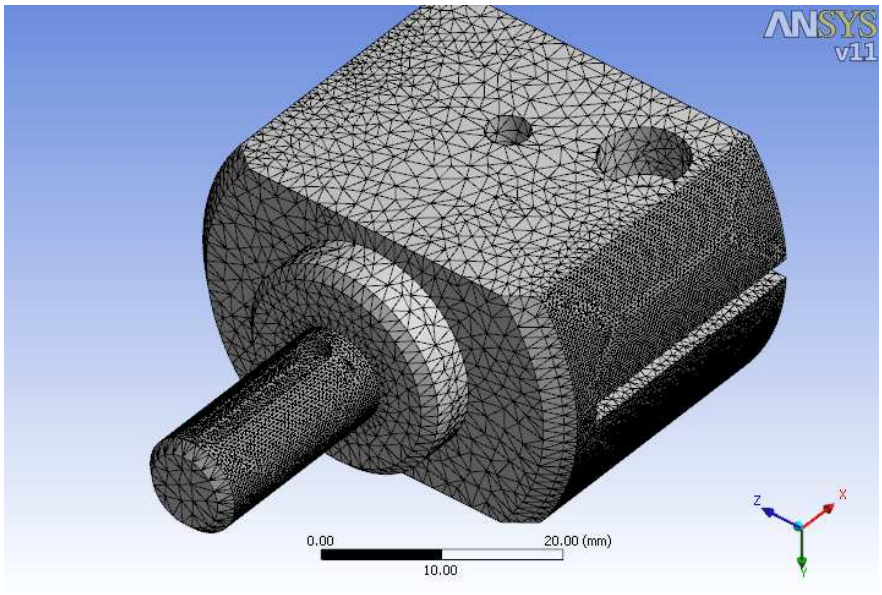
Şekil 3.7 Malzeme özelliği atanması ve yapısal çeliğin mukavemet değerleri

Sonlu elemanlar yaklaşımıyla yapılan çözümlemelerde, analiz yapılacak modellerin üzerinde mesh adı verilen ve birbirlerine nodlar (noktalar) ile bağlanmış olan ağlar üretilir. Bu ağların başlangıç noktalarından bitiş noktalarına kadar sınır koşullar ışığında sonlu elemanlar algoritmaları döndürülerek parçalar üzerindeki gerilme, uzama, ısı dağılımı v.b. analizlerin yapılması mümkün olmaktadır. Sonlu elemanlar yaklaşımı, karmaşık geometrili modellerin bu tip analizlerinin yapılabilmesini mümkün kılar.

Parça üzerinde sonlu elemanlar analizinin daha hassas olarak yapılmasını istediğimiz yerlere “mesh” bölümünden “face sizing” alt özelliği eklenerek istenen yüzeyler seçilir. Daha sonra istenen ağ aralığı da girilerek ayarlanır.



Şekil 3.8 Mesh bölümünden face sizing bölümü



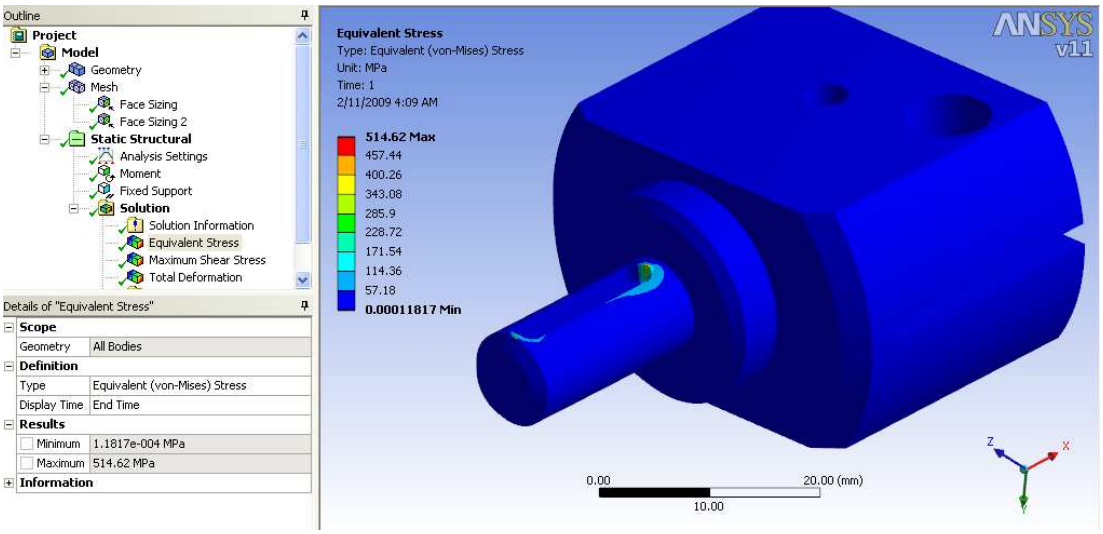
Şekil 3.9 Modelimizin sonlu elemanlar ağı (mesh), oluşturulduktan sonraki hali.

Analiz işlemi Ansys menüsünde yer almakta olan “solve” düğmesine basılarak veya model ağacının üzerinde bulunmakta olan “geometry” üzerine sağ tıklanarak gelen menüden “solve” düğmesi seçilerek yapılır.

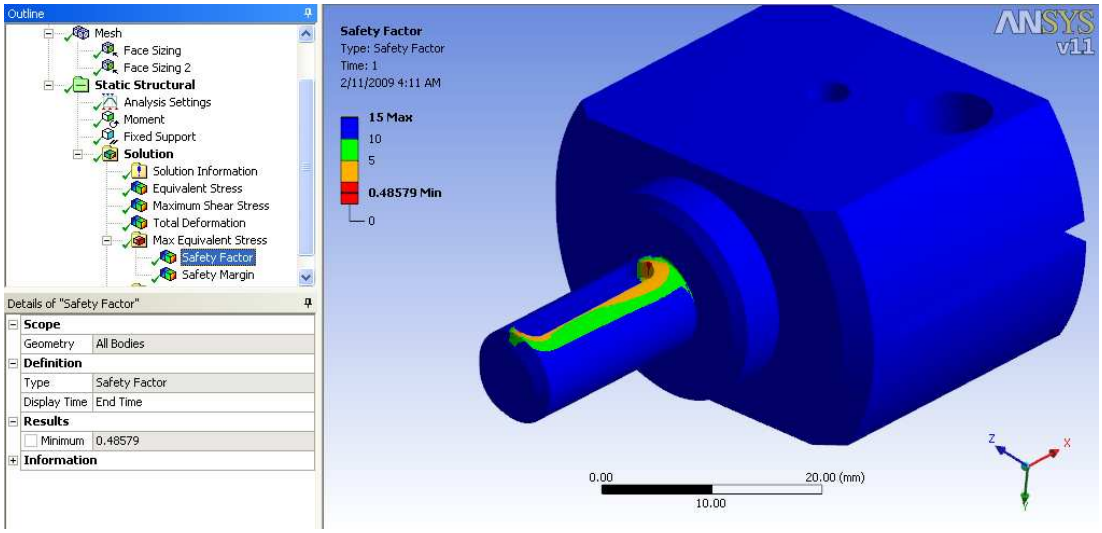
İlk analiz yapıldıktan sonra dikkat edilmesi gerekenler; değerler verdiğimiz sınır koşulları ve atanan malzeme özelliklerine göre programın hesaplamış olduğu güvenlik katsayısı değerleridir.

Parçamızın analizinden elde edilecek güvenlik katsayısının değerlendirilebilmesi için yüklemelerin doğru olarak tayin edilebildiği ve sistem yüksek hızlarda kullanılmayacağından katsayısı 1-1,5 olarak kabul edilmiştir. Güvenlik katsayısının 1'in altında çıkması kabul edilmeyecektir.

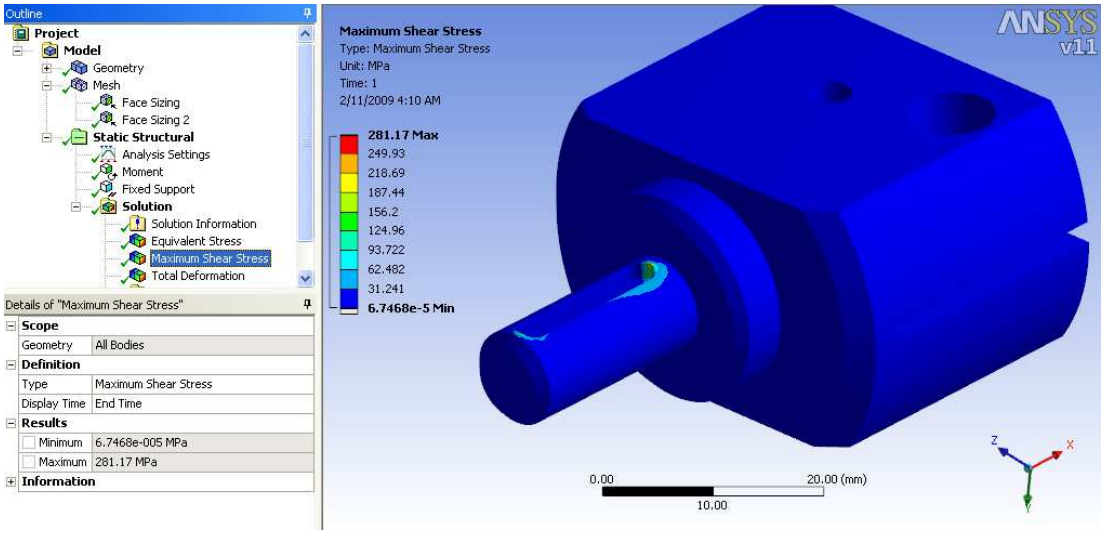
Parçanın verilen sınır koşulları ve yapısal çelik malzeme değeri için çıkan analiz sonuçları şekillerdeki gibidir.



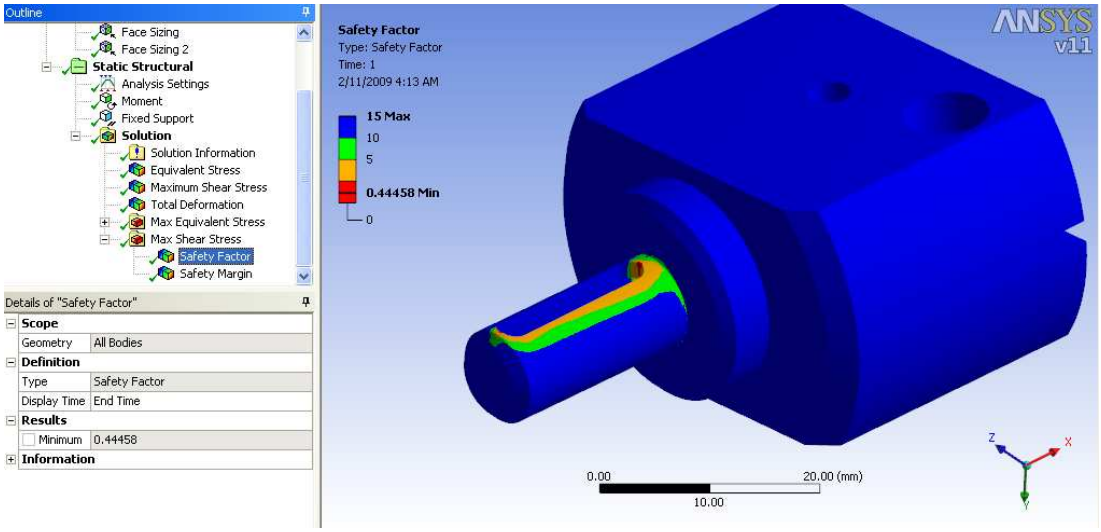
Şekil 3.10 Birinci analiz için Von-Mises kriterine göre eşdeğer gerilme değeri (MPa cinsinden)



Şekil 3.11 Birinci analiz için eşdeğer gerilmeye göre emniyet katsayısı

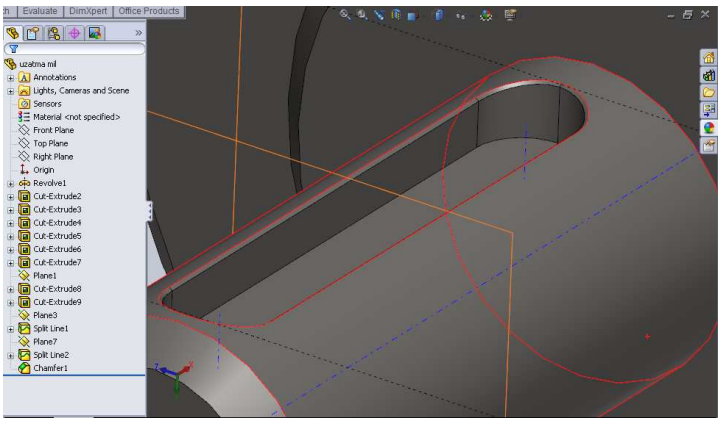


Şekil 3.12 Birinci analiz için maksimum kayma gerilmesi (MPa cinsinden)



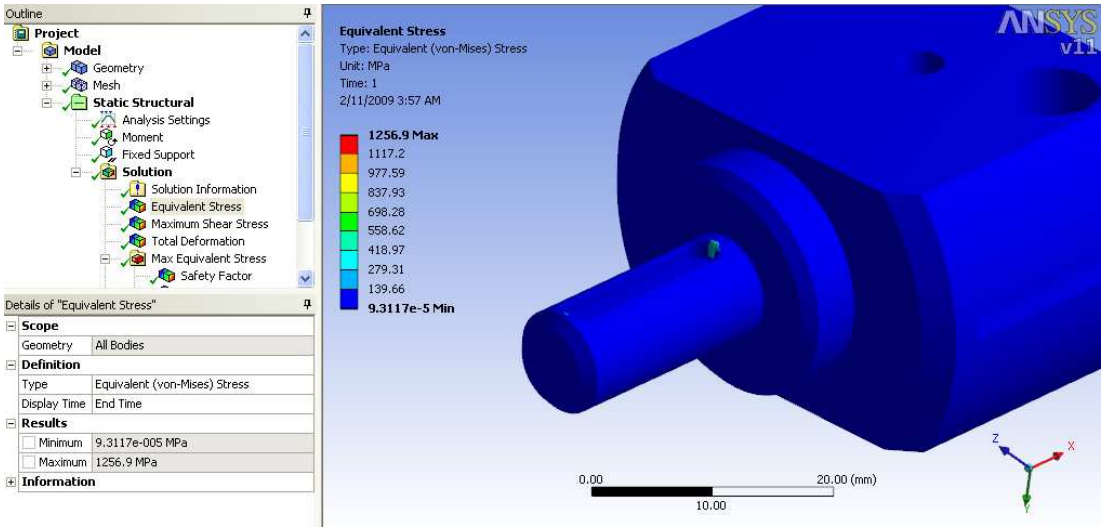
Şekil 3.13 Birinci analiz için maksimum kayma gerilmesine göre emniyet katsayısı

Şekillerde verilen analiz sonuçlarından elde edilmiş güvenlik katsayı değerleri, ilk başta açıklaması yapılmış olduğumuz, güvenlik katsayısı değerleri sınırlarında değildir. Oluşturduğumuz geometrinin üzerindeki çıkış mili çapı, giriş delik çapı gibi fiziksel ölçüleri elimizde hazır bulunan tezgah ve motor ölçülerine bağlı olduğundan ölçü değişimi gibi bir durum yapılmayacaktır. Fakat analizden de anlaşılacağı gibi tek bir noktada gerilmeler çok yüksek değerlere ulaşmakta parçanın geri kalan kısmına göre çok daha yüksek değerler göstermektedir. Bu gerilmelerin oluştuğu kısımda keskin köşelerin ortadan kaldırılması parça üzerinde bulunan muhtemel tekil noktaların önüne geçeceği düşünülmüştür. Yüksek gerilme değerinin düzelmesi için geometri üzerinde pah kırılarak iyileştirme yapılmaya çalışılmıştır.



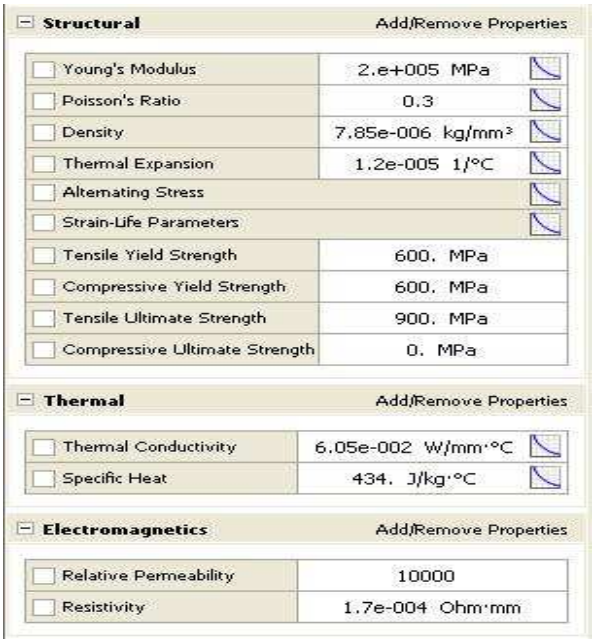
Şekil 3.14 Kama çevresine pah kırılmış yeni modelin görünümü

Parçaya bu iyileştirilme yapıldıktan sonra aynı malzeme özellikleri için tekrar analiz yapılmıştır.



Şekil 3.15 İkinci analiz için Von-Mises kriterine göre eşdeğer gerilme değeri MPa cinsinden

Şekilden de görüleceği gibi gerilme değeri ilk analizde elde edilmiş değerin iki katı değere ulaşmıştır. Bu iyileştirme gerçekçi bir sonuç değildir. Daha önceden yapılan iyileştirme kaldırıldıktan sonra üçüncü bir analiz, malzeme özellikleri iyileştirilerek tekrar yapılmıştır. Modelin üretileceği malzemenin mukavemet açısından uygun özelliklerde olması gerekir. Yeni kullanılan malzeme olarak piyasada kolay çokça bulunan yüksek karbonlu bir çelik olan C1050 kullanılmıştır.

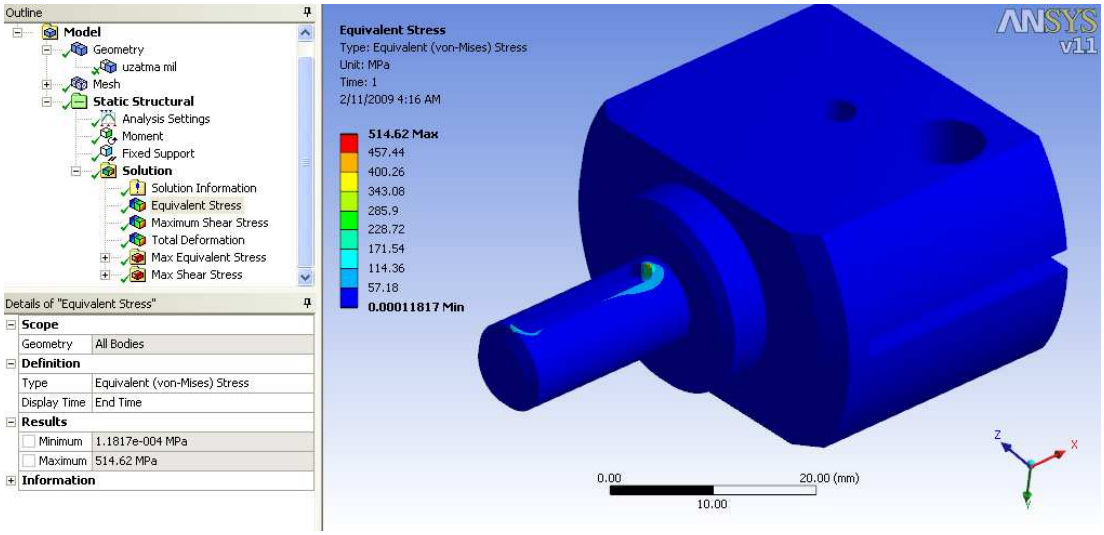


Structural	Add/Remove Properties
Young's Modulus	2.e+005 MPa
Poisson's Ratio	0.3
Density	7.85e-006 kg/mm³
Thermal Expansion	1.2e-005 1/°C
Alternating Stress	
Strain-Life Parameters	
Tensile Yield Strength	600. MPa
Compressive Yield Strength	600. MPa
Tensile Ultimate Strength	900. MPa
Compressive Ultimate Strength	0. MPa

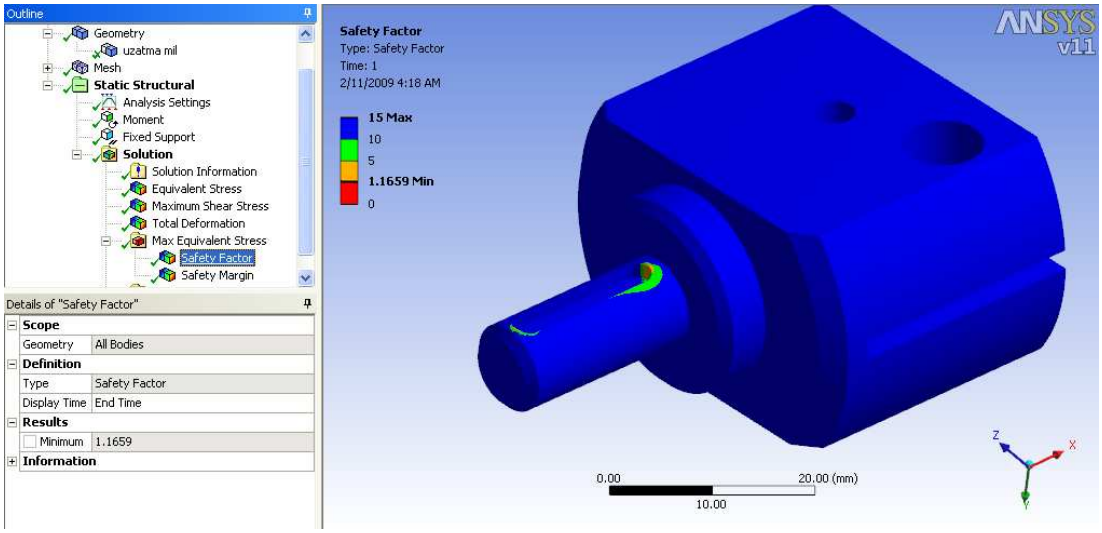
Thermal	Add/Remove Properties
Thermal Conductivity	6.05e-002 W/mm·°C
Specific Heat	434. J/kg·°C

Electromagnetics	Add/Remove Properties
Relative Permeability	10000
Resistivity	1.7e-004 Ohm·mm

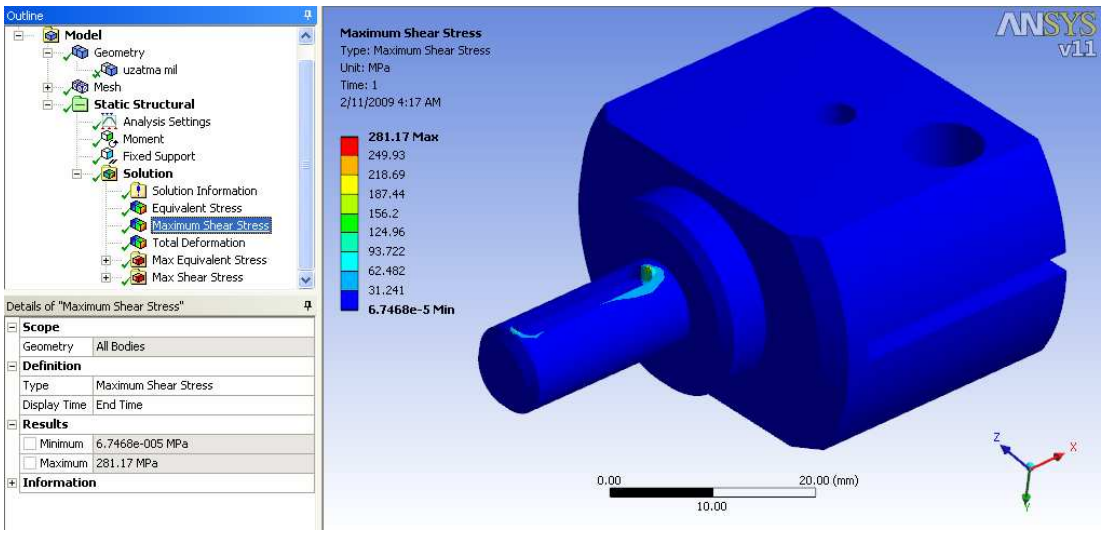
Şekil 3.16 C1050 karbonlu çelik malzeme mukavemet değerleri



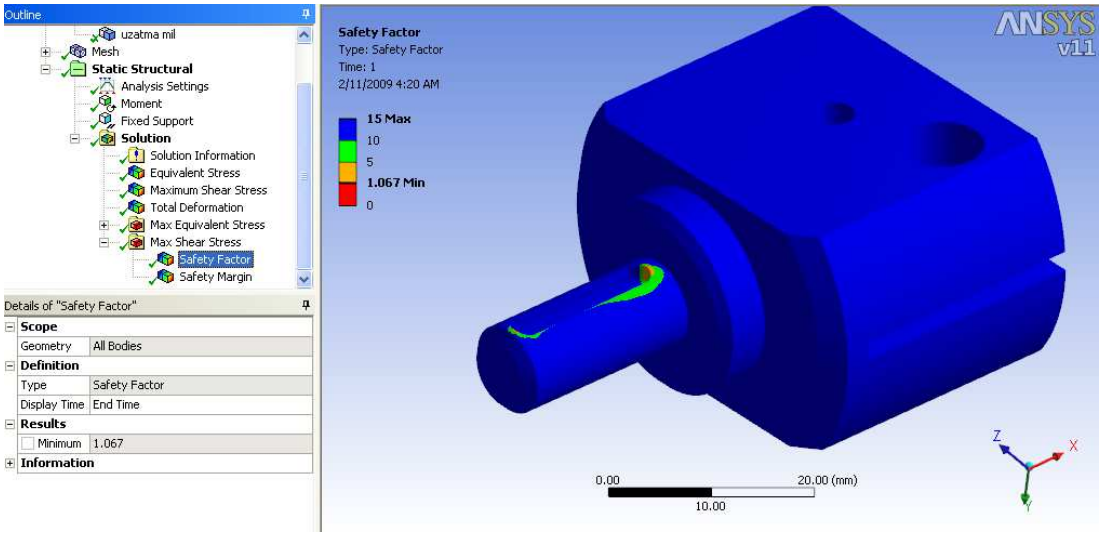
Şekil 3.17 Üçüncü analiz için eşdeğer gerilme MPa cinsinden (Von-Misses kriterine göre)



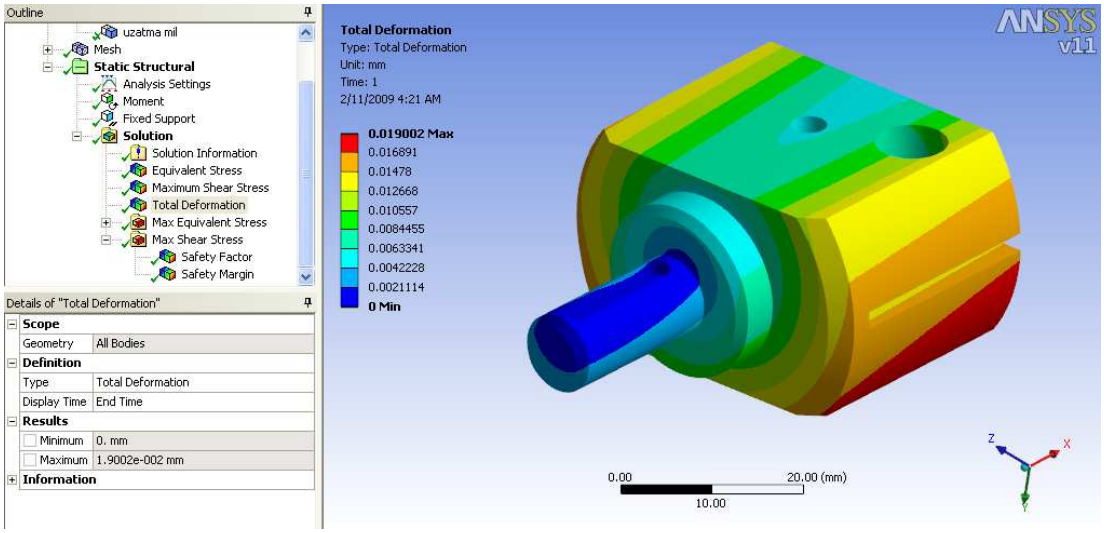
Şekil 3.18 Üçüncü analizin eşdeğer gerilmeye göre güvenlik katsayısı



Şekil 3.19 Üçüncü analiz için maksimum kayma gerilmesi (MPa cinsinden)

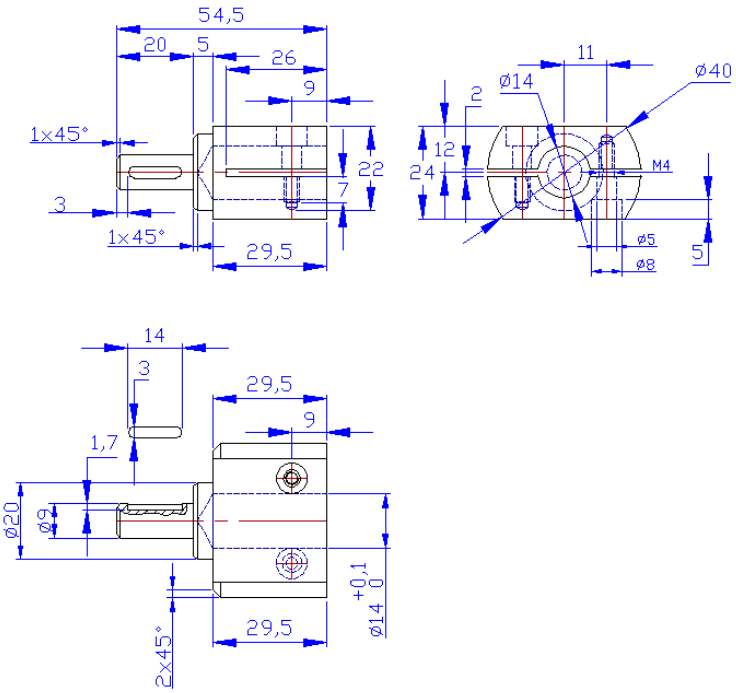


Şekil 3.20 Üçüncü analiz için maksimum kayma gerilmesine göre emniyet katsayısı



Şekil 3.21 Üçüncü analiz için mm cinsinden toplam şekil değiştirme.

Analizden çıkan sonuçlar doğrultusunda istenilen değerlere yakın emniyet katsayıları elde edilmiştir. Bu sayede en son seçilen malzemenin uygun olduğuna ve modelin bu malzemeden imalatının yapılmasına karar verilmiştir.

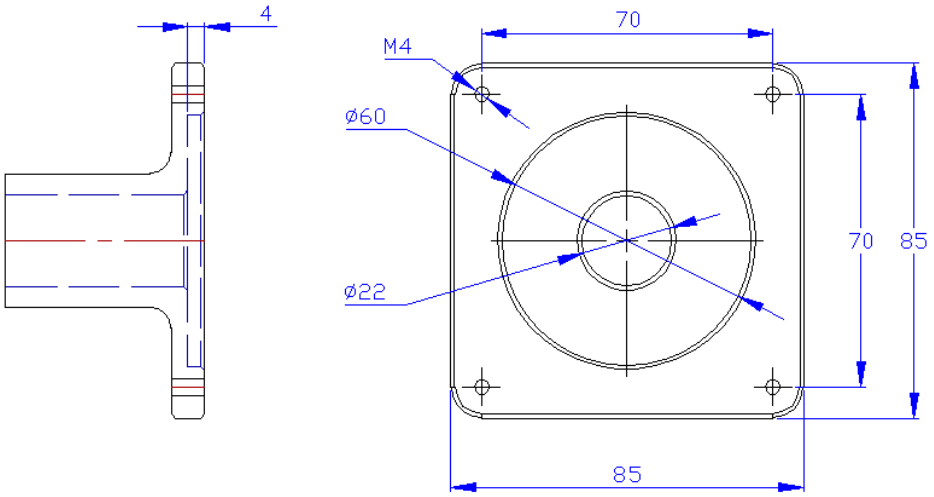


Şekil 3.22 Tasarım sonucunda üretilecek olan parçanın imalat resmi

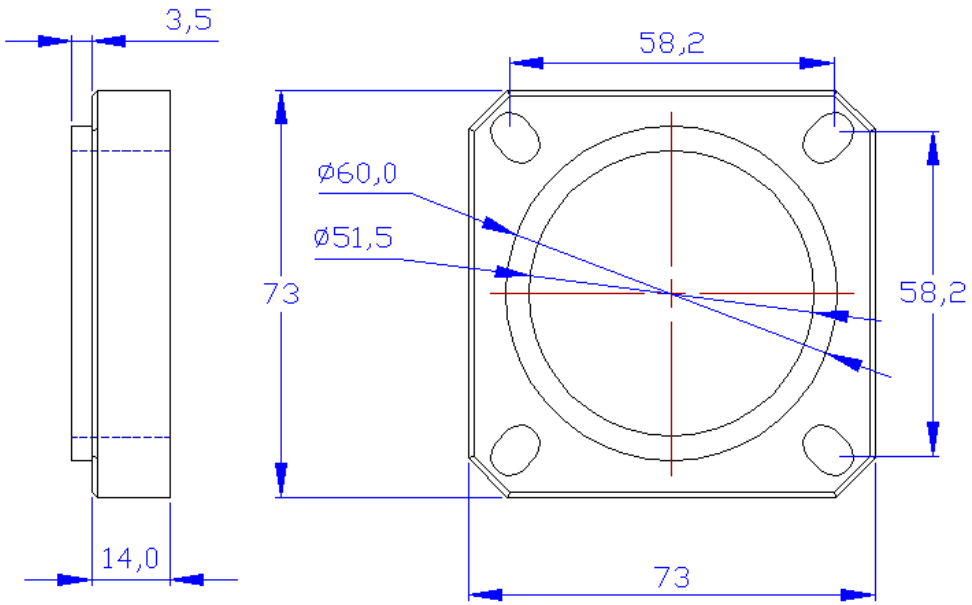
3.2 Bağlama Flanşı Tasarımı

Tezgah üzerinde daha önceden step motorların takılı bulunduğu kare şeklindeki motor bağlama flanşı ile yeni motorun gövdesinin birbirlerine bağlanabilmesi için bu iki parçanın arasına ölçülerine göre tasarlanmış, içerisine redüksiyon milinin girebileceği uzunluk ve çapta kare şeklinde bir flanşla hepsi bir araya getirilmek istenmiştir. Bu doğrultuda yapılan çalışma ile bu flanş tasarlanmıştır.

Bu flanş, mukavemet bakımından hassas hesapların yapılmasını gerektirecek bir yapıda olmayıp üzerine etkiyecek olan 1,15Nm'lik momenti kolaylıkla üzerinde kullanılacak olan civatalar yardımı ile taşıyacaktır. Bu nedenle flanş sonlu elemanlar analizi ile değil diğer parçaların geometrisine göre modellenmiştir.

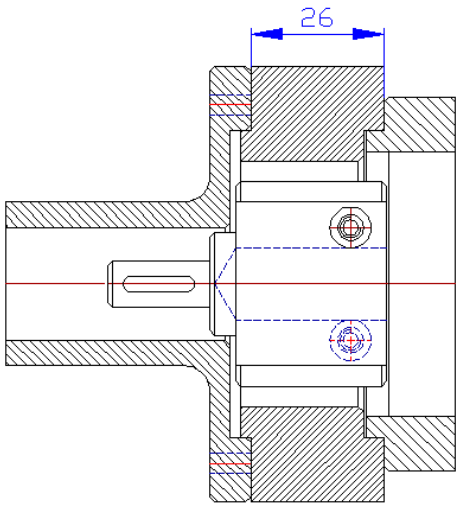


Şekil 3.23 Tezgah üzerinde mevcut olan kare şeklindeki flanş

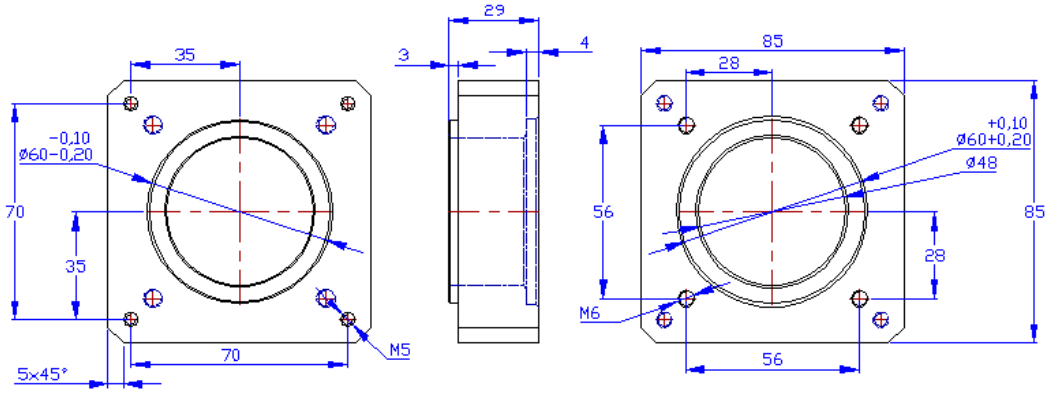


Şekil 3.24 Servo motorun ön bağlama flanşı

Redüksiyon mili tasarımı, tezgah bağlama flanşı ve motor flanşı bir araya getirilerek oluşturulan montaj resmi ile gereken ölçüler belirlenmiştir.



Şekil 3.25 Ölçü çıkarmak için oluşturulmuş montaj resmi

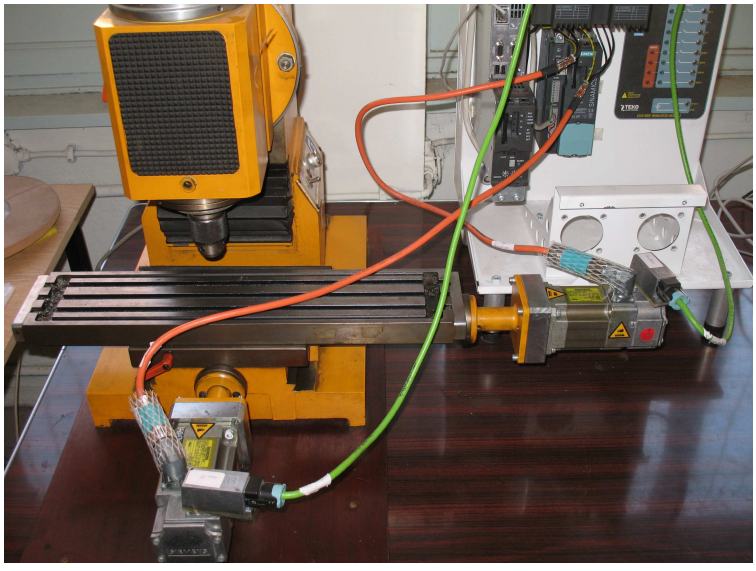


Şekil 3.26 Kare flanşın ölçüleri

3.3 Tezgahın Montajdan Sonraki Son Durumu

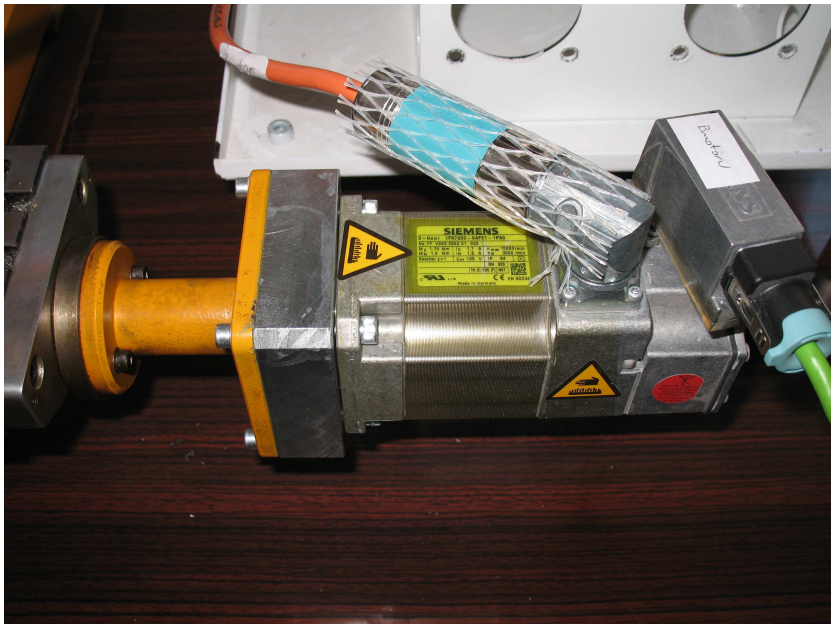
Tasarımı ve imalatı yapılan parçaların, servo motorların, kontrol ünitesinin ve bilgisayarın hepsi bir araya getirilerek sistem montaj edilmiştir.

Şekil 3.27 Montajdan sonraki durum



Şekil 3.28 Montaj sonrası yakın görünüm

Şekil 3.29 Redüksiyon milinin motora bağlanmış hali



Şekil 3.30 Bütün parçaların tezgaha montajlanmış hali

BÖLÜM DÖRT
PROJENİN YAZILIMSAL TASARIMI

4.1 Scout Ara Yüzü, Motorların Sisteme Tanıtılması ve Diğer Ayarlar

Structural Text programında yazılan ara fonksiyonlar, dışarıda şekil bilgilerinden interpole edilmiş hareket komutlarını sistem içerisine aktardıktan sonra okunabilmesi için düzenlenmiştir.

Scout programının ayarlarının yapılması ve sistemin çalışır hale gelmesini sağlamak, bilgi sahibi olmayan biri için tek başına araştırılıp, bulunup ve öğrenilmesi uzun bir süreçtir. Siemens'in birçok özelliği tek program altına modüler olarak koyduğu Scout programında istenen özellikler paketler halinde yüklenmektedir. Bu beraberinde birçok komut bilgisi sahibi olmayı gerektirir. İnternet üzerinde ve Siemensin uygulamalarında yeteri kadar örnek bulunmamaktadır. Yalnız Scout programının çalışma mantığının bir defa anlaşılmasından sonra aslında yapısının basit fakat içerisinde çok fazla ayrıntının olduğu görülür.

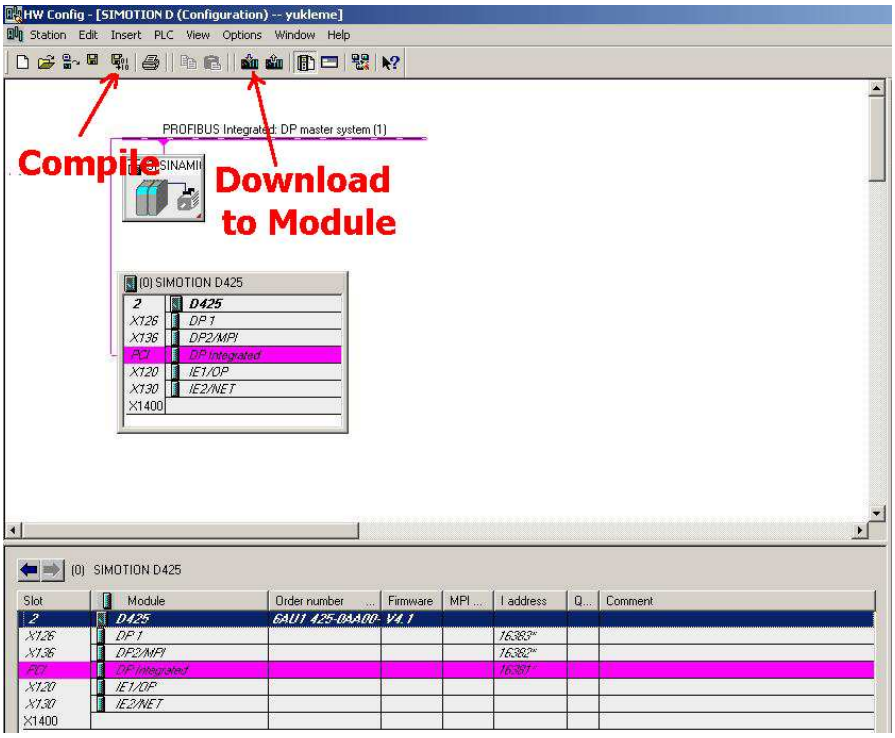
4.1.1 Scout İçerisine Sistemin Yüklenmesi ve Ayarlarının Yapılması

Scout programında yeni bir proje açılarak işleme başlanır. Yeni bir donanımın yüklenebilmesi için karşımıza gelen ağaçtan "Create new device" daha sonra gelen menüden "CPU type: D425 v4.1" ve "Variant" kısmından "Sinamics S120 integrated V2.5 in D425" seçilerek onaylanır.

Bu işlem bittikten sonra soldaki sistem ağacının içerisine yüklemek istediğimiz D425 ünitesi ve Sinamics S120'nin yüklenmiş olduğunu görürüz. Bu yüklemenin tamamlanabilmesi için ağ ayarlarının yapılarak sistem ile bilgisayarın haberleşebilir hale getirilmesi gerekmektedir. Bilgisayara daha önceden sistem ile aynı seviyeli bir IP adresinin statik olarak tanıtılmış olması gereklidir. Örnek verecek olursak D425'in IP adresini ileride 169.254.11.22 şeklinde tanımlayacağız, buna göre bilgisayarın IP

adresinin daha önceden 169.254.11.25 gibi bir IP olarak atanması gereklidir.

Bu işlemler yapıldıktan sonra karşımıza “interface selection” bölümü gelir. Buradan sistemle bilgisayarı hangi arabirim üzerinden haberleştireceğimizin seçilmesi gerekmektedir. Biz “Ndiswanlp” arabirimini seçerek onaylarız sistem otomatik olarak “HW config” arabirimine yönlenir.



Şekil 4.1 HW config arabirimi, compile ve donwload to module butonu

“HW config” bölümünde ayarlar otomatik oluşturulduğu için bilgilerin derlenip D425’e aktarılması gerekmektedir. Bunun için yukarıda ekranın solunda “Save and Compile” düğmesine tıklanarak yapılan donanımsal çalışmanın saklanarak derlenmesi sağlanır. Daha sonra bu sistem bilgilerinin D425’e yüklenmesi gereklidir. Bunun için “Download to Module” butonuna tıklanarak sistem bilgisinin D425’e gönderilmesi için gereken ayarların yapılmasına başlanmış olunur. Bu durumlar olurken D425 ünitesi üzerinde 6 rakamı yanıyor ve yanında bir nokta yanıp sönüyor ise sistem normal durumda komut alımına hazır bekliyor demektir.

Tablo 4.1 D425 üzerinde bulunan LED'lerin gösterimi ve anlamları tablosu (Siemens, 2007)

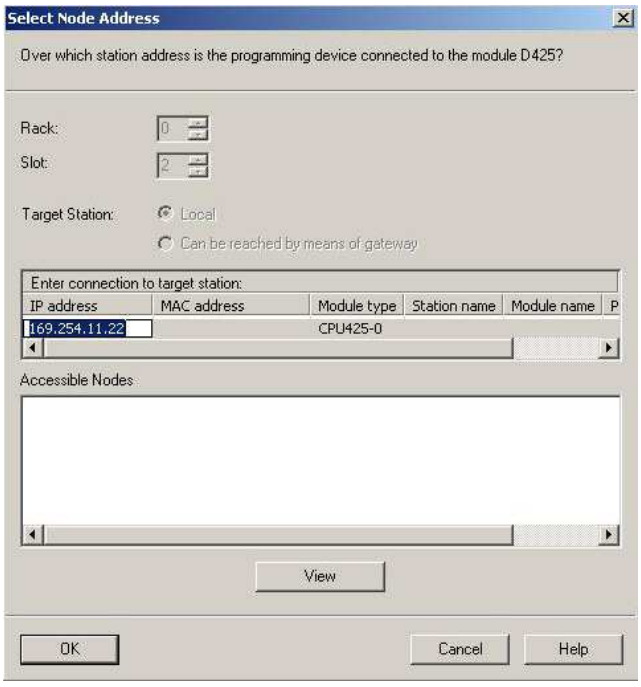
Anlamı	**LED gösterimi**							
	RDY	**RUN**	**STOP**	**STOPU**	**SF**	**DP1**	**DP2**	**OPT**
Hız verme	1 (Sarı)	1 (Sarı)	1 (Sarı)	1 (Sarı)	1 (Sarı)	1 (Sarı)	1(Sarı)	1(Sarı)
CF kartı takılmamış veya kayıp	1 (Sarı)	1 (Sarı)	1 (Sarı)	1 (Sarı)	1 (Sarı)	1 (Sarı)	1(Sarı)	1(Sarı)
Herhangi bir hata oluştuğunda CF kartı baştan başlatılmalı.(Hatalı CW)	0.5/1 (Krmz.)	0	0	0	0	0	0	0
D4*5 çalışmaya hazır: SIMOTION hizmet sistemi çalışıyor ve SINAMICS entegresi hizmet etmeye hazır.	1 (Yeşil)	x	x	x	x	x	x	x
SINAMICS entegresi başlatılmamış(SINAMICS gömülü yazılımı hazır değil veya hatalı), veya bir hata oluşmuştur.	1 (Krmz.)	x	x	x	x	0	0	x
CF kartı okuma veya yazma erişimi.	^ (Sarı)	x	x	x	x	x	x	x
RUN	x	1 (Yeşil)	0	0	x	x	x	x
RUN-STOPU bağlantısı	x	1 (Yeşil)	0	2/1 (Sarı)	x	x	x	x
STOPU- RUN bağlantısı	x	2/1 (Yeşil)	0	1 (Sarı)	x	x	x	x
STOPU	x	0	0	1 (Sarı)	x	x	x	x
Servis durumu	x	2/1 (Yeşil)	0	2/1 (Sarı)	x	x	x	x
STOPU- STOP bağlantısı	x	0	2/1 (Sarı)	1 (Sarı)	x	x	x	x
STOP	x	0	1 (Sarı)	0	x	x	x	x
STOP- STOPU bağlantısı	x	0	1 (Sarı)	2/1(Sarı)	x	x	x	x
Durum seçicisi yolu ile veya D4*5'in kendi kendine tamamen baştan başlatılması isteği	x	0	0.5/1 (Sarı)	0	x	x	x	x
Tamamen baştan başlatılması süreci	1	0	0	0	0	0	0	0
Tamamen baştan başlatılmasının tamamlanması	1 (Yeşil)	0	1 (Sarı)	0	x	x	x	x
Bir uyarı (alarm, mesaj,not)beklemede olarak tanınır	1 (Yeşil)	x	x	x	1 (Krmz.)	x	x	x
Kullanılan (SIMOTION) programının cevap veremediği bir hata arz ederse, hatayı düzeltmek için takip eden işlemleri yapmak gerekir: • Güç AÇIK/KAPALI • CF kartı kontrol etmek • Yeni prosedürü devreye sokmak • D4*5'i tekrar yerleştirmek	^ (Krmz.)	^ (Krmz.)	^ (Krmz.)	^ (Krmz.)	^ (Krmz.)	^ (Krmz.)	^ (Krmz)	^ (Krmz)
Teknolojiyi alt lisanslama/ opsiyonel objeler	1 (Yeşil)	x	x	x	0.5/1 (Krmz)	x	x	x

LED durum anahtarı: LED'ler farklı çalıştırma durumları ve Simotion D4*5'de meydana gelen herhangi bir hatayı göstermekte kullanılmaktadır. Bunları aydınlatma, flaş çakma veya farklı renklerde yanıp sönme şeklinde olabilir. Tabloda kullanılan sembollerin anlamları Tablo 4.2'deki gibidir. Her LED her renkte yanabilir.

Tablo 4.2 LED anahtarı (Siemens, 2007)

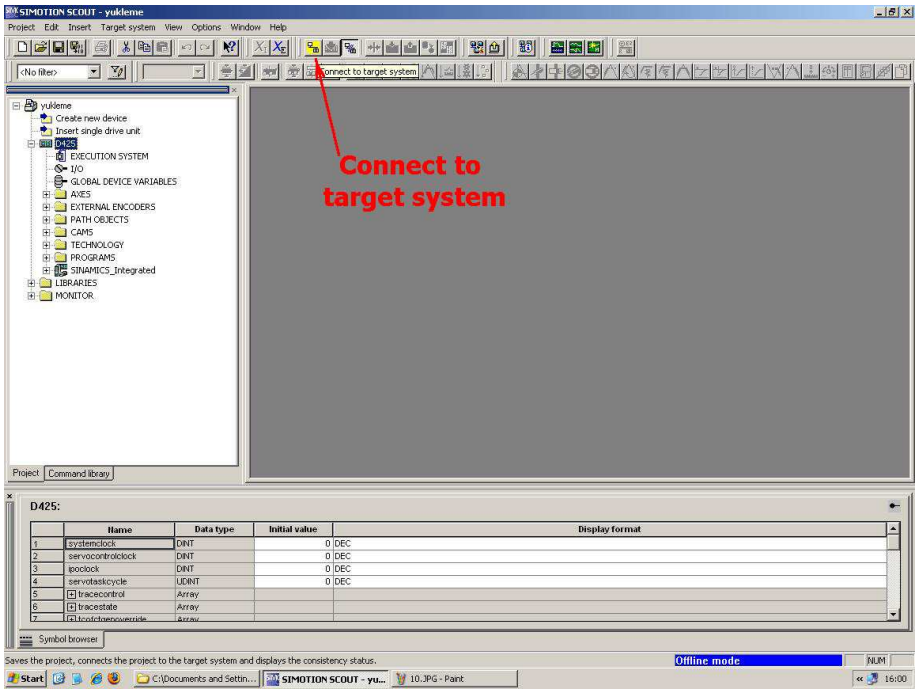
1	LED açık
0	LED kapalı
0.5/1	flaş çakan LED (0.5 Hz)
2/1	flaş çakan LED (2 Hz)
^	yanıp sönen LED
X	LED aydınlatabilir

Gelen “Select Target Module” menüsünden D425 seçilir ve onaylanır. Karşımıza “Select Node Address” menüsü gelir. Bu bölümden D425’in verecek olduğumuz IP adresi 169.254.11.22 girilir. “Select Node Address” menüsü onaylandıktan sonra D425’in “stop mod” da olduğunu başlatılıp başlatılmayacağını soran ekran gelir. Bu menü de onaylanır. Bu şekilde ağ tanımlaması bitirilir.



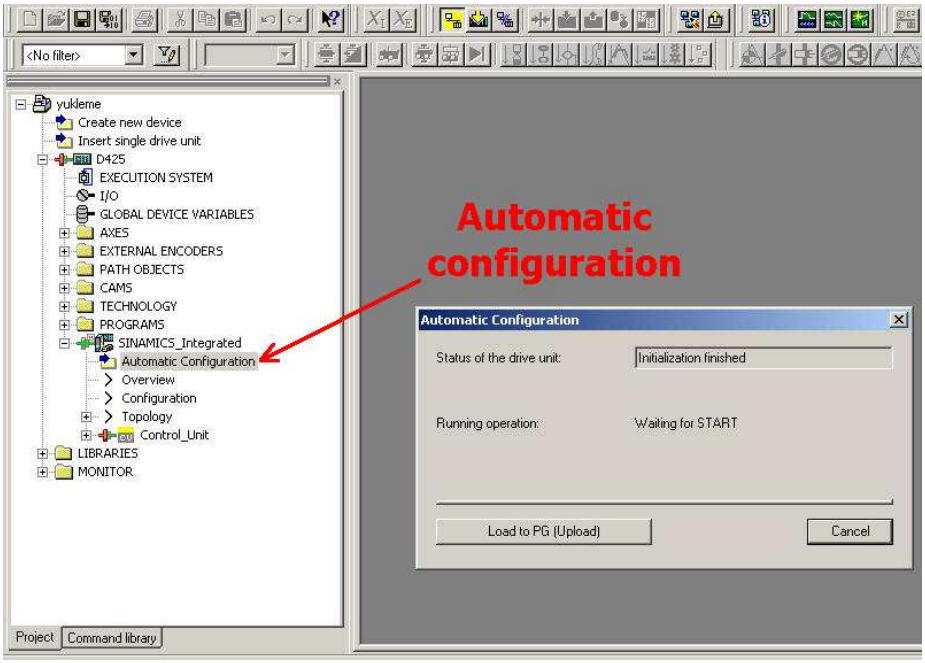
Şekil 4.2 D425 IP adresi tanımlaması

D425'e gereken ayarlar yapıldıktan sonra "HW Config" menüsü kapatılarak D425'in bilgisayar iletişiminin kurulması işlemine geçilir. Bu işlem için "Simotion Scout" proje ekranından "Connect to target system" düğmesine basılır ve iletişim sağlanır.



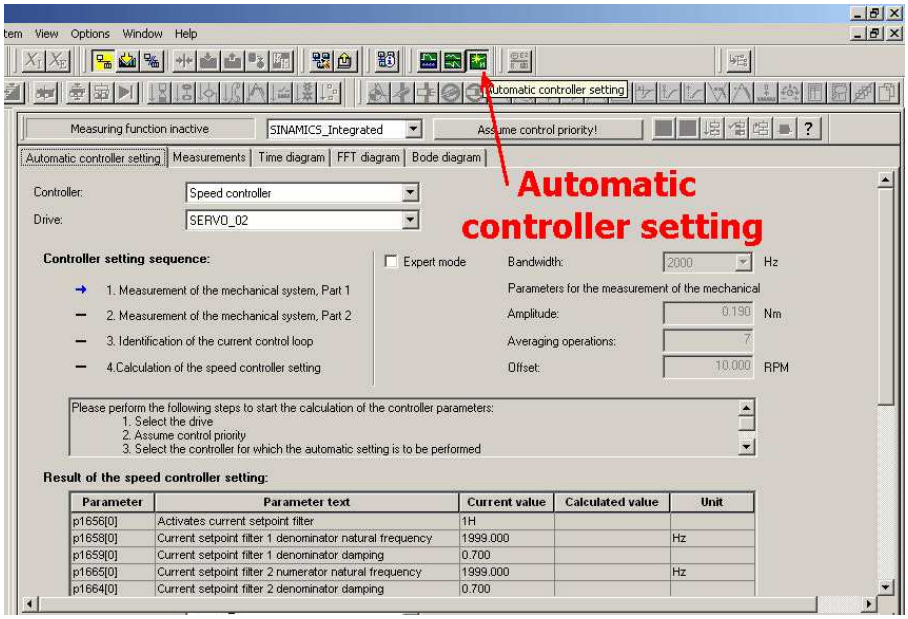
Şekil 4.3 Simotion Scout proje ekranı connect to target system düğmesi

Sistemin otomatik olarak kendi kendini konfigüre etmesi için, üzerinde bağlı donanımların tanınması işleminin başlatması gereklidir.



Şekil 4.4 (Automatic configuration) otomatik ayarlama menüsü

“Automatic configuration” düğmesine tıklandıktan sonra sistem otomatik olarak üzerinde takılı motorları ve sürücüleri bulur. Bu işlemden sonra artık takılmış olan servo motorların bağlanan mekanik sisteme göre kazanç katsayısı gibi sistemden sisteme değişecek değerlerin belirlenmesi için “Automatic controller setting” menüsünden girilen menüden ayarlar otomatik olarak yaptırılmaya başlanır.



Şekil 4.5 Servo motorların ve sürücülerinin kazançlarının hesaplatıldığı automatic controller setting menüsü

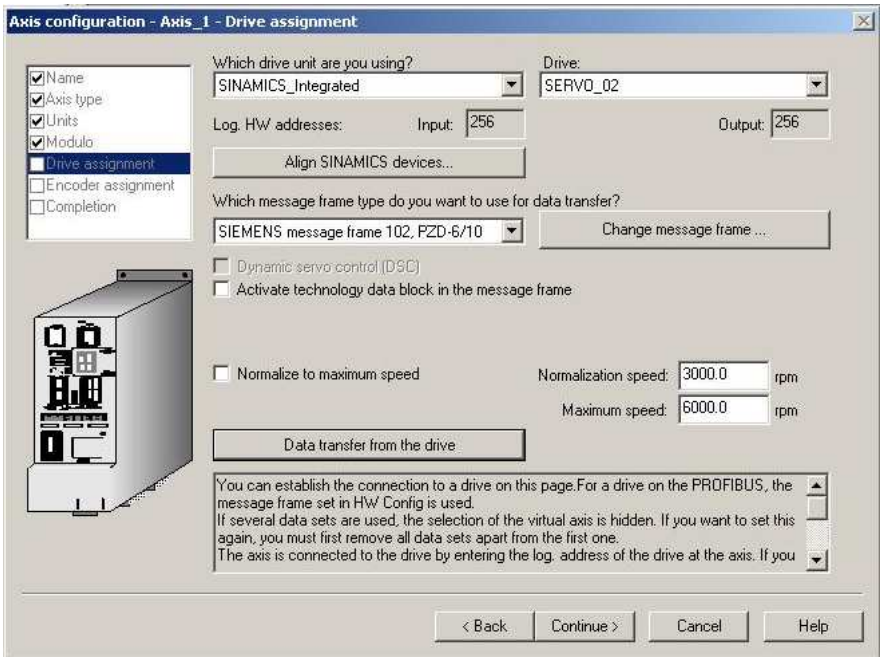
Gelen menüde de yazdığı gibi öncelikle motor seçimi “Drive” bölümünden yapılır. Daha sonra “Controller” bölümünden “Speed Controller” seçilir. Bu işlemler yapıldıktan sonra “Assume control priority” düğmesine basılarak makinenin sürücülerin kontrolünü devralması “1” düğmesine basılarak sağlanır. Bundan sonra “0” düğmesinin sağ tarafında bulunan “Perform all Steps” adlı düğmeye basılır ve sistemin test edilmesi için gönderilecek komut sinyallerinin frekansının yazılı olduğu menü onaylanarak işlem süreci başlatılır. Bu işlemler yapılırken sistemdeki zorlamaların hesaba katılabilmesi için motorların tezgah üzerinde bağlı olması şarttır.

Mekanik sistemin ölçümü 1 ve 2 işlemleri tamamlandıktan sonra gelen menüde onaylanır ve bu sefer “Identification of Current Control Loop” ve son adım olan “Calculation of The Speed Controller Setting” işlemleri de tamamlanır. Bu işlem adımları sürerken motor düzensiz titremeler ve hareketler yapacaktır.

Tanımlama işlemi bittikten sonra bulunulan menünün en alt kısmında bulunan “Accept” düğmesine basılarak yapılan ayarların kabulü sağlanır. Aynı işlemler diğer servo motor için de yapılarak “Accept” düğmesine basılır. “Give up Control Priority”

düğmesine basılarak işlemler sonlandırılır ve daha sonra sistem online duruma geçirilerek iki taraftaki bilgilerin de eşitlenebilmesi için “Download to Module” düğmesine basılarak yapılan işler D425’in hafızasına yazılır.

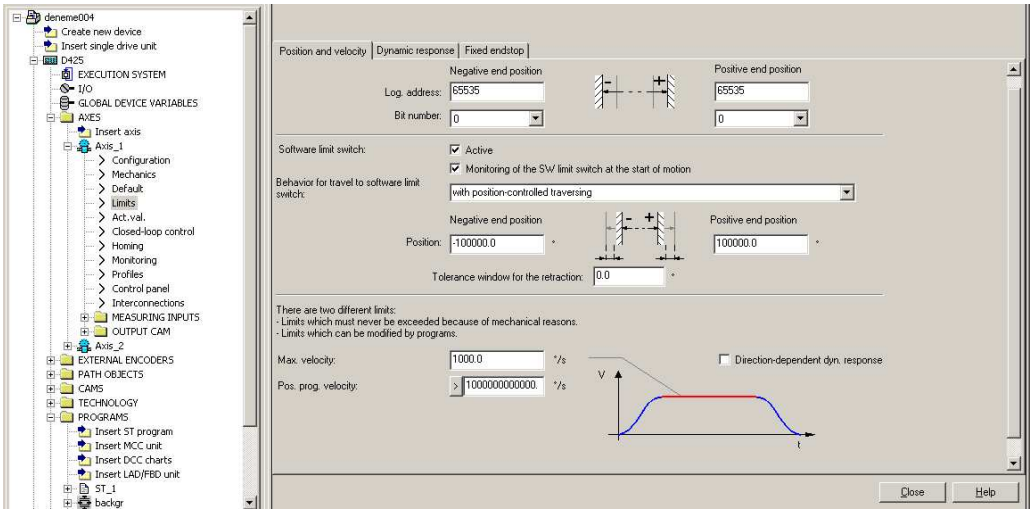
Bu işlemler tamamlandıktan sonra motorların işlemsel olarak kullanılabilmesi için birer eksen olarak tanımlanmaları şarttır. Bundan önce programda kullanacak olduğumuz teknoloji paketini belirlememiz gereklidir. Bunu konfigürasyon ağacındaki D425’in üzerine sağ tıklayarak gelen menüden “Select Technology Packages” bölümünden “CAM(V4.1)”in seçili olduğundan emin oluruz. D425’in alt dalı olan “Axis” bölümünden “Insert Axis” bölümü seçilerek buradan motorların eksenlere atamaları yapılmaya başlanır. “Insert Axis” düğmesine tıklandıktan sonra karşımıza gelen menüden “Speed Control” ve “Positioning” seçilir ve onaylanır. “Axis Configuration”dan “Axis Type: Rotary”, “Electric, Mode: Standart”, “Motor Type: Standart Motor” seçildikten sonra “Continue” tıklanır. Diğer bölümlerde birim olarak metrik birim sistemi seçilir. “Drive Assignment“dan “Align Sinamics Devices” buradan gelen menüden “Sinamics Integrated Align” düğmesine basılarak seçilir. Bundan sonra gelen menüde motorun yapacağı işleme göre belirlenebilen tipi atanabilmektedir. Bu bölüme açıklama getirilecek olursa: “Massage Frame Type” kısmında bulunan tipler; 102 yazan dinamik servo kontrol olmadığı zaman, 103 bu işlem için ikinici bir enkoder yüklenecekse seçilmelidir. 105 dinamik servo kontrol varsa ve 106 dinamik servo uygulaması için ikinci bir enkoder yüklenecekse seçilmelidir. Yapacağımız işlemde dinamik servo kontrol kullanmadığımızdan gereksiz yere CPU kullanılmaması için 105 veya 106 seçilmemelidir. “Data Transfer From The Drive” düğmesine tıklanarak motorun daha önceden otomatik olarak hesaplanan değerleri burada eksene yüklenir. Daha sonra gelen “Encoder Assigment” bölümünden yine sisteme sürücüden bilgi transferi yaparak motor üzerinde takılı olan resolver burada sisteme tanıtılır.



Şekil 4.6 Drive assigment bölümündeki massage frame tipleri

Eksen tanımlaması yapıldaktan sonra tezgahın güvenliği için eksenlerin gidebileceği maksimum mesafeler veya ivme, hız gibi değişkenlerin alabilecekleri maksimum değerler olan limit değerleri, her eksen için “Axis” kısımlarının “Limits” bölümünde tanımlanabilmektedir.

Artık sisteme programların yazılması ve yapılacak işler (Execution tasks) bölümüne programların atanması işlemleri yapılabilir.



Şekil 4.7 Eksen limitlerinin tanımlandığı “limits of axis” kısmı

4.2 Scout’a Programların Yazılması:

Scout ara yüzüne dışarıda yazmış olduğumuz çizim bilgilerinin sistem içerisinde okunabilir hale gelmesi için bazı programların yazılması gerekmektedir. Bu programların yazılması Scout içerisinde birçok yolla olabilmektedir. Bunlar Structural Text dili, MCC dili, Ladder gibi PLC’lerde çokça kullanılan programlama diliyle programlama yapılabilmektedir. Bu özellik, uygun yerde uygun dillerin kullanılması konusunda çok kolaylaştırıcı bir özelliktir. Mesela MCC programlama dili görsel programlama esasına dayandığı için çok zor olmayan, iç içe karmaşık döngülere girilmeyecek durumlarda hızlı sonuç alınmasını sağlar. Projemiz içerisinde de belli yerlerde bu dilden istifade edilmiştir. Structured Text (ST) programlama dili ise Pascal tabanlı olduğu için metinsel programlama dillerini kullanmaya alışmış olanlarca daha kolay anlaşılıp hâkim olunabilir. Projede D425’i yönlendiren programın yazımı, bu dil kullanılarak yapılmıştır.

Scout’un içerisinde “Execution Tasks” adı verilmiş olan, hangi durumda hangi programın çalıştırılması gerektiğini komuta eden bir alt program mevcuttur. Bu program içerisine yazılıp kullanılması istenen her program kayıt edilir.

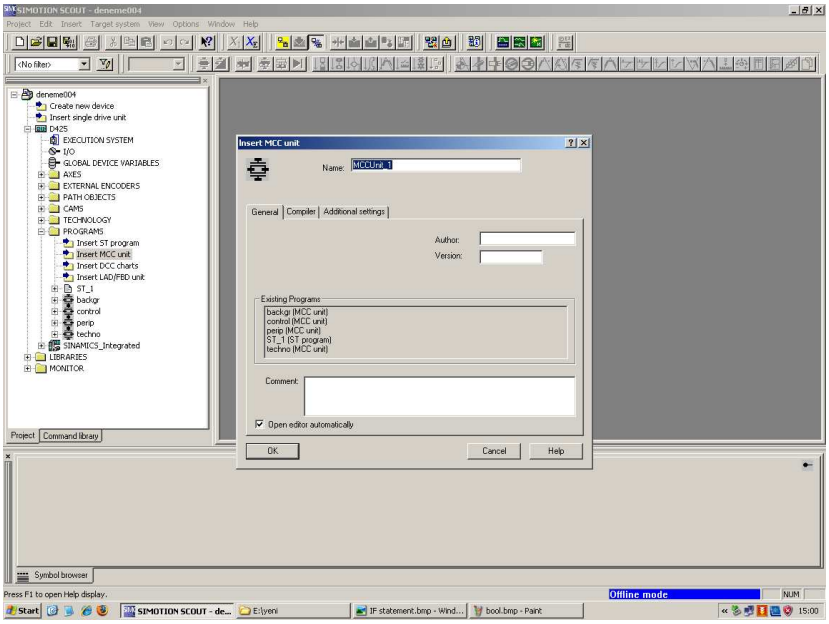
“Execution Task” bölümünde yapılan tanımlamalardan “Peripheral Fault Task”, “Technology Fault Task” ve “Background Task” bölümlerine mutlaka birer programın atanmış olması gerekmektedir. Devam eden bölümde bu atamaların nasıl yapıldığı ve program içeriklerine değinilecektir.

4.2.1 MCC Programları

Kısa bir açıklama yapılacak olursa MCC içerisinde döngü kontrolü, eksen hareketlerinin düzenlenmesi, vb. birçok özelliğin görsel olarak düzenlenebildiği ve bu düzenlemenin bir diyagram şeklinde yapılabildiği bir programla dili türevidir. Daha sonra anlatılacak olan Structured Text adlı programlama dilinde dosyaların kaydedilebilmesini desteklemektedir.

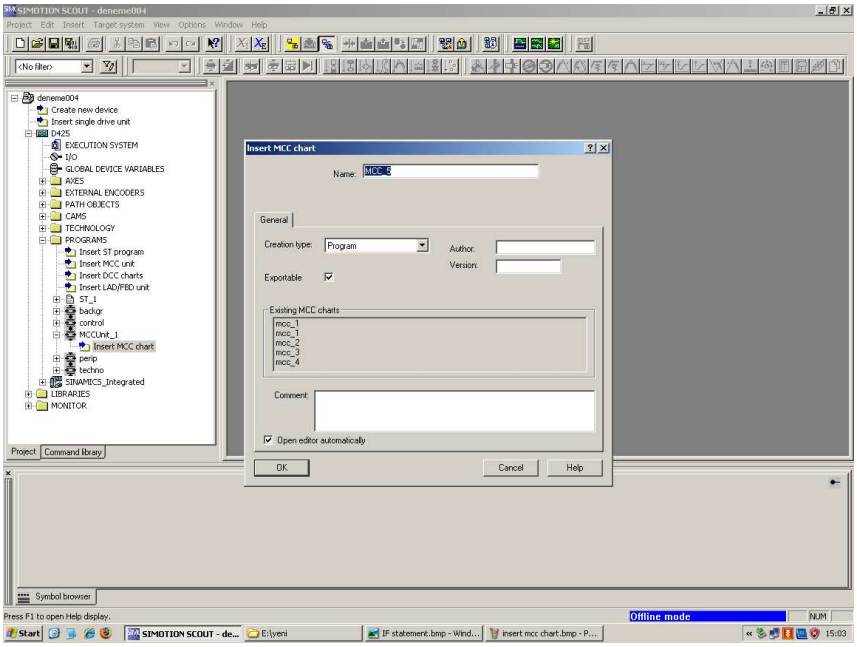
“Peripheral Fault” ve “Technology Fault Task”ları Simotion’ın bu sınıflamalar içerisinde yer alan elemanların hata vermeleri veya bir sorunla karşılaşmaları durumunda Scout’un ne gibi bir işlem yapması gerektiği ile ilgili olan durumları düzenleyerek, oluşabilecek kazaların önüne geçilmesini hedeflemektedir. Simotion’ın bu programlar tanımlanmadan çalışmamasından dolayı bu programlar içerisine oluşturulabilecek minimum program olan Start-End komutlarını içeren birer program atanacaktır.

Bu programı yazmak için, öncelikle “Programs” bölümünden insert “MCC Unit” diyerek yeni bir MCC programı oluşturulur. Bu programa gereken görevin ismi verilir. Örnek olarak Peripheral Fault için perip adlı bir program oluşturuyoruz.



Şekil 4.8 Insert MCC Unit

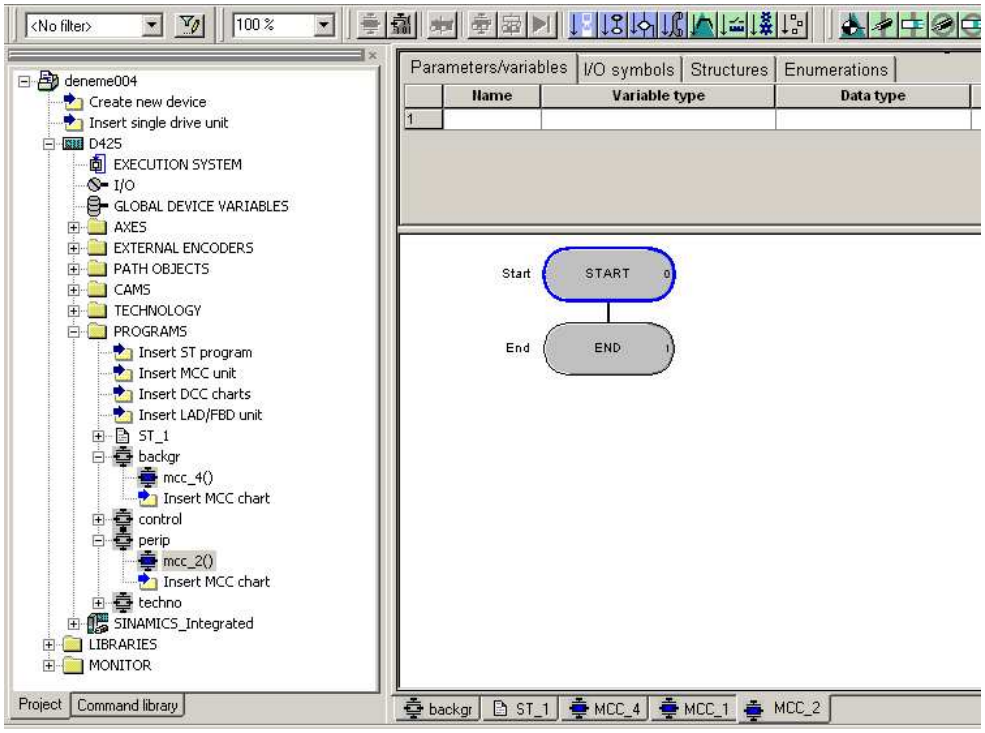
İsmini bu şekilde düzenlediğimiz program içerisinde bulunan insert MCC Chart'ı seçerek programı yazabileceğimiz bir grafik ekranı getiririz.



Şekil 4.9 MCC Chart'ın eklenmesi

MCC Chart'ın eklenmesiyle içerisinde hazır bulunan Start-End programı işimizi görecek durumdadır. Başka bir komut grafiği girilmesine gerek yoktur.

Bu program içerisinde herhangi bir değişken atanması işlemi yapılmadığı için değişkenlerin eklendiği kısımda bir değişiklik yapılmasına da gerek yoktur.

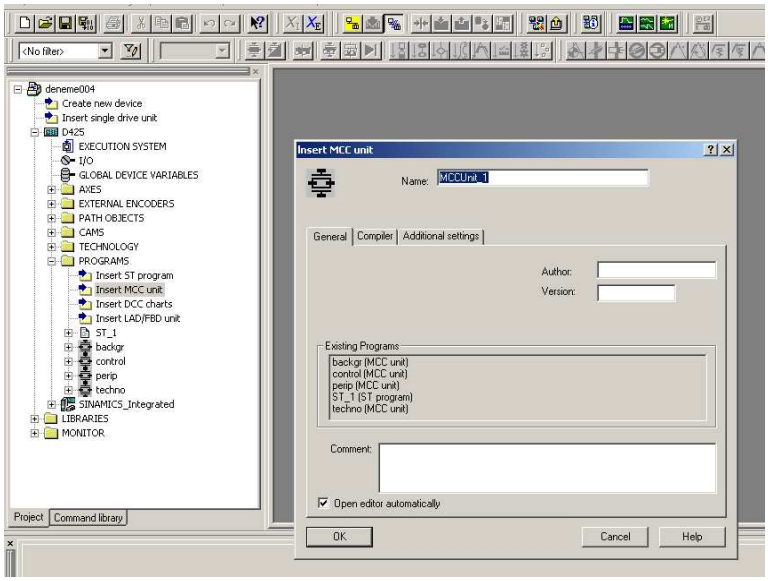


Şekil 4.10 Peripheral Task için oluşturulmuş bir program örneği

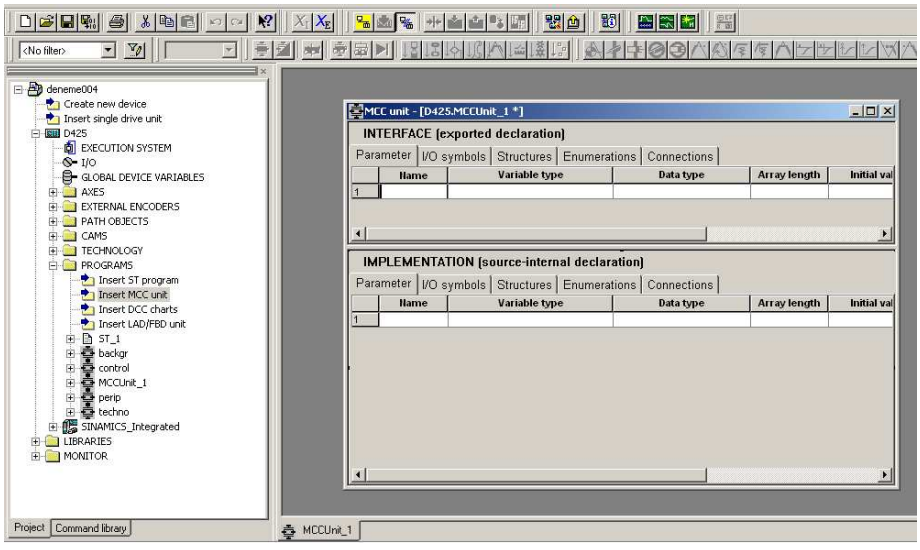
Background Task programı içerisine ise daha karmaşık bir program yazmamız gerekmektedir. Bunun sebebi normalde D425 çalışmaya başladığı andan itibaren arka planda çalışan program "Background Task"a yüklenmiş olan programdır ve bu program sürekli çalışmaya devam eder. Bu program içerisinde, bizim çalıştırmak istediğimiz ana program tanımlanmalı ve bu sayede bizim komutlarımızı hareket işlemlerine çeviren program çalışır hale getirilmelidir. Bu yönlendirme "Background" program içerisinde yapılacaktır. Programın içerisinde bir değişken tanımlanmıştır.

Bu değişken, backgr adlı MCC programını başlarda oluşturduğumuz diğer programlar gibi oluşturup içerisinden yer alan değişkenlerin girildiği “Interface (exported declaration)” kısmına tanımlanmıştır.

Bu programın ve değişkenin tanımlanması sıra ile verilecek olursa

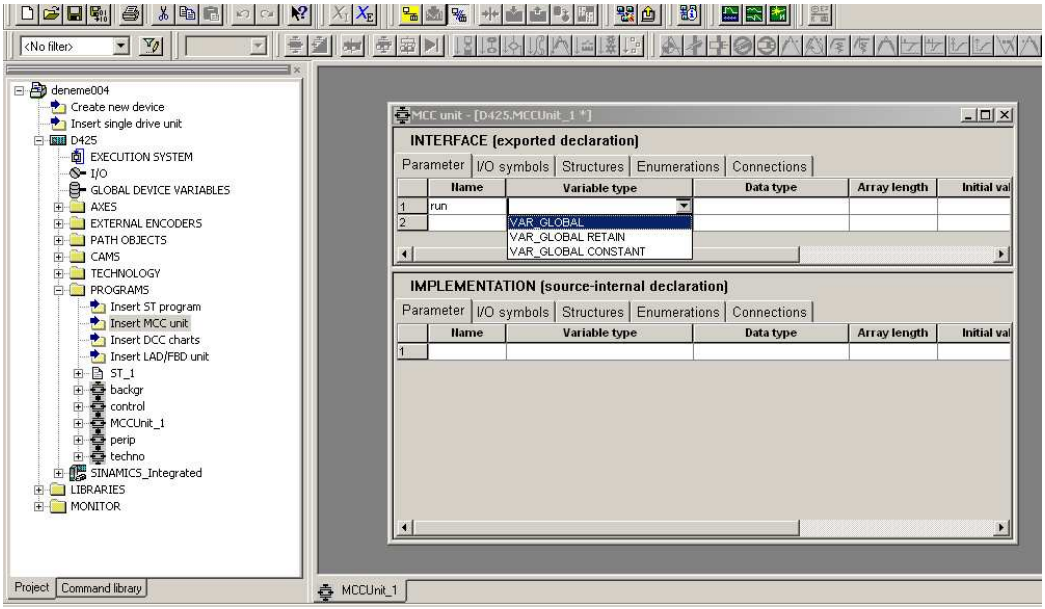


Şekil 4.11 MCC Unit eklenmesi

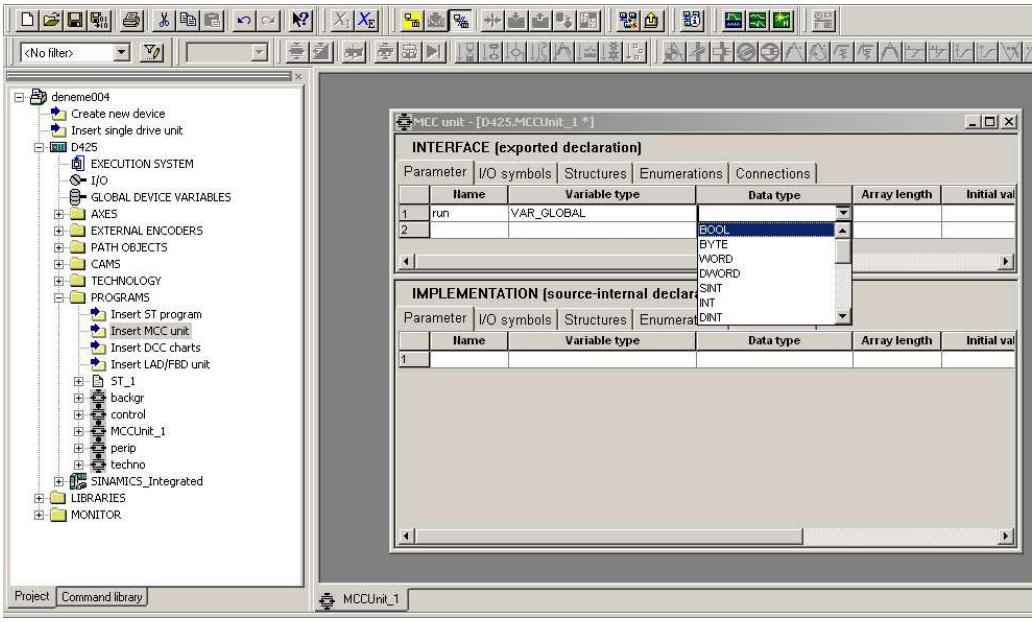


Şekil 4.12 Değişkenin tanımlandığı interface bölümü

İlk olarak “run” adında bir değişken bütün program içerisinde okunabilmesi için Var_Global olarak tanımlanır.



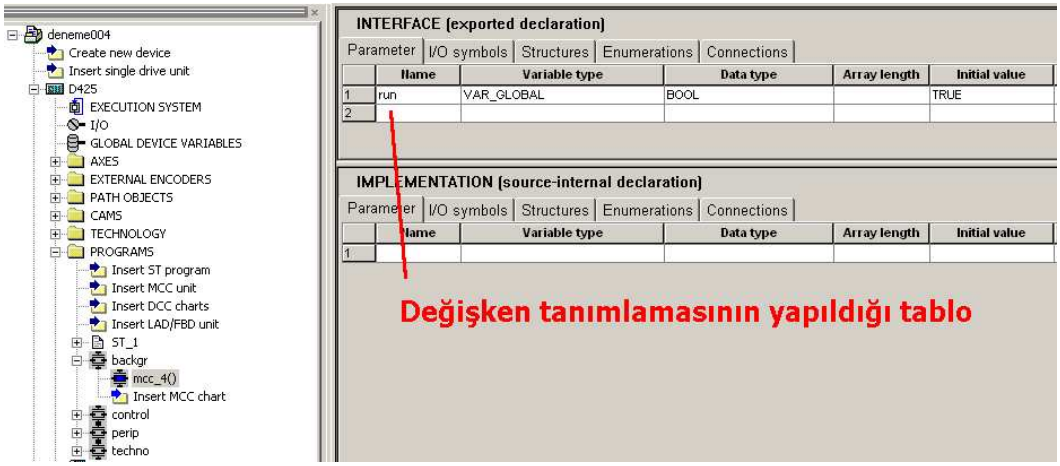
Şekil 4.13 Global değişken tanımlanması



Şekil 4.14 Değişken tipi tanımlaması

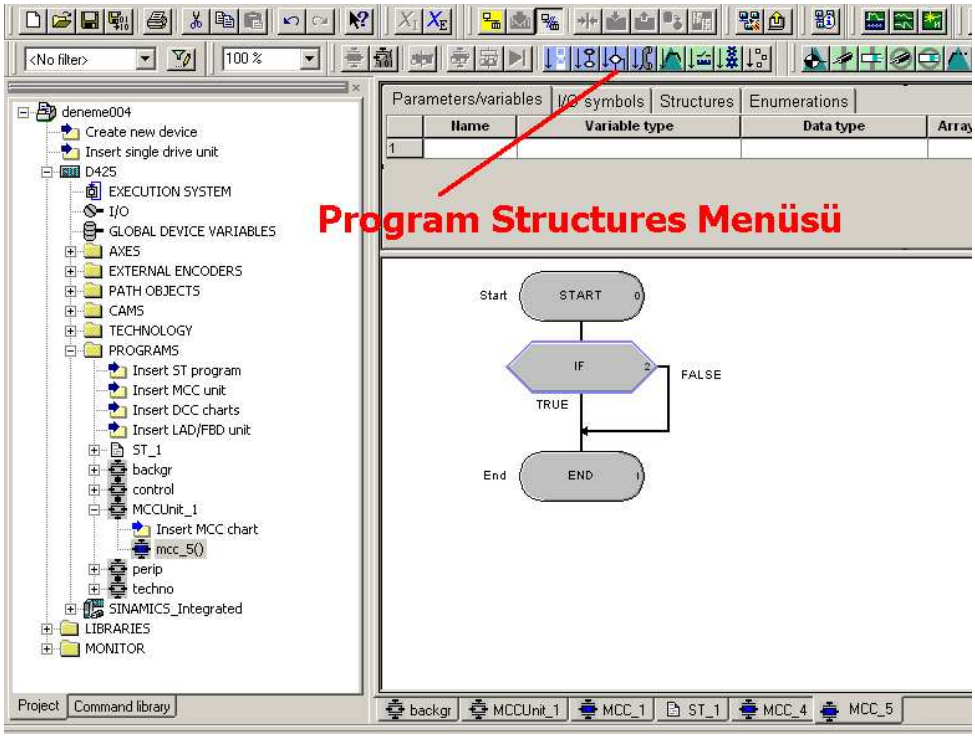
Değişken tipi olarak ise Bool tanımlanır.

Son olarak ilk değer olan “True” tanımlanmasının yapılması yeterlidir.



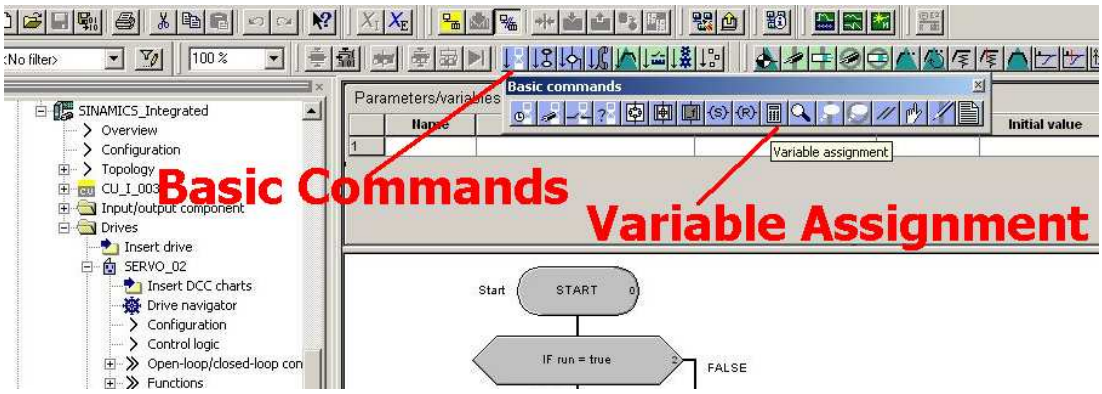
Şekil 4.15 Background Task için yazılan program da değişkenin atandığı tablo

Daha sonra eklemiş olduğumuz MCC Chartın içerisine program yazılmaya başlanır.

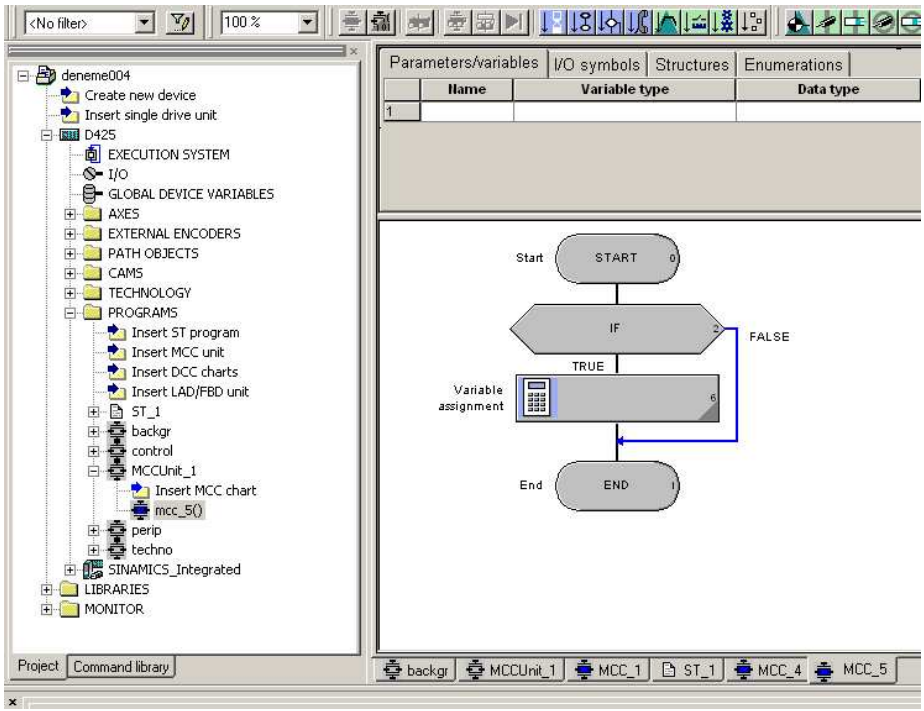


Şekil 4.16 If komutunun eklenmesi

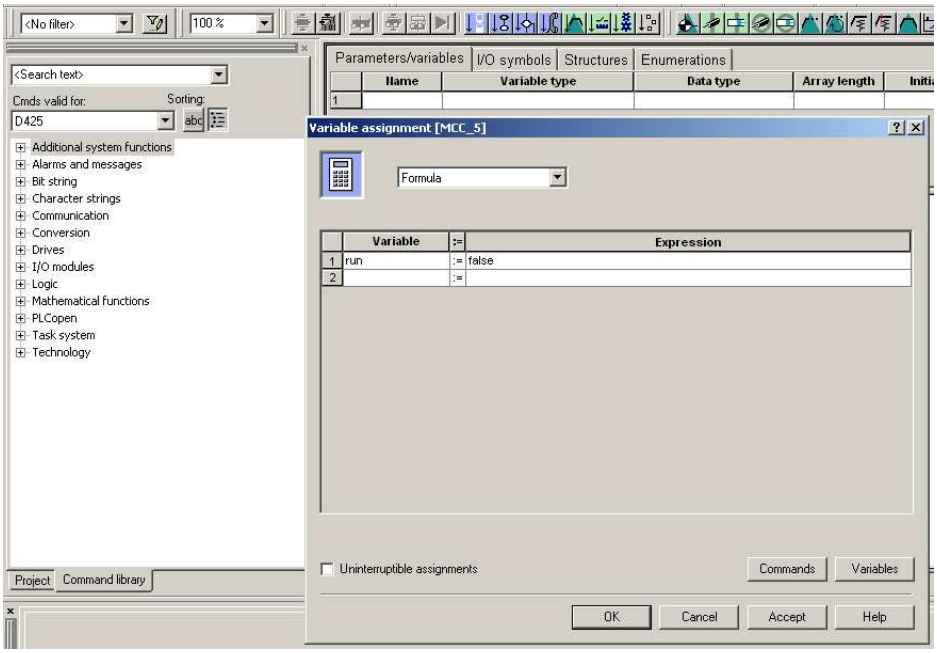
Scout programının üst kısmında yer alan “Program Structures” menüsünden if komutu “Start-End” yazan programın içerisine eklenir. Daha sonra önceden tanımladığımız “run” değişkenine atanan değerin değişmesi için If komutundan sonra “Basic Commands” menüsünde yer alan “Variable Assignment” komutu eklenerek içerisine değişken olarak “run” ve sonra ki değer olarak “False” yazılmalıdır.



Şekil 4.17 Variable Assignment komutu



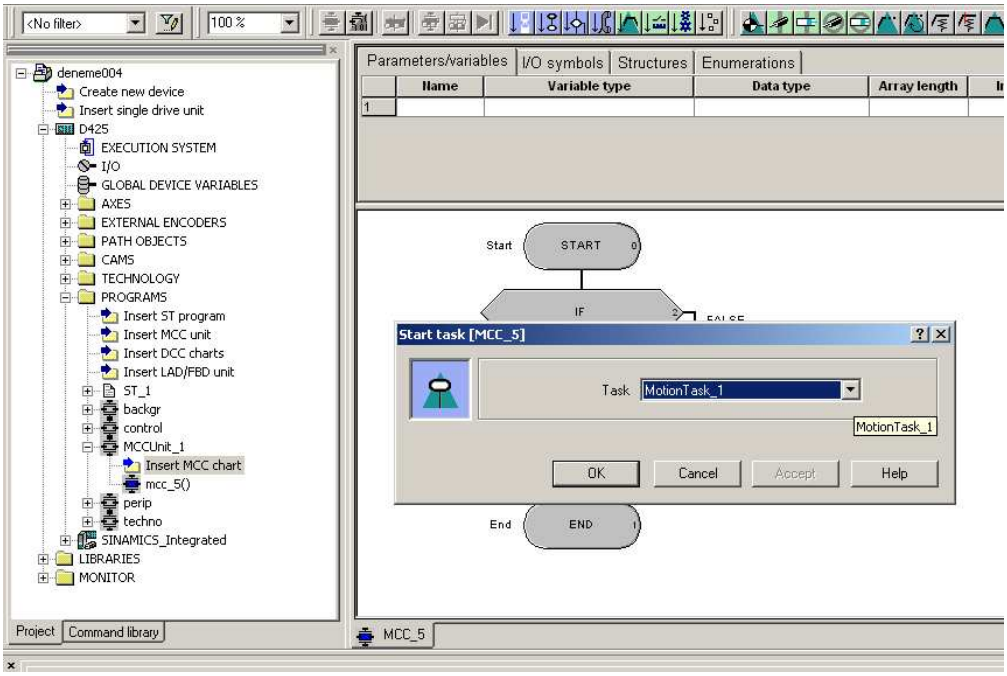
Şekil 4.18 Variable Assignment eklendikten sonraki görünüm



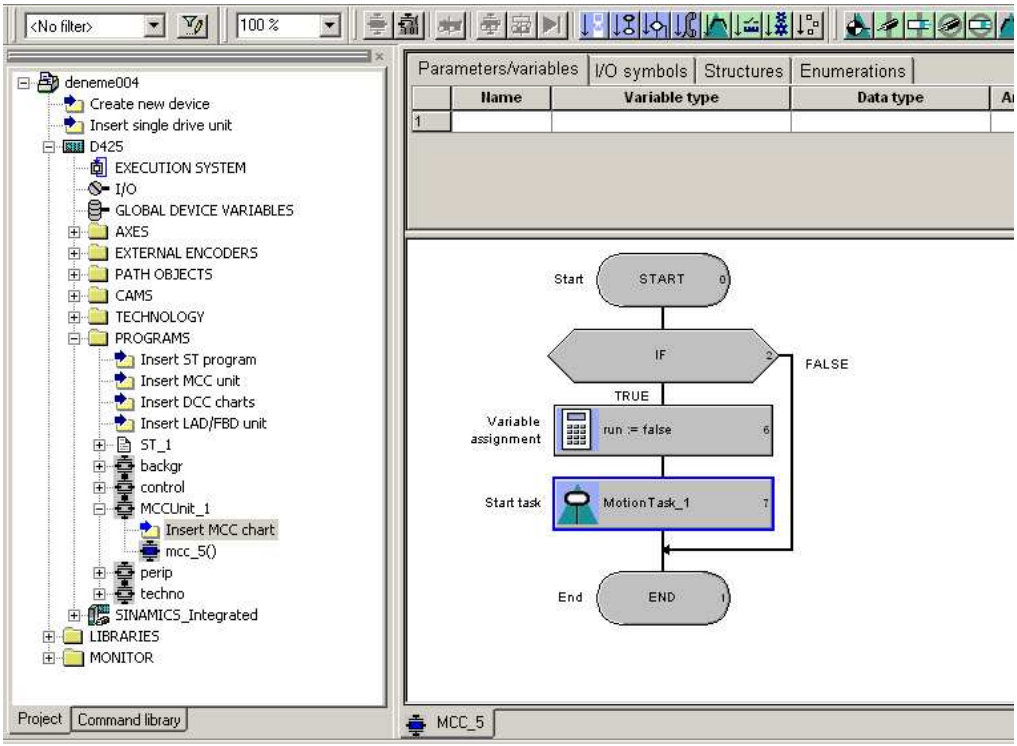
Şekil 4.19 Değişkendeki değerin karşılaştırılması istenen değerin girilmesi

"Variable Assignment" dan sonra "Start Task" komutu eklenerek "Motion Task_1" adlı program görevlendirilmelidir.

Şekil 4.20 Start Task komutu



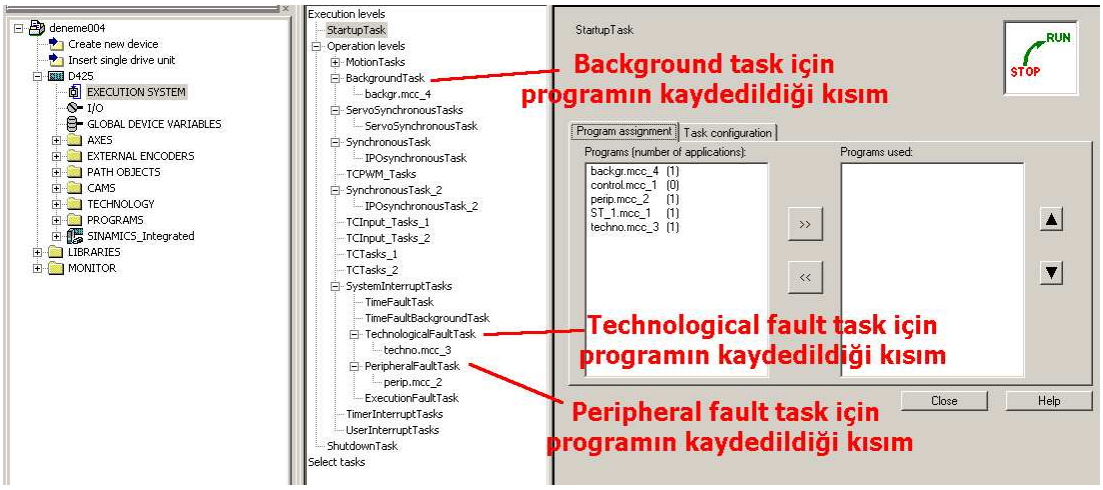
Şekil 4.21 Start Task içerisine Motion Task1'in atanması



Şekil 4.22 Programın yazımından sonraki görünüm

Programın açıklamasını yapacak olursak; D425'in Run modu aktif hale gelecek olursa "run" adlı değişken "False" olarak atanacak ve ardından "Motion Task_1" adlı "Execution System"de atanmış olan program çalışır hale getirilecektir. Daha sonra program bitirilecektir.

"Peripheral Fault" için oluşturduğumuz programın aynısından bir tane de "Technological Fault" için de oluşturarak, bunları D425 ağacının içerisinden yer alan "Execution System" de gereken yerlere kaydederiz.



Şekil 4.23 Execution system bölümünde programların atanacağı bölümler

"Execution System" içerisinde yazmış olduğumuz program, "Program Assigment" kısmında görünür. Bu programın sistem içerisinde kaç defa görevlendirilmiş olduğu program isminin yanında parantez ile ifade edilmiştir. Programın atanacağı bölüme girilerek istenen program seçilir ve üzerinde ">>" ifadesi bulunan düğmeye tıklanarak istenilen kısma program atanır. Programların "Execution System" içerisinde ifade edilerek görevlendirilmeleri işlemi bu kadardır.

4.3 Structured Text Programı

Yapılandırılmış Metin (Structured Text) ST Pascal tabanlı, yüksek seviyeli bir programlama dilidir. Bu dil IEC 61131-3 standardına dayanmakta olup, bu standart

programlanabilir mantık kontrolörleri (PLC) için olan programlama dillerini düzenler. ST bu standartın Structured Text (yapılandırılmış metin) kısmına dayanır.

ST gibi yüksek seviyeli programla dillerini kullanmak kullanıcıya büyük imkanlar sağlar. Örnek olarak; veri yönetimi, işlem optimizasyonu, matematiksel veya istatistiksel işlemler bunların arasındadır. Aynı zamanda programlamayı yapan kişiye, programın işleyiş hakimiyetine, bir programlamacı yaklaşımı ile yaklaşabilmesini kolaylaştırır.

Ek olarak IEC 61131-3 standardına uyularak, SIMOTIN ST programlama dili Simotion donanımları içi özel eklenmiş komutlar da içerir. Bu komutlar donanımların birbiri ile iletişimi, hareketlerin düzenlenmesi gibi daha farklı birçok işlemler için tasarlanmıştır. Bu özellikler sistemimize bağlı olan donanımların hareketlerini başlatmak, bitirmek, çalıştırılacak motorlara enerjinin verilmesi gibi daha birçok değişkeni ve komutu içerisinde barındırır. (Siemens, 2007)

4.3.1 ST Dilinin Elemanları

ST programlama dili isim verilerek değişken atanmasına izin vermektedir. Değişken seçerken değişken isminin bir harf ile başlaması zorunludur. Fakat daha sonra istenilen kombinasyonda sayılar, harfler veya semboller kullanılabilinir. Değişken isimleri büyük küçük harf duyarlı olmadığı için istenilen kombinasyonda büyük küçük harf barındırabilir. Fakat sistem içerisinde daha önceden programlama dilinde başka anlamlar ifade eden isimler değişken olarak atanamazlar bunlara örnek verecek olursak: DATA, LADDER, FORCE, CONFIG, YES, NO, PLC vb. isimler kullanılamaz ama: TESTER, calisma, Cal/17 gibi değişken isimleri kullanılabilir.

Değişken tanımlamaları bir program içerisinde VAR ile başlayıp END_VAR ile bitirilen bölüm arasında yapılmaktadır. Bu şekilde tanımlama değişkenlerin genel değişken olarak tanımlanmasını sağlar. Değişken tanımlama da kullanılan komutlar tabloda yer almaktadır. (Hugh J. 2008)

Tablo 4.3 Değişken tanımlama komutlarının gösterilişleri ve açıklamaları (Siemens. 2007)

Tanımlama	**Açıklama**
VAR	Genel değişken tanımlaması için kullanılır.
VAR_INPUT	Fonksiyonun giriş değişkenlerinin tanımlaması için kullanılır.
VAR_OUTPUT	Fonksiyonun çıkış değişkenlerinin tanımlaması için kullanılır.
VAR_IN_OUT	Fonksiyonda hem giriş hem çıkış değerleri için kullanılacak değişkenler için kullanılır.
VAR_EXTERNAL	
VAR_GLOBAL	Global değişkenler için
VAR_ACCESS	
RETAIN	Gücün gidip gelmesinde değer hafızada tutulur.
CONSTANT	Sabit değerlerin tanımlanmasında kullanılır.
AT	Değişken hafızada istenilen özel bir bölgede tutulur
END_VAR	Değişken tanımlamasının bitimini işaret eder.

Tanımlama Örnekleri:

```
VAR AT  %B3:0 : WORD ; END_VAR
```

(hafızaya bir kelime alır.)

```
VAR A: BOOL; END_VAR
```

(A adında bir boolean, değişken olarak atar)

Tablo 4.4 Değişken tipleri (Siemens 2007)

Tip	Ayrılmış kelime	Bit Genişliği	Değer aralığı
Bit data tipi Bu data tipindekiler 1 bit, 8 bit, 16 bit veya 32 bit aralığı kullanır. Bu data tipinin ilk değeri 0'dır.			
Bit	BOOL	1	0, 1 veya FALSE, TRUE
Byte	BYTE	8	16#0 dan 16#FF a kadar
Kelime	WORD	16	16#0 dan 16#FFFF a kadar
İki misli kelime	DWORD	32	16#0 dan 16#FFFF_FFFF a kadar
Sayısal tip Bu data tipi sayılsal değerlerin atanması için kullanılır. Bu data tipindeki değişkenler için başlangıç değeri 0 (tam sayılar için) veya 0.0 (bütün kayan noktalı sayılar için).			
Kısa tam sayı	SINT	8	-128 den 127 (-2**7 dan 2**7-1'e kadar) a kadar
İşaretsiz tam sayı	USINT	8	0dan 255'e kadar (0dan 2**8-1' e kadar)
Tam sayı	INT	16	-32_768 den 32_767 (-2**15 dan 2**15-1'e
İşaretsiz tam sayı	UINT	16	0 dan 65_535'e kadar (0 dan 2**16-1'e kadar)
İki misli tam sayı	DINT	32	-2_147_483_648 den 2_147_483_647' e kadar (-2**31 / 2**31-1)
İşaretsiz iki misli tam sayı	UDINT	32	0dan 4_294_96_729' e kadar (0 dan 2**32-1'e kadar)
Kayan noktalı sayılar (IEEE -754'e göre)	REAL	32	-3.402_823_466E+38 den -1.175_494_351E−38'e kadar, 0.0, +1.175_494_351E−38 den +3.402_823_466E+38'e kadar Doğruluk: 24-bit , 6 ondalık sayıya tekabül eder
Uzun kayan noktalı sayılar (IEEE-754'e uygun olarak)	LREAL	64	−1.797_693_134_862_315_8E+308 den −2.225_073_858_507_201_4E308'e kadar, 0.0, +2.225_073_858_507_201_4E−308'den +1.797_693_134_862_315_8E+308'e kadar Doğruluk: 53-bit, 15 ondalık sayıya tekabül eder

Tablo 4.5 İşleçler ve matematik fonksiyonları (Siemens 2007)

Matematik Fonksiyonları	
:=	Değişkene bir değer atamak için
+	Eklemek için
-	Çıkarmak İçin
/	Bölüm işlemi için
*	Çarpım işlemi için
MOD(A,B)	A değerinin B değerine bölümünden kalan sayıyı verir
SQR(A)	A sayısının kare kökünü verir
FRD(A)	2'lik tabandan 10'luk taban çevrim
TOD(A)	10'luk tabandan 2'lik tabana çevrim
NEG(A)	Sayının işaretini çevirir
LN(A)	A'nın Doğal logaritma değerini hesaplar
LOG(A)	10'luk tabanda logaritmasını hesaplar
DEG(A)	Radyan'dan Dereceye çevrim yapar
RAD(A)	Derece'den Radyana çevrim yapar
SIN(A)	Sinüs fonksiyonu
COS(A)	Kosinüs fonksiyonu
TAN(A)	Tanjant fonksiyonu
ASN(A)	Sinüs fonksiyonunun tersi
ACS(A)	Kosinüs fonksiyonunun tersi

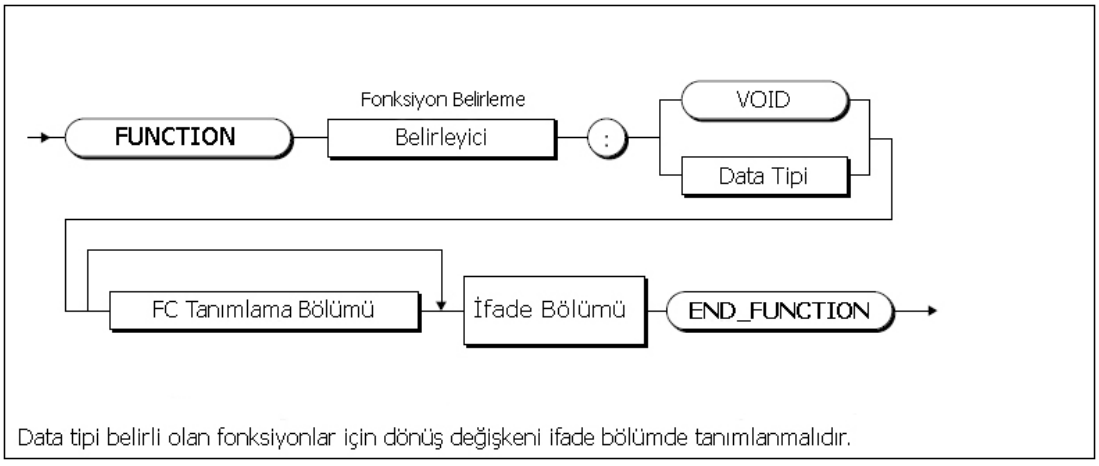
ATN(A)	Tanjant fonksiyonunun tersi
XPY(A,B)	A üzeri B
A**B	A üzeri B
Karşılaştırma İşleçleri Listesi	
>	Büyüktür
>=	Büyük eşittir
=	Eşittir
<=	Küçük eşittir
<	Küçüktür
<>	Eşit değildir
Mantık İşleçleri	
AND(A,B)	Mantıksal and
OR(A,B)	Mantıksal or
XOR(A.B)	Exclusive or
NOT(A)	Mantıksal not
!	Mantıksal not

Fonksiyon oluşturmak; bir program içerisinde program ve kod karmaşasını engellemek, programın daha hızlı çalışmasını sağlamak, çok karışık parametrelerin ve kodların sürekli girilmesi gereken veya sürekli karmaşık matematiksel işlemlerin yapılması gereken programlama kısımlarında, sadece fonksiyon ismini çağırıp gereken parametreleri girerek yapma açısından büyük kolaylıklar sağlamaktadır.

Tez içerisinde de çokça kullanmış olduğumuz fonksiyon tanımlamaları, programın işleyişini, gönderdiğimiz motor değişkenlerinin dışarıdan okunması ve içten işlenmesi kısımlarında faydalı kısaltmalar sağlamıştır. Bu fonksiyonlar oluşabilecek kısmi hataların önüne geçmiş ve programın işleyebileceği komut sayısını içerisine özet şekilde komutlar gönderebildiğimizden, arttırmıştır.

Fonksiyonların tanımlanması uygulama bölümünün bildiri kısmının içinde, fonksiyonu kullanacağımız bölümden önce yapılmalıdır.

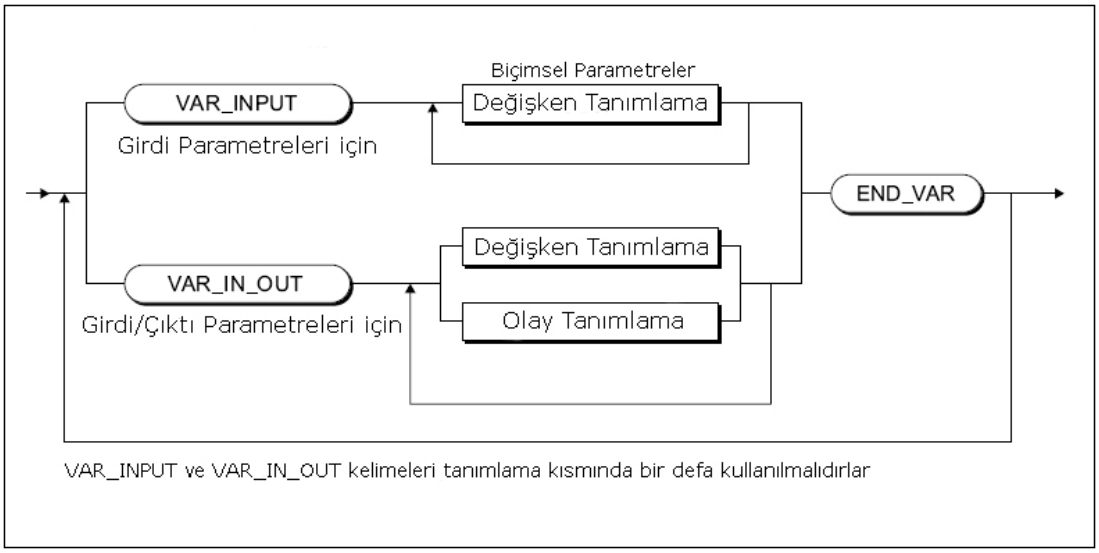
FUNCTION kelimesinden sonra fonksiyonu ifade eden kelime gelmelidir. Daha sonra fonksiyonun geri dönüşte barındırdığı bilgi için data tipi yazılır. Eğer geri dönüş ifadesi bir değer içermiyorsa VOID yazılır. Daha sonra da tercihen bildiri kısmı ve durum kısmı gelir. Bu ifadeler END_FUNCTION ile bitirilir.



Şekil 4.24 Fonksiyon tanımlanmasının şematize çizimi (Siemens 2007)

Tanımlama kısmı birçok alt parçaya ayrılır. Sabitler, yerel kullanılan değişkenler ve parametreler burada tanımlanır.

Fonksiyon içerisinde girdi çıktı değişkenlerinin tanımlanması Şekil 4.25 ile görsel olarak ifade edilmiştir.



Şekil 4.25 Fonksiyon değişkenlerinin tanımlanma şematiği (Siemens 2007)

Biçimsel girdi parametreleri o anki girdi değerlerini alır. Biçimsel çıktı parametreleri çıkış değerlerini transfer için kullanılır. Fakat yalnız fonksiyon bloklarında kullanılabildiklerinden tezde anlatılmayacaktır.

Tablo 4.6 Fonksiyon içerisinde değişken tanımlamada kullanılan tanımlama şekilleri (Siemens 2007)

Data Tipi	Yazımı
Sabit	VAR CONSTANT Bildirim Listesi END_VAR
Giriş Parametreleri	VAR_INPUT Bildirim Listesi END_VAR
Giriş /Çıkış parametreleri	VAR_IN_OUT Bildirim Listesi END_VAR
Yerel değişkenler	VAR Bildirim Listesi END_VAR
Bildirim Listesi: Tipi belirtilecek tanımlayıcıların listesi	

4.3.2 ST Programının Yazılması

Scout programının içerisinden ST derleyicisi kullanılarak yazılacak olan bir programın yapısını inceleyecek olursak; aşağıda verilen şekilde en kısa olarak anlatabilir.

```
INTERFACE
   USEPACKAGE CAM;(*$_ GridID:ffffffff $_*)
   PROGRAM yazımıturamı;
END_INTERFACE

IMPLEMENTATION

   FUNCTION yazıtura : BOOL
      VAR_INPUT
      ……
      END_VAR
   END_FUNCTION

   PROGRAM yazımıturamı
      VAR
      ……
      END_VAR
      .
   END_PROGRAM
END_IMPLEMENTATION
```

Oluşturulan bir program “Interface” kısmıyla başlar, bu kısım içerisinde Scout programında hangi teknoloji paketinin kullanılacağını belirleyen “Usepackage“ komutu ve içerisinde oluşturulacak olan programın isminin tanıtıldığı kısım yer alır. Daha sonra “Implementation” kısmı yer alır ki bu kısım içerisinde fonksiyonları, değişkenleri, ana programı ve tabi ki ana programın içerisinde yer alan döngü kontrol komutlarını barındırır.

Scout programında, bir servo motorun kontrol edilebilmesi için, öncelikle o motorun atanmış olduğu eksene gücün verilmesi gereklidir. Simotion, bu işlemin ST dilinden yapılabilmesi için teknoloji işlevleri barındırmaktadır. Bu komut “_enableAxis” şeklinde yazılmaktadır ve bu komut içerisinde eksene ait parametreler tayin edilebilmektedir.

```
_enableAxis(axis:=Axis_1, movingMode:=DO_NOT_CHANGE, enableMode:=ALL,
servoControlMode:=ACTIVE, servoCommandToActualMode:=INACTIVE,
nextCommand:=WHEN_COMMAND_DONE, commandId:=_getCommandId(),
forcecontrolMode:=INACTIVE);
```

“_enableAxis” komutu, program içerisinde yukarıda yazan şekilde ifade edilmiştir. İkinci eksen üzerinde aynı komut, “axis:=Axis_1” yerine “axis:=Axis_2” yazılarak kullanılmıştır.

Bir eksen aktif duruma getirildikten sonra yapacağı hareketi belirleyen değişkenlerin kontrolöre gönderilmesi gerekmektedir. Motoru pozisyon kontrolü modunda kullandığımız için, motora gitmesi gereken bilgi; döneceği yeni açı miktarı ve bu açının dönmesi gereken negatif veya pozitif yöndür. Bu bilgi de kontrolöre “_pos” adlı bir teknoloji komutu kullanılarak gönderilebilmektedir.

```
_pos(axis:=Axis_1, direction:=POSITIVE, positioningMode:=RELATIVE,
position:=amount, velocityType:=DIRECT, velocity:=500,
blendingMode:=INACTIVE, mergeMode:=IMMEDIATELY,
nextCommand:=WHEN_MOTION_DONE, commandId:=_getCommandId());
```

Her hareket üretilmesi istenen durumda bu kadar fazla metin içeren bir program satırının motora gönderilmesi, yazılması açısından çok zorlayıcı bir durumdur. Bu tez içerisinde yapılan çalışmada, komutların hepsi bir araya getirilerek fonksiyonlar

içerisinde toplanmışlardır. Öncelikle tekil +y,-y,+x,-x yönlerinde hareketlerin oluşmasını sağlayacak hareket fonksiyonları, daha sonradan her eksende yapılacak hareketleri tek bir fonksiyon içerisinden toplayacak ana bir eksen fonksiyonu çatısı altında toplanmışlardır. +y ve –y ekseninde yapılan hareketi denetleyen “move_vertical(değer);” şeklinde bir fonksiyon hazırlanmıştır. +x ve -x ekseninde yapılan hareketi denetleyen “move_horizontal(değer);” şeklinde diğer fonksiyon tanımlanmıştır. Visual Basic programı içerisinde hareket komutları bu fonksiyonların sözdizimi şeklinde oluşturularak program içerisinde okunur hale getirilmişlerdir.

```
FUNCTION move_up : DINT
    VAR_INPUT
        amount : INT;
    END_VAR
        move_up := _pos(axis:=Axis_2, direction:=POSITIVE,
positioningMode:=RELATIVE, position:=amount, velocityType:=DIRECT,
velocity:=500, blendingMode:=INACTIVE, mergeMode:=IMMEDIATELY,
nextCommand:=WHEN_MOTION_DONE, commandId:=_getCommandId());
END_FUNCTION

FUNCTION move_down : DINT
   .
   .
END_FUNCTION


FUNCTION move_left : DINT
    VAR_INPUT
        amount : INT;
    END_VAR
        move_left := _pos(axis:=Axis_1, direction:=NEGATIVE,
positioningMode:=RELATIVE, position:=amount, velocityType:=DIRECT,
velocity:=500, blendingMode:=INACTIVE, mergeMode:=IMMEDIATELY,
nextCommand:=WHEN_MOTION_DONE, commandId:=_getCommandId());
END_FUNCTION

FUNCTION move_right : DINT
   .
   .
END_FUNCTION
```

Fonksiyonları, eksen hareketlerinin ifadelerini, kısaltacak şekilde düzenlemiştir.

```
FUNCTION move_vertical : VOID
    VAR_INPUT
        amount : INT;
    END_VAR

    VAR
        tmp : DINT;
    END_VAR

    IF (amount < 0) THEN
        tmp := move_down(ABS(amount));
    ELSE
        tmp := move_up(amount);
    END_IF;
END_FUNCTION

FUNCTION move_horizontal : VOID
   .
   .
END_FUNCTION
```

Fonksiyonlarıyla, bu eksen hareketleri iki ayrı fonksiyon içerisine toplanmış olunur.

Scout içerisine, Visual Basic ile yazdığımız programla üretmiş olduğumuz komut satırlarını, kopyalayarak komutların programın içerisine dahil program satırları gibi görünmesi sağlanmıştır. Bu sayede dışarı ile veri alış verişi kısıtlı olan Scout programının içerisine dışarıda üretilmiş komutların eklenmesiyle Scout'un yapması için tasarlanmamış görevler yapılabilir duruma getirilmiş olmaktadır.

```
PROGRAM INTERPOLATOR
    VAR
        _MccRetDINT : DINT;
    END_VAR
        ;
        (* Eksen Aktif duruma getiriliyor *)
        _MccRetDINT := _enableAxis(axis:=Axis_1, movingMode:=DO_NOT_CHANGE,
enableMode:=ALL, servoControlMode:=ACTIVE,
servoCommandToActualMode:=INACTIVE, nextCommand:=WHEN_COMMAND_DONE,
commandId:=_getCommandId(), forcecontrolMode:=INACTIVE);

        (* Eksen Aktif duruma getiriliyor) *)
        _MccRetDINT := _enableAxis(axis:=Axis_2, movingMode:=DO_NOT_CHANGE,
enableMode:=ALL, servoControlMode:=ACTIVE,
servoCommandToActualMode:=INACTIVE, nextCommand:=WHEN_COMMAND_DONE,
commandId:=_getCommandId(), forcecontrolMode:=INACTIVE);
//----------------------------
VISUAL BASİC PROGRAMINDA YAZDIĞIMIZ PROGRAMLA ÜRETİLMİŞ KOMUTLAR BURAYA
KOPYALANIP YAPIŞTIRILIR
//----------------------------
    END_PROGRAM
 END_IMPLEMENTATION
```

4.4 G-Kodu ve İnterpolasyon Mantığı

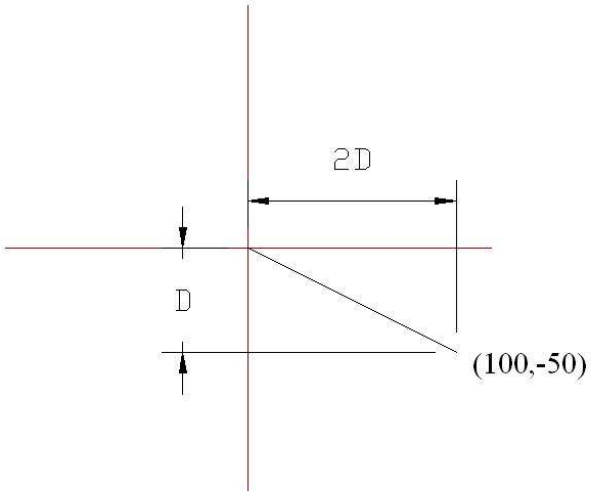
Visual Basic ile yazılan programda ulaşılmak istenen nokta, programın işleme mantığının, CNC tezgahların programlanması mantığını düzenlemekte olan G kodu standardındaki, girilen koordinat verilerini işleyerek, istenen çizimlerin tezgaha yaptırılabilmesidir.

Bu koordinat verilerinin işlenmesi interpolasyon mantığıyla, gidilmesi gereken ΔX ve ΔY değişimlerinin hesaplatılarak, motorlara bu verilerin ulaştırması işlemi ile olur.

Sistemimizde kullanılan G kodu mantığı 2 farklı yapıya sahiptir. Birincisi çizgisel hareketleri düzenleyen çizgisel interpolasyon. İkincisi ise dairesel hareketleri düzenleyen dairesel interpolasyon mantığıdır.

Çizgisel interpolasyonda gidilmesi gereken ilk nokta ile ikinci noktanın koordinatları G kodundan alınarak, çizginin oluşabilmesi için her eksenin bir tam harekette ne kadar ilerlemesi gerektiğinin hesaplatılmasıyla olur. G kodunda bu hareketleri G01 adlı komut düzenler. Örnek verilecek olursa 45 derece açı yapan bir çizginin ΔX ve ΔY değişimleri birbirlerine eşit miktarda olacaktır. Bu artımların miktarı da çizginin son ulaşacağı boyu belirler. G kodundan nasıl okunacağının mantığı ise; bir önceki bitiş noktası bir sonraki çizginin başlangıç noktasıdır. “G01 X100 Y-50” gibi bir komut kendisinden önce bir ifade olmadığı için başlangıç noktası olarak (0,0) noktasını kabul ederek, bitiş noktasını (100,-50) kabul edecektir. Böylece çizmesi gereken çizgi şekil 4.26’daki gibi olacaktır. Bu sayede X eksenini hareket ettiren motor 2D kadar pozitif yönde gittiği zaman Y eksenini kontrol eden motor D kadar negatif yönde hareket edecektir.

CNC tablamızın üzerindeki vidalı milin hatvesi p=2mm diş başı çapı ø12mm olarak ölçülmüştür. Tablanın ilerleme katsayısını olarak mildeki 1º dönüş 0.00556979 mm ye denk gelmektedir. Programlarda hesaplar buna göre yapılacaktır.



Şekil 4.26 G01 örnek koduna ait çizim

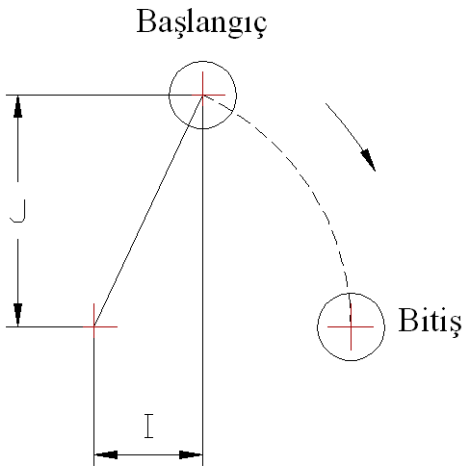
Visual Basic dilinde yazılmış olan programda bu işlemleri yapan algoritma ifadesi şu şekildedir.

```
işlev merdiven(gidilecek_x, gidilecek_y)

   tam_x = 0
   tam_y = 0

   yeni_x = 0
   yeni_y = 0

   x = 0
   y = 0

   adım_x = 0
   adım_y = 0

   eğer (gidilecek_x > gidilecek_y) ise
      adım_x = işaret(gidilecek_x)
      adım_y = işaret(gidilecek_y) * mutlak(gidilecek_x / gidilecek_y)
   değilse
      adım_y = işaret(gidilecek_y)
      adım_x = işaret(gidilecek_x) * mutlak(gidilecek_x / gidilecek_y)
   eğer sonu

   döngü (sıralıdır([0, x, gidilecek_x]) ve sıralıdır([0, y, gidilecek_y]))
ise
      yeni_x = tamsayı(x)
      yeni_y = tamsayı(y)

      eğer (yeni_x <> tam_x) ise
         yatay_ilerle(yeni_x - tam_x)
```

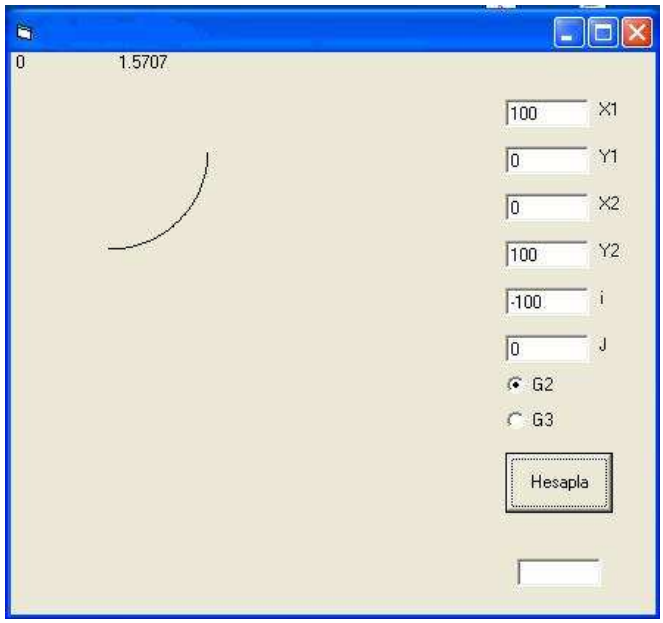
```
         tam_x = yeni_x
      eğer sonu

      eğer (yeni_y <> tam_y) ise
         dikey_ilerle(yeni_y - tam_y)
         tam_y = yeni_y
      eğer sonu

      x = x + adım_x
      y = y + adım_y
   döngü sonu

   eğer tam_x <> gidilecek_x ise yatay_ilerle(yeni_x - tam_x)
   eğer tam_y <> gidilecek_y ise dikey_ilerle(yeni_y - tam_y)

işlev sonu
```

Şekil 4.27 Visual Basic ile yazılmış G01 çizgisel hareket komutları için yazılmış olan programın arayüzü

Dairesel hareketlerin oluşabilmesi için yapılan pozisyon hesabına dairesel interpolasyon adı verilir ve G kodunda bu işlemlere G02 saat yönünde dairesel interpolasyon, G03 ile de saat yönünün tersi dairesel interpolasyon komutları aracılığı ile erişilebilir. Örnek olarak G02 kodunun yazılışı şekil 4.28'deki gibi ifade edilir.



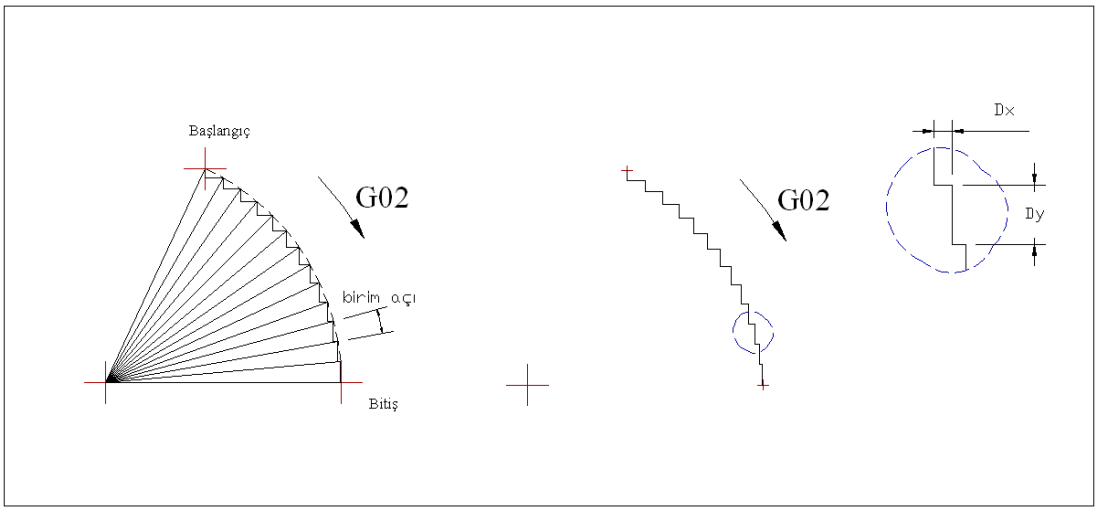
Şekil 4.28 G02 dairesel interpolasyon



Şekil 4.29 Çember programının arayüzü

Bu yöntem, sırasıyla bu işlemleri gerçekleştirmektedir:

- Önce, dairesel hareketin başlangıç noktasının konumu kartezyen koordinatlarda bulunur.
- Sonra, dairesel hareketin bitiş noktasının kartezyen koordinatlardaki konumu bulunur.
- Daha sonra bu konumları hesaplanmış olan noktalar arasındaki açı hesaplanır.
- Daha sonra, kutupsal koordinat uzayındaki bu iki açı arasındaki yayın üzerindeki, belirli açısal aralıklarla noktalarının koordinatları hesaplanır.
- Bir noktadan diğerine ulaşarak dairesel hareketi sağlamak için, aradaki x ve y farkı hesaplanır ve bu farkları kapatacak dikey ve yatay hareketler üretilir.



Şekil 4.30 Dairesel interpolasyon mantığı

Visual Basic programlama dilinde yazılmış olan çemberin başlangıç ve bitiş noktalarını hesaplayan programın komut satırları aşağıdaki gibi ifade edilmeye çalışılmıştır.

Tablo 4.7 Çemberin başlangıç ve bitiş noktalarını hesaplayan program (Çimen O., 2006)

```
Eğer X1 > mx1 Ve Y1 = my1 Ardından
     basla_açı = 0
Eğer sonu
Eğer X1 = mx1 Ve Y1 > my1 Ardında 'başlangıç noktası +y ekseni üzerinde
    basla_açı = 1.5707
Eğer sonu
Eğer X1 = mx1 Ve Y1 < my1 Ardından
    basla_açı = 4.7124
Eğer sonu
Eğer X1 < mx1 Ve Y1 = my1 Ardından
    basla_açı = 3.1416
Eğer sonu
Eğer X1 > mx1 Ve Y1 > my1 Ardından 'başlangıç noktası 1.bölgede
    basla_açı = Atn(tam değer(j) / tam değer(I)) '
Eğer sonu
Eğer X1 < mx1 Ve Y1 > my1 Ardından 'başlangıç noktası 2.bölgede
   basla_açı = Atn(tam değer(I) / tam değer (j)) + 1.5707
Eğer sonu
Eğer X1 < mx1 Ve Y1 < my1 Ardından 'başlangıç noktası 3.bölgede
    basla_açı = Atn(tam değer (j) / tam değer (I)) + 3.1416
Eğer sonu
Eğer X1 > mx1 Ve Y1 < my1 Ardından 'başlangıç noktası 4.bölgede
    basla_açı = Atn(tam değer (I) / tam değer (j)) + 4.7124
Eğer sonu
xfark = tam değer (X2 + (-1 * mx1))
yfark = tam değer (Y2 + (-1 * my1))
Eğer X2 > mx1 Ve Y2 = my1 Ardından 'bitiş noktası +x ekseni üzerinde
    bit_açı = 0
Eğer sonu
Eğer X2 = mx1 Ve Y2 > my1 Ardından  'bitiş noktası+y ekseni üzerinde
    bit_açı = 1.5707
Eğer sonu
Eğer X2 = mx1 Ve Y2 < my1 Ardından 'bitiş noktası -y ekseni üzerinde
    bit_açı = 4.7124
Eğer sonu
Eğer X2 < mx1 Ve Y2 = my1 Ardından 'bitiş noktası -x ekseni üzerinde
    bit_açı = 3.1416
Eğer sonu
Eğer X2 > mx1 Ve Y2 > my1 Ardından 'bitiş noktası 1.bölgede
    bit_açı = Atn(yfark / xfark)
Eğer sonu
Eğer X2 < mx1 Ve Y2 > my1 Ardından 'bitiş noktası 2.bölgede
    bit_açı = Atn(xfark / yfark) + 1.5707
Eğer sonu
Eğer X2 < mx1 Ve Y2 < my1 Ardından 'bitiş noktası 3.bölgede
    bit_açı = Atn(yfark / xfark) + 3.1416
Eğer sonu
Eğer X2 > mx1 Ve Y2 < my1 Ardından 'bitiş noktası 4.bölgede
    bit_açı = Atn(xfark / yfark) + 4.7124
Eğer sonu
```

Program içerisinde interpolasyonu yapan kod satırları

```
Eğer g2=1 Ardından

Eğer basla_açı < bit_açı ardından
 aradeg1 = basla_açı
 basla_açı = bit_açı
 bit_açı = aradeg1
Eğer sonu

   Döngü I_1 = basla_açı’dan bit_açı’ya Adım -0.01745

   xkoordnat = mx1 + r * Cos(I_1)
   ykoordnat = my1 + r * Sin(I_1)
   xkoordnat2 = mx1 + r * Cos(I_1 + 0.01745)
   ykoordnat2 = my1 + r * Sin(I_1 + 0.01745)

   motoradım_x = ((xkoordnat2 - xkoordnat) / 0.00556979) 'ara değişken
   motoradım_y = ((ykoordnat2 - ykoordnat) / 0.00556979)

   motorbil_x = (yuvarla ((motoradım_x), 1)) 'adım bilgisi
   motorbil_y = (yuvarla ((motoradım_y), 1))

   Print #2, "move_horizontal(" + Str$(motorbil_x) + ");"
   Print #2, "move_vertical(" + Str$(motorbil_y) + ");"

Döngü I_1 sonu
Eğer sonu
Kapat #2

Eğer g3 = 1 Ardından
   Aç "C:\CNC\G3HesabıX_Y.st" çıkış için As #2

   bit_açı = basla_açı + alfa
   Döngü I_2 = basla_açı’dan bit_açı’ya Adım 0.01745 '0.01745 radyan olarak
                                                      1 dereceye denk

   xkoordnat = mx1 + r * Cos(I_2)
   ykoordnat = my1 + r * Sin(I_2)
   xkoordnat2 = mx1 + r * Cos(I_2 + 0.01745)
   ykoordnat2 = my1 + r * Sin(I_2 + 0.01745)

   motorad_x = ((xkoordnat2 - xkoordnat) / 0.00556979) 'ara değişken
   motorad_y = ((ykoordnat2 - ykoordnat) / 0.00556979)

   motorbil_x = (yuvarla((motorad_x), 1)) 'adım bilgisi
   motorbil_y = (yuvarla((motorad_y), 1))

   Print #2, "move_horizontal(" + Str$(motorbil_x) + ");"
   Print #2, "move_vertical(" + Str$(motorbil_y) + ");"

Döngü I_2 sonu
Eğer sonu
```

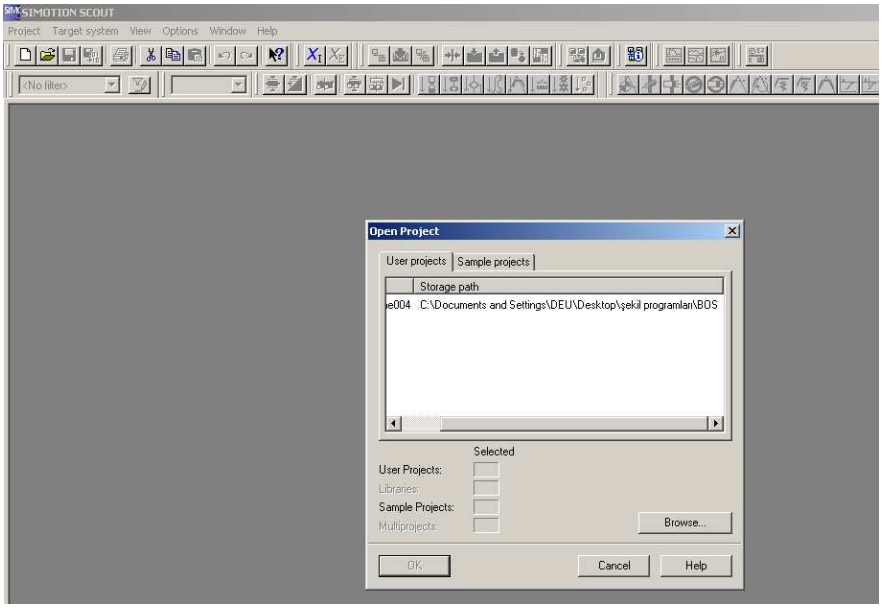
4.5 CNC Dik İşlem Tezgahına Program Komutların Yüklenmesi ve Tezgahın Çalıştırılması

Tezde şimdiye kadar yaptıklarımızı içeren proje dosyası kalıp oluşturacak şekilde içerisinde herhangi bir pozisyon komutu bulunmayarak saklanmalıdır.

Kalıp olan bu dosyaya daha sonraları işlemek istediğimiz parçaların komut satırları programın içerisine dışarıdan kopyalanarak çalışır duruma getirilebilir.

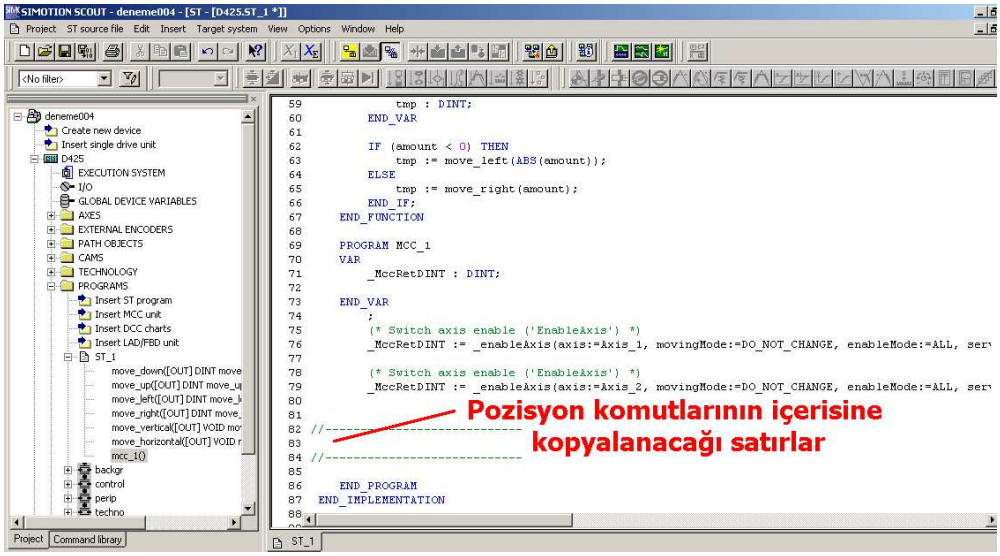
Visual Basic'de yazdığımız programdan elde edilen komut satıları dosyalarından kopyalanarak Scout içindeki ST programımızın içerisine daha önceden ifade ettiğimiz program satırları arasına, yapıştırılır.

Bunu örnek bir çalışma için yapacak olursak

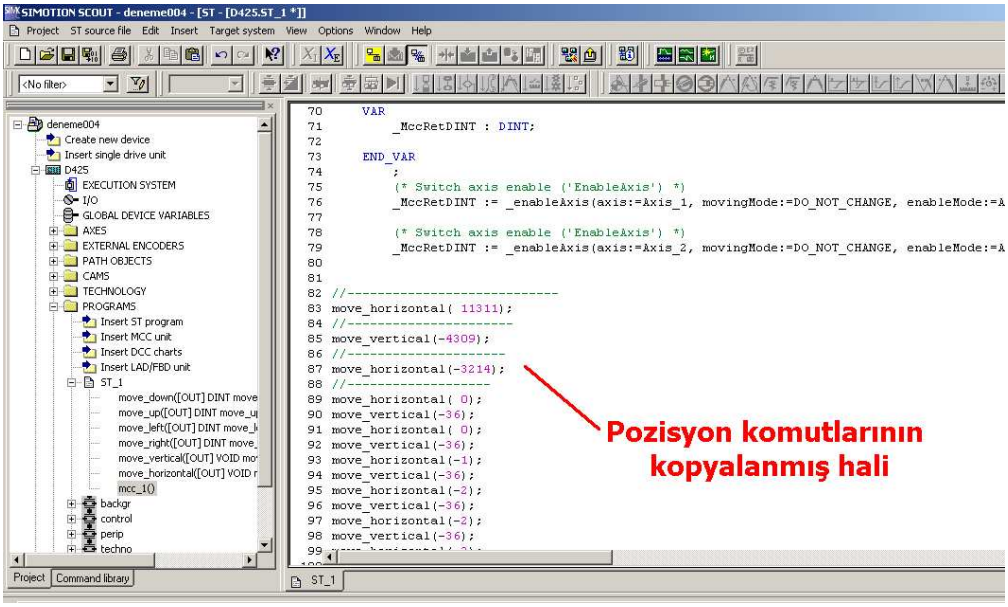


Şekil 4.31 Boş kalıp proje dosyasının Scout programında açılması

Boş kalıp dosyası Scout tarafından sistem içerisinde açılır.

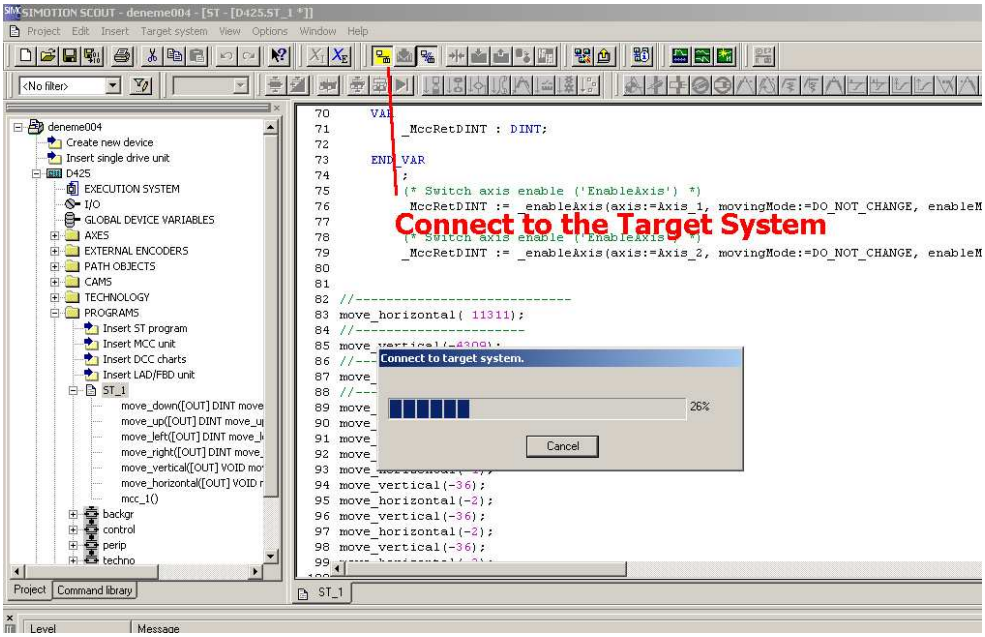


Şekil 4.32 Pozisyon komutlarının içerisine kopyalanacağı satırların gösterimi



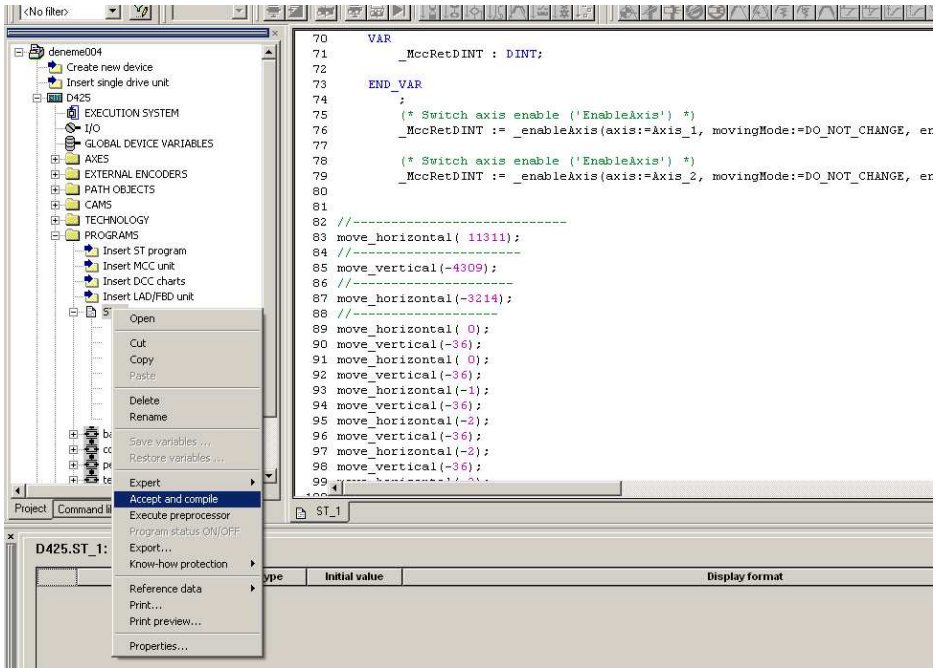
Şekil 4.33 Pozisyon komutları ST programı içerisine eklendikten sonraki görünüm

Pozisyon komutları Scout içerisine kopyalanır.



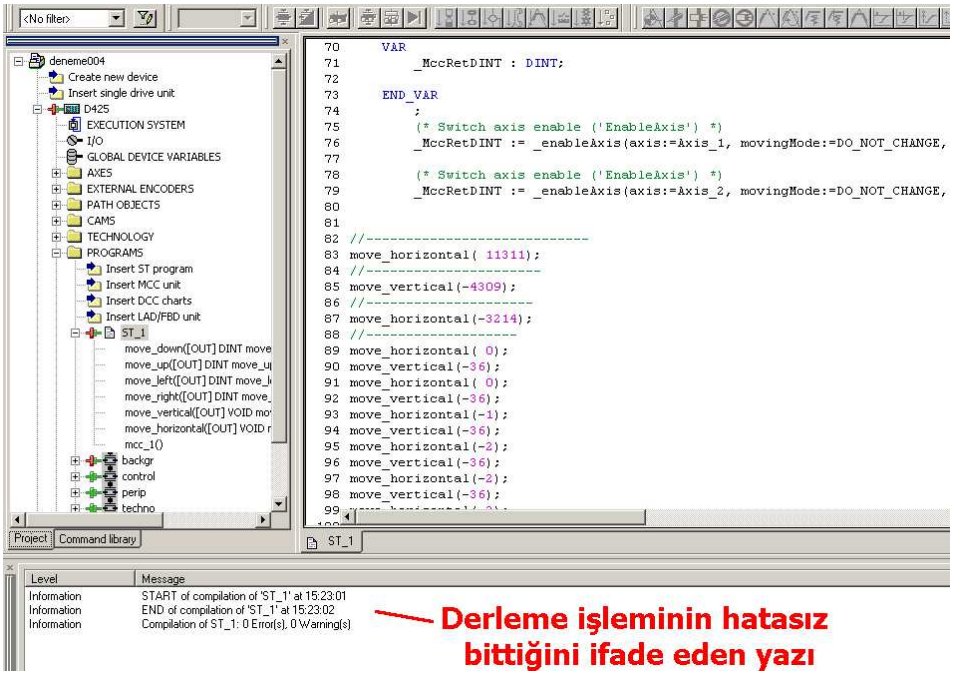
Şekil 4.34 D425'in bilgisayarla bağlantı kurması işlemi

"Connect to target system" düğmesine basılarak sistemin online duruma gelmesi sağlanır. Bu sayede derlenecek olan programın sisteme yüklenmesi hazır hale getirilir.

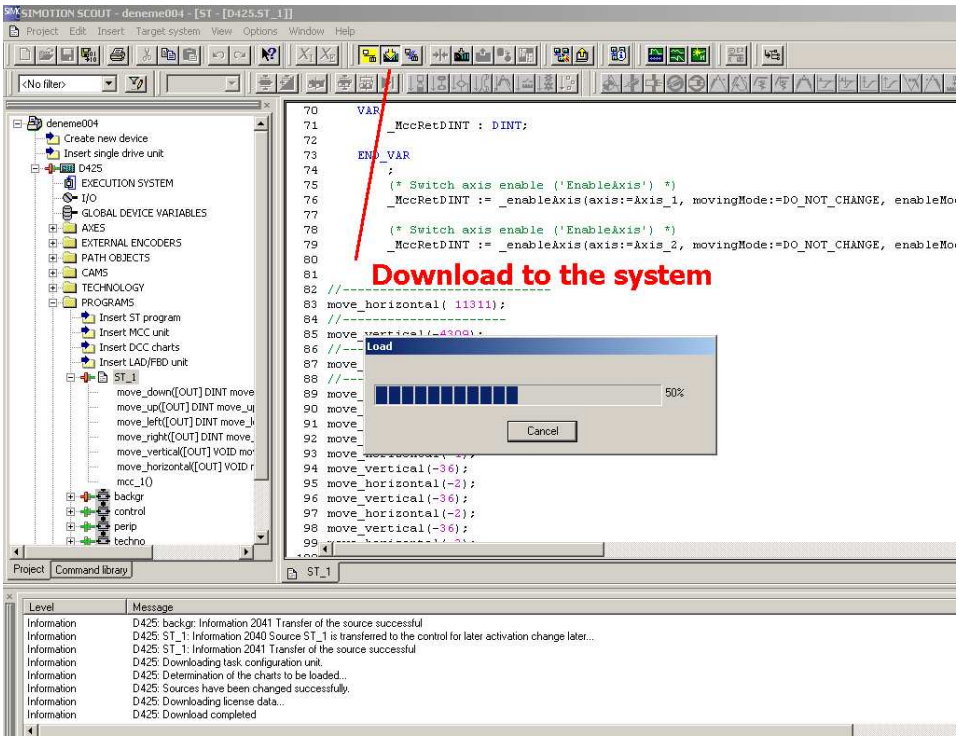


Şekil 4.35 Programın derlenmesi

Program derlendikten sonra sisteme yüklenmeye hazır haldedir.

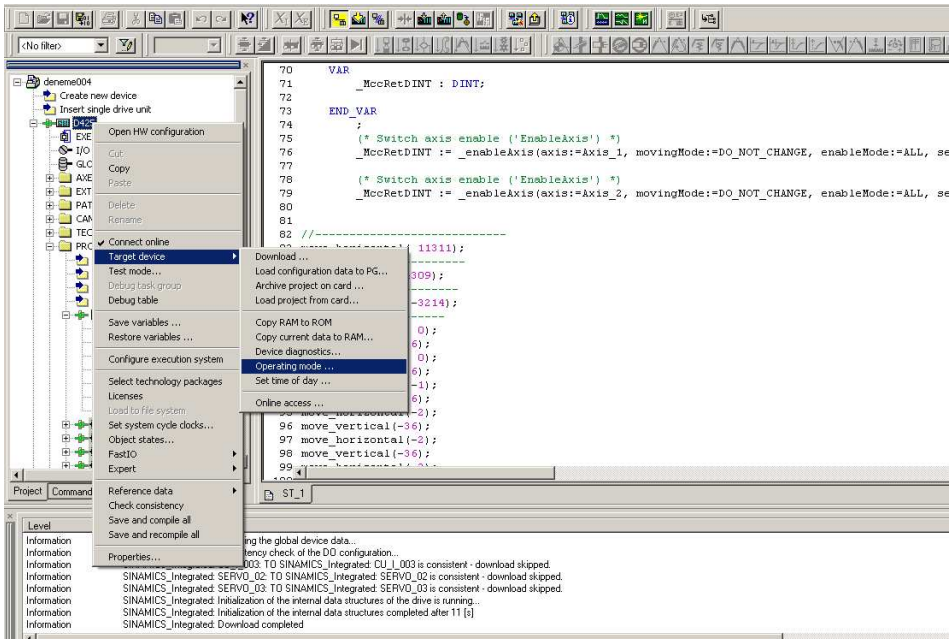


Şekil 4.36 Derleme işleminin hatasız bitişini ifade eden program yazıları

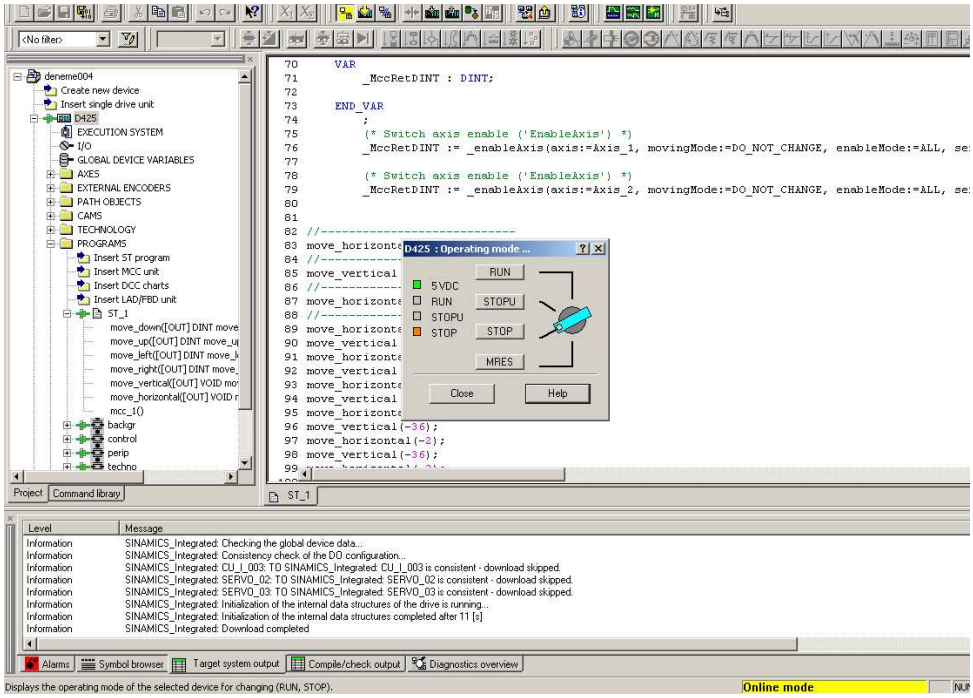


Şekil 4.37 Derlenen programın sisteme yüklenmesi

“Download to the Module” düğmesine basılarak derlenmiş olan program sistem içerisine yüklenir. Sistem içerisine yükleme bittikten sonra, artık servo motorlar komutları çalıştırabilecek duruma gelmiştir. Programı çalıştırmak için D425’in üzerinden sağ tıklanarak açılan menüden “Target Device” ve daha sonra “Operating Mode” seçilir. Bu işlemlerden sonra karşımıza “Run, Stop” düğmelerinin bulunduğu menü gelir. Bu menü’den “Run” seçildiği takdirde sistemimiz hemen bulunduğu konumdan çalışmaya başlayacaktır.



Şekil 4.38 Programın çalıştırılması için girilmesi gereken menü

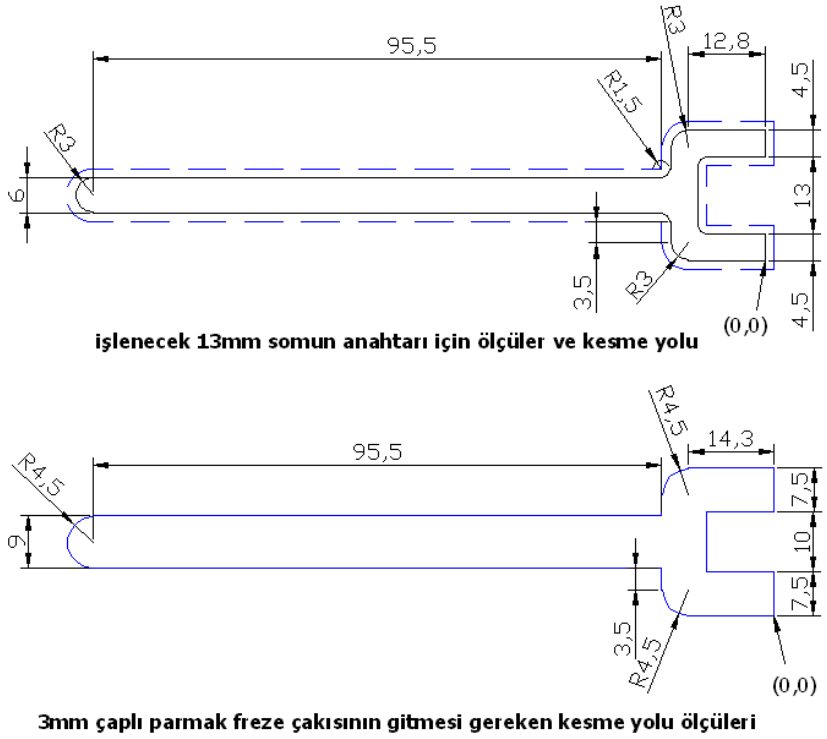


Şekil 4.39 Operating mode menüsü

BÖLÜM BEŞ
ÖRNEK ÇALIŞMALAR

Bu bölümde, tezgah ile işlenmiş örnek parçaya ait fotoğraflar ve bu parçaların oluşturulduğu G kodu ile ifade edilmiş program örneklerine yer verilmiştir.

5.1 Örnek Çalışmalar



Şekil 5.1 13mm somun anahtarı işleme ölçüleri

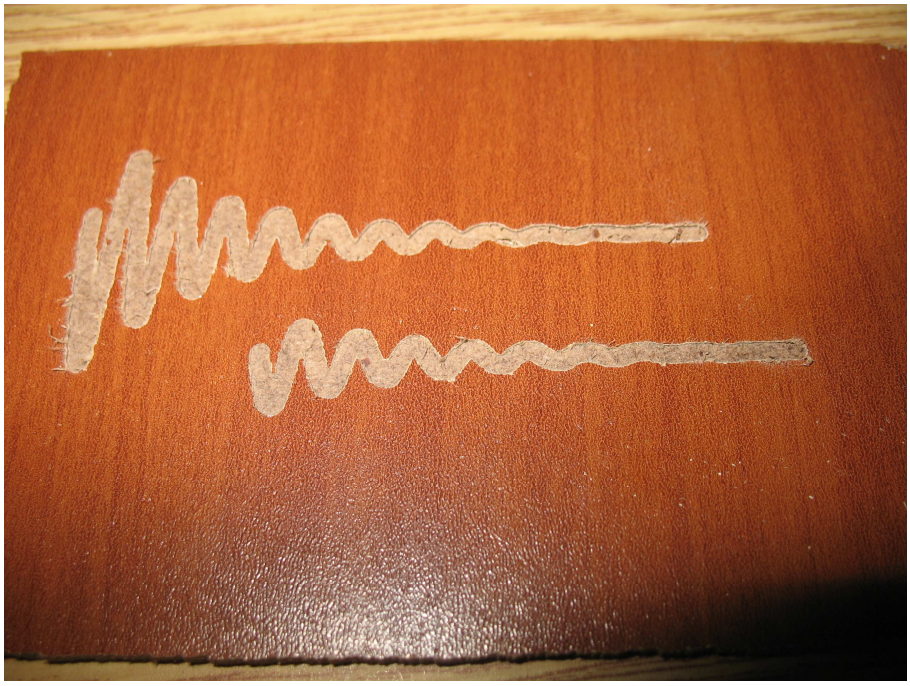
Şekil 5.2 Parçanın işlenmiş hali

Programın G-kodunda ifadesi

```
G01 X0 Y7,5
G01 X-11,3 Y7,5
G01 X-11,3 Y17,5
G01 X0 Y17,5
G01 X0 Y25
G01 X-14,3 Y25
G03 X-18,8 Y20,5 I0 J-4,5
G01 X-18,8 Y17
G01 X-114,3 Y17
G03 X-114,3 Y8 I0 J-4,5
G01 X-18,8 Y8
G01 X-18,8 Y4,5
G03 X14,3 Y0 I-4,5 J0
G01 X0 Y0
```

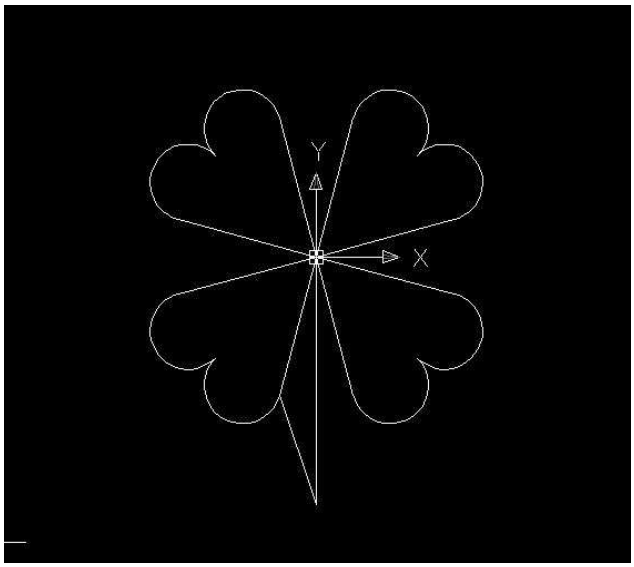
Bu program, hazırlanan Visual Basic programlarından faydalanılarak, komut dosyalarına çevrilir ve sistem içerisine kopyalanır. Komut dosyalarının sayısı çok fazla olduğu için burada başlangıcından bir kısmı örnek olarak verilmiştir.

```
move_vertical( 1500);
move_horizontal(-2260);
move_vertical( 2000);
move_horizontal(2260);
move_vertical( 1500);
move_horizontal(2860);
```



Şekil 5.3 Exponansiyel bir fonksiyonun sönümlü zaman cevabına ait bir örnek

Şekil 5.4 Yıldız örneği



Şekil 5.5 Yonca örneği cad çizimi

Şekil 5.6 Yonca örneğinin ahşaba işlenmiş hali

BÖLÜM ALTI
SONUÇLAR

Bu tezde, üniversitede bulunmakta olan eski ve kullanılamaz durumdaki bir CNC dik işleme tezgahının modernizasyonu yapılmıştır. Eklenen servo ekipman ile, iki ekseni servo kontrolle kumanda edilen dik işlem tezgahının nasıl oluşturulduğu anlatılmıştır. Siemens'in Simotion D425 adlı servo motor kontrol ürününün yanında Siemens Scout adlı kontrol programı kullanılmış, fakat bu programın içerisinde G kodu okuma özelliği olmadığı için, Scout içerisindeki Structured Text adlı programlama dilini kullanarak, işlemlerin gerçekleştirilmesini sağlayacak fonksiyonlar tanımlanmıştır. Visual Basic'te yazılan bir program aracılığıyla, Scout'un içinde tanımlanan bu fonksiyonların çağırılması ve G koduyla belirtilmiş hareketlerin gerçekleştirilmesi sağlanmıştır.

Tezin genelinden anlaşılabileceği gibi, yapılan bu çalışmada yazılmış olan bu program, sanayide kullanılan CNC tezgahların sahip olduğu kapasite ve özelliklerde değildir, bu amaçla da yazılmamıştır. Karmaşık hareketlerin uygulanması gereken yerlerde, işlem adımlarının fazla bulunduğu parça programlarında program cevap verebilecek yapıda değildir. Fakat uygulama örneklerinden görüleceği gibi, fazla karmaşık olmayan ve sürekli yapılacak olan işlemler, tezgahın hassasiyeti oranınca iyi olabilecektir. Aynı zamanda, bu yapılan çalışma sayesinde, dışarıdan kontrolü mümkün olmayan bir sistemin içerisine, bu yazılımın anlayacağı dilde bir program hazırlayarak, istenilen hareketlerin kolaylıkla yapılabilmesi sağlanmıştır. Ayrıca içerisinde istenilen bir özelliğin olmadığı servo kontrol programının içerisine yazılan başka bir programla, dışarıda üretilmiş komutlar kullanılarak bu özellik programa kazandırılmıştır.

Tezin sağladığı avantajlar yönünden Siemens’in Scout adlı yazılımını programlamanın kolay bir yolu gösterilmiştir. Siemens’in bir ürünü üzerinde çalışılmış olsa da, bu tez aslında başka firmaların benzer yazılım ve donanım ürünlerini kolayca kontrol etmenin örneğini sunmaktadır. Belli bir üretim safhasında servo kontrollerle yapılacak olan otomasyon çalışmalarında bu tarz bir programlama metodu, programı yazacak kişi için büyük kolaylık sağlayacaktır. Ayrıca desteklenen programlama dillerini bilmeyen biri için cihazın programlamasını görsel ve bilinen grafiksel komutları kullanarak programlamanın bir yolu da bu tez sayesinde gösterilmiştir.

KAYNAKLAR

3 phase sync, (b.t). Şubat 2009, http://www.eu.necel.com/applications/industrial/01_motor_control/images/3phasesync.gif.

CNC Tezgahlarının Tarihçesi, (b.t). Şubat 2009, http://www.turk-cad.com/index.php/Cnc-Nedir-/CNC-TEZGAHLARININ-TARiHcESi.html.

CNC tezgahlarının tarihçesi, (b.t). Şubat 2009, https://www.veribaz.com/viewdoc.html?cnc-tezgahlarinin-tarihcesi-353194.html.

Çimen,O. (2006). *Cnc tezgah tasarımı ve uygulaması*. Lisans tezi. Manisa: Celal .Bayar.Üniversitesi Makine Mühendisliği Bölümü.

G3102 Vertical Mill, (b.t). Şubat 2009, http://www.grizzly.com/products/Vertical-Mill/G3102.

Hugh, J. (2008). *Structured text programming*, Şubat 2009, www.eod.gvsu.edu/~jackh/books/plcs/chapters/plc_st.pdf.

Hyatt, R., Dayton, Jr. ve Dayton, D., (1999). *The measurement, instrumentation and sensors handbook*. New York: Crc Press

IMAS CNC Tezgah, (b.t). Şubat 2009, http://www.quality-ss.com/library/images/haasvf-4.jpg.

İmalat, (b.t). Şubat 2009. http://tr.wikipedia.org/wiki/%C4%B0malat.

İmalat, (b.t). Şubat 2009. http://www.mkn.itu.edu.tr/bolumler/imalat/meslek/meslek.htm.

Nakamura, M., Goto, S. ve Kyura, N., (1998). *Mechatronic servo system control-problems in industries and their theoretical solutions*. New York: Springer

Servo control facts, (b.t). Şubat 2009, www.baldor.com/pdf/manuals/1205-394.pdf.

Siemens simotion ST Structured Text programming and operating manual, (03/2007). https://a248.e.akamai.net/cache.automation.siemens.com/dnl/jM/jM0OTk0MQAA_27002409_HB/ST_Programming_en-US.pdf.

Simotion, Sinamics S120 and motors for production machines catalog PM 21, (2008). Şubat 2009, http://www.automation.siemens.com/mc-app/eclog/log.gif?owner=admc&folder=99a87a12-7df7-4b4e-86af-ce8532cd7f43&object=413c1bdc-0626-45b8-b176-67af3d8c0a0c&type=files&parent=efed2098-1e0c-4e92-a80e-b00ee72ade39&label=PM21_2008_complete_E2.pdf&lang=en&share=mc-sol&level=Internet&referer=%2fmarketing%2finternet%2fmc%2fmc-sol%2fen%2f99a87a12-7df7-4b4e-86af-ce8532cd7f43%2findex.aspx&curl=%2fmc%2fmediadb%2f413c1bdc-0626-45b8-b176-67af3d8c0a0c%2fPM21_2008_complete_E2.pdf&reference=download&redirect=%2fmc%2fmediadb%2f413c1bdc-0626-45b8-b176-67af3d8c0a0c%2f download%2fPM21_2008_complete_E2.pdf.

EKLER

ŞEKİLLER LİSTESİ

Şekil Sayfa

ŞEKİLLER LİSTESİ (devam)

Şekil Sayfa

ŞEKİLLER LİSTESİ (devam)

Şekil Sayfa

TABLO LİSTESİ

Tablo Sayfa

Diaresel İnterpolasyonVisual Basic Program Kodları

```
Public Function RoundTo(ByVal dblNumber As Double, ByVal _
   dblRoundTo As Double) As Double
'Examples:
'MsgBox RoundTo(2.231312312312, 0.1) = 2.2
'MsgBox RoundTo(2.261312312312, 0.1) = 2.3
    On Error Resume Next
    RoundTo = Round(dblNumber / dblRoundTo) * dblRoundTo
End Function
Function ArcSin(X As Double) As Double
    ArcSin = Atn(X / Sqr(-X * X + 1))
End Function
Private Sub Command1_Click()
X1 = Val(Text1.Text)
Y1 = Val(Text2.Text)
X2 = Val(Text3.Text)
Y2 = Val(Text4.Text)
I = Val(Text5.Text)
j = Val(Text6.Text)
g2 = Option2.Value
g3 = Option3.Value
If g2 = True Then
g2 = 1
g3 = 0
End If
If g2 = False Then
g2 = 0
g3 = 1
End If
r = Sqr((I) ^ 2 + (j) ^ 2)
a = Sqr((X2 + (-1 * X1)) ^ 2 + (Y2 + (-1 * Y1)) ^ 2)
10
'radyan cinsinden açı değeri
If (a / (2 * r)) = 1 Or (a / (2 * r)) = -1 Then
  alfa = 3.1416
  Else:
  alfa = 2 * ArcSin((a * 0.5) / r)
End If

mx1 = X1 + I
my1 = Y1 + j

If a = 0 And g3 = 1 Then
basla_açı = 0
bit_açı = 6.2831
End If

If a = 0 And g2 = 1 Then
basla_açı = 6.2831
bit_açı = 0
End If

If a = 0 Then GoTo 100
'çemberin kaç derece olduğunu hesapla
If X1 > mx1 And Y1 = my1 Then
     basla_açı = 0
End If
```

```
If X1 = mx1 And Y1 > my1 Then 'başlangıç noktası +y ekseni üzerinde
    basla_açı = 1.5707
End If

If X1 = mx1 And Y1 < my1 Then
    basla_açı = 4.7124
End If

If X1 < mx1 And Y1 = my1 Then
    basla_açı = 3.1416
End If

If X1 > mx1 And Y1 > my1 Then 'başlangıç noktası 1.bölgede
    basla_açı = Atn(Abs(j) / Abs(I)) '
End If

If X1 < mx1 And Y1 > my1 Then 'başlangıç noktası 2.bölgede
   basla_açı = Atn(Abs(I) / Abs(j)) + 1.5707
End If

If X1 < mx1 And Y1 < my1 Then 'başlangıç noktası 3.bölgede
    basla_açı = Atn(Abs(j) / Abs(I)) + 3.1416
End If

If X1 > mx1 And Y1 < my1 Then 'başlangıç noktası 4. bölgede
    basla_açı = Atn(Abs(I) / Abs(j)) + 4.7124
End If

xfark = Abs(X2 + (-1 * mx1))
yfark = Abs(Y2 + (-1 * my1))


If X2 > mx1 And Y2 = my1 Then 'bitiş noktası +x ekseni üzerinde
    bit_açı = 0
End If

If X2 = mx1 And Y2 > my1 Then  'bitiş noktası +y ekseni üzerinde
    bit_açı = 1.5707
End If

If X2 = mx1 And Y2 < my1 Then 'bitiş noktası -y ekseni üzerinde
    bit_açı = 4.7124
End If

If X2 < mx1 And Y2 = my1 Then 'bitiş noktası -x ekseni üzerinde
    bit_açı = 3.1416
End If

If X2 > mx1 And Y2 > my1 Then 'bitiş noktası 1.bölgede
    bit_açı = Atn(yfark / xfark)
End If

If X2 < mx1 And Y2 > my1 Then 'bitiş noktası 2.bölgede
    bit_açı = Atn(xfark / yfark) + 1.5707
End If

If X2 < mx1 And Y2 < my1 Then 'bitiş noktası 3.bölgede
    bit_açı = Atn(yfark / xfark) + 3.1416
End If
```

```
If X2 > mx1 And Y2 < my1 Then 'bitiş noktası 4. bölgede
    bit_açı = Atn(xfark / yfark) + 4.7124
End If

100
Print basla_açı, bit_açı

If g2 = 1 Then
    Open "C:\CNC\G2Hesabı.nc" For Output As #1
    Open "C:\CNC\G2HesabıX_Y.st" For Output As #2
    Open "C:\CNC\denemeG2.txt" For Output As #4

If basla_açı < bit_açı Then
aradeg1 = basla_açı
basla_açı = bit_açı
bit_açı = aradeg1
End If
   For I_1 = basla_açı To bit_açı Step -0.01745

   xkoordnat = mx1 + r * Cos(I_1)
   ykoordnat = my1 + r * Sin(I_1)
   xkoordnat2 = mx1 + r * Cos(I_1 + 0.01745)
   ykoordnat2 = my1 + r * Sin(I_1 + 0.01745)
   Print #4, "x"; xkoordnat, "y"; ykoordnat
   Line (1000 + xkoordnat2 * 10, 1000 + ykoordnat2 * 10)-(1000 + xkoordnat *
10, 1000 + ykoordnat * 10)

   motoradım_x = ((xkoordnat2 - xkoordnat) / 0.00556979) 'ara değişken
   motoradım_y = ((ykoordnat2 - ykoordnat) / 0.00556979)

   motorbil_x = (RoundTo((motoradım_x), 1)) 'adım bilgisi
   motorbil_y = (RoundTo((motoradım_y), 1))

   Print #1, "X"; motorbil_x, "Y"; motorbil_y
   Print #2, "move_horizontal(" + Str$(motorbil_x) + ");"
   Print #2, "move_vertical(" + Str$(motorbil_y) + ");"

Next I_1
End If
Close #1
Close #2
Close #4

If g3 = 1 Then
   Open "C:\CNC\G3Hesabı.nc" For Output As #3
   Open "C:\CNC\G3HesabıX_Y.st" For Output As #2
   Open "C:\CNC\denemeG3.txt" For Output As #5

    bit_açı = basla_açı + alfa
    For I_2 = basla_açı To bit_açı Step 0.01745 '0.01745 radyan olarak 1
dereceye denk

   xkoordnat = mx1 + r * Cos(I_2)
   ykoordnat = my1 + r * Sin(I_2)
   xkoordnat2 = mx1 + r * Cos(I_2 + 0.01745)
   ykoordnat2 = my1 + r * Sin(I_2 + 0.01745)

   Print #5, "x"; xkoordnat, "y"; ykoordnat

   Line (1500 + xkoordnat2 * 10, 1500 + ykoordnat2 * 10)-(1500 + xkoordnat *
```

```
10, 1500 + ykoordnat * 10)

   motorad_x = ((xkoordnat2 - xkoordnat) / 0.00556979) 'ara değişken
   motorad_y = ((ykoordnat2 - ykoordnat) / 0.00556979)

   motorbil_x = (RoundTo((motorad_x), 1)) 'adım bilgisi
   motorbil_y = (RoundTo((motorad_y), 1))

   Print #2, "move_horizontal(" + Str$(motorbil_x) + ");"
   Print #2, "move_vertical(" + Str$(motorbil_y) + ");"

   Print #3, "X"; motorbil_x, "Y"; motorbil_y
Next I_2
End If
Close #3
Close #2
Close #5

End Sub
```

Çizgisel İnterpolasyon Visual Basic Program Kodları

```
Function IsBetween(ByVal x As Double, ByVal limit1 As Double, ByVal limit2
As Double) As Boolean
    If limit2 > limit1 Then
        IsBetween = (x >= limit1) And (x <= limit2)
    Else
        IsBetween = (x <= limit1) And (x >= limit2)
    End If
End Function

Sub move_ladder(x As Long, y As Long)

    Dim myX As Double, myY As Double
    myX = 0
    myY = 0

    Dim intX As Long, intY As Long
    intX = 0
    intY = 0

    Dim stepX As Double, stepY As Double

    Dim tmpX As Integer, tmpY As Integer

    If Abs(stepX) > Abs(stepY) Then
        stepX = Sgn(x) * 1
        stepY = Sgn(y) * (Abs(y) / Abs(x))
    Else
        stepX = Sgn(x) * (Abs(x) / Abs(y))
        stepY = Sgn(y) * 1
    End If

    While IsBetween(myX, 0, x) And IsBetween(myY, 0, y)
        myX = myX + stepX
        myY = myY + stepY

        tmpX = CInt(myX)
        tmpY = CInt(myY)

        If tmpX <> intX Then

            move_horizontal (tmpX - intX)
            intX = tmpX

        End If

        If tmpY <> intY Then

            move_vertical (tmpY - intY)
            intY = tmpY

        End If
    Wend

   If intX <> x Then

        move_horizontal (x - intX)
```

```
    End If

    If intY <> y Then

        move_vertical (y - intY)

    End If
End Sub

Public Sub Command1_Click()
Dim adiX As Long, adiY As Long, adimX As Long, adimY As Long

X1 = Val(Text1.Text)
Y1 = Val(Text2.Text)
X2 = Val(Text3.Text)
Y2 = Val(Text4.Text)

deltaX = X2 - X1
deltaY = Y2 - Y1

adiX = RoundTo((deltaX / 0.00557), 1)
adiY = RoundTo((deltaY / 0.00557), 1)

absdeltaX = Abs(deltaX)
absdeltaY = Abs(deltaY)

adimX = RoundTo((absdeltaX / 0.005), 1)
adimY = RoundTo((absdeltaY / 0.005), 1)

If adimY = 0 And adimX = 0 Then End

If adimY = 0 Then
Open txtFileName.Text For Output As #1
    Print #1, "move_horizontal(" + Str$(adiX) + ");"
    Print #1, "move_vertical(" + Str$(adiY) + ");"

Close #1
End If

If absdeltaX = 0 Then
   Open txtFileName.Text For Output As #1
   Print #1, "move_horizontal(" + Str$(adiX) + ");"
   Print #1, "move_vertical(" + Str$(adiY) + ");"
   Close #1
End If

If absdeltaX = absdeltaY Then
    Open txtFileName.Text For Output As #1
    sayY_1 = 1
    For sayX_1 = 1 To adimX Step 10
       sayY_1 = sayY_1 + 10

       Print #1, "move_vertical("; Sgn(deltaX) * 10; ");"
       Print #1, "move_horizontal("; Sgn(deltaY) * 10; ");"

    Next sayX_1

    Close #1
End If
```

```
If absdeltaX < absdeltaY And absdeltaX <> 0 Then

    move_ladder adiX, adiY

End If

If absdeltaX > absdeltaY And absdeltaY <> 0 Then

 move_ladder adiX, adiY
     End If
End Sub
```

Structured Text Programı Kaynak Kodları

```
INTERFACE
USEPACKAGE CAM;(*$_ GridID:ffffffff $_*)
PROGRAM MCC_1;

END_INTERFACE
IMPLEMENTATION

    FUNCTION move_down : DINT
        VAR_INPUT
            amount : DINT;
        END_VAR
        move_down := _pos(axis:=Axis_2, direction:=NEGATIVE,
positioningMode:=RELATIVE, position:=amount, velocityType:=DIRECT,
velocity:=30, blendingMode:=INACTIVE, mergeMode:=IMMEDIATELY,
nextCommand:=WHEN_MOTION_DONE, commandId:=_getCommandId());
    END_FUNCTION

    FUNCTION move_up : DINT
        VAR_INPUT
            amount : DINT;
        END_VAR
        move_up := _pos(axis:=Axis_2, direction:=POSITIVE,
positioningMode:=RELATIVE, position:=amount, velocityType:=DIRECT,
velocity:=30, blendingMode:=INACTIVE, mergeMode:=IMMEDIATELY,
nextCommand:=WHEN_MOTION_DONE, commandId:=_getCommandId());
    END_FUNCTION

    FUNCTION move_left : DINT
        VAR_INPUT
            amount : DINT;
        END_VAR
        move_left := _pos(axis:=Axis_1, direction:=POSITIVE,
positioningMode:=RELATIVE, position:=amount, velocityType:=DIRECT,
velocity:=30, blendingMode:=INACTIVE, mergeMode:=IMMEDIATELY,
nextCommand:=WHEN_MOTION_DONE, commandId:=_getCommandId());
    END_FUNCTION

    FUNCTION move_right : DINT
        VAR_INPUT
            amount : DINT;
        END_VAR
        move_right := _pos(axis:=Axis_1, direction:=NEGATIVE,
positioningMode:=RELATIVE, position:=amount, velocityType:=DIRECT,
velocity:=30, blendingMode:=INACTIVE, mergeMode:=IMMEDIATELY,
nextCommand:=WHEN_MOTION_DONE, commandId:=_getCommandId());
    END_FUNCTION

    FUNCTION move_vertical : VOID
        VAR_INPUT
            amount : DINT;
        END_VAR

        VAR
            tmp : DINT;
        END_VAR

        IF (amount < 0) THEN
```

```
            tmp := move_down(ABS(amount));
        ELSE
            tmp := move_up(amount);
        END_IF;
    END_FUNCTION

    FUNCTION move_horizontal : VOID
        VAR_INPUT
            amount : DINT;
        END_VAR

        VAR
            tmp : DINT;
        END_VAR

        IF (amount < 0) THEN
            tmp := move_left(ABS(amount));
        ELSE
            tmp := move_right(amount);
        END_IF;
    END_FUNCTION

    PROGRAM MCC_1
    VAR
        _MccRetDINT : DINT;

    END_VAR
        ;
        (* Switch axis enable ('EnableAxis') *)
        _MccRetDINT := _enableAxis(axis:=Axis_1, movingMode:=DO_NOT_CHANGE,
enableMode:=ALL, servoControlMode:=ACTIVE,
servoCommandToActualMode:=INACTIVE, nextCommand:=WHEN_COMMAND_DONE,
commandId:=_getCommandId(), forcecontrolMode:=INACTIVE);

        (* Switch axis enable ('EnableAxis') *)
        _MccRetDINT := _enableAxis(axis:=Axis_2, movingMode:=DO_NOT_CHANGE,
enableMode:=ALL, servoControlMode:=ACTIVE,
servoCommandToActualMode:=INACTIVE, nextCommand:=WHEN_COMMAND_DONE,
commandId:=_getCommandId(), forcecontrolMode:=INACTIVE);

//----------------------------

//----------------------------

    END_PROGRAM
 END_IMPLEMENTATION
```