T.C.
SAKARYA ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

ENDÜSTRİYEL KONTROL SİSTEMLERİ İLE KESME MAKİNESİ OTOMASYONU

YÜKSEK LİSANS TEZİ

Elektrik ve Elektronik Müh. Cihan KARAMAN

Enstitü Anabilim Dalı	**:**	**ELEKTRİK VE ELEKTRONİK MÜH.**
Enstitü Bilim Dalı	**:**	**ELEKTRİK MÜH.**
Tez Danışmanı	**:**	**Y.Doç. Dr. Yılmaz UYAROĞLU**

Haziran 2007

T.C.
SAKARYA ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

ENDÜSTRİYEL KONTROL SİSTEMLERİ İLE KESME MAKİNESİ OTOMASYONU

YÜKSEK LİSANS TEZİ

Elektrik ve Elektronik Müh. Cihan KARAMAN

Enstitü Anabilim Dalı : ELEKTRİK ve ELEKTRONİK MÜH.

Enstitü Bilim Dalı : ELEKTRİK MÜH.

Bu tez 06 / 06 /2007 tarihinde aşağıdaki jüri tarafından Oybirliği ile kabul edilmiştir.

Yrd. Doç. Dr. Yılmaz UYAROĞLU	**Yrd. Doç. Dr. İlyas ÇANKAYA**	**Yrd. Doç. Dr. Fedai ÇAVUŞ**
Jüri Başkanı	**Üye**	**Üye**

TEŞEKKÜR

Tezin hazırlanması aşamasında bana her türlü desteği veren danışman hocam sayın Y.Doç. Dr. Yılmaz Uyaroğlu'na ve şirketinin imkanlarını sunan LSE Otomasyon firması'na ve bana herzaman destek olan aileme teşekkürü bir borç bilirim.

İÇİNDEKİLER

SİMGELER VE KISALTMALAR LİSTESİ

Bus	: Bütün cihazların ulaşabileceği haberleşme topolojisi
Cm	: Santimetre
Can	: Denetleyici alan ağı
DC	: Doğru akım
GND	: Toprak
HI	: Can bus sinyali
LOW	: Can bus sinyalinin tersi
IGBT	: Tetikleme devresi
IO	: Dijital giriş ve çıkış
KB/s	: Saniyedeki kilobyte sayısı
M	: Metre
MEL	: Negatif son limit
Mm	: Milimetre
Ms	: Milisaniye
PEL	: Pozitif son limit
PDO	: Proses veri nesnesi
Pulse	: Darbe
PC	: Bilgisayar
ORG	: Sıfır sinyali
OPC	: Proses kontrolü için nesne bağlama
Reset	: Değerin false değerine eşitlenmesi
SCADA	: Denetleyici kontrol ve veri denetimi
SDO	: Servis veri nesnesi
Set	: Değerin true değerine eşitlenmesi
RS	: Reset ve set
VDC	: Doğru akım voltaj seviyesi

ŞEKİLLER LİSTESİ

TABLOLAR LİSTESİ

ÖZET

Anahtar kelimeler: Can Bus, PLC, Servo Sürücüler, Endüstriyel Bilgisayarlar, Eksen Kontrol Kartı, SCADA, Delphi, Makine Otomasyonu.

Günümüzde otomasyon sistemlerinde kullanılan cihazların çeşitleri ve fonksiyonları artmıştır. Bu artış sistemlerin kontrol edilebilme gücünü arttırarak, daha karmaşık uygulamalara otomasyon çözümleri sunmuştur. Sistemin optimum tasarımı için, sistemdeki cihazların fonksiyon kabiliyetlerinin bilinmesi ve farklı cihazların bir sistem içerisinde çalışması sağlanmalıdır.

Bu çalışmanın amacı otomasyon sistemlerinde en çok kullanılan cihazların karmaşık iç yapılarına girilmeden sistemler içerisindeki görevlerinin ve haberleşme sistemlerinin belirlenmesidir. Bu tezde endüstriyel kontrol sistemlerinin birçok elemanını içerisinde bulunduran "kesme makinesinin" çalışma prensibi araştırılarak otomasyon sistemlerinin daha iyi anlaşılması amaçlanmıştır.

Bu çalışmanın sonucunda servo sürücüler, PLC, eksen kontrol kartı ve endüstriyel bilgisayar kullanılarak "kesme makinesinin" otomasyonu yapılmıştır. Otomasyon sistemini oluşturan bileşenlerin özellikleri incelenerek sistemler için optimum tasarım yöntemi oluşturulmuştur. Cihazlar arasında sistemin ihtiyaçlarına cevap verebilecek haberleşme sistemi kurulmuştur. PLC içerisinde sistemin çalışmasını sağlayacak lojik algoritma oluşturulmuştur. Kesme işleminin yapılabilmesi için hareket yöntemi belirlenmiştir. Eksen kontrol kartı kullanılarak istenilen hareket yöntemini sağlayacak referans üretilmiştir. Motorların üretilen referansı hatasız harekete dönüştürebilmesi için servo motor sürücüleri programlanmıştır. Delphi tabanında oluşturulan SCADA yazılımı ile sistemin kontrolü ve izlenmesi sağlanmıştır.

CUTTING MACHINE AUTOMATION WITH INDUSDUSTRIAL CONTROL SYSTEMS

SUMMARY

Key words: Can Bus, PLC, Servo Drives, Industrial Computers, Axis Control Card, SCADA, Delphi, Machinery Automation,

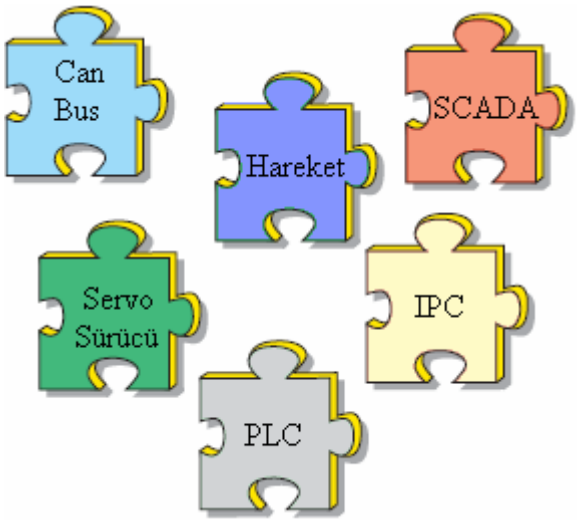
Nowadays, variety and capability of devices which are using in automation systems are improved. This improvement rise the control capability of devices and offer solutions to more complex applications. For optimum planning of system, capability of devices should known and different devices should work together.

The aim of this study is to clarify function of devices in complex automation systems and comminication systems between devices. Furthermore, device's complicated internal structure is not mentioned. While subject is clarfing, cutting machine which includes common automation devices is given an example. Thus, system could be understand better.

Automation system of cutting table is made conclude of this study by using servo drives, PLC, axis controller card and industrial computer. Features of devices which are used in automation systems, are studied for optimum desing metot for systems. Comminication system which could cover system requirements is established between devices. A logic algorithm is builded to run the system. Motion metod is determined for cutting process. A reference for required motion is produced by axis controller card. Servo drives are programmed for converting reference to faultless motion with motors. For controlling and observing of system is provided with a SCADA software which based on Delphi.

BÖLÜM 1. GİRİŞ

Otomasyon sistemlerinde kullanılan cihazların çeşitleri fazlalaşmıştır. Cihazların ihtiyaç duyulan fonksiyonları birarada çalıştırabilmesi için yüksek seviyeli dillerde haberleşmesi gerekmektedir. Otomasyon sistemleri tasarlanırken karşılaşılan en büyük sorun sistemde kullanılacak cihazların belirlenmesidir. Fazla özellikli ürünler seçildiğinde, maliyet artmakta, düşük özellikli ürünler seçildiğinde, istenilen fonksiyonlar yapılamamaktadır. Tasarımcının en optimum seçimi yapabilmesi için cihazların bütün özelliklerini ve otomasyon sistemlerindeki kullanım alanlarını iyi bilmesi gerekmektedir. Tasarımcılara sunulan kaynaklar, genellikle cihazların özelliklerini en ince detayına kadar açıklamakta fakat sistem içerisinde başka cihazlar ile birlikte kullanılması konusunu yeteri kadar aydınlatmamaktadır.



Şekil 1.1. Otomasyon sistemi içerisindeki bileşenler

Otomasyon sistemlerini oluşturan bileşenler şekil 1.1'deki gibi bir yap boza benzetilebilir. Otomasyon sistemlerini oluşturan servo sürücüler, PLC'ler, endüstriyel bilgisayarlar, eksen kontrol kartları, SCADA yazılımları ve haberleşme sistemleri bu yap bozun birer parçasıdır. Yap bozu oluşturan aynı donanımlar kullanılarak farklı birçok sisteme çözüm üretilebilir. Endüstriyel sistemler aynı temel

noktada birleşirler. Her makinede kontrol edilen motorlar mekanizmayı tahrik eder. Otomasyon sistemlerinde çözüm bekleyen temel problem motorların kontrol edilme yöntemidir. Bu çalışmada motorların optimum kontrol edilme yöntemi oluşturulmuştur.

Otomasyon sistemlerinde ayrıca sistemi izlemek ve kontrol etmek için bilgisayarlar, lojik algoritmaların kurulabilmesi için PLC'ler, tahrik etmek için motorlar ve valfler, hareketin profilini oluşturabilmek için kontrol kartları bir arada kullanılmaktadır. Bu konuları anlatan birçok kaynak bulunmaktadır fakat kaynaklar sadece bir cihaz üzerine yoğunlaşmıştır. Otomasyon sistemlerini anlatan kaynaklarda PLC konusunda birçok çalışma bulunmaktadır fakat PLC'nin SCADA sistemi içerisinde kullanımı veya sürücüler ile haberleşmesi konuları tam olarak aydınlatılmamıştır. Endüstriyel bilgisayarlar ve SCADA programları ile ilgili birçok kaynak bulunmaktadır, fakat edüstriyel bilgisayarlar içinde çalışan SCADA sistemleri ile PLC'nin veya motorların, kontrol edilmesi konuları tam olarak aydınlatılmamıştır. Bu çalışmada otomasyon sistemlerini oluşturan bileşenlerin sistem içerisideki görevleri açıklanmıştır. Cihazların teknik özelliklerinin yerine sistem içerisindeki çalışma yöntemleri incelenmiştir. Bu çalışma ile karmaşık otomasyon sistemlerinde bileşenlerin özellikleri incelenerek sistemler için optimum tasarım yöntemi oluşturulmuştur.

Otomasyon sistemlerinde haberleşmenin kullanılması ile sistemin kontrol edilebilme yeteneği artar. Sistemlerde veri alışveşi için kullanılan donanımlar azalır. Sistemin izleme kabiliyeti artar. Böylece sistemlerin kurulum ve işletme maliyetleri düşer. Ayrıca sistemdeki arızaların uzaktan izlenmesi sonucunda bakım maliyetleride düşer. İşletme içerisindeki uzak noktalardan haberleşme kanalları üzerinden dijital veri iletişimi yapılarak kablo maliyetleri de düşer. Haberleşme sistemlerinin tasarımları yapılırken sistemin genişleyeceği gözönünde bulundurulmalıdır. Eklere olanak tanımayan haberleşme sistemleri bir süre sonra sistemin ihtiyaçlarına cevap veremeyebilirler. Bu çalışmada sistemlerin ihtiyaçlarına cevap verebilecek haberleşme sistemi kurulmuştur.

Otomasyon sistemlerinde kontrol edilecek sistemin konumu elektriksel sinyaller aracığı ile PLC'lere iletilmelidir. Mekanizmanın konumu, PLC içerisinde yazılmış olan algoritma ile değiştirilebilmelidir. PLC içerisindeki algorima sonuçları dijital sinyaller olarak üretir. Oluşan dijital sinyaller tahrik elemanlarını tetikleyerek, mekanizmanın istenilen pozisyonu alması sağlar. PLC içerisindeki lojik algoritmaya giren ve algoritmadan çıkan değerler haberleşme kanalları ile bir panel üzerine taşınmalıdır. Böylece operatör sistemin çalışmasını tam olarak izleyebilir. Sistemde kullanılacak PLC seçimi önemlidir. Seçilen PLC haberleşme protokollerini desteklemelidir, arabirimleri kolay anlaşılmalıdır, kullanıcı dostu olmalıdır, ortak PLC dil standartlarına uymalıdır, maliyet açısından rekabetçi olmalıdır, haberleşme kanalları kullanılarak IO ilave edilebilmelidir ve ürünün uzun yıllar temin edilmesi üretici tarafıdan garanti edilmelidir. Bu özellikleri taşımayan PLC'ler zaman içerisinde geçerliliğini yitirmektedir. Bu çalışmada bütün bu özellikler gözönünde bulundurularak bir PLC seçilmiştir ve haberleşme kanalları kullanılarak IO ilavesi yapılmış ve gerekli yazılımların yapılma yöntemi açıklanmıştır. PLC sistem ile haberleştirilmiştir.

Sistemde eksenleri hatasız ve dinamik olarak hareket ettirebilmek için servo motorlar kullanılmalıdır. Servo motorların görevlerini hatasız yerine getirebilmeleri için servo sürücüler tarafından kontrol edilmelidir. Sürücüler içerisinde, PLC gibi geniş olmasada, lojik algoritmalar üretilebilmektedir. Bu lojik algoritmalar içerisinde gerekli fonksiyonlar PLC'lere taşınmadan çözülerek, sistemin çalışması için gerekli programın yazılması kolaylaşmaktadır. Yüksek fonksiyon kabiliyeti gerektiğinde sürücüler içerisine PLC kartları eklenebilmektedir. Sisteme motorların adaptasyonu için servo sürücüler üzerinde birçok ayar yapılması gerekmektedir. Dolayısı ile servo sürücülerin programlanması için en önemli nokta, programlamayı sağlayan ara yazılımdır. Bu ara yazılım eğer kullanıcı tarafından iyi anlaşılamaz ise motorlar verimli çalışamazlar. Sahada servo sürüler kullanılmadan önce deney ortamlarında ayar yapılabilecek noktalar belirlenmelidir. Servo sürücü seçilirken sistemin ihtiyaçları dikkate alınmalıdır. Sistemin kontrolünde oluşan hatalar programlama eksikliğinden kaynaklanabileceği gibi cihazların eksikliğinden de oluşabilir. Sistemin ihtiyaçlarından fazlasını karşılayabilecek bir sürücü seçildiğinde sistemin kurulum maliyeti artmaktadır. Servo sürücüler haberleşme sistemleri ile anlık durumlarını

PLC'lere iletebilmelidirler. Haberleşme sistemlerinin kullanılmadığı sürücülerde veri iletişimi için dijital sinyaller kullanılmakta ve sistemin maliyeti artmaktadır. Bu çalışmada motorları hatasız kontrol edebilecek sürücüler programlanmıştır ve sistem ile arasındaki haberleşme kurulmuştur.

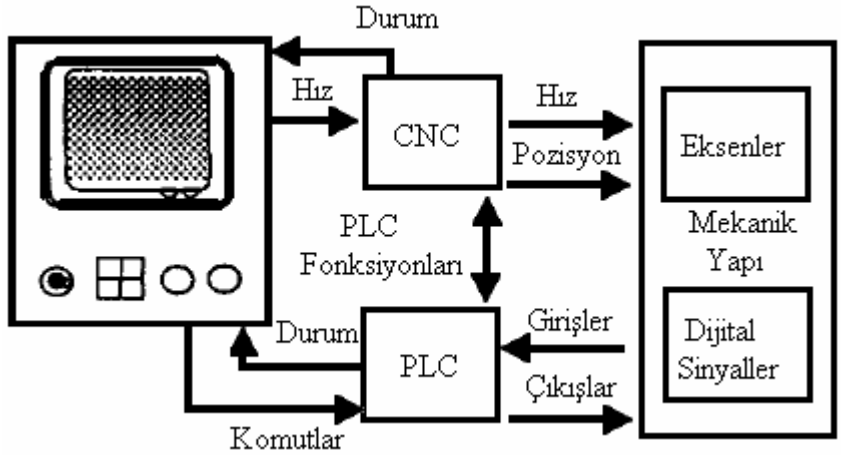
PLC'ler ve oparatör panelleri karmaşık otomasyon sistemlerinin bütün ihtiyaçlarını karşılayamamaktadır. Bilgisayarlar işletmelerdeki verilerin veri tabanı içerisine alınması, karmaşık makine otomasyonları ve uzaktan erişim taleplerine PLC'lerden daha fazla cevap verebilmektedir. Ayrıca bilgisayarların teknolojik kısıtlamaları PLC'lerden daha azdır. Bilgisayarların PLC'lere göre fonksiyon kabiliyetleri fazlalığı ile orantılı olarak, programlanmaları da karmaşıktır. Lojik algoritmalar PLC'deki gibi kolay ve hızlı oluşturulamaz ve ortamlardaki gürültüye karşı PLC'ler kadar dayanıklı değildir. Bu sebeplerden ötürü, sistemler içerisinde lojik fonksiyonların üretilmesi için PLC'ler kullanılmalıdır. Üretilen sonuçların kullanıcıya iletilmesi ve veri tabanlarında saklanması için de bilgisayarlar kullanılmalıdır. Sahada kullanılacak bilgisayarlar kötü koşullara karşı dayabilmesi için özel olarak üretilmelidir. Ortamda bulunan neme ve toza karşı dayanıklı olmalıdırlar. Kontrol kartları üzerinde kullanılan komponentler yüksek ısılara karşı dayanıklı olmalıdır. Sistemlerde oluşabilecek titreşime dayanıklı olmalıdır. Bilgisayarlar üzerinde çalıması için SCADA yazılımları Delphi gibi programlama dilleri ile üretilebilir veya hazır yazılımlar satın alınabilir. Kullanılacak yazılım sistemin ihtiyaçlarına göre maliyet analizi yapılarak belirlenmelidir. SCADA yazılımı ile bilgisayar, sistemde bulunan bütün cihazları ortak bir protokolde birleştirmektedir. Bu çalışma ile bir SCADA yazılımı üretilmiş ve sistem kontrol edilmiştir. SCADA yazılımı ile sistemdeki cihazların durumları izlenmiştir.

Sürücülerin içerisinde motorları kontrol edecek algoritmalar oluşturulabilir. Bu algoritmalar tek eksen kontrolünde yeterli düzeyde olsada birden fazla eksenin bir arada çalışabilmesi için yeterli değildir. Birden fazla sürücü senkronizasyon veya pozisyonlama gibi grup hareketlerini hatasız yapabilmektedir. Sürücüler içerisindeki algoritma kullanılarak senkronizasyon uygulamalarında birden fazla sürücü aynı anda hızlanabilir veya yavaşlayabilir. Pozisyonlama uygulamalarında ise sürücüler aynı anda harekete başlayabilir fakat farklı zamanlarda hareketi bitirir. CNC,

malzeme işleme veya robot uygulamalarında eksenleri oluşturan motorlar pozisyonlama işlemlerinine aynı anda başlamalı ve bitirmelidir. Bu uygulamalarda oluşturulacak hareket bir motorun kendi başına hareketi değildir. İstenilen hareket motorların birarada uzayda oluşturduğu vektörel harekettir. Birçok eksen ile vektörel hareket oluşturulmak için sürücülerin içerisindeki algoritmalar yetersiz kalmaktadır ve bu hareketi oluşturabilecek eksen kontrol kartları kullanılmalıdır. Eksen kontrol kartları sadece motorların hareketi için gerekli referans kaynağını üretir. Bu referans kaynağı sürücülere iletilerek motorların hareketi sağlanır. Eksen kontrol kartlarında ayrıca daire ve teğet hareketleri için gerekli referanslar da üretilebilir. Bu çalışmada eksen kontrol kartı kullanılarak sürücüler tahrik edilmiştir ve uzayda istenilen vektörel hareket oluşturulmuştur.

Bu çalışmada kesme makinesinin çalışma yöntemi açıklanarak otomasyon sistemlerinin daha iyi anlaşılması amaçlanmıştır. Makinenin oluşturulabilmesi için ilk olarak kumanda edilecek cihazlar ve ölçme yapılacak sensörler belirlenmiştir. Kesme yapacak mekanizmanın hareketi hava ile kontrol edileceğinden tahrik için pnomatik valfler kullanılmıştır. Elektriksel sinyaller aracılığı ile bobinlere bağlanan piston üzerindeki hava basıncı kontrol edilerek pistonun hareketi sağlanmıştır. Kesme masasının yukarı ve aşağı hareketi fazla güç gerektirdiğinden hareket için hava yerine hidrolik pistonlar kullanılmıştır. Hidrolik motoru ile gerekli basınç ayarlandıktan sonra hidrolik pistonlara verilen elektrik sinyalleri ile hareket sağlanmıştır. Hidrolik ve pnomatik pistonların pozisyonlarını elektriksel sinyallere çevirebilecek sensörler kullanılmıştır. Makine üzerinde ileri ve geri yönde değişken hareketli üç adet eksen vardır. Eksenler dinamik ve hassas kontrol edilebilmesi için servo motorlar tarafından tahrik edilmiştir. Eksenlerin hareketlerinin mekanizmaya zarar vermemesi için eksenlerin başlangıç ve bitiş noktalarına limit sensörler koyulmuştur. Makinedeki durumun kullanıcı tarafından daha kolay izlenebilmesi için yeşil, sarı ve kırmızı lambalar kullanılmıştır. Makinenin kontrolünün kolaylaştırılması için lojik komutlar pano üzeriden tuşlar ile sisteme iletilmektedir. İkinci aşamada kumanda edecek cihazlar bir pano içerisine yerleştirilerek dış ortamdan yalıtılmıştır. Makine üzerindeki tahrik elemanlarının kabloları kumanda edecek cihazlara bağlanmıştır. Hidrolik ve pnomatik sistemleri kontrol eden valflerin bobinleri ile pozisyonlarını bildiren sensörler, lojik komutları veren tuşlar ve uyarı

lambaları PLC'ye bağlanmıştır. Eksenleri kontrol edecek motorların uçları servo sürücülere bağlanmıştır. Mekanizmadaki limit ve başlangıç noktalarını belirten sensörler eksen kontrol kartına bağlanmıştır. Eksen kontrol kartı bilgisayara takılarak pano içerisine yerleştirilmiştir. Bilgisayarın ekranı panonun kapağına yerleştirilerek kullanıcının sistemi izlemesi ve kontrol etmesi sağlanmıştır. Makinenin çalışması için gerekli lojik algoritma PLC'nin içerisinde oluşturulmuştur. Üçüncü aşamada Can Bus protokolü ile sistemde bulunan sürücüler, PLC ve bilgisayar arasında haberleşme kurulmuştur. Dördüncü aşamada Delphi programı kullanılarak bilgisayar içerisinde SCADA yazılımı oluşturulmuştur. SCADA yazılımı içerisinde OPC programı kullanılarak sistemdeki diğer cihazlar ile haberleşme kurulmuştur. Beşinci aşamada SCADA yazılımının sistemi kontrol etmesi ve izlemesi sağlanmıştır. Kesilecek malzemenin boyutuna göre oluşturulan Gcode dizisi, SCADA yazılımı üzerinden eksen kontrol kartına gönderilerek, kesme işlemi yapılmıştır. Makinede kesme işlemi yapılmak istendiğinde oluşturulan SCADA yazılımı, Gcode dizilerinden pozisyonları okur ve gerekli hareket komutlarını eksen kontrol kartına gönderir. Eksen kontrol kartı komutların gerçekleşmesi için her eksene özel referans üretir. Referanslar arasındaki interpolarizasyonu eksen kontrol kartı yapmaktadır. Kesme makinesinin çalışma şeması şekil 1.2'de gösterilmiştir.



Şekil 1.2. Kesme makinesinin çalışma şeması [10]

PLC, sürücü veya bilgisayar gibi farklı cihazların içerisindeki fonksiyonların bir arada çalışması için, bütün cihazların ortak bir protokolde iletişim kurması gerekmektedir. Haberleşme sistemleri kullanılarak, farklı cihazlardaki fonksiyonlar,

birbirleri ile ardışık çalışabilirler. Haberleşme protokollerinden Can Bus, cihazlar arasında kullanılacak ortak haberleşme protokolü olarak seçilmiştir. Can Bus haberleşme sistemi içerisinde kullanılan terimlerin anlamları bulunmuştur [1]. Otomasyon sistemlerindeki Can Bus mimarisi [2], genel otomasyon sistemlerinde, System Bus haberleşme sistemlerinin kullanılması ve cihazların birbirleri ile haberleşmesinin kurulması incelenmiştir [3]. Can Bus haberleşme sisteminin servo sürücüler ve eksen kontrol sistemleri ile kullanılması araştırılmıştır [4]. Can bus haberleşme sisteminde kullanılan entegreler belirlenmiştir [5]. Can Bus sisteminin katman yapısı ve telegram içeriği [6] ve Can Bus sinyallerinin yapısı anlaşılmıştır. [7].

Sistemler oluşturulurken dijital sinyalleri kullanarak bir çok lojik algoritmanın üretilmesi gerekmektedir. Lojik algoritmaların en kolay oluşturulabileceği cihaz PLC'dir. PLC tabanlı sistemlerin, röle ve bilgisayar tabanlı sistemler ile arasındaki farklılıklar anlaşılarak [8], PLC'nin otomasyon sistemleri içerisinde kullanılma yöntemleri, programlanması ve haberleştirme yöntemleri incelenmiştir [9]. CNC makinelerinin çalışma sistemi içerisinde eksen kontrol kartının ve PLC'nin kullanımı araştırılmıştır [10]. PLC tabalı motor kontrol algoritması anlaşılmıştır [11]. PLC'nin farklı programlama mantıkları incelenmiştir [12-14].

Sistemin hassas bir şekilde tahrik edilmesi için servo motorlar ve sürücülerin kullanılması gerekmektedir. Lenze firmasının ürettiği Servo sürücülerin kullanılması hakkında eğitim notları alınmıştır [15]. Böylece servo motorlar ve sürücüler hakkında pratik bilgiler edinilmiştir. Servo sürücülerin içerisindeki fonksiyon bloklarının yapısı anlaşılmıştır[16].

Sistemin en üst seviyeli konrtrolü için bilgisayar tabanlı bir yazılımın kullanılması gerekmektedir. Bilgisayar tabanlı kontrol sistemi olan SCADA sistemlerini oluşturan bileşenler ve bu bileşenler ile yapılan çalışmalar hakkında bilgi alınmıştır [18]. Piyasada kullanılan bir SCADA yazılımının içeriğini incelenmiştir [19]. Sahada SCADA sistemlerinin çalıştırabilecek bilgisayarlar, özel olarak tasarlanmalıdır. Endüstriyel bilgisayarların kullanım alanları ve otomasyon sistemindeki yeri araştırılmıştır [20]. SCADA sisteminin PLC'ler ile çalışma alanları incelenmiştir

[21]. İnterpolarizasyon gibi kontrol yöntemlerini gerçekleştirmek için eksen kontrol kartı kullanılmalıdır. Eksen kontrol sistemi ile ilgili genel bilgi edinilmiştir. Tasarlanan hareketin kontrol kartı üzerinden sistemi tahrik etmesi için gerekli yöntem araştırılmıştır [22]. Delphi yazılımı ile sistemin izlenebileceği ve kontrol edilebileceği bir SCADA programı üretme yöntemi araştırılmıştır [23-24].

Bölüm 2'de Can Bus haberleşme sistemi araştırılmıştır. Haberleşme sisteminin otomasyon alanında kullanılma sebepleri belirtilmiştir. Kesme makinesinde kullanılan cihazlar örnek gösterilerek, Can Bus sisteminin haberleşme alt yapısı ile haberleşme sistemi içerisindeki kullanılabilecek haberleşme kanalları açıklanmıştır. Haberleşme sistemi için gerekli olan kablo yapısı ve topraklama yöntemi anlaşılmıştır. Haberleşme sistemi içerisindeki cihazların birbirleri ile haberleştirilme yöntemi açıklanarak ve örnek bir SDO telegramının içeriği belirlenmiştir.

Bölüm 3'de PLC'nin sistemler içerisindeki çalışma yöntemleri ve kesme makinesinde oluşturulan yazılım anlaşılmıştır. PLC'nin çalışma sistemi açıklanarak diğer kontrol sistemleri ile arasındaki farklar karşılaştırılmıştır. PLC'nin sistem içerisindeki çalışma yöntemi belirlenmiştir. Kesme makinesinin çalışması için PLC programı oluşturulmuştur. PLC ile ek IO modüllerinin haberleştirilmesi için bir yöntem geliştirilmiştir. Sistemin otomatik olarak çalışabilmesi için gerekli PLC algoritması kurulmuştur. Bu algoritma içerisinde otomatik başlangıç şartlarının oluşturulması, farklı noktalardan gelen sinyaller ile sistemin bir noktadan kontrol edilmesi sağlanmıştır. PLC'nin "home" işlemini yapabilmesi için gerekli algoritma oluşturulmuştur. Sistem çalışmadan önce "home" işleminin gerekliliği belirtilmiştir. Farklı noktalardan gelen "home" sinyalleri bir noktada toplanarak ve başlangıç şartları da gözetilerek "home" işlemi için gerekli algoritma kurulmuştur. Sistemin manuel çalışması için gerekli algoritmalar oluşturulmuştur. Manuel olarak kumanda edilecek pnomatik valflerin kontrolü için gerekli algoritma oluşturulmuştur. Sistemde oluşabilecek arızalar bir noktada toplanmış ve kullanıcıyı arıza durumunda uyarabilecek bir algoritma tasarlanmıştır. Haberleşme sisteminde oluşan gecikme sürelerini belirlemek için bir algoritma oluşturulmuştur. PLC'nin OPC haberleşme sisteminde çalışabilmesi için gerekli kodlama sistemi oluşturulmuştur.

Bölüm 4'de servo motor sürücülerinin çalışma sistemleri anlaşılmıştır. Servo motor sürücüleri hakkında genel bilgiler, servo sürücülerin kullanılma sebepleri, kullanılacak servo motorda olması gereken özellikler, sürücülerin üzerindeki bağlantı noktaları, sürücüleri programlayabilmek için gerekli arayüzler, sürücülere verilebilecek hız referanslarının kaynakları ve cihazlara enerji verilmeden önce yapılması gerekenler açıklanmıştır. Servo sürücülerinin fiziksel olarak çalışması ve yazılımsal olarak programlanması seviyelere ayrılmış ve bu seviyeler açıklanmıştır. Servo sürücülere bağlanabilecek motorların özellikleri belirlenmiştir. Kesme makinesinde servo sürüler için bir program oluşturulmuştur.

Bölüm 5'da hareket çeşitleri ve hareketi oluşturma yöntemleri açıklanmıştır. İnterpolarizasyon kavramı ve interpolarizasyon içeren hareketleri oluşturabilmek için gerekli donanımlar ortaya konulmuştur. Hareketin oluşturulabilmesi için gerekli encoder sinyalleri ve mekanizma üzerindeki limitlerin algılanabilme yöntemleri belirlenmiştir. "Home" işlemini kontrol kartı üzerinden gerçekleştirebilecek algoritma belirlenmiştir. Hareketin oluşturulabilmesi için kontrol kartına verilmesi gereken kodlar anlaşılmıştır. Lineer ve S rampaları ile kalkış ve duruş arasıdaki farklar belirtilmiştir. Lineer ve dairesel interpolarizasyonun farkları açıklanmıştır. Kontrol kartı ile fiziksel sinyaller arasında oluşturulabilecek bağlantı yöntemi belirlenmiştir.

Bölüm 6'de endüstriyel bilgisayarlar ve kullanım alanları araitırılmıştır. Endüstriyel bilgisayarların özellikleri, standart bilgisayarlar ile endüstriyel bilgisayarlar arasındaki farklar belirtilmiştir. Kesme makinesinde endüstriyel bilgisayarın kullanım amacı belirtilmiştir. SCADA sisteminin yapısı ve otomasyon uygulamalarındaki yeri belirlenmiştir. Sistemin çalışması için endüstriyel bilgisayar üzerinde gerekli SCADA yazılımı oluşturulmuş, oluşturulan yazılım ile eksen kontrol kartı kontrol edilmiş, eksen kontrol kartının ürettiği pulse dizisi motor sürücülerine gönderilerek hareket oluşturulmuştur. Sistemdeki cihazlar ile endüstriyel bilgisayar içindeki SCADA yazılımı arasındaki haberleşme, OPC programı kullanılarak sağlanmıştır. Sistemlerin genel özellikleri belirlenmiştir. Sistemin çalıştırılabilmesi ve izlenebilmesi için gerekli SCADA yazılımı Delphi programlama dilinde üretilmiştir. SCADA yazılımı için Delphi programının

kullanılma sebepleri ortaya koyulmuştur. Delphi programlama dili hakkında bazı temel noktalar belirtilmiştir. Program içerisinde formlar oluşturulmuş ve formların açılış ve kapanış prosedürleri belirlenmiştir. Gcode satırlarını değişkenlere ayırabilecek bir algoritma oluşturulmuştur. Gcode satırlarını, Delphi içerisinde belirlenmiş değişkenlere atayabilecek bir algoritma oluşturulmuştur. Gcode'ları dosyadan okuyabilecek ve değişkenlere atayabilecek bir prosedür üretilmiştir. Sistem üzerinde mm cinsinden girilen değerleri gerekli pulse sayısına çevirebilecek fonksiyonlar tanımlanmıştır. Değişkenlere atanmış olan veriyi eksen kontrol kartında harekete dönüştürebilecek komutlar seçilerek, istenilen hareketin oluşturulması sağlanmıştır. Gcode satırlarının ardışık çalışması için gerekli algoritma üretilmiştir. Sistemin otomatik çalışması ve durması için gerekli komutlar üretilmiştir. Anlık pozisyonların yazılım üzerinde izlenebilmesi için, kontrol kartından pulse değerlerini okuyacak ve okunan pulse değerlerini mm cinsine çevirerek ekranda gösterebilecek bir algoritma oluşturulmuştur. Home işleminin oluşturulabilmesi için home fonksiyonunu seçebilecek ve gerçekleştirebilecek bir algoritma oluşturulmuştur. Sistemin manuel hareket etmesi için Delphi tabanında hazırlanmış form üzerinde gerekli fonksiyonlar belirlenmiştir. OPC programı, Delphi içerisinde kullanılarak SCADA yazılımı ile sistemin haberleşmesi sağlanmıştır. SCADA yazılımına sistemdeki cihazlardan okunmak istenen kodların tanımlaması yapılmıştır. Sistemden komut alınırken ve sisteme komut gönderilirken sistemin optimum kullanımı için bit dizileri ile word değişkenleri arasında bir algoritma oluşturulmuştur. PLC üzerindeki dijital giriş ve çıkış sinyalleri, SCADA yazılımı üzerinde gösterebilecek bir algoritma oluşturulmuştur. Sistem üzerinde bulunan kısıtlamalar sistemin çalışmasına eklenmiştir. Kısıtlamalar eklendikten sonra sistemi kontrol edecek komutlar için yeni tanımlamalar üretilmiştir. OPC üzerinden veri okunabilmesi ve yazılabilmesi için prosedürler üretilmiştir. Uyarı sinyallerinin SCADA yazılımında gösterilmesi için bir yöntem oluşturulmuştur.

BÖLÜM 2. CAN BUS HABERLEŞME SİSTEMİ

2.1. Giriş

Günümüzde otomasyon sistemleri içerisinde haberleşme sistemlerini kullanmak çok yaygınlaşmıştır. Otomasyon sistemleri içerisinde sistemi izlemek için bilgisayarlar, kontrol etmet için PLC'ler, motorları tahrik etmek için sürücüler kullanılmaktadır. Aynı sistem içerisinde farklı cihazlar bir arada kullanıldığında, bir fonksiyon birden fazla cihazın bir arada çalışması ile oluşturulabilir. Bu durumda, cihazlar arasında veri iletişiminin oluşturulması gerekir. Örnek olarak bilgisayar ekranından PLC'ye gönderilen hız değerinin, sürücüye ulaşması gerektiğinde, bilgisayar ile sürücü arasında bir iletişim sisteminin kurulması gereklidir. İletişim gerektiren işlemler için dijital sinyaller veya haberleşme sistemleri kullanılabilir. Dijital sinyaller kullanıldığında hem komutun gönderileceği kaynak tarafından sinyal üretilebilecek, hemde komutu algılayacak cihaz tarafından sinyali algılayabilecek donanımlar gereklidir. Ayrıca bu sinyalleri taşıyacak kablolar da kullanılmalıdır. Bu sinyallerin fazla olduğu sistemlerde sinyal alışverişi için birçok ek donanımda gerekebilir. Haberleşme sistemleri kullanıldığında bütün veri alış verişi, haberleşme kabloları aracılığı ile haberleşme donanımları kullanılarak yapılabilir. Kesme makinesinde sürücüler üzerindeki arıza sinyallerini PLC'ye iletebilmek için Can Bus haberleşme protokolü kullanılarak sürücü üzerindeki dijital çıkışlar, PLC üzerindeki dijital girişler ve cihazlar arasında sinyalleri taşıyacak kabloların kullanılmasına ihtiyaç kalmamıştır. Haberleşme sistemlerinin kullanılmasının en büyük avantajı, sistemlerin kontrolünün daha az donanım kullanılarak yapılabilmesi ve böylece sistemin kurulum maliyetinin düşmesidir. Haberleşme sistemleri kullanıldığında, cihazlar arasında istenildiği gibi veri alışverişi sağlanabildiğinden, PLC'nin sistem üzerindeki kontrolü arttırılabilir, sistem daha esnek bir yapı ile kontrol edilebilir ve izlenebilir, sistemdeki hatalar daha kolay görülebilir. Haberleşme sistemleri kullanılarak cihazların durumları uzak mesafelerden de izlenebilir. Üretilen makinelerin arızaları,

makinenin yanına gidilmeden, haberleşme kanalları ile bağlantı kurularak uzaktan bulunabilir. Haberleşme sistemlerine ek dijital giriş ve çıkış modülleri eklenerek, uzak mesafedeki sinyaller, sadece haberleşme kablosu kullanılarak PLC'lere taşınabilirler. Böylece sistemlerde kullanılan kablolar azaltılarak, sistemin maliyeti düşürülebilir.

2.2. Haberleşme Alt Yapısı

Haberleşme alt yapısını kurabilmek için ilk belirlenmesi gereken sistemde kaç cihaz olduğudur. Haberleşme hattındaki cihazlardan biri, haberleşmeyi başlatması için master seçilmelidir. Birden fazla master cihaz seçilebilir fakat bu haberleşme sisteminin yükünü arttırır. Can Bus ile haberleşme kurulmadan önce cihazlar arasında haberleştirilecek sinyallerin belirlenmesi gerekmektedir. Can Bus haberleşme sisteminde üç adet PDO kanalı bulunmakta, her bir PDO kanalından 4 word gönderilebilmektedir. SDO kanalı ile gönderilecilecek verinin sınırı olmasa da, PDO kanalına göre veri iletim hızı çok düşüktür. Haberleşme kısıtlamalarında dolayı, haberleştirilecek cihazlar arasındaki veri trafiği düzenlenmelidir. Cihaz sayısı arttıkça haberleşme kanalının yoğunluğu arttığından, haberleşme hızının düşürülemesi gerekmektedir. Haberleşme kablosunun uzaması ve çapının azalması haberleşme hızının düşmesine sebep olur. Cihaz sayısına ve haberleşme kablosunun uzunluğuna göre kullanılacak haberleşme hızı belirlenmelidir [2].

Can Bus içerisinde en fazla 63 cihaz arasında haberleşme kurulabilir. Her cihazın adresi farklı olmalıdır. Kullanılabilecek en uzun kablo uzunluğu 1 km'dir ve bu uzunlukta haberleşme hızı 50 kBit/sec'dir. En yüksek haberleşme hızı ise 1000 kBit/sec'dir ve en uzun 25 m kablo üzerinde kullanılabilir. Bus uzunluğuna göre veri hızı tablo 2.1'de belirtilmiştir. Haberleşme erişim protokollerinden CSMA/CA'yı kullanarak verilerin haberleşme hattında çarpışmasını en aza indirir. Verilerde bir hata oluştuğunda, alıcı tarafından veriyi gönderen cihaza hata telegramı gönderilir, veriyi gönderen cihaz veriyi tekrar gönderir. Kullanılabilecek en uzun telegram 106 bit'ten oluşur [7].

Tablo 2.1. Bus uzunluğuna göre veri hızı [7]

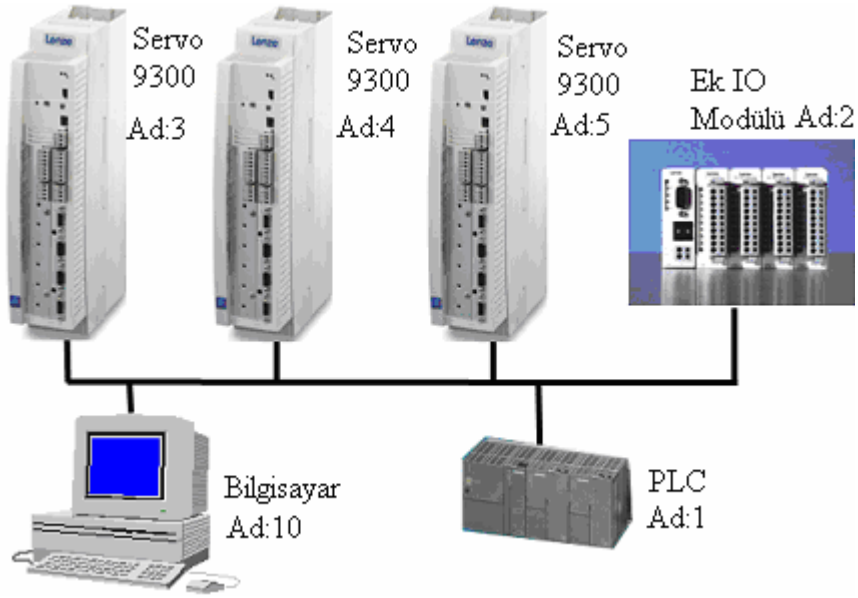
Bus Uzunluğu m	Veri Hızı kbit/s	Veri süresi µs
25	1000	1
50	800	1.25
100	500	2
250	250	4
500	125	8
650	100	10
1000	50	20

Can Bus haberleşme sistemi üç aşamadan oluşur. Açılış aşaması, elektronik kartlara ilk enerji uygunadığı anda başlar ve devre haberleşmeye hazır olduğunda biter. Bu süre içerisinde kart ile herhangi bir haberleşme kurulamaz. Açılış aşaması tamamlandıktan sonra, haberleşme devresi kendini otomatik olarak çalışma öncesi aşamasına geçirir. Bu aşamada cihaz tam olarak haberleşmeye başlamış sayılmaz. Sadece SDO kanalı ile gelen telegramlara cevap verir. Çalışma aşamasına geçmek için master cihaz tarafından komut bekler. Master cihazından çalışma komutu aldıktan sonra haberleşme devresi kendini çalışma aşamasına geçirir ve PDO haberleşmesini başlatır. Haberleşme devresi bu aşamada tamamen haberleşmeye başlamıştır. Kesme makinesinde kullanılan cihazlarda 352 numaralı kod değeri bir seçildiğinde, cihaz master yapılmış olur. Master cihazın görevi haberleşme sistemindeki cihazlara çalış komutu göndererek, cihazların haberleşmeye başlamasını sağlamaktır. Haberleşme sisteminin sıfırlanması gerektiğinde, bütün hattı sadece master cihaz sıfırlayabilir.

Kesme makinesinin haberleşme hattında altı cihaz bulunmaktadır. Sürücüler ile PLC arasındaki haberleşme sistemi için gerekli veri uzunluğu bir PDO kanalını aşmamaktadır. Haberleşme hattındaki en büyük yoğunluk, bilgisayar ile OPC programı ile hattaki veri değişimlerini incelerken oluşturmaktadır. Kullanılan haberleşme kablosunun uzunluğu 5 m'dir. Sistem üzerindeki cihaz sayısı ve kablo uzunluğu dikkate alındığında, 1000 kBit/s haberleşme hızını kullanmak mümkündür.

Haberleşme hattının, enerji hattından kaynaklanan parazitlere karşı daha dayanıklı olması için, veri hızı 500 kBit/s seçilmiştir.

Kesme makinesinde PLC, bilgisayar, ek IO modülü ve üç tane servo sürücü bulunmaktadır. Cihazların donanımları üzerindeki Can Bus entegresi ile, birbirleri arasında haberleşme kurulmalıdır. Sistemdeki bütün cihazların adreslerinin farklı olması gerekmektedir. PLC adresi bir, ek IO modülünün adresi iki, X ekseninin sürücüsünün adresi üç, Y ekseninin sürücüsünün adresi dört, Z ekseninin sürücüsünün adresi beş olarak ayarlanabilir. Bilgisayar sadece izleyici olacağından, sabit büyük bir adres verilebilir. Kesme makineside kullanılan Can Bus sisteminin alt yapısı şekil 2.1'de gösterilmiştir.

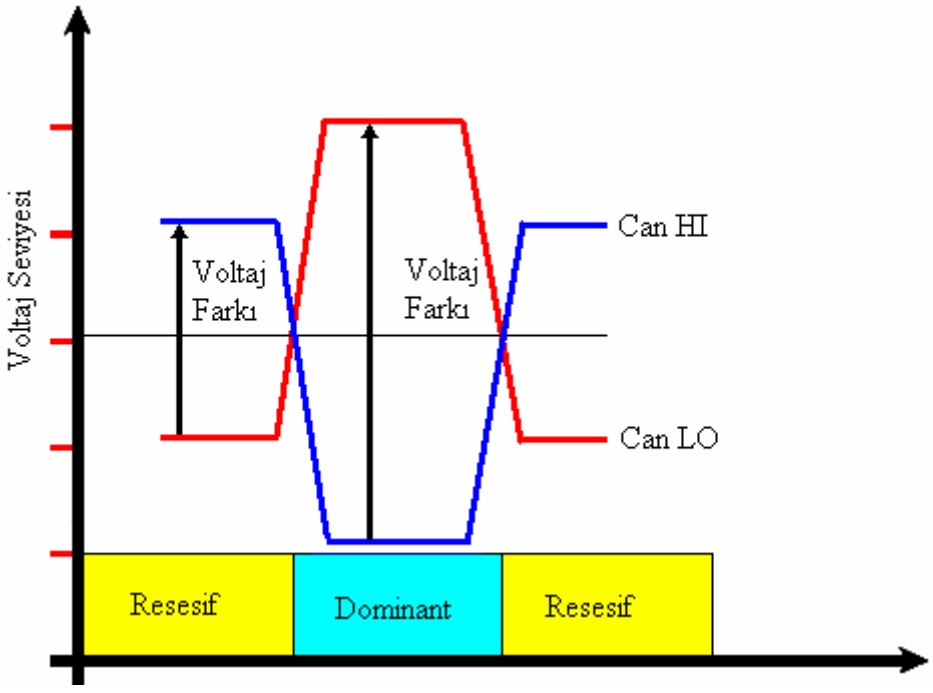


Şekil 2.1. Kesme makinesinde kullanılan Can Bus haberleşme sisteminin alt yapısı

Can Bus'in haberleşme alt yapısını kurmak karmaşıktır. Lenze firması, Can Bus protokolünün bazı özelliklerini değiştirerek, yeniden tasarlamış, haberleşme alt yapısının kurulumunu kolaylaştırmış ve ismine System Bus adını vermiştir. System Bus, Can Bus protokolünün özelliklerinden sadece SDO ve PDO telegramları ile, haberleşme sistemini içerir. Can Bus ve System Bus protkolleri açık bir ISO standartıdır.

2.3. Haberleşme Sisteminin Kablo Yapısı

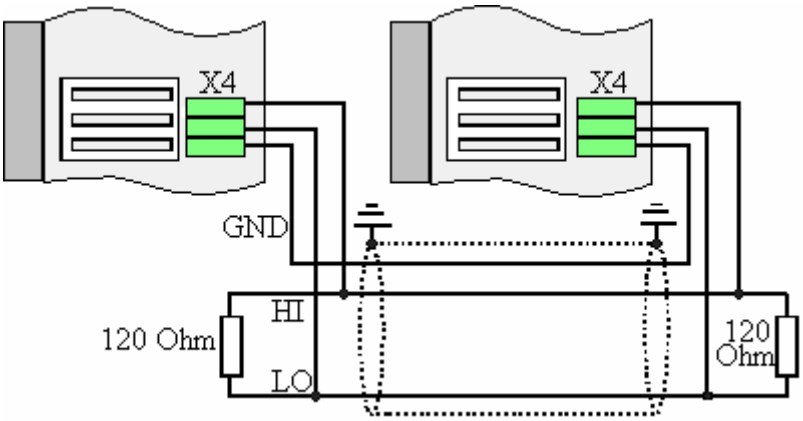
Sytem Bus haberleşme sisteminin kurulmasında en önemli noktalardan biri haberleşme kablolarının bağlantı şeklidir. Haberleşme üç farklı kablo üzerinden sağlanır. GND kablosu aracılığıyla haberleşecek iki devre potansiyel olarak eşitlenir. Sürücünün içerisindeki haberleşme entegresi ile topraklama hattı arasındaki direnç 100 ohm'dur. HI kablosu üzerinden haberleşecek sinyal, LO kablosu üzerinden de haberleşecek sinyalin tersi taşınır. Sinyalin kendisinin ve tersinin gönderilerek, haberleşecek sinyal parazitlerden daha az etkilenmiş olur, daha uzun mesafelere taşınabilir ve alıcı tarafından hata sezinimi daha etkili yapılabilir. Diferansiyel sinyal toprak noktaları farklı olan sistemlerde kullanılmalıdır. Sinyal seviyeleri şekil 2.2'de gösterilmiştir.



Şekil 2.2. Can Bus haberleşme sistemindeki sinyal seviyeleri [7]

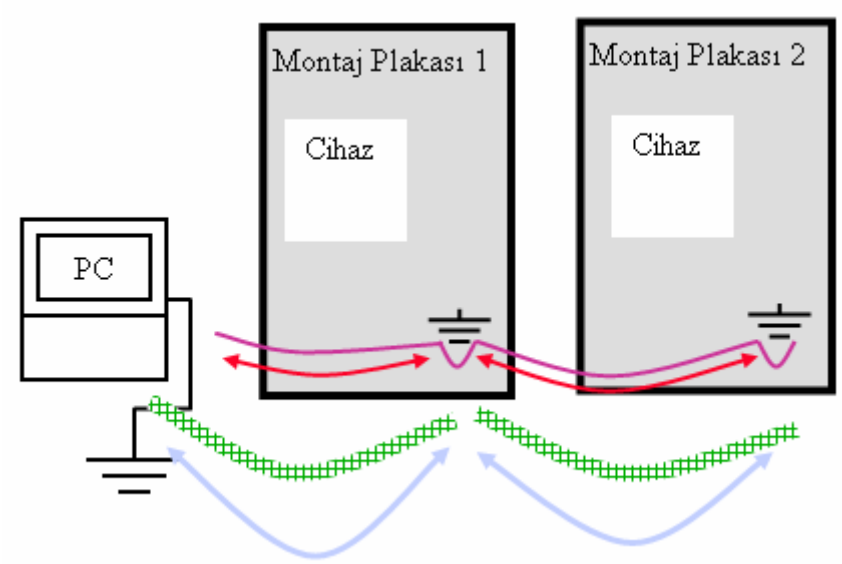
Haberleşme kablosunun başlangıç ve bitiş uçlarındaki HI ve LO uçları arasına 120 ohm değerinde sonlandırma dirençlerini bağlamak gerekir. Bu dirençler kablo üzeride enerji birikmemesini ve haberleşme kabloları üzerinde sürekli aynı seviyede enerji olmasını sağlarlar. Haberleşme kabloları mutlaka burulmuş ve ekranlanmış

olmalıdır. Standartlar haricindeki kablolar kullanıldığında, sağlıklı bir haberleşme sağlanamamakta ve haberleşme sürekli çökmektedir. Haberleşme kablolarının bağlantı şeması şekil 2.3'de gösterilmiştir.



Şekil 2.3. Haberleşme kablolarının bağlantı şeması

Haberleşmenin kullanılmış olduğu sistemlerdeki en büyük sorunlardan biri, hatalı yapılan kablolama sonucunda cihazlarda oluşan potansiyel farklardır. Haberleşme hattındaki farklı cihazlar aynı topraklama hattına farklı noktalardan bağlandığında, cihazlar arasında potansiyel fark oluşabilmektedir. Servo sürücülerin motorları dinamik olarak kontrol etmeleri sonucunda sürücülerin enerji taşıma hattına gönderdiği parazitler, potansiyel farkın oluşmasının temel sebebidir. Oluşan potansiyel farklar cihazlar arasında kompanzasyon akımları oluşturur ve System Bus gibi frekansı yüksek haberleşme sistemlerini olumsuz etkileyebilir, cihazların sayısının artması sonucunda haberleşme sistemerinde sürekli kopmalar meydana getirebilir, kontrol sinyallerinde parazitler oluşturabilir veya PLC'lerin veya haberleşme modüllerinin arızalanmasına sebep olabilir. Potansiyel farklardan oluşan olumsuz sonuçları önlemek için bütün cihazlara bağlanan topraklama hattı bir noktada birleştirilerek cihazların potansiyel farkları eşitlenmelidir. Potansiyel farkı eşitlemek için kullanılan kablolar bağlantı yüzeyine mümkün olduğunca geniş olarak bağlanmalı ve kullanılacak kablo çapları da enerji için kullanılan çaplardan büyük olmalıdır. Şekil 2.4'de, haberleşme sistemi içerisinde kullanılacak cihazlar arasındaki oluşabilecek gerilim farkları kırmızı oklar ile, birbirine bağlanması gereken noktalar mor çizgiler ile, kullanılacak kablonun bağlantı ayakları yeşil renkler ile gösterilmiştir.



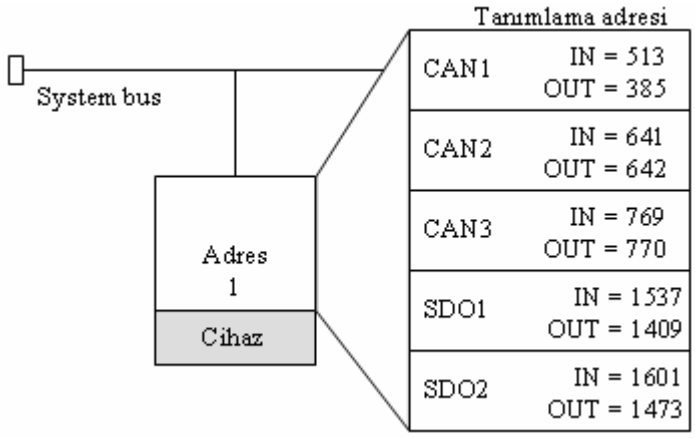
Şekil 2.4. Haberleşme sistemlerindeki oluşabilecek potansiyel farklar

2.4. Cihazların Arasıda Haberleşme Kurulması

Bilgisayar, Can Bus haberleşme sistemine bir arabirim aracılığı ile bağlanılabilir. Bu arabirim bilgisayarın paralel veya Usb portu ile Can Bus protokolü arasında köprü görevi görür. PLC veya sürücü programlama arayüzleri bu arabirimler üzerinden cihazlar ile haberleşirler. Can Bus çeviricilerinin en kullanıcı dostu, bilgisayarın USB portu ile haberleşenidir. Lenze'nin ürettiği cihazlara, Usb arabirim modülü olan 2177IB modülünü kullanılarak, bilgisayar ile haberleşebilir.

Can Bus haberleşme protokolünde SDO ve PDO olmak üzere iki farklı ana haberleşme kanalı vardır. System Bus haberleşme protokolünde, Can Bus içerisindeki SDO haberleşme kanalından iki tanesi kullanılabilir. SDO kanallarının öncelikleri düşüktür ve döngüsel değildir. Belirli parametrelerin değerlerini okumak ve yeni değerler yazmak için kullanılır. Bilgisayarlar ve operatör panelleri, sistemdeki cihazlar ile SDO kanalından iletişim kurarak, sistemin genel haberleşme yoğunluğunu arttırmazlar. Sisteminde bir yoğunlaşma olduğunda, ilk olarak SDO haberleşmesi sonlanır. System Bus haberleşme protokolünde, Can Bus içerisindeki PDO haberleşme kanalından üç tanesi kullanılabilir. PDO kanalları ile döngüsel

zamana bağlı veya verinin değişiminine bağlı haberleşme kurulabilir. Zamana bağlı döngüsel haberleşme yönteminde verinin değişip değişmediğine bakılmadan belirli zaman aralıkları ile cihazlar arasında telegram alışverişi yapılır. PDO kanallarında ilki, en fazla önceliğe sahip olan kanaldır. Haberleşme kanalında bir sorun olduğunda en son PDO1 kanalının bağlantısı kesilir. Genellikle daha yüksek seviyeli bir cihazla haberleşmek için kullanılır. Birinci kanal haberleşmeye, senkronizasyon telegramı ile başlar. Senkronizasyon telegramını alan cihaz haberleşme hattına kendi telegramını yollar ve gelen telegramları değerlendirir. PDO2 kanallı normal bir önceliğe sahiptir. İşlem zamanı 1-2 milisaniye'dir. PDO3 kanalı en az önceliğe sahip olan kanaldır. Sürücüler içerisindeki PDO kanalları CAN1, CAN2 ve CAN3 olarak adlandırılmıştır. Sürücülere gelen sinyaller için IN, sürücüden gönderilecek olan sinyaller içinde OUT, kanalların isimlerinin ardından gelir. Kesme makinesindeki cihazlar CAN2 kanalından birbirleri ile haberleştirilmiştir. CAN2 OUT için tanımlama adresi PLC ve sürücülerde 354/3 numaralı kod içerisinden ayarlanır ve 355/3 kodundan izlenir. CAN2 IN için tanımlama adresi, PLC ve sürücülerde 354/4 numaralı kod içerisinden ayarlanır ve 355/4 kodundan izlenir. Can Bus sistemi içerisindeki kullanılabilecek haberleşme kanalları, örnek tanımlama aresleri ile birlikte Şekil 2.2'de gösterilmiştir.



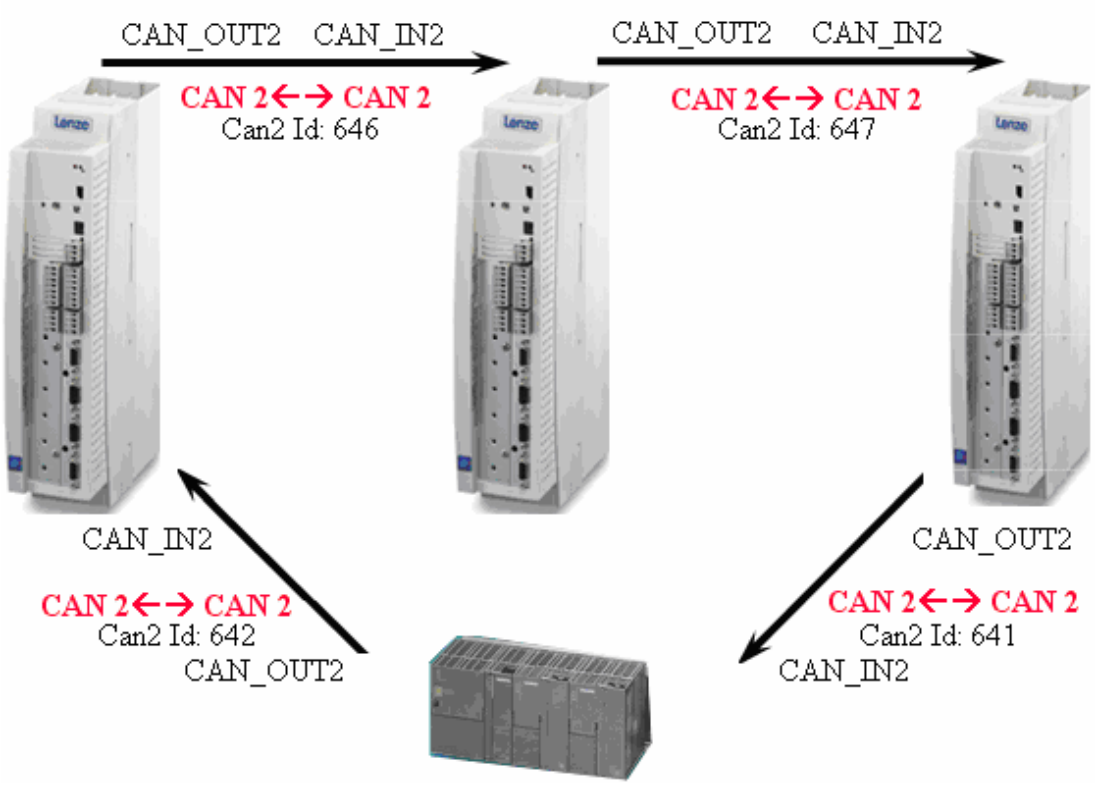
Şekil 2.5. Can Bus haberleşme sisteminin içerisindeki kanallar

Can Bus protokolünde bir cihazdan diğerine veri yollamak için gönderici, alıcının tanımlama adresini telegrama ekler. Hattaki bütün cihazlar bu telegramı okur, telegramdaki tanımlama adresi ile kendi tanımlama adresi aynı olan cihaz, veriyi değerlendirir. Bir telegram birden fazla cihaza gönderilebilir fakat birden fazla

cihazdan bir cihaza gönderilemez. Haberleşme hattında birden fazla master cihaz olabilir. Cihazlar kendileri arasında haberleşme kurabilirler. Mesajın önceliğini tanımlama adresi belirler. Genellikle haberleşme hattındaki en kritik cihazlar en düşük adresleri aldığından, en yüksek tanımlama adresine sahip olan telegram en düşük öneme sahiptir. Eğer aynı hat üzerinden iki telegram aynı anda haberleşmek isterse, yüksek önceliği olan telegram önce gönderilir [2].

Bir cihazın çıkış tanımlama adresi ile diğer cihazın giriş tanımlama adresi eşitlenerek, cihazlar arasında haberleşme kurulur. Kesme makinesinde haberleşecek veri dizisinin uzunluğu bir word'den küçük olduğundan, bütün cihazlar arasında bir PDO kanalı üzerinden haberleşme kurulabilir. PLC üzerinden haberleşme başlatılması için PLC master seçilmelidir. Veriler PLC üzerinde toplanacağı için, haberleşme sisteminin kurulmasına PLC üzerinden başlanmalıdır. PLC'nin CAN2 OUT tanımlama adresi 642'dir ve X ekseninin sürücüsünün CAN2 IN adresi 642 yapılarak, PLC ile X ekseni sürücüsü arasında bir haberleşme kurulur. X ekseninin sürücüsünün CAN2 OUT adres tanımlaması ile Y ekseninin sürücüsünün adres tanımlaması 646 yapılarak, aralarında haberleşme kurulur. Y ekseninin sürücüsünün CAN2 OUT adres tanımlaması ile Z ekseninin sürücüsünün adres tanımlaması 647 yapılarak, aralarında haberleşme kurulur. Z ekseninin sürücüsünün CAN2 OUT adres tanımlaması ile PLC'nin adres tanımlaması 641 yapılarak, aralarında haberleşme kurulur. Böylece PLC'den çıkan veri, tekrar PLC'ye geri dönebilir.

Sistemdeki adreslerin tanımlamaları yapıldıktan sonra enerjinin tamamen kapatılması ve üç dakika beklenilmesi gerekir. Bu süre içinde, ayarlanmamış haberleşme sisteminin bütün hata mesajları haberleşme hattına bağlanmış olan dirençler üzerinde yok olur. Enerji tekrar açıldığında haberleşmedeki master cihaz olan PLC tarafından haberleşme tekrar başlatılır. Şekil 2.5'de kesme makinesinde kullanılan haberleşme yönteminin şeması gösterilmiştir.

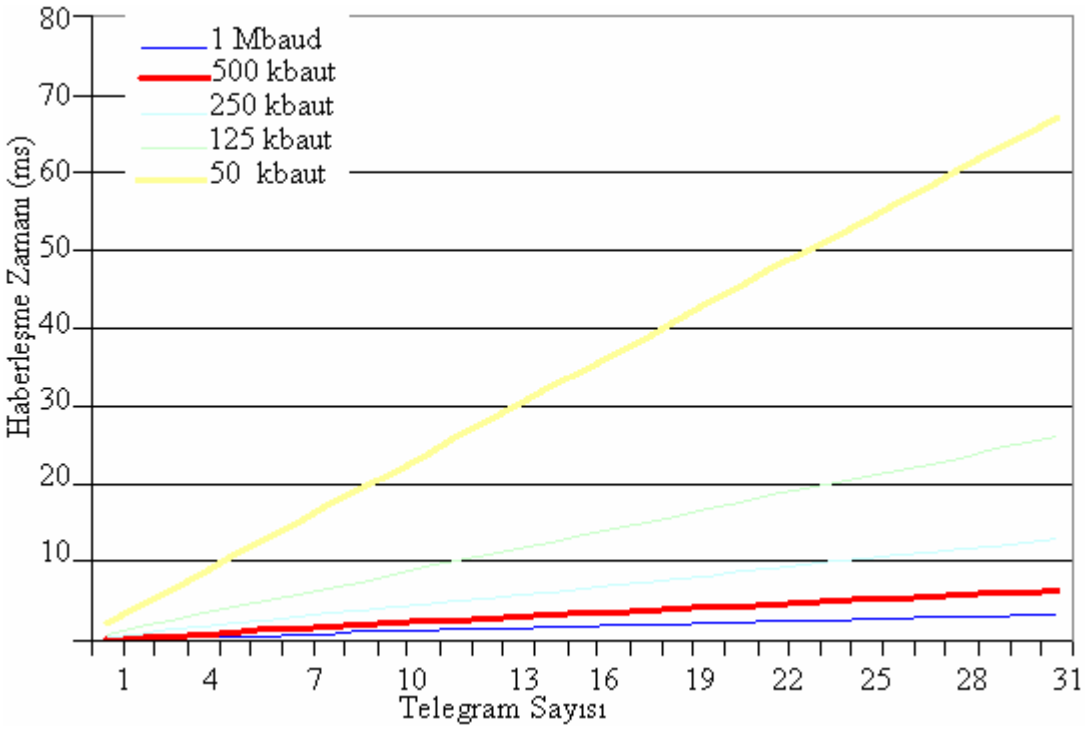


Şekil 2.6. Cihazlar arasındaki haberleşme şeması

Can Bus sistemlerine telefon sinyalleri kullanılarak uzaktan erişilebilir. Bir modem/CAN dönüştürücü modülü ile telefon sinyalleri, Can Bus telegramlarına dönüştürülebilir. Sistemlerin uzaktan izlenmesi ile sistem üreticileri, üretip dünya çapında pazarladıkları sistemlerin arızalarına veya yapılan kullanıcı hatalarına bir noktadan erişebilirler.

Haberleşme sistemindeki cihazların durumu 359 numaralı kodun değeri izlenerek kontrol edilebilir. Haberleşme sistemi sorunsuz çalışıyor ise cihaz üzerindeki 359 numaralı kod sıfır değerini alır. Eğer cihaz çalışma öncesi aşamasıda ise kod'un değeri 1 olacaktır. Cihaza birçok hata telegramlar ulaştığında, kod'un değeri 2 olur. Cihaz hattaki hata telegramları bitene kadar, hatta veri yollamaz. Cihaz haberleşme sisteminden koptuğu anda, kodun değeri 3 olur. Bu durumdan itibaren cihaz haberleşme sistemine tekrar bağlanamaz. Cihazın hattan kopmasının en büyük sebebi, sistemde oluşan elektriksel parazitler, sonlandırma dirençlerinin kullanılmamış olması veya tanımlama adreslerinin hatalı yapılmış olmasıdır. Cihaz üzerinden 358 numaralı kod bir seçilerek veya enerjinin açılıp kapatılması ile

haberleşme sıfırlanır ve sürücü tekrar haberleşmeye başlar. Haberleşme hattının yoğunluğu, 361 numaralı koddan izlenebilir. Haberleşme yoğuluğunu oluşturabilecek en büyük sebepler kablolama hatası veya cihaz sayısına göre haberleşme hızının yüksekliğidir. Şekil 2.6'da, haberleşme hızı ile telegram sayısı arasındaki ilişki, farklı haberleşme hızlarına göre verilmiştir. Kullanılan her telegram 8 byte uzunluğunda kullanıcı verisi taşımaktadır ve Can Bus yoğunluğu %80'dir.



Şekil 2.7. Haberleşme hızı, telegram sayısı, haberleşme zamanı arasındaki ilişki [3]

Haberleşme yoğunluğunu düşürmek için haberleşme sistemi, en optimum şekilde tasarlanmalıdır. Bir PDO kanalının dört word'ünden her biri, farklı bir cihazın referans değeri olacak şekilde, dört ayrı cihaza gönderilerek, bir PDO üzerinden dört cihazın referans değeri ayarlanabilir. Cihazların telegramları haberleşme kanallarında arka arkaya bağlanarak bir PDO üzerinden bütün cihazlara komut gönderilebilir. Anlık değerlerin okunma hızı düşük tutulabilir. Tasarım aşamasında kullanılan bu yöntemleri ile haberleşme yoğunluğu düşürülebilir.

SDO kanalı için kullanılan çerçeve, 11 bit'lik tanımlama adresi ile başlar ve arkasından komut kodu gelir. Birinci SDO kanalı için tanımlama adresi sürücüye veri

göndermek için 1536 sayısının üzerine sürücünün adresi, sürücünün verdiği cevap için 1408 sayısının üzerine sürücünün adresi eklenerek bulunabilir. Komut kodu çerçevenin gönderiliş amacını belirler. Kodun hexadecimal olarak 23 değeri yazma isteğini, 60 değeri yazma isteğinin cevabınını, 40 değeri okuma isteğini, 43 okuma isteğinin cevabını, 80 hata oluştuğunu belirtir. Komut kodunun ardınan, işlem yapılacak kodun hexadecimal değeri, ardından alt kod numarası ve son olarak 4 byte uzunluğunda kullanıcı veririsi gelir. Tanımlama adresi haricinde, telegram 8 byte uzunluğundadır. X ekseni sürücüsünün hızlanma zamanını PLC tarafından değiştirilmelidir. Hızlanma zamanının kod değeri sürücü üzerinde onikidir. PLC, sürücü üzerindeki oniki numaralı kodun değerini 1 saniyeden 20 saniye değiştirmek için bir telegram üretmelidir. Sürününün Can Bus adresi 5 olduğundan, SDO1 için PLC'deki telegram için tanımlama adresi 1541 olmalıdır. bu bir yazma isteği olduğundan komut kodu 23 olmalıdır. Kod numarasındaki değer Can Bus protokolüne göre, istenilen kod değeri 24575 değerinden çıkartılarak, hexadecimal formata çevirilmeli, ardından ilk iki hanesi ile son iki hanesindeki değerler yer değiştirmelidir. 12 numaralı kod için, 24563 değerinin hexadecimal karşılığı 5FF3'tür ve çerçeveye F3 5F yazılmalıdır. Hızlanma rampasının alt kod değeri olmadığından, alt kod değeri için sıfır yazılmalıdır. Kod'un yazılacak değeri 10000 sayısı ile çarpılmalı, çıkan sonuç hexadecimal formata çevirilmeli ve ikişer karakter olarak sondan başa doğru dizilmelidir. Yazılması istenilen 20 değeri, 10000 sayısı ile çarpılıp hexadecimal formata çevirildiğinde, 030D40 sayısı çıkmaktadır, ve telegrama 400D0300 olarak yazılmalıdır. Sürücünün cevabı ise 1413 tanımlama adresi ile başlayacaktır. Komut kodunda, yazma cevabı olduğu için hexadecimal formatta, 60 değeri olmalıdır. Kod numarasındaki değer F3 5F olarak kalmalıdır. Çerçevenin içeriği şekil 2.7'de gösterilmiştir.

PLC'nin X eksenine gönderdiği veri :

Tanımlama Adresi	Komut Kodu	Kod İkinci ikili	Kod İlk ikili	Alt Kod	Veri 1	Veri 2	Veri 3	Veri 4
1537	23	F3	5F	00	40	0D	03	00

X'nin ekseninin PLC'ye gönderdiği veri :

Tanımlama Adresi	Komut Kodu	Kod İkinci ikili	Kod İlk ikili	Alt Kod	Veri 1	Veri 2	Veri 3	Veri 4
1409	60	F3	5F	00	00	00	00	00

Şekil 2.8. Telegram içeriği [3]

Can bus ile üst seviyeli bir sistem arasında haberleşme kurabilmek için, OPC gibi bir program kullanılarak ortak bir haberleşme standartına geçilmelidir. Ortak standartlarda yapılan haberleşme sistemlerinde, üretici firmaların cihazları bir noktada buluşabilmekte, farklı üretici firmaların ürettikleri cihazlar birbirleri arasında haberleşme kurabilmektedir. Kesme makinesinde, sistemdeki cihazlar ile Delphi yazılımının, Can Bus protokolü üzerinden haberleşmesi gerekmektedir. Cihazları haberleştirebilmek için OPC programı ile, Can Bus protokü ortak bir standarta çıkarıtılmıştır. Delphi programı da bu standart üzerinden sistem içerisindeki cihazlar ile haberleşme kurmuştur. Böylece basit bir yoldan, üretici firmanın önemi olmadan, farklı sistemler aynı ortak haberleşme tabanından iletişim kurabilirler. OPC server programı ile, bir noktadan bütün cihazlara erişilerek, cihazların parametrelerin değerleri izlenebilir veya değiştirilebilir. Böylece cihazların üzerindeki parametreler, Delphi yazılımı üzerinden de değiştirilebilir.

BÖLÜM 3. PLC SİSTEMLERİ

3.1. Giriş

Sistemin çalışması için, mekanik sistemin durumu dijital sinyallere çevirilerek bir kontrolöre bildirilmeli ve bu kontrolör de mekanik sistemin durumunu verilen komutları içinde hazırlanmış algoritmaya göre değerlendirip, sistemin konumunu değiştirmek için dijital sinyalleri üretmeli ve bu sinyaller de tahrik elemanları aracılığı ile mekanik sistemin yeni durumunu belirlemelidir. Bu algoritmaya pnomatik bir piston örnek gösterilebilir. Pistonun pozisyonu üzerindeki sensörler ile bir kontrolöre iletilir, sensölerden gelen sinyallere göre kontrolör gelen komutları değerlendirir ve içinde kurulan algoritmaya göre, eğer pistonun pozisyonunun değişmesi gerekiyor ise, pistonun tahrik elemanı olan valfe sinyal verip, hava kanalının açılmasını sağlayarak pistonun pozisyonunu değiştirebilir. Bir mekanizmada eğer şartlar tam olarak elektriksel sinyallere dönüştürülüp bir kontrolöre iletillir, kontrolör içerisinde bu şartlara göre bir algoritma yazılır, kontrolörün çıkışına elektriksel sinyaller ile çalışabilecek tahrik elemanları bağlanırsa mekanizma istenilen şekilde kontrol edilebilir.

Bu çalışma ile araştırılacak olan sistem bir kesim makinesidir. Kesim makinesinde mekanizmanın konumu elektriksel sinyaller kullanılarak kontrolöre iletilmektedir. Mekanizmanın konumu değiştirilmek istendiğinde elektriksel sinyaller aracılığı ile tahrik elemanları kontrol edilmektedir. Konrolör içerisinde yapılacak olan algoritma, bu sinyal trafiğini kontrol ederek mekanizmanın istenildiği gibi çalışması sağlayacaktır. Bu algoritmayı çalıştırabilmek için birçok cihaz kullanılabilir. Bu cihazlardan en çok tercih edileni PLC cihazıdır. PLC'nin tercih edilmesinin en büyük sebebi kullanıcı dostu olmaları ve programlanabilmelerinin çok basit olmasıdır. Günümüzde üretilen PLC'ler ve ara birim yazılımlarını kullanarak sistemlerin kontrolünü PLC ile yapmak önceki yıllara göre oldukça yaygınlaşmıştır. Bu yaygın

kullanımın sonucunda üretici firmalar PLC'lerin teknolojilerini oldukça geliştirmiştir, kullanılabilecek giriş çıkış sayılarını teorik olarak kullanılabilecek en üst sayılara çıkarmış, çevrim zamanlarını çok düşürerek encoder gibi frekansı yüksek sinyalleri bile algılanabilmesini sağlamış, kullanıcıların kolaylıkla anlayabilecekleri PLC programlama arayüzleri geliştirmişlerdir. PLC pazarında üretici firmaların fazla olması rekabeti iyice arttırmıştır. PLC üreticileri, kullanıcılara her geçen gün daha fazla özlellikli PLC cihazları sunmaktadır. Fazla sayıda üretici olması ve her üreticinin kendine özgü PLC programlama arayüzünü kullanıcıya sunması, kullanıcıları zor durumda bırakmaktadır. Genel olarak üretici firmalar tarafından kullanılan algoritmalar aynı tabanda oluşturulabilmektedir ve aynı programlama yöntemlerini desteklemektedir. Örnek olarak bütün PLC'ler ladder yöntemi ile programlanabilirler. Fakat özel olarak kullanılan değişken tanımlama yöntemleri, haberleşme sistemleri, encoder sayıcısı veya alt program gibi fonksiyonların oluşturulabilme özelliklerinin her üretici tarafından farklı yöntemler ile kullanıcıya sunulmaktadır. Üretici firmaların en büyük hedeflerinden bir de kullanıcıya kendi tarzlarındaki PLC donanımlarına ve arayüzlerine alıştırarak, kullanıcının farklı marka ürün kullanmamasını sağlamaktır. Büyük PLC üreticiler bu konuda oldukça başarılı olmuşlardır, kendi ara yazılımlarıda gereksiz yere karmaşıklık yaratacak yöntemler kullanmışlar, bu yöntemleri kendi tarzları olarak kullanıcılara kabul ettirmişlerdir. Bu yöntemi benimseyen kullanıcılara, diğer üreticilerin sunduğu basit arayüzler, karmaşık gelmektedir. PLC yazılımlarında zamanla bir standartlaşmaya gidilmektedir. Büyük PLC üreticileri bu standartlaşmaya yanaşmasada, genel PLC üreticileri bu standartlara yaklaşmaktadır. Bu standartlaşmanın en büyük amacı ortak bir dilde bütün PLC'leri programlayabilmektir. Lenze PLC ve programlama yazılımı, bu ortak standarta en yakın olan firmalardan biri olduğundan, bu proje için tercih edilmiştir.

Son yıllarda endüstriyel uygulamalarda bilgisayar tabanlı teknoloji, PLC'lerden daha hızlı ilerlemiştir. Fabrikalarda içerisinde geniş tabana yayılan sistemlerdeki verilerin izlenmesi, sahadan toplanan bilgilerin veri tabanlarında birleştirilebilmesi, makine üretici firmaların her geçen gün daha fazla fonksiyonu içeren makineleri üretmesi, uzak noktalardan internet gibi bağlantılar kullanarak bilgisayarlara çok rahat erişilebilmesi gibi sebeplerden dolayı bilgisayarın endüstriyel ortamda kullanılması

çok artmıştır. Ayrıca PLC'lerdeki teknolojik kısıtlamalar bilgisayarlara göre çok fazla olduğundan programcının yapabileceği fonksiyonları kısıtlıdır. Bilgisayarlarda teknolojinin son noktasındaki gelişmeler kullanılabilir ve programcılar tarafından teknolojinin izin verdiği ölçüde özgürce programlar yazılabilir. Bilgisayarlarda yazılım yapabilmek için programlama dillerinden birinin kullanılması gerekmektedir. Bilgisayarlarda kullanılan programlama dilleri, PLC'lerde kullanılan programlama dillerine göre karmaşıktır. Bilgisayarlarda programlama kabiliyetinin yüksek olması, programlama dilinin daha karmaşık olması sonucunu doğurmuştur. Buna karşılık PLC'lerde çok basit lojik fonksiyonlar ile programlama yapabilmek mümkündür. Bu sebeple iki sistemin bilgisayarların ve PLC'lerin kendilerine üstün özelliklerini, PLC'lerin lojik fonksiyonları programlamaktaki kolaylığını, bilgisayarların da hafıza ve programlanabilme üstünlüğünü kullanmak en doğru olan yöntemdir. PLC içerisinde çözülebilecek fonksiyonlar, PLC programının içerisinde çözümlenmeli, daha karmaşık fonksiyonlar ise bilgisayar programı tarafından çözülmelidir. Sistemlerin özelliklerine göre avantajları ve dezavantajları tablo 3.1'de gösterilmiştir.

Tablo 3.1. Kontrol sistemlerin avantajlarının karşılaştırılması [8]

Sistemler				
Özellik	Röle Sistemleri	Bilgisayarlar	PLC Sistemleri	Mikroişlemci
Fonksiyon başına düşen maliyet	Çok düşük	Yüksek	Düşük	Düşük
Fiziksel Ölçüleri	Çok büyük	Küçük	Çok Küçük	Küçük
Çalışma Hızları	Yavaş	Çok hızlı	Hızlı	Hızlı
Gürültüye Karşı Direnç	Çok iyi	Kötü	İyi	Kötü
Sistemi tasarlamak için gerekli süre	Fazla	Çok Fazla	Az	Çok Fazla
Sistemi kurmak için gerekli süre	Fazla	Az	Çok az	Fazla
Karmaşık sistemler için	Yetersiz	Yeterli	Yeterli	Yeterli
Sistemde değişim yapılması	Çok zor	Zor	Çok Basit	Çok zor
Bakım ve arızalı parçanın değişimi	Zor	Zor	Kolay	Zor

3.2. PLC'nin Sistem İçerisinde Kullanılması

PLC kesme makinesinde programlanacak, dijital girişler ile mekanizmanın durumu hakkında bilgi alınacak, yazılan program doğrultusunda dijital çıkışlar ile tahrik elemanları kontrol edilecektir. PLC olarak Lenze markasının üretmiş olduğu EPL-

10200 model PLC kullanılacaktır. Bu PLC, fiyat olarak rakiplerine göre avantajlı olduğu, programlamak için kullanılan Drive PLC Developer isimli yazılımının kullanıcı dostu olduğu, üzerinde System Bus entegresinin gömülü bulunduğu için seçilmiştir. PLC ile bir sistem kurmadan önce ilk belirlenmesi gereken nokta sistemde kullanılacak dijital giriş ve çıkış sinyallerinin sayısıdır. Bu belirlendikten sonra kullanılacak PLC'nin giriş çıkış sayısı ihtiyaca göre seçilmelidir. Kullanılan PLC'nin üzerinde 8 adet dijital giriş, dört adet dijital çıkış bulunmaktadır. Kesme makinesinde 18 dijital giriş, 12 dijital çıkış ihtiyacı bulunmaktadır. PLC'nin üzerindeki dijital giriş ve çıkış sayıları yetersiz olduğundan, sisteme ek giriş çıkış modülü ileve edilecektir. Bu modül Can Bus standartlarında haberleşebilen bir modül olmalıdır. Bu modül için Lenze firmasının ürettiği EPM-T110 gateway modülü kullanılacaktır. Modüle iki adet EPM-T210 dijital giriş modülü ile 16 dijital giriş, iki adet EPM-T220 dijital çıkış modülü ile 16 dijital çıkış ilave edilecektir. Giriş ve çıkış modülleri gateway ile haberleşecek, gateway'de PLC ile Can Bus aracılığı ile haberşecektir. Ek modüller ile PLC'nin toplam dijital giriş sayısı 24 ve dijital çıkış sayısı da 20 olmuştur. Bu durumda 6 adet dijital giriş ve 8 adet dijital çıkış fazla olmuştur. Kesme makinesinde kullanılmış olan PLC ve Ek IO modülleri şekil 3.1'de gösterilmiştir. Giriş ve çıkış sayılarında fazlalık sistemde yedek olarak bırakılmalıdır. Yedek bırakmanın en büyük avantajı, ileride sisteme fonksiyon eklenmesi durumunda, PLC'nin donanım özelliklerini değiştirmeden cevap verilebilmesidir.

Şekil 3.1. PLC ile ek IO modüllerinin bir arada kullanılması

Yeterli dijital giriş ve çıkış oluşturulduktan sonra, PLC'nin programlanmasına geçilmelidir. PLC'nin programlanması için, Lenze firmasının Drive PLC Developer Studio programı kullanılacaktır. Program açıldığında ilk olarak programlanacak olan cihazı sorar, buradan kullanılacak PLC donanımı seçilir. Bu projede kullanılan PLC'nin versiyonu 7.0 olduğundan, bu menüden Drive PLC V7.x seçilmelidir. Donanım seçildikten sonra program, kullanıcının hangi dil ile PLC'nin programlanmak istediğini sorar. Bu bölümde FBD seçilerek, onaylanmalıdır. Bu seçim programcının kendi seçimidir. Her PLC yazılımı farklı yöntemler ile yapılabilir. Yöntemlerden en az karmaşığı ve programı takip etmenin en kolay olduğu yöntem fonksiyon blok yöntemidir. Bu yöntemde bütün yazılım fonksiyon blokları ile yapılabilir. Ladder diline göre üstünlüğü, bir satıra istenildiği kadar fonksiyon bloğu eklenebilmesidir. Böylece birçok şart bir satırda toplanabilmektedir böylece sistemde bir arıza olur ise, hangi şartın gerçekleşmediğinden kaynaklandığı kolaylık ile görülebilmektedir. Yazı ile programlanan PLC dilleri, görsellikten uzaktır. Programı takip ekmek için sürekli yazı satıları okunmalıdır. Fonksiyon bloğu yönteminde ise yönteminde ise şartlar kutular içerisinde daha kolay izlenebilir. Fonsiyon bloğu yöntemi, yazı ile programlanan PLC dillerine göre, görsellik açışından üstünlük sağlamaktadır. Kesme makinesini PLC programlama yöntemi olarak üstünlüklerinden dolayı fonsiyon bloğu yöntemi kullanılacaktır.

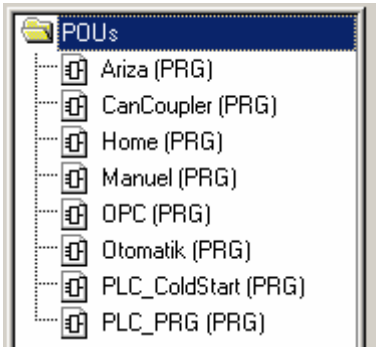
3.3. PLC İçerisinde Kesme Makinesinin Programı

Programda kullanılan her fonksiyon bloğu PLC içerisine tanımlanmalıdır. PLC programında her ifade bir satırda yazılır, yeni ifadeler için yeni satırlar eklenmelidir. İfadelerin fazla olması durumunda satırlar da fazlalaşır. Programda satır sayısının fazlalaşması, sistemin çalışmasını takip etmeyi karmaşıklaştırır, yapılan hatanın bulunmasını zorlaştırır. Yazılımı daha kolaylaştırmak için, sistem çalışma mantığı göz önüne alınarak alt bölümlere ayırılmalı, her bölümnün algoritması kendi içerisinde kurulmalıdır. Örnek olarak bu kesme makinesinde sistem manuel ve otomatik çalışma olarak iki bölüm olmalıdır. Otomatik kesme işlemi sırasında hiçbir manuel fonksiyon çalışmamalı, manuel işlemler sadece sistem üzerinde manuel seçildiğinde çalışmalıdır. Ek dijital girişlerin ve çıkışların haberleşmesi için kullanılan EPM-T110 modülü ile haberleşmek için bir bölüm ayrılmalı, bu bölümde

sadece haberleşme ile ilgili algoritma kurulmalıdır. PLC'nin, OPC ile haberleşmesi için ayrı bir bölüm oluşturulmalı, bu bölümde sadece PLC'deki veriler OPC'de kullanılacak wordlere taşınmalıdır.

Sistemin home işlemini yapması için bir bölüm oluşturulmalı, bu bölümde sadece home işlemi ile ilgili algoritma kurulmalıdır. Arıza şartlarının toplandğı bir bölüm oluşturulmalı, bu bölümde sadece sistemin arıza vermesi ile oluşabilecek durumlar ele alınmalıdır. Bütün bu alt bölümlerin çalışmasını sağlayan ve belirtilen konu başlıklarının içerisine giremeyen fonksiyonların çalıştırıldığı bir ana bölümde oluşturulmalıdır.

Farklı bölümlerin PLC içerisinde oluşturulmasının en basit yöntemi alt programlara ayırma yöntemidir. Bütün bölümler için ayrı bir alt program tanımlanmalıdır. Arıza için "ariza" isimli, ek modüller ile haberleşebilmek için "CanCoupler" isimli, home fonksiyonu için "Home" isimli, manuel işlemleri için "Manuel" isimli, OPC işlemleri için "OPC" isimli, otomatik işlemleri için "Otomatik" isimli, haberleşmenin açılması için "PLC_ColdStart" isimli alt programlar ve bütün alt programların çalışmasını düzenleyen ana program için "PLC_PRG" isimli ana program tanımlanmalıdır. Alt programların tanımlanması şekil 3.2'de gösterilmiştir.



Şekil 3.2. PLC'de kullanılan alt programların gösterilmesi

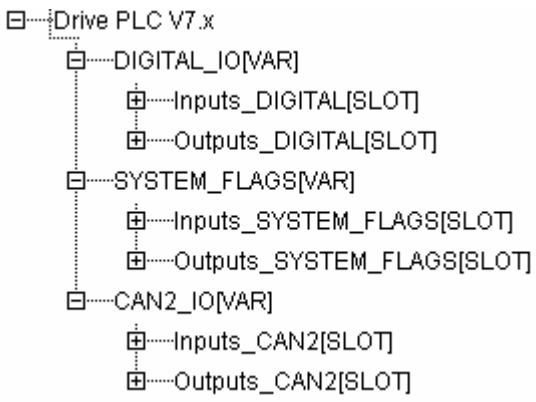
Programlamaya başlamadan önce PLC'nin dijital girişleri ve çıkışları tanımlanmalıdır. PLC donanımın üzerindeki sekiz adet dijital giriş kullanılmıştır. Bu girişler programın "Resources" bölümününün, "PLC configuration" bölümündeki "DIGITAL_IO" içerisine eklenmiştir. Dijital giriş sinyallerinin değişkenlerin

isimleri, sinyalin PLC'ye giriş kaynağı ve sinyalin kullanım amacını kısaca içermelidir. Böylece program içerisinde değişkenler kullanılırken, nereden ve ne amaç için geldiği kolayca görülebilir. Sistemdeki dijital girişlerin listesi tablo 3.2'de gösterilmiştir.

Tablo 3.2. Sistemdeki dijital girişlerin listesi

No	PLC içerisindeki değiken ismi	Açıklama
1	DIGIN_bIn1_Makine_acil_stop	Acil duruş tuşu
2	DIGIN_bIn2_makine_otomatikte	Otomatik, manuel seçimi
3	DIGIN_bIn3_home_yap	Home işlemini başlatma isteği
4	DIGIN_bIn4_kesme_start_butonu	Otomatik kesim başlangıç isteği
5	DIGIN_bIn5_gonye_yukari_butonu	Gönyelerin yukarı kaldırma isteği
6	DIGIN_bIn6_gonye_asagi_butonu	Gönyeleri aşşağıya indirme isteği
7	DIGIN_bIn7_tabla_yukari_butonu	Tablayı yukarı kaldırma isteği
8	DIGIN_bIn8_kesme_asagi_butonu	Kesme elmasını aktifleştirme isteği

"PLC configuration" bölümüne ayrıca PLC'de kullanılacak bütün dış fonksiyonlar eklenmelidir. Bu fonksiyonlardan biri haberleşme'dir. PLC sistemdeki diğer sürücüleri ile haberleşmelidir. Bunun için ikinci Can PDO kanalı en uygunudur. Bu kanalın PLC'de çalışabilmesi için "PLC configuration" bölümüne CAN2 eklenmiştir. Ayrıca PLC programında frekans kaynaklarınında kullanılması gerekir. Bu frekans kaynakları da SYSTEM_FLAGS olarak eklenmelidir. Bölüme ayrıca kullanılacak diğer kaynaklar da eklenebilir. PLC'deki kaynaklar şekil 3.3'de gösterilmiştir.



Şekil 3.3. PLC içerisinde kaynakların oluşturulması

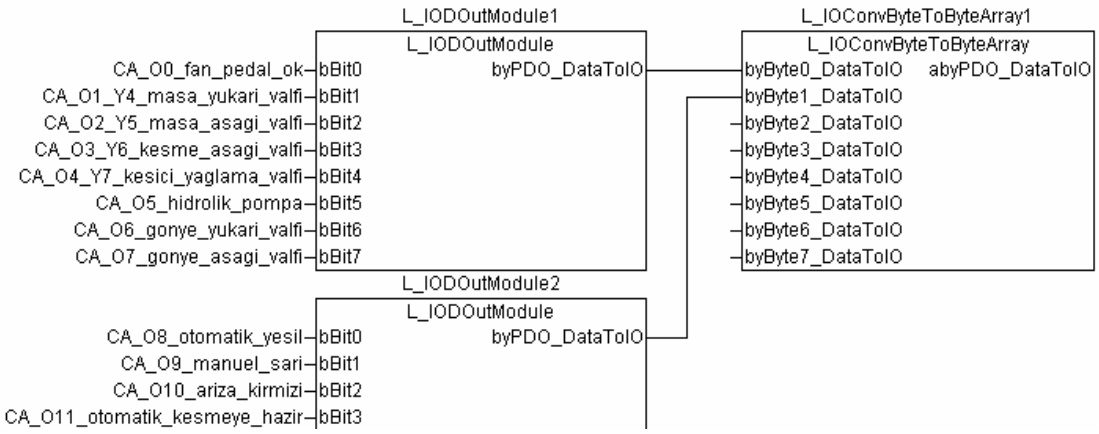
3.3.1. PLC ile IO modülü arasındaki haberleşmenin oluşturulması

PLC ile ek dijital giriş çıkış modülünü haberleştirmek için gerekli olan program, "CanCoupler" alt programının içerisine yazılmalıdır. Giriş çıkış modülleri ile haberleşmek için çok sayıda algoritma oluşturmak gerekir. Bu algoritmalar PLC üretici firmalar tarafından özel fonksiyon blokları içerisinde toplanmıştır. Böylece PLC programcıların haberleşme sistemi içerisindeki cihazlar arasında haberleşmenin kurulması kolaylaşır. Dijital IO sinyalleri için her bir verinin uzunluğu bir bittir. Birçok bit uzunluğundaki sinyalin alış verişi için en kolay yol, bit'lerden bir byte dizisi oluşturmak, ve byte dizisini gönderip almaktır. Böylece daha az değişken sayısı ile, veri alışverişi sağlanmış olur. Ek dijital giriş çıkış modülünde de bit'lerden byte dizileri oluşturulacak ve byte dizileri habeleştirilecektir. Dijital çıkış sinyalleri için iki byte tanımlanacaktır. Bu byte'daki değişkenler CA_O ile başlamalıdır. CA_O'nun anlamı, Can Bus ile göderilecek, _O'nun anlamı ise sinyalin dijital çıkış olduğudur. CA_O'nun ardından çıkış numası ve kısaca sinyalin kullanma açıklaması yer almalıdır. Böylece program içerisinde, sinyallerin anlamları kolaylık ile belirlenebilir. Tablo 3.3'de PLC'den Ek IO modülüne gönderilecek sinyallerin listesi bulunmaktadır.

Tablo 3.3. PLC içerisinde, Ek IO'ya gönderilecek sinyallerin listesi

No	Değiken ismi	Açıklama
1	CA_O0_fan_pedal_ok	Fan ve Pedallara çalışmaları için sinyal üretir
2	CA_O1_Y4_masa_yukari_valfi	Masayı yukarıya kaldıran valfi tetikler
3	CA_O2_Y5_masa_asagi_valfi	Masayı aşağıya kaldıran valfi tetikler
4	CA_O3_Y6_kesme_asagi_valfi	Kesici elmasın valfini tetikler
5	CA_O4_Y7_kesici_yaglama_valfi	Yağlama valfini tetikler
6	CA_O5_hidrolik_pompa	Hidrolik pompayı kontaktörü aracılığı ile çalıştırır
7	CA_O6_gonye_yukari_valfi	Gönyelerin yukarı kalkması için valfi tetikler
8	CA_O7_gonye_asagi_valfi	Gönyelerin aşşağıya inmesi için valfi tetikler
9	CA_O8_otomatik_yesil	Yeşil bir lamba ile makinenin çalıştığı belirtir
10	CA_O9_manuel_sari	Sarı bir lamba ile makinenin manuel konumda olduğunu belirtir
11	CA_O10_ariza_kirmizi	Kırmızı bir lamba ile makinenin arızada olduğunu belirtir
12	CA_O11_otomatik_kesmeye_hazir	Sisteme kesme işlemine hazır olduğunu bildirir

Dijital çıkışlar için hazılanan değişkenler önce "L_IODOutModule" fonksiyon bloklarında, bitlerden toplanarak byte dizilerine çevirilmiştir, ardından bu bytelar sekizlik byte dizilerine "L_IOConvByteToByteArray" fonkisyon bloğu aracılığı ile çevirilmiştir. Bu yöntem ile sekiz adet "IOConvByteToByteArray" fonksiyon bloğu ile 64 adet dijital çıkış tanımlanabilir. Daha fazla çıkış gerektiğinde sisteme, "L_IOConvByteToByteArray" fonksiyon bloğu ilave edilerek dijital çıkışların sayısı attırılabilir. Şekil 3.4'de fonksiyonların birbirleri ile bağlantısı gösterilmiştir.



Şekil 3.4. PLC içerisinde dijital çıkışların word dizisine çevirilmesi

Dijital girişleri algılamak için byte dizileri, byte'lara bölünmeli, byte'lar da bit'lere bölünmelidir. Kullanılması gereken algoritma dijital çıkışlar için kullanılanın tam tersidir. Dijital giriş sinyalleri için iki byte tanımlanmıştır. Bu byte'daki değişkenler CA_I ile başlamalıdır. CA_I'nun anlamı, Can Bus'dan gelecek, _I'nun anlamı ise sinyalin dijital giriş olduğudur. CA_I'nun ardından giriş numası ve kısaca sinyalin kullanma açıklaması yer almalıdır. Böylece program içerisinde, sinyallerin anlamları kolaylık ile belirlenebilir. Masayı yukarıya kalıracak valf için değişkene verilebilecek isim CA_O1_masa_yukari_valfi olmalıdır. Ek modülden gelen sinyallerin listesi tablo 3.4'de gösterilmiştir.

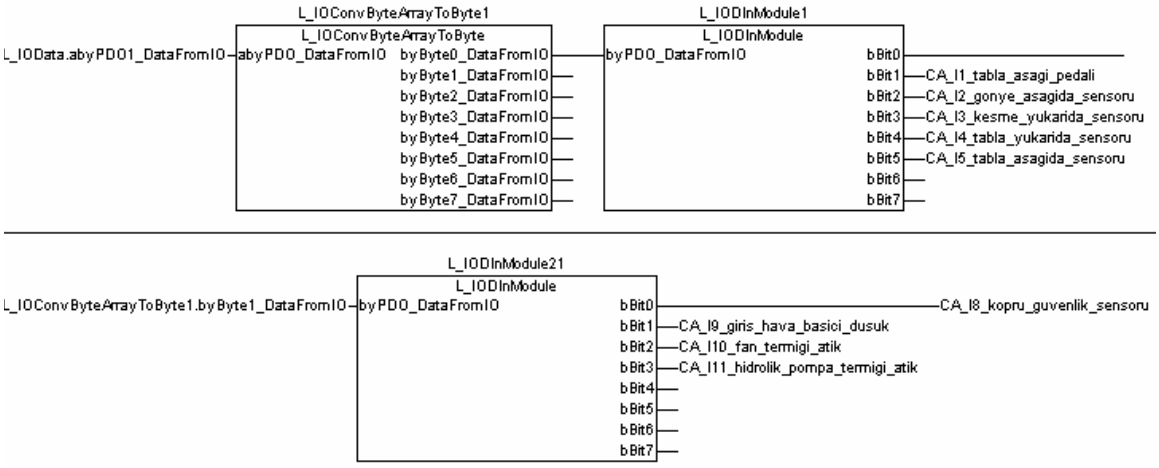
Tablo 3.4. Ek modülden gelen dijital girişler

No	Değiken ismi	Açıklama
1	CA_I0_kesme_asagi_iptal	Kesme elmasının fonksiyonunun iptali isteği
2	CA_I1_tabla_asagi_pedali	Tablayı yatay düzleme indirme isteği

Tablo 3.4. Devam

3	CA_I2_gonye_asagida_sensoru	Gönyelerin aşşağıda olduğunu berliten sinyal
4	CA_I3_kesme_yukarida_sensoru	Kesme elmasının havada olduğunu belirten sinyal
5	CA_I4_tabla_yukarida_sensoru	Tablanın dikey düzlemde olduğunu belirten sinyal
6	CA_I5_tabla_asagida_sensoru	Tablanın yatay düzlemde olduğunu belirten sinyal
7	CA_I8_kopru_guvenlik_sensoru	Köprü güvenlik sensörü uyarı sinyali
8	CA_I9_giris_hava_basici_dusuk	Hava basıncının düşük olduğu uyarısı sinyali
9	CA_I10_fan_termigi_atik	Fan termiğinin atmış olduğu uyarı sinyali
10	CA_I11_hidrolik_pompa_termigi_atik	Pompa termiğinin atmış olduğu uyarı sinyali

Dijital girişler için gelen byte dizisi önce "L_IOConvByteArrayToByte" fonksiyon bloğu ile byte'lara bölünmüştür. Bytelar ise "L_IODInModule fonksiyon" bloğu ile bit'lere bölünmüştür. Bu yöntem ile sekiz adet L_IODInModule fonksiyon bloğu ile 64 adet dijital giriş tanımlanabilir. Daha fazla çıkış gerektiğinde sisteme, L_ L_IOConvByteArrayToByte fonksiyon bloğu ilave edilerek dijital çıkışların sayısı attırılabilir. Haberleşme kanalından gelen word değerinden, dijital girişleri oluşturacan bitlere bölünme algoritması şekil 3.5'de gösterilmiştir.



Şekil 3.5. PLC içerisinde dijital girişlerin word dizilerinden alınması

Dijital girişler ve çıkışlar için tanımlanan byte dizileri haberleşme kanalına gönderilmelidir. "L_IOData15" fonksiyon bloğu gelen ve giden byte dizilerini düzene sokar. Dijital girişler için 8 byte veriyi haberleşme kanalından alır, dijital çıkılar için 8 byte veriyi haberleşme kanalına gönderir. Haberleşmeden gelen ve haberleşme kanalına gönderilecek byte verilerine aracılık eder.

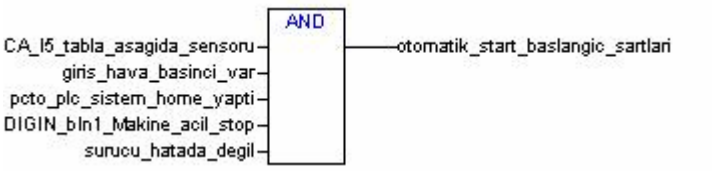
Tablo 3.5. PDO kanalındaki telegramın içeriği

	Byte 0	Byte 1
PDO1-Rx	Giriş Byte 1	Giriş Byte 2
PDO1-Tx	Çıkış Byte 1	Çıkış Byte 2

"L_IOData15" fonksiyon bloğu, "L_IOParPDO15" fonksiyon bloğu ile haberleşir. "L_IOParPDO15" fonksiyon bloğu'da Can Bus ile haberleşmeyi sağlar. Haberleşilecek olan modülün Can Bus adresi, PDO1-Rx ve PDO1-Tx adresleri, "L_IOParPDO15" fonksiyon bloğuna gildikten sonra blok ek modül ile haberleşmeye başlar. Ek modülden gelen byte dizilerini "L_IOData15" fonksiyon bloğuna gönderir. "L_IOData15" fonksiyon bloğundan gelen verileri de ek modüle gönderir. Böylece ek modül ile haberleşme sağlanmış olur. "L_IOParPDO15" fonksiyon bloğunun içerisindeki, PDO telegramının iç yapısı tablo 3.5'de gösterilmiştir.

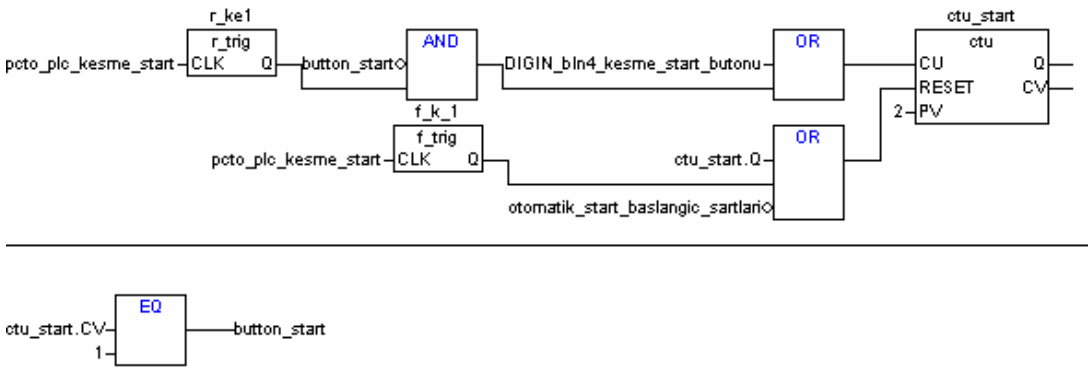
3.3.2. Sistemi otomatik çalıştırabilecek PLC algoritması

Sistemin otomatik olarak çalışabilmesi için birçok şart sağlanması gerekmektedir. Bu şartlardan herhangi biri oluşmadığında sistem çalışmamalıdır. Bu şartlar tablanın yatay düzlemde olması, hava basıncının yeterli olması, sistemin home işlemini gerçekleştirmiş olması, acil duruş'a geçilmemiş olması ve sürücülerin hatada olmamasıdır. Bu şartların hepsi bir AND operatöründe birleştirilmeli ve bir çıkış alınmalıdır. Bu çıkış aktif değil ise sistem otomatik çalışmamalı, sistem çalıştıktan sonra herhangi bir durum değişti ise çalışma derhal durdurulmalıdır. Bu çıkışın değişken ismi otomatik_start_baslangic_sartlari olmalıdır. Başlangıç şartlarını oluşturan algoritmanın yapısı şekil 3.6'da gösterilmiştir.



Şekil 3.6. PLC içerisinde başlangıç şartlarının oluşturulması

Makineyi otomatik olarak çalıştırmak için iki yol vardır. Birincisi bilgisayar içerisindeki SCADA yazılımından gelebilir, ikincisi ise pano üzerindeki sistemi çalıştırma tuşundan dijital olarak gelebilir. Bilgisayardan sistemi çalıştırmak için "pcto_plc_kesme_start" değişkenine, bilgisayar tarafından bir değeri, durdurmak için ise sıfır değeri atanmaktadır. Pano üzerinden çalıştırmak için "DIGIN_bIn4_kesme_start_butonu" değişkeni bir kez tetiklendiğinde sistem çalışmalı, ikici kez tetiklendiğinde sistem durmalıdır.

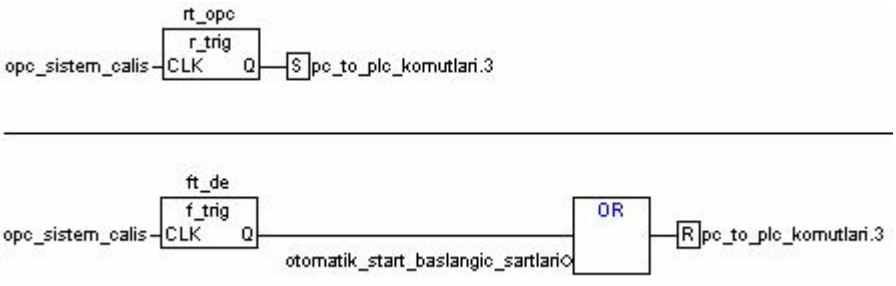


Şekil 3.7. Pano üzerindeki tuş üzerinden çalışma sinyalini oluşturan algoritma

Pano üzerinde çalıştırmak ve durdurmak için sadece bir tuş vardır. Bir tuşla iki işlem yapabilmek için kullanılabilecek metotlardan en kolayı, işlemi sayıcı üzerinden yapmaktır. Sayıcının sayma değeri en fazla iki olmalıdır ve değer iki olduğunda sayıcı kendini resetlemelidir. Sayıcının çıkış değeri karşılaştırıcı operatör kullanılarak bir değeri ile karşılaştırılmalı, oparatör değer bire eşit olduğunda anda çıkış vermelidir. İlk açılışta sayıcının değeri sıfır olduğundan karşılaştırıcı operatör çıkış vermeyecektir. Sayıcı bir sayma sinyali aldığında, çıkış değerini bir yapacak, karşılaştırıcı operatör çıkış verecektir. Sayıcıya bir sayma sinyali daha gönderildiğinde, sayıcının çıkış değeri iki olacak ve kendini resetleyerek, çıkış değerini sıfır yapacaktır. Çıkış değeri sıfır olduğunda karşılaştırıcı oparatör çıkışı kesecektir. Böylece bir tuş ile sistemi çalıştırma ve durdurma işlemleri gerçekleştirilebilir. Bu durumu karmaşıklaştıran bilgisayar üzerinden çalıştırma komutunu geldikten sonra durdurma komutunun pano üzerindeki dijital tuş üzerinden gelmesidir veya pano üzerindeki dijital tuşdan çalıştırma komutu geldikten sonra durdurma komutunun ise bilgisayar üzerinden gelmesidir. Bu durumda sinyallerin çalışmasının hataya uğramaması için en kolay yol, bilgisayardan gelen sinyalin de,

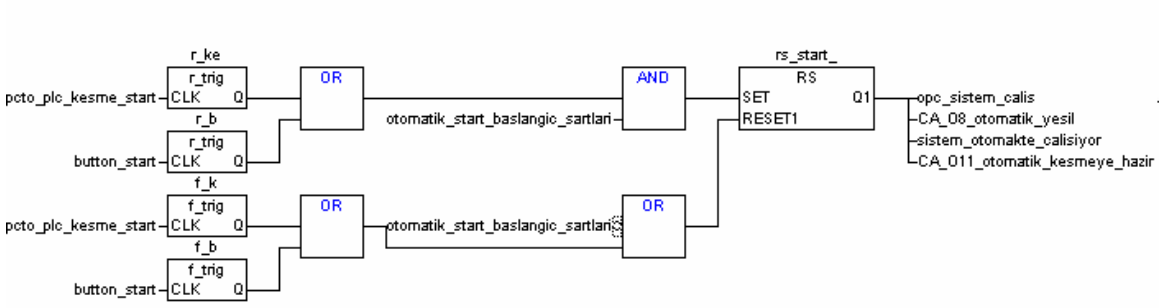
sanki pano üzerinden gelen dijital sinyal gibi, sayıcının değerini arttırmasıdır. Panodan gelen çalıştırma komutu da aynı şekilde, bilgisayardan gelen sinyal gelmiş gibi, PLC içerisindeki bilgisayardan gelen sinyal komutunu aktifleştirmelidir. Sonuç olarak iki farklı sinyal için PLC'de iki farklı algoritma tanımlanmalı, iki sinyal de birbirlerini tetiklemelidir. Karşılaştırıcının çıkışına "button_start" isimli bir değişken tanımlanmalı, bu değişkenin sıfır olması durumunda, yani pano üzerinden çalış komutunun gelmediği durumda, bilgisayadan bir çalış komutu gelir ise, bilgisayardan gelen komut sayıcıyı tetikleyerek "button_start" değişkeninin değerini bir yapmalıdır. Eğer pano üzeriden bir çalış komutu gelir ise, sayıcıyı tetiklemeli ve "button_start" değişkeninin değerini bir yapmalıdır. Eğer button_start değişkeninin değeri bir olduktan sonra, çalış komutu veya bilgisayardan dur komutu gelir ise sistem durdurmalı, yani "button_start" değişkeninin değerini sıfır yapılmalıdır. Bilgisayardan gelen çalış sinyali için "pcto_plc_kesme_start" değişkeninin yükselen kenar tetiklemesi, durdurmak için ise düşen kenar tetiklemesi kullanılmalıdır. Bunun amacı değerin sadece değiştiği anda komut'a dönüşmesi, değer sabit durumda ise, sistemin çalışmasını etkilememesidir. Şekil 3.7'de sayıcı algoritmasının yapısı gösterilmiştir.

Şekil 3.8'de gösterilen algoritma ile sistem pano üzerindeki tuş tarafından çalıştırılır ise, çalışma sinyalinin yükselen kenar tetiklemesi ile bilgisayar üzerinden gelen çalış komutunun değerini bir yapılmalıdır. Eğer sistemin çalışması durdurulur ise çalışma sinyalinin düşen kenarı ile bilgisayar üzerinden gelen çalış komutunun değeri sıfır yapılmalıdır.



Şekil 3.8. Set ve reset devresi

Sistemin çalış sinyalini üretmesi için bir RS fonksiyon bloğu kullanılacaktır. RS fonksiyon bloğunun çalışma algoritması flip-flop devresinin çalışma algoritması ile aynıdır. Bu fonksiyon bloğuna çalışma komutu gönderildiğinde çıkış değerini bir'e eşitleyecek, dur sinyali geldiğinde ise çıkış değerini sıfır'a eşitleyecektir. Çalış sinyali "pcto_plc_kesme_start" veya "button_start" değişkelerinden gelebilir. Bu değişkenlerin değerlerinin sadece değiştiği anda RS fonksiyon bloğunu etkilemelidir. Kenar tetiklemesinin kullanılma amaçlarından biri de, programın geliştirilmesi gerektiğinde şartların ayırılabilmesine kolaylık sağlamaktır. RS fonsiyon bloğunun çıkış değeri bir sayısına eşit olduğunda, opc_sistem_calis, CA_O8_otomatik_yesil, sistem_otomakte_calisiyor ve CA_O11_otomatik_kesmeye_hazir değişkenlerinin değeri bir olacaktır. Biçok çıkış değişkenin kullanılması, tek bir noktada değişlerin izlenmesini kolaylaştırmaktadır. Sistemin çalışmasını sağlayan algoritma şekil 3.9'da gösterilmiştir.

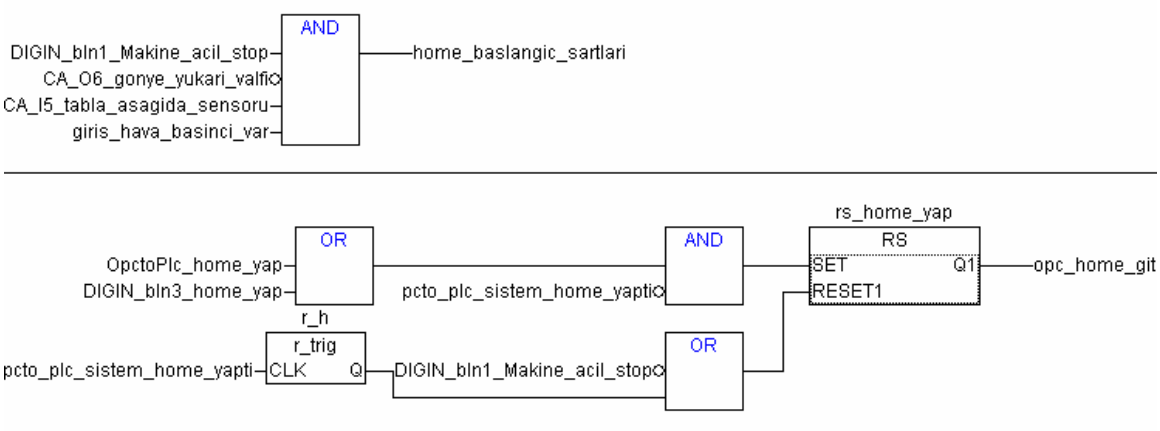


Şekil 3.9. Sistemi çalıştırmasını sağlayan lojik algoritma

3.3.3 PLC'nin home işlemini yapması için gerekli algoritma

Enerji kapatılıp açıldıktan sonra, kontrol kartı bulunduğu noktayı kendi sıfır noktası olarak kabul etmektedir. Bu durumdaki hataları önlemek için gerekli algoritma, "Home" alt programının içerisine yazılmalıdır. Enerji olmadığında bile pozisyonunu koruyabilen motorlar vardır, fakat yüksek maliyetli olduklarından, tercih edilmemektedir. Sistem çalıştırılmadan önce, mekanizma başlangıç noktasına gönderilmelidir. Mekanizma başlangıç noktasına gönderildikten sonra kontrol kartının anlık pozisyon'u sıfırlanmalıdır. Böylece mekanizmanın başlangıç noktası ile, kontrol kartının sıfır noktası eşitlenmiş olur. Sıfırlama işlemi, yani home işlemi başlangıçta yapılmaz ise, yapılan kesimlerin ölçüleri doğru olsa bile, başlangıç

noktası referans aldındığında yanlış olur veya mekanizma mekanik limitlerin dışına çıkabilir. Home işleminin yapılabilmesi için başlangıç şartarı vardır. Bunlar acil duruş tuşuna basılmamış olması, gönyelerin yukarıda olmaması, tablanın aşağıda olması ve hava basıncının yeterli olmasıdır. Bu durumlar için PLC'de tanımlanan değişkenler sırasıyla DIGIN_bIn1_Makine_acil_stop, CA_O6_gonye_yukari_valfi, CA_I5_tabla_asagida_sensoru ve giris_hava_basinci_var'dır. Bu değişkenler bir AND operatörüne bağlanmalıdır. Operatördeki şartlar uygun ise home_baslangic_sartlari değişkeninin değerini bir yapmalıdır. Home işlemi ancak home_baslangic_sartlari değişkeninin değeri bir ise yapılabilmelidir. Home işlemi enerji açıldıktan sonra sadece bir kez yapılmasına izin verilmelidir. Pano üzerindeki acil duruş tuşuna basıldığında, güvenlik için, motor kontrol cihazlarının motora giden enerjileri kesilmektedir, bu sebep ile sistemin ataletinden kaynaklanan kontrolsüz küçük hareketler olabilmektedir. Bu durumda sistemin pozisyonunu kayıp etmesine sebep olabilir. Home işlemi bir kez yapıldıktan sonra ancak acil duruş tuşuna basıldıktan sonra, tekrar işleminin yapılmasına izin verilmelidir. Home isteği iki yol ile OpctoPlc_home_yap değişkeni ile bilgisayardan, DIGIN_bIn3_home_yap değişkeni ile pano üzerindeki home tuşundan PLC'ye gelebilir. Bu iki istekten biri geldiğinde, eğer başlangıç şartları sağlanıyor ise opc_home_git çıkış değişkenin bir yapılarak, PLC tarafından eksen kontrol kartına home yapma emri gönderilir. Acil duruş tuşuna basıldığında veya home işlemi tamamlandığında çıkış değeri sıfır yapılarak sistem yeni home emri için hazı bekler. Eksen kontrol kartı home işlemini tamamladığını pcto_plc_sistem_home_yapti değişkeninin değerini bir yaparak PLC'ye iletir. Bu değişkenin değeri bir olduğu sürece tekrar home yapma komutları engellenmelidir. Bu işlemler için bir RS fonksiyon bloğu kullanılmalı ve çıkışına opc_home_git değişkeni bağlanmalıdır. Bu değişkenin değeri bir olduğunda, bilgisayara home işlemi başlatması emri gönderilmelidir. Mekanizma başlangıç noktasına gönderilmeli, ve daha sonra pcto_plc_sistem_home_yapti değişkeninin değerini bir yaparak PLC'ye bildirmelidir. Acil duruş tuşuna basıldığında ise, bu değişkenin değeri sıfır yapılmalıdır. Home sistemini oluşturan lojik algoritma, şekil 3.10'da gösterilmiştir.



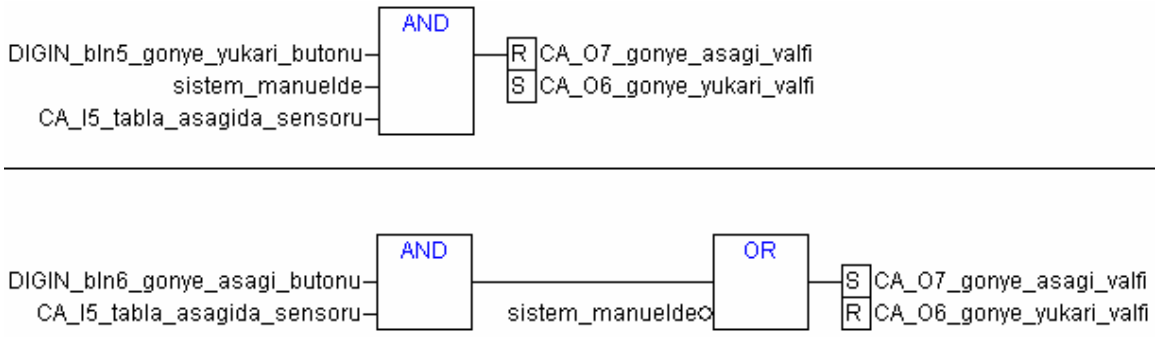
Şekil 3.10. Home sisteminin lojik algoritması

3.3.4. Sistemin manuel çalıştırabilecek PLC algoritması

Otomatik çalışma sisteminde bazı hareketler, sistem tarafında kontrol edilir ve kullanıcının müdahalesi engellenir. Sistemin bütün hareketleri kullanıcı tarafından belirlenebileceği bir manuel çalışma sistemi oluşturulmalıdır. Kullanıcının belirlediği hareketler, sistemin otomatik çalışma algoritmasına uymayabileceğinden, çalışma şartları da farklı olabilir. Manuel işemler için gerekli algoritma "Manuel" alt programının içerisine yazılmalıdır. Önce sistemin manuelde olduğu anlaşılmalıdır. Bunun için sistem_manuelde adında bir değişken tanımlanmalı, değerine de panodan gelen dijital otomatik seçim sinyali bağlanmalıdır. Programda DIGIN_bIn2_makine_otomatikte değişkeninin tersini kullanmaktansa, tersine sistem_manuelde değişkeni atamak ve programda onu kullanmak, sistemi izlemeyi kolaylaştırır. Ayrıca sistem otomatik olarak çalışıyor ise, işlemleri bitirdikten sonra manuel konuma geçmelidir. Sistemin manuel'de olduğunu gösteren algoritma, şekil 3.11'de gösterilmiştir.

Şekil 3.11. Sistemin manuelde olduğunun bildiren algoritma

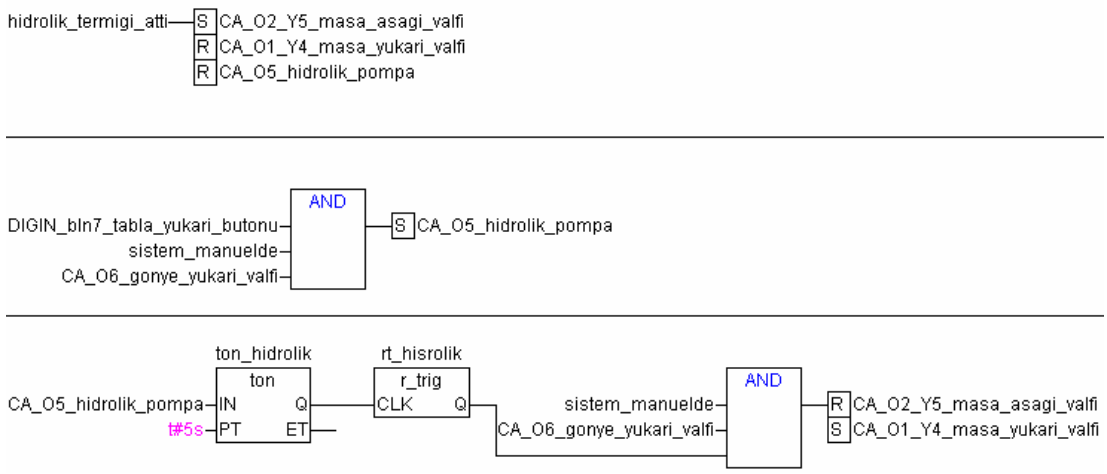
PLC ile pnomatik pistonların kontrolü için en uygun yöntem set ve reset yöntemleridir. Sistemde, masaya koyulan camı X ve Y yönlerinde hizalamak için gönyeler vardır. Bu gönyeler pnomatik pistonlara bağlıdır. Pistonu yukarı ve aşağı hareket ettirmek için iki adet bobin kullanılmaktadır. Hareket ettirilmek istenilen yöndeki bobin enerjilenerek, pistona hava girişi sağlanır ve pistonda gelen hava ile hareket eder. Piston, yukarı yönde hareket ettirilmek istendiğinde aşağı yöndeki piston, aşağı yönde hareket ettirilirilmek istendiğinde yukarı yöndeki pistona hareket veren sinyale reset yapılmalıdır. İki sinyal aynı anda set yapılır ise valf hiçbir yöne hareket edemez. Sistem manuel çalışma modunan çıktığında gönyeler mutlaka aşağıya inmiş olmalıdır. Gönye fonksiyonu sadece sistem manuel'de iken çalışmalıdır. Gönye sistemini çalıştıracak, şartlara bağlı set ve reset devresi için gerekli olan algoritma şekil 3.12'de gösterilmiştir.



Şekil 3.12. Gönye sisteminin çalıştıran algoritma

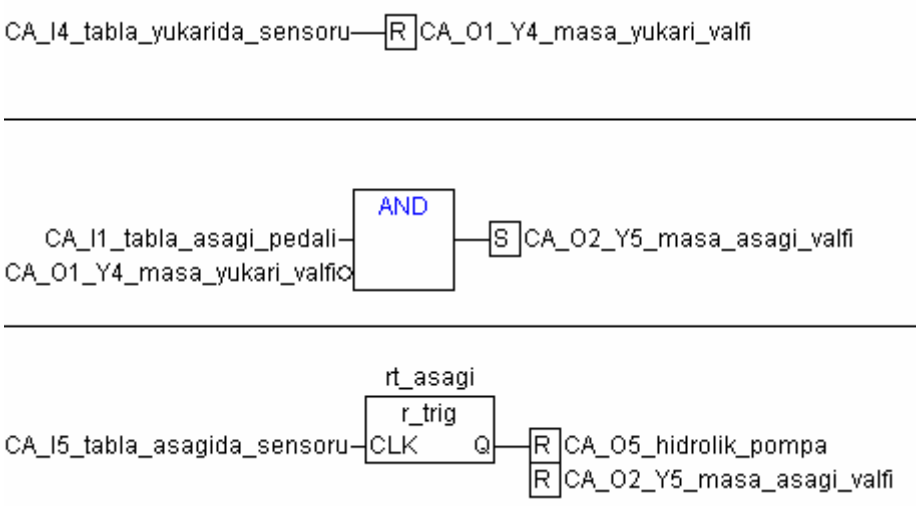
PLC sistemlerinde bir hareketin arkasından diğer bir hareket komutu verilebilir. Bu iki sisitem arasına belirli bir zaman da koyulabilir. Bu konudaki en iyi örnek masanın yukarı kalkması fonksiyonudur. Masayı yukarı kaldıran valfi tetikleyen bobine enerji verilmeden önce, hidrolik motorunun beş saniye boyunca çalışması gerekmektedir. Masa yukarı kalkması için başlangıç şartları vardır. Sistem manuel konumunda olmalı ve masa yukarı kalktığında koyulan camın düşmemesi için gönyeler mutlaka yukarıda olmalıdır. Başlangıç şartları oluştuğunda, masa yukarı tuşuna basıldığında, hidrolik pompayı çalıştıran CA_O5_hidrolik_pompa değişkeninin değerine bir atanmalı ve dijital çıkış ile hidrolik motorununu çalıştıran kontaktörün bobinine enerji gönderilmelidir. Kontaktör kapalı konuma geçince hidrolik motoruna besleme enerjisi verilmelidir. Hidrolik pompa beş saniye çalıştıktan sonra, gönyelerde

yukarıda ve sistem manuelde ise, masayı yukarı kaldıracak valfin bobinine enerji verilmelidir. Bobine enerji verebilmek için, CA_O1_Y4_masa_yukari_valfi değişkeninin değeri bir'e set edilmeli, CA_O2_Y5_masa_asagi_valfi değişkeninin değeri sıfır'a resetlenmelidir. Bu şartlar oluşurken hidrolik pompa motorunun terimiği atar ise, hidrolik pompa çıkışı ve masayı kaldıran valfin çıkışı kesilmeli, masayı aşşağıya indiren valfe sürekli enerji verilmelidir. Sistemin çalışma algoritması şekil 3.13'de gösterilmiştir.



Şekil 3.13. Hidrolik pompo ile masa hareket valfi arasındaki ilişki

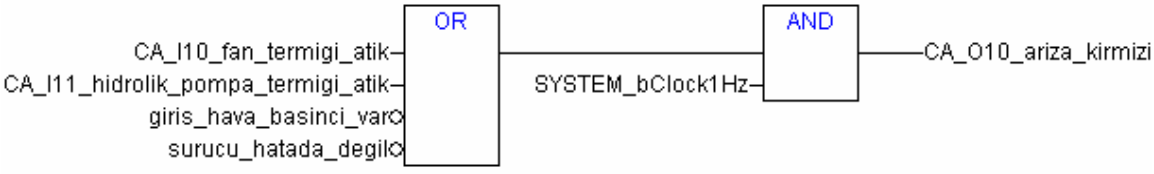
Tabla yukarı kalktıktan sonra, masa yukarı valfi görevinin tamamlamıştır, bobinine giden enerjinin kesilmesi gerekir. Masanın yukarı kalktığını PLC'ye gelen bir dijital giriş sinyali ile anlaşılabilir. CA_I4_tabla_yukarida_sensoru değişkenin değeri bir'e eşit olduğunda, tabla yukarı kalkmış demektir ve bu değişken CA_O1_Y4_masa_yukari_valfi değişkeninin değerini sıfıra resetleyerek, masanın yukarı kalkmasını sağlayan valfin enerjisini kesmelidir. Tabla aşağı pedalına basıldığında ve masayı yukarı kaldıran valfte enerji yok ise, masa aşağı valfinin bobinine enerji, CA_O2_Y5_masa_asagi_valfi değişkeninin değeri bire set edilerek verilmelidir. Tabla aşağıya indiğinde, yani CA_I5_tabla_asagida_sensoru değişkeninin değeri bir olduğunda, hidrolik pompanın çalışması durdurulmalıdır. Masayı aşağıya indiren valfin bobinine giden enerjide, CA_O2_Y5_masa_asagi_valfi değişkeninin değeri sıfıra set edilerek kesilmelidir. Sistemin çalışması için gerekli algoritma şekil 3.14'de gösterilmiştir.



Şekil 3.14. Hidrolik pompa ve masa aşağı valfinin resetleyen algoritma

3.3.5. Arıza kontrolü yapılabilmesi için gerekli PLC algoritması

PLC'de kontrol edilecek sistemin arızaları için bir alt program oluşturulmalıdır, bu alt programa "Ariza ismi" verilmelidir. Sistem'de bir arıza oluşunca, kırmızı bir lambaya enerji verilerek, kullanıcılar ikaz edilmelidir. Bu ikaz'ın kullanıcılar tarafından daha iyi algılabilmesi için yarım saniye uyarı vermeli, yarım saniye uyarı vermemesi gerekir. Bir saniye periyot'lu sinyal PLC'nin içirisinde, zaman saatleri bölünümde bulunmaktadır. SYSTEM_bClock1Hz değişkeni PLC'nin bir iç değişkenidir ve bir saniyelik frekansa sahiptir. Fan termiği veya hidrolik motor termiği attıklarında veya hava basıncı düştüğünde veya sürücüler hataya geçtiğinde, CA_O10_ariza_kirmizi değişkeninin değeri, bir saniyelik frekans ile bir'e eşitlenmesi ile, dijital bir çıkış üretilmiş olur ve bu çıkışta kırmızı arıza lambasına enerji verilerek kullanıcı uyarılmalıdır. Arıza çıkış sinyalinin oluşturabilecek algoritma şekil 3.15'de gösterilmiştir.

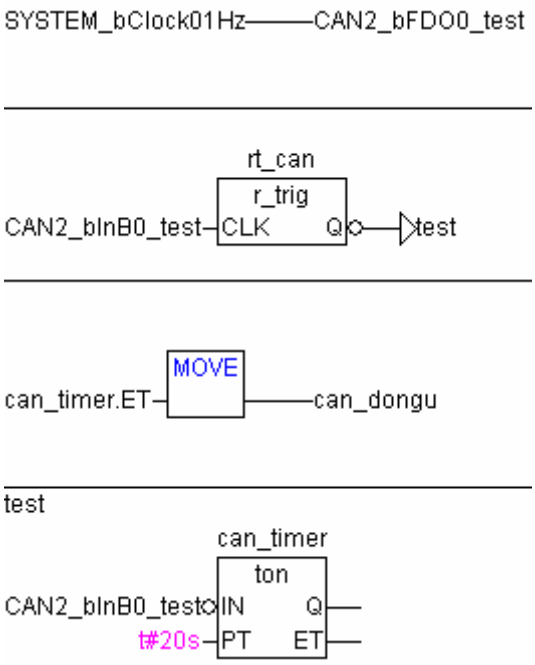


Şekil 3.15. Arıza sinyalini oluşturan algoritma

3.3.6. Sistemdeki haberleşme hızının ölçülmesi

Sistemdeki cihazları birbirleri arasında Can Bus protokolü ile haberleşmektedir. Bu haberleşmede oluşan zaman kayıplarını ve hataları anlamak için, sürekli hattı kontrol etmek gerekir. Kontrol hızı sistemin haberleşmesini yavaşlatmayacak düzeyde olmalıdır. Beş saniye boyunca bir sinyali ve beş saniye boyunca sıfır sinyali, haberleşecek cihazlar arasında arka arkaya gezdirilmeli, ve en son PLC'ye geri dönmelidir. Sinyalin üretilip hatta verildikten sonra, bir sayıcı çalıştırılmaya başlanmalı ve sinyal tekrar PLC'ye geri döndüğünde sayıcını zaman değeri kayıt edilmeli, kayıt işelemi bittikten sonra zaman sayıcısının sayma değeri yeniden başlatılmalıdır. PLC içindeki SYSTEM_bClock01Hz değişkeni beş saniye bir, beş saniye sıfır sinyali üretmektedir. Bu değişken CAN2_bFDO0_test değişkenine atanarak, sinyal ikinci PDO kanalının birinci bit'i üzerinden haberleşme kanalına iletilir. Haberleşme kanalı üzerinden CAN2_bInB0_test değişkeni ile, sinyal tekrar PLC'ye geri döner. Bu dijital sinyali PLC'nin göndermesi ve alması arasında geçen süre hesaplanmalıdır. Bu sürede, bir sayıcı'ya yüksek frekanslı sinyal saydırılarak geçen zaman hesaplanabilir.

Yüksek frekanslı bir sinyali saymak, PLC için önemli bir işlem yükü getirir. Bunun yerine aynı zamanın hesabı bir Ton fonksiyon bloğu ile de yapılabilir. Ton fonksiyon bloğu, PLC içerisinde dijital sinyalin bir değerini diğer bir değişkene belirli bir süre geciktirdikten sonra atamak için kullanılır. Değişkenin değerine sıfır atamak için ise herhangi bir geciktirme uygulamaz. Ton fonksiyon bloğunun içerisinde sinyali geciktiren bir zaman saati vardır. Bu zaman saati için PLC kartı üzerinde entegreler tanımlıdır ve PLC için basit bir fonksiyondur. Bu zaman saatinin kullanılması, PLC'ye küçük bir işlem yükü getirir. Ton fonksiyon bloğunun geciktirme süresi, algılanacak sinyalin ortalaması 5 saniye olduğu için, 5 saniyeden fazla olmalı, en uygunu ise haberleşme sistemindeki anlık kopmaların algılanmaması için de 20 saniye olmalıdır.



Şekil 3.16. Haberleşme hızını ölçen algoritma

PLC, fonksiyonları satırları arka arkaya işleyerek çalışır. Hangi satır aktif ise, sadece o satırdaki fonksiyon işlemcide işlenir. Atlama yani jump fonksiyonu ile, satırların ardışık işlenmesi bozulabilir, istenen satırdaki işlemler yapılmadan, geçilebilir. Zaman saatinin değeri, sadece haberleşme kanalından gelen sinyal tekrar PLC'ye döndüğü anda yani CAN2_bInB0_test değişkeninin değeri bir olduğu anda, bir başka değişkene atanmalıdır. İşlemin sadece PLC'de bir döngü süresi boyunca olması için, sinyalin yükselen kenar tetiklemesi alınmalıdır. CAN2_bInB0_test değişkeninin yükselen kenar tetiklemesinde, zamanlayıcının değeri kayıt edilmelidir. İşlemin gerçekleşebilmesi için sinyalin yükselen kenar tetiklemesinde, can_timer.ET değişkeni içerisindeki ton fonksiyon bloğunun zaman saati değişkenini, can_dongu değişkenine atanmalıdır. PLC, CAN2_bInB0_test değişkeninin yükselen kenarı gelmedikçe, test satırına geçerek atama işlemi olan satırı atlamalıdır, yükselen kenar tetiklemesi geldiğinde ise atama işlemini yapmalıdır. Sinyal gecikmesini gösteren algoritma şekil 3.16'de gösterilmiştir.

3.3.7. PLC'nin OPC sisteminde çalışması

Plc'de "OPC" isimi ile bir alt program tanımlanmalı ve bu alt programın içirisinde OPC ile PLC'den endüstriyel bilgisayara gönderilecek olan veriler yer almalıdır. Bilgisayara gönderilecek verilerin her birinin uzunluğu bir bit'tir. Toplam veri alışverişi için 42 tane bit tanımlacaktır. Bu bitlerin OPC için her birine bir değişken tanımlanması durumunda 42 adet değişkenin tanımlanması gerekmektedir. Bunun yerine bit'ler word gruplarında toplanmalıdır. Bir word 16 bit'ten oluşmaktadır. Her bit, word dizisinin bir elemanı olacak şekilde tanımlanacaktır. Word dizilerinin oluşturulma sırasına göre bilgisayar üzeride de word'ler tekrar bit dizilerine ayrılacaktır. PLC'de 18 adet dijital giriş bulunmaktadır. 18 bit, iki word üzerinde toplanabilir. İlk 14 dijital giriş için, plc_dijital_cikislar_0_15 isimli bir word değişkeni tanımlanmıştır. 4 adet dijital giriş için de plc_dijital_girisler_16_32 isimli bir word değişkeni tanımlanmıştır. PLC'de 12 adet dijital çıkış bulunmaktadır, bütün dijital çıkışları tanımlamak için bir word değişkeni tanımlanmalıdır ve bu değişkene plc_dijital_cikislar_0_15 ismi verilmelidir. Bit'ler word'lerin elemanları olacak şekilde atanmıştır.

PLC'de oluşturulan her word değişkenlerine ayrı bir kod atanmalıdır. OPC bu kodları okuyarak ve yazarak haberleşme kuracaktır. PLC programının "Resources" menüsünün içerisindeki "Instance parameter manager" bölümü içinde kodlar tanımlanmıştır. PLC içinde haberleşme için birçok kod mevcuttur, bu kodlardan 3000 ile 5000 arasındaki kodlar kullanıcıya ayırılmıştır. Kodlar, 3000 sayısından başlanarak arka arkaya tanımlanmalıdır. Tanımlanan kodları OPC programı, Delphi içerisinde yazılmış olan SCADA yazılımına aktaracaktır. Kesme makinesinde OPC tarafından kullanılacak olan kodların PLC üzerinde oluşturulduğundaki görüntü şekil 3.16'da gösterilmiştir.

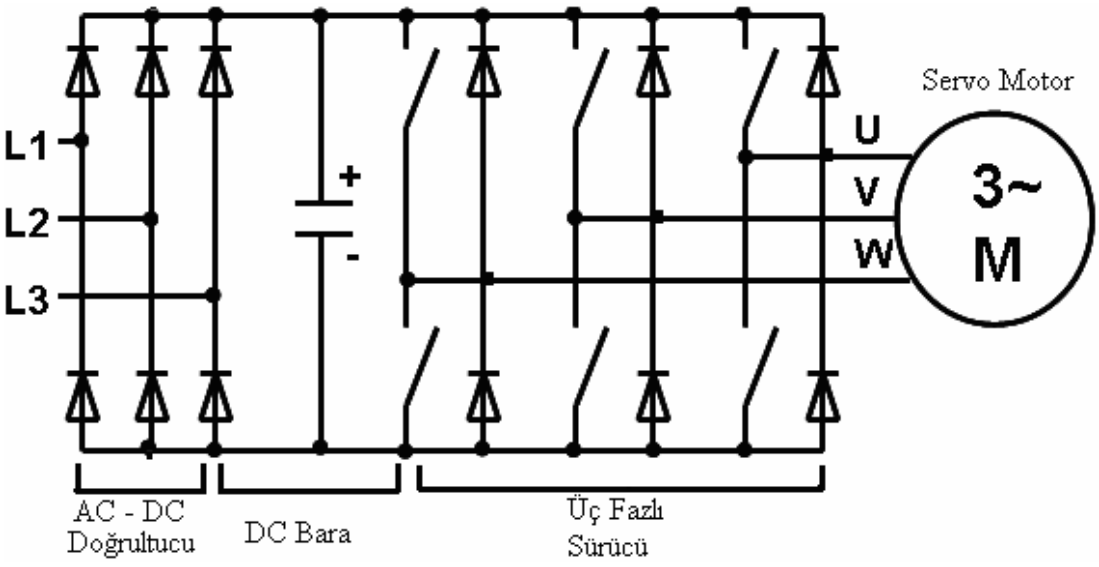
Code	Program unit	Variable	Parameter text	Var.T...
C3000	Global Variables	plc_dijital_girisler_0_15	PLC_in_0_15	WORD
C3001	Global Variables	plc_dijital_girisler_16_32	PLC_in_16_32	WORD
C3002	Global Variables	plc_dijital_cikislar_0_15	PLC_out_0_15	WORD
C3003	Global Variables	plc_to_panel_0_16	PLC_to_Panel	WORD
C3004	Global Variables	pc_to_plc_komutlari	Pc_to_PLC	WORD

Şekil 3.17. PLC içerisindeki değişkenlere kod verilmesi

BÖLÜM 4. SERVO MOTOR SÜRÜCÜLERİ

4.1. Giriş

Sistemde mekanizmayı tahrik edilecek üç adet X,Y ve Z eksenleri bulunmaktadır. Bu eksenler için üç adet motor kullanılmıştır. Bu motorlar hassas ve dinamik hareketleri gerçekleştirebilmeleri için senkron servo olmalıdır. Senkron servo bir motoru kontrol etmek için de servo sürücüler kullanılmalıdır. Servo sürücülerin en önemli özellikleri motorları çok dinamik çalıştırabilmeleri, hatasız hız ve açı kontrolü yapabilmeleridir. Servo sürücüler motorları kontrol ederken PLC veya diğer cihazlarla da haberleşebilirler. Servo sürücüleri programlayabilmek için en önemli nokta programlamayı sağlayan arabirimlerdir. Servo sürücüler motorları kontrol etmek için programlanmaları gerekir. Servo sürücüler şebekeden gelen üç fazı doğrultarak, içerisinde DC barayı oluşturur. Oluşturduğu DC bara üzerinden de motora gönderdiği enerjiyi IGBT'ler üzerinde tetikleyerek motorun istenildiği gibi tahrik edilmesi sağlar. Sürücünün şeması şekil 4.1'de gösterilmiştir [14].



Şekil 4.1. Servo sürücünün şeması [16]

Sistemler doğru analiz edilip mekanizmaya uygun güçlerde motor ve sürücü seçilmelidir. Güç seçimi mekanik tasarımcılar tarafından yapılmalıdır. Motorlar mekanizmaya bağlandıktan sonra, sürücüler programlanmalı ve motorlara istenilen komutlara uygun şekilde enerji gönderecek şekilde ayarlanmalıdır. Sistemin kontrolü tamamen programcının elindedir. Programcı kullanılan ara yazılımın ve motorların özelliklerini ne kadar iyi kullanabilir ise sistem okadar kusursuz çalışır. Sistemin kontrolünde oluşan hatalar kullanılan cihazların fonksiyonlarının eksikliğinden olabileceği gibi, programcının eksik bilgisinden de kaynaklanabilir. Sürücüler sahada kullanılmadan önce programcı tarafından deney ortamlarında mutlaka test edilmeli, fonksiyon kabiliyetleri anlaşılmalıdır. Sistemlerdeki fonsiyon ihtiyaçları çok farklıdır. Bütün sistemlerin ihtiyacını karşılayacak bir motor kontrol cihazı üretilebilir. Bu cihaz birçok fonksiyonu içerdiği için programlanması çok karşık olacaktır. Ayrıca bütün fonksiyonları içerdiği için fiyatı da yüksek olacaktır. Bunun yerine sürücü üretici firmalar, belirli sistemler için özel sürücü kartları geliştirmişlerdir. Bu sürücülerin yazılımları sadece bir çözüme yönelik olarak geliştirilmiştir. Sisteme özel yazılımlarda, sistem için gerekli fonksiyonlar daha önceden üretici firma tarafından geliştirilmiştir ve programcı sadece bu fonksiyonları kullanabilir. Programcı, bütün algoritmayı oluştumak yerine, üretici tarafından hazırlanmış fonksiyonlar arasıdan bağlantı kurarak sistemi daha hızlı ve kolay bir şekilde çalıştırır. Sisteme özel sürücüler programla açısından kolaylık sağlarken, kullanılacak sürücünün seçimini zorlaştırmıştır. Sistemlerin fazlalığı, sürücü sayısının da fazlalığı sonucunu oluşturmuştur. Piyasada birçok marka ve model servo sürücü bulabilmek mümkündür. Aynı markanın bile birçok model servo sürücüsü bulunabilmektedir. Doğru sürücü, mekanizmada istenilen fonksiyonları gerçekleştirebilecek en ucuz sürücüdür. Sürücü seçiminde ilk olarak sistemin ihtiyaçları belirlenmelidir. Kesim makinesinde sürücülerin temel ihtiyaçları aşağıdaki gibidir;

i)-Kullanılacak sürücüler Can Bus protokolü ile haberleşebilmelidirler. Bu haberleşme protokolünü aracılığı ile PLC ve OPC arasında haberleşme kurularak sistemin izlemesi ve kontrolü yapılacaktır.

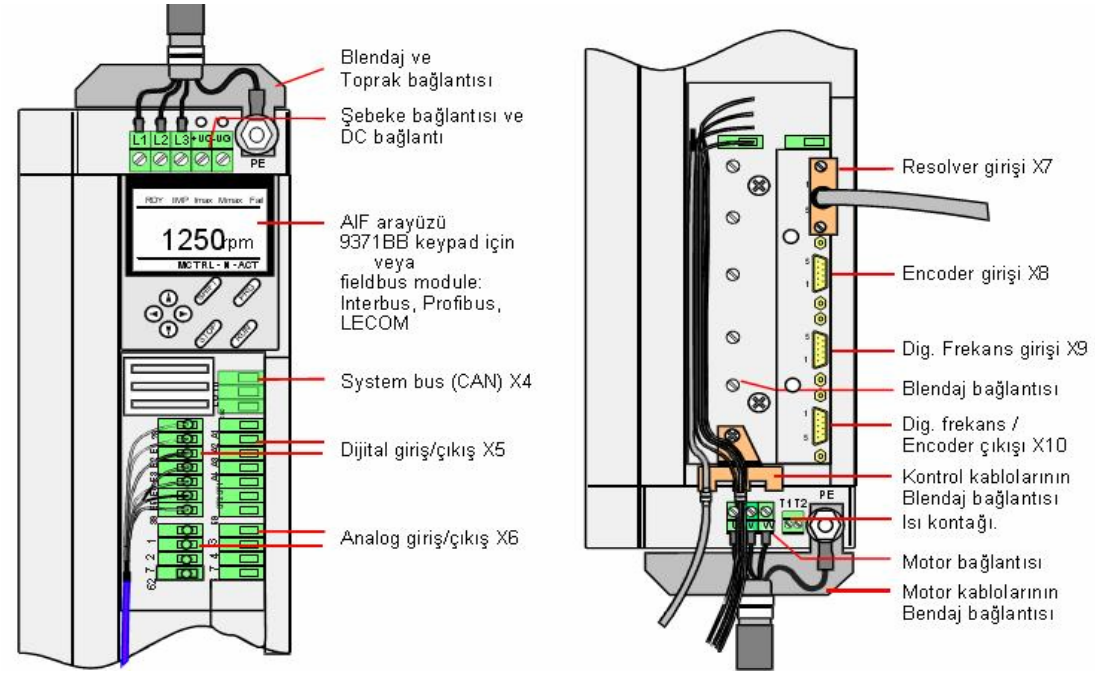
ii)-Eksen kontrol kartı A ve B olmak üzere iki kanallı pulse üretmektedir. Kullanılacak sürücünün bu pulse'lere göre pozisyon ve hız ayarı yapması gerekmektedir.

iii)- Sürücüler referansı takip ederken verilen hedef hız ve pozisyona göre, anlık olarak yaptığı hatayı bulması, bir sonraki hareket hamlesinde bu hatayı yok edecek şekilde çalışması, yani gelen referansı hatasız bir şekilde takip etmesi gerekmektedir.

iv)-Sürücü gelen referansı takip ederken PID değerlerine göre çalışmalıdır. Bu PID değerleri sistemdeki diğer cihazlardan değiştirilebilmelidir.

v)-Sürücüler içerisinde programlanabilecek lojik fonksiyonlar olmalıdır.

Bu ihtiyaçlara göre kullanılabilecek en uygun sürücü Lenze firmasının ürettiği 9300 model sürücüdür. Sitemdeki üç eksen için 3, 1.5 ve 0.37 kW güçlerinde sürücüler kullanılacaktır. Sürücüleri programlamak için Lenze firmasının ürettiği GDC, yani Global Drive Contol programı kullanılacaktır. Bu programın kullanım kolaylığı diğer markaların yazılımlarına göre yüksektir ve fonksiyon kabiliyeti fazladır. Sürücüler sahada kullanılmadan önce bir pano üzerine takılmalıdır. Gerekli elektriksel bağlantılar yapılmalıdır. Kullanılacak olan Lenze marka sürücünün elektriksel bağlantı noktaları şekil 4.2'de gösterilmiştir.



Şekil 4.2. Lenze marka servo sürücünün bağlantı noktaları [16]

Sürücü ile bilgisayar arasında haberleşme yapılabilmesi için ara cihazlara ihtiyaç duyulur. Bu cihazlar bilgisayarlarda kullanılan haberleşme protokolü ile sürücünün kontrol kartının haberleşmesi için bir köprü görevi görür. Kullanılan sürücüler ile RS 232 ve Can Bus protokolleri ile haberleşilebilir. RS 232 protokolü ile en hızlı 19200 kB/s hızında haberleşme sağlanabilir ve en fazla bir sürücü ile haberleşilebilir. Can Bus protokolü ile 1000 kB/s hızında haberleşme sağlanabilir ve birden fazla sürücü ile aynı anda haberleşilebilir. Bilgisayar üzerindeki Usb çıkışı ile Can Bus protokolünde haberleşme yapılmasını sağlayan dönüştürücüler kullanılarak bilgisayar ile sürücüler haberleştirilebilir. GDC yazılımı, Lenze firmasının ürettiği Usb çeviriciyi otomatik olarak tanımaktadır. Aynı markanın Usb çeviricisi ve yazılımı kullanıldığında sadece kullanıcı ayarları yapılarak, haberleşme kurulabilir. Bilgisayar ile usb çevirici arasındaki donanımsal ayarları, GDC programı otomatik olarak yapacaktır.

GDC programı ile sürücüler arasında Can Bus protokolü ile haberleşme kurulduktan sonra, sürücülerin programları bilgisayar araclığı ile yapılabilir. Öncelikle X eksenini tahrik eden motorun sürücüsünü programlamak uygundur. Üç eksenin programı aynı olmalı, programlardaki fark sadece eksenlerin bağlı olduğu mekanik ölçülerin ve motorların farklılığı olmalıdır. Sürücü ile bilgisayar arasında haberleşme kurulduktan sonra, bilgisayar ilk olarak sürücü üzerinde bulunan fabrika parametrelerini okur. Bu parametreler başlangıç ayarlarıdır, istenildiği zaman bu ayarlara geri dönülebilir.

Sürcülerde bir kod listesi vardır. Kullanılabilecek bütün foksiyonlar için kodlar tanımlanmıştır. Bu kodların değerleri değiştirilerek istenilen ayar yapılabilir. Lenze marka sürücülerde hızlanma rampasının süresini değiştirmek için 12 numaralı kod, duruş rampasının süresini değiştiemek için de 13 numaralı kod atanmıştır. Rampa değerleri değiştirilmek istendiğinde, kodların değerlerini değiştirmek gerekir. Bütün kodların anlamları, üretici tarafından hazırlanmış olan kod tablolarında mevcuttur [15]. Sürücüler gelen elektriksel sinyalleri, içersinde yapılmış olan programa göre değerlendirir ve motora gönderdiği elektrik enerjisini ayarlayarak, motorun sistemi istenildiği gibi tahrik etmesini sağlar. Bütün bağlantı noktaları sürücü üzerinde numaralandırılmıştır. Sürücülere, 24 VDC seviyesindeki E1…E5 klamenslerindeki

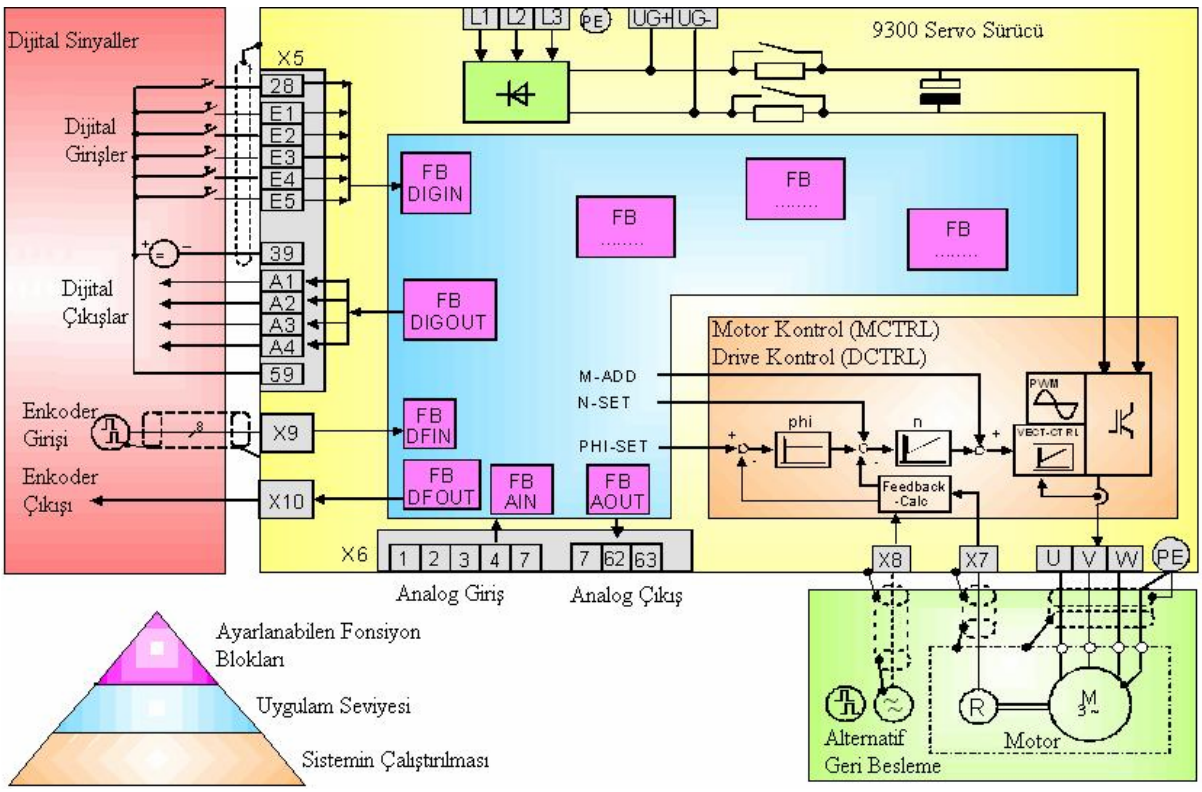
dijital girişler kullanılarak elektriksel komutlar gönderilebilir ve bu komutların sonuçları A1…A4 klamenslerindeki 24 VDC seviyesinde dijital çıkışlarından alınabilir. Sürücülerin dijital giriş ve çıkışları bir PLC gibi programlanabilir fakat fonksiyon kabiliyetleri PLC'ler kadar yüksek değildir. Basit lojik algoritmalar, sürücü üzerinde çalıştırılabilir. Sürücü üzerinde 1, 2, 3, 4 klamenslerindeki bağlantılar ile 10 VDC seviyesinde iki adet analog giriş, 62, 63, 7 klamenslerindeki bağlantılar ile de iki adet analog çıkışa sahiptir. Analog giriş sinyalleri ile sürücüye tork ve hız referans değerleri gönderilebilir. Analog çıkış sinyalleri ile de, anlık tork ve hız değerleri diğer sistemlere iletilebilir. Sürücü üzerindeki X9 klamensinden enkoder sinyali sürücüye gönderilebilir, X10 klamensi üzerinden de sürücü içerisinden encoder sinyali alınabilir. Encoder sinyalleri hız referansını oluşturmak için kullanılabilecek en hassas sinyallerdir. Analog sinyalleri kullanan en hassas devreler bile %0.05 hata yapılabilmektedir. Bu hata birçok uygulamada tolerans değerinin içerisinde kalmış olsa da, hassas uygulamalar için yetersizdir. Kesme makinesi gibi, kesilmiş malzeme üzerindeki toleransın 0.1 mm olduğu sistemlerde mutlaka enkoder sinyalleri kullanılmalıdır.

Kesim makinesi uygulamasında sürücünün motoru bir tur döndürmesi için X9 klamensinden 8196 pulse alması gerekmektedir. Ayrıca sürücü motoru bir tur çevirdiğinde, X10 klamensi üzerinden 8196 pulse gönderebilir. Böylece kontrol sistemi X9 klamensinden verdiği pulse'lar ile X10 klamensinden gelen pulse'leı karşılaştırarak sistemin yapmış olduğu hatayı izleyebilir. Sürücüye enerji verilmeden önce sürücünün elektriksel bağlantıları kontrol edilmelidir. Sürücüler 380 VAC besleme gerilimi ile çalıştıklarından, bağlantılarda yapılabilecek bir hata, önemli hasarlara neden olabilir. Öncelikle sürücünün L1..L3 klamenslerine gelen enerjinin 380 VAC olduğu kontrol edilmelidir. Enerji kablosu blendajlı olmalı ve blendaj ile toprak kablosu uygun yerlere bağlanmalıdır. Motor'a enerji iletecek kablolar U,V,W klamenslerine bağlanmalıdır. Sürücü ve motor üzerindeki kablo sıraları aynı olmalıdır. Kabloların sırasında oluşabilecek bir hata motorun istenilenin dışında bir hareket yapmasına sebep olacaktır. Motor'a giden enerji kablosu birçok frekansta harmonik sinyal taşır, bu sinyallerin sistemdeki diğer sinyalleri etkilememesi için kullanılacak kablo mutlaka blendajlı olmalıdır ve blendaj ile toprak kablosu uygun yere bağlanmalıdır. Sürücülerde farklı tip geri beslemeler için farklı tip klamensler

bulunur, sistemdeki geri beslemenin çalışabilmesi için, bağlantılar doğru klamense yapılmalıdır. Dijital sinyallerin bağlı olduğu klamensler kontrol edilmelidir. 39 numaralı klamense, dijital sinyallerin potansiyel seviyelerini eşitlemek için kullanılan 24 VDC geriliminin eksi ucu bağlanmalıdır. Bütün kontroller yapıldıktan sonra sürücüye enerji verilmelidir.

4.2. Servo Sürücülerin Fiziksel ve Yazılımsal Çalışma Seviyeleri

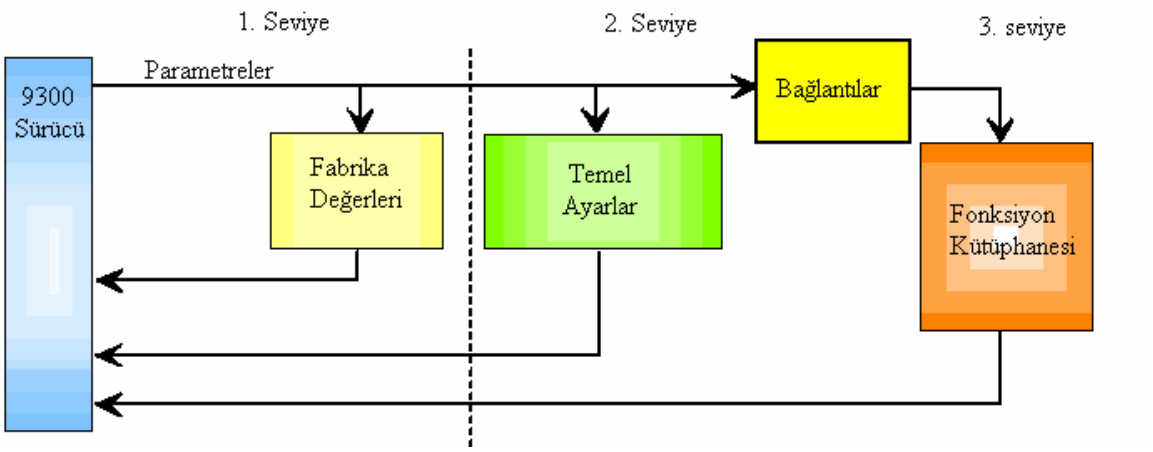
Servo sürücünün çalışması, üç temel seviye üzerinde fonsiyon blokları, uygulama ve sistemin çalışması seviyelerinde toplanabilir. Sürücüye dışardan gelen sinyaller klamensler aracılığı ile ayarlanabilen fonksiyon blokları kullanılarak uygulama seviyesi içerisine alınır. Uygulama seviyesinde, fonsiyon blokları arasında bağlantılar kurularak motorun dönmesi gereken değerler hesaplanır ve değerler sistemin çalıştıracak bölüme iletir. Sistemi çalıştıracak bölümde motorun istenilen değerlerde hareket etmesi için motora enerji uygulanır ve geri besleme ile uygulanan değerlerin doğruluğu kontrol edilir.



Şekil 4.3. Servo sürücülerin çalışma seviyeleri

Fonsiyon blokları yazılan programa göre, sürücü üzerindeki kodların değerlerini değiştirir. Kodların değerlerini teker teker manuel değiştirmek yerine fonksiyon bloklarını kullanmak, sürücüyü programlama işlemini hızlandırır. Servo sürücülerin çalışma seviyeleri şekil 4.3'de gösterilmiştir [15].

Sürücüye GDC programı ile bağlandıktan sonra istenilen ayarlamalar parametre menüsünün içerisindeki kodların değerleri değiştirilerek yapılabilir. Sürücüyü programlama işlemi üç temel seviyeye ayrılabilir. Birinci seviye sürücüyü fabrika değerleri ile çalıştırması içerir. Bu üretici tarafından yapılmış olan programdır ve sürücüyü hız modunda çalıştırmak için tasarlanmıştır. Hız modu, en basit programdır, programda sürücü dönüş referansını birinci analog girişten alır, dijital sinyaller ile dönüş yönünü belirleyerek motoru sürer. Sürücü üzerinde üretici tarafından yazılmış beş adet program vardır. Uygulamaya göre bu sabit programın seçilmesi ve çalıştırılması ikici seviyededir. Bu programlar hız modu, tork modu, ve üç farklı senkronizasyon programı içerir. Üçüncü seviye ise programda kullanılan fonksiyon bloklarının bağlantılarının, uygulamaya yönelik olarak kullanıcı tarafından yapılmasıdır. Sistemlerin özel yazılım gereksinimleri, üretici tarafından yapılmış olan programlar ile çözülmesi mümkün değildir. Programcının, fonksiyon bloklarındaki bağlantıları kullanarak, özel çözümler üretmesi gerekir. Servo sürücülerdeki programlama seviyeleri şekil 4.4'de gösterilmiştir [15].



Şekil 4.4. Servo sürücülerin programlama seviyeleri

4.3. Servo Sürücülere Motorların Tanıtılması

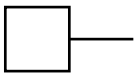
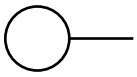
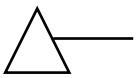
Servo sürücülerde en önemli noktalardan biri kullanılacak motorun sürücüye tanıtılmasıdır. Motorun bütün özelliklerinin sürücüye tanıtılması gerekir. Aynı marka sürücü ve motor kullanıldığında, sürücü içerisinde motor değerleri bulunmaktadır. Farklı marka motor ve sürücü kullanıldığında, kullanılan motorun plakası üzerinde yazılı olan değerler tanıtma işlemi için yetersiz kalmakta, motorun eş değer devresindeki bütün değerlerin üretici firmadan istenmesi gerekmektedir. Sürücülerde kullanılacak servo motorların bilinmesi gereken değerlerinin listesi ile bu değerlerin Lenze sürücülerde girilmesi gereken kod numaraları tablo 4.1'de gösterilmiştir [14]. Lenze marka sürücüde, Lenze marka motor tanıtımı 86 numaralı kod'dan yapılmaktadır. Motorun dönüş yönü kontrol edilmeli, eğer yanlış ise sürücü üzeirnden değiştirilmelidir. Motor yönünü değiştirmek için, motor kablolarının sırası değiştirilmemelidir. Motor ayarlandıktan sonra, kullanılan geri besleme sürücüye 25 numralı kod'dan tanıtılmalıdır. Motor ve geri besleme tanıtıldıktan sonra sürücü motoru kontrol etmek için hazırdır.

Tablo 4.1. Sürücüler ile kullanılacak motorlarda bilinmesi gereken değerler tablosu

Kod	Açıklaması
C0006	Motor tipi
C0022	Maksimum akım limiti
C0070	Hız kontrolü için P kazancı
C0071	Hız kontrolü için Ti zamanı
C0075	Akım kontrolü için P kazancı
C0076	Akım kontrolü için Ti zamanı
C0081	Motor gücü
C0084	Motorun stator direnci
C0085	Motorun Leakage endüktansı
C0087	Morun devri
C0088	Motorun akımı
C0089	Motorun frekansı
C0090	Motorun voltajı
C0091	Motorun cos phi değeri

Servo sürücüler içerisinde dört temel sinyal tipi vardır. Bu sinyaller dijital sinyaller, analog sinyaller, hız sinyalleri ve açı sinyalleridir. Açı sinyali motorun hızının integrali alınmış halidir ve motorun ilerlediği açıyı verir. Servo sürücülerde kullanılan sinyal tipleri, sembolleri ve değer aralıkları, tablo 4.2'de gösterilmiştir.

Tablo 4.2. Servo sürücülerde kullanılan sinyal tipleri

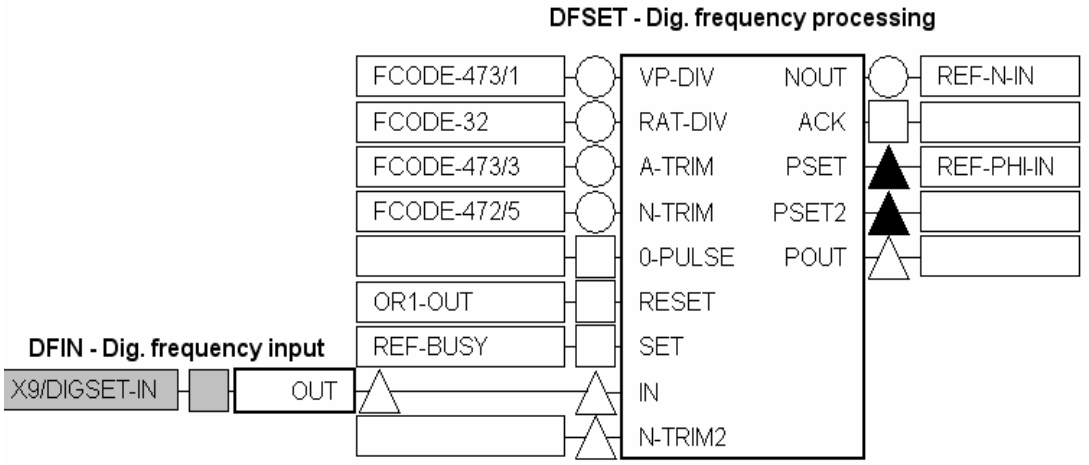
Sinya tipi	Sembol	Değer aralığı
Dijital		0 veya 1 sinyali
Analog		16 Bit (-32767… + 32767) 100 % = 14 Bit, ±16384 Çözünürlük: 1 = 0.0061 %
Hız		16 Bit, -32767 … + 32767 (Incr /ms) 15000 rpm = 16384 Incr / ms Çözünürlük : 1 Incr / ms = 0.9155 rpm
Açı		32 Bit, - 231 … + 231 (Incr) 1 Tur = 65536 Incr

4.4. Kesme Makinesindeki Servo Sürücünün Programı

Kesme makinesinde sürücüler, eksen kontrol kartından gelen pulse miktarına göre hareket etmelidir. Hassas bir kontrol için eksen kontrol kartı, motorun bir tur dönüşü için 8196 pulse üretmektedir. Sürücünün 425 numaralı kodu 5 seçilerek, 8196 pulse bir tur'a eşitlenmelidir. Kontrol kartı enkoder sinyali olarak A kanalından hız ve B kanalından da yön göndermektedir. Sürücünün 427 numaralı kodu 1 seçilerek, gelen dijital sinyali sürücünün algılaması sağlanmalıdır.

Fonksiyon blokları "Tool" menüsünden "FB Editor" seçilerek açılır. FB editörü ile istenilen fonksiyon blokları arasında bağlantı kalem simgesi seçilerek kurulabilir veya makas simgesi seçilerek bağlantılar silinebilir. Motorlar kontrol kartının ürettiği enkoder sinyallerinin frekansına göre hareket etmelidir. X9 klamensine bağlı olan enkoder sinyali, DFIN fonksiyon bloğu ile uygulama katmanının içerisine alınır.

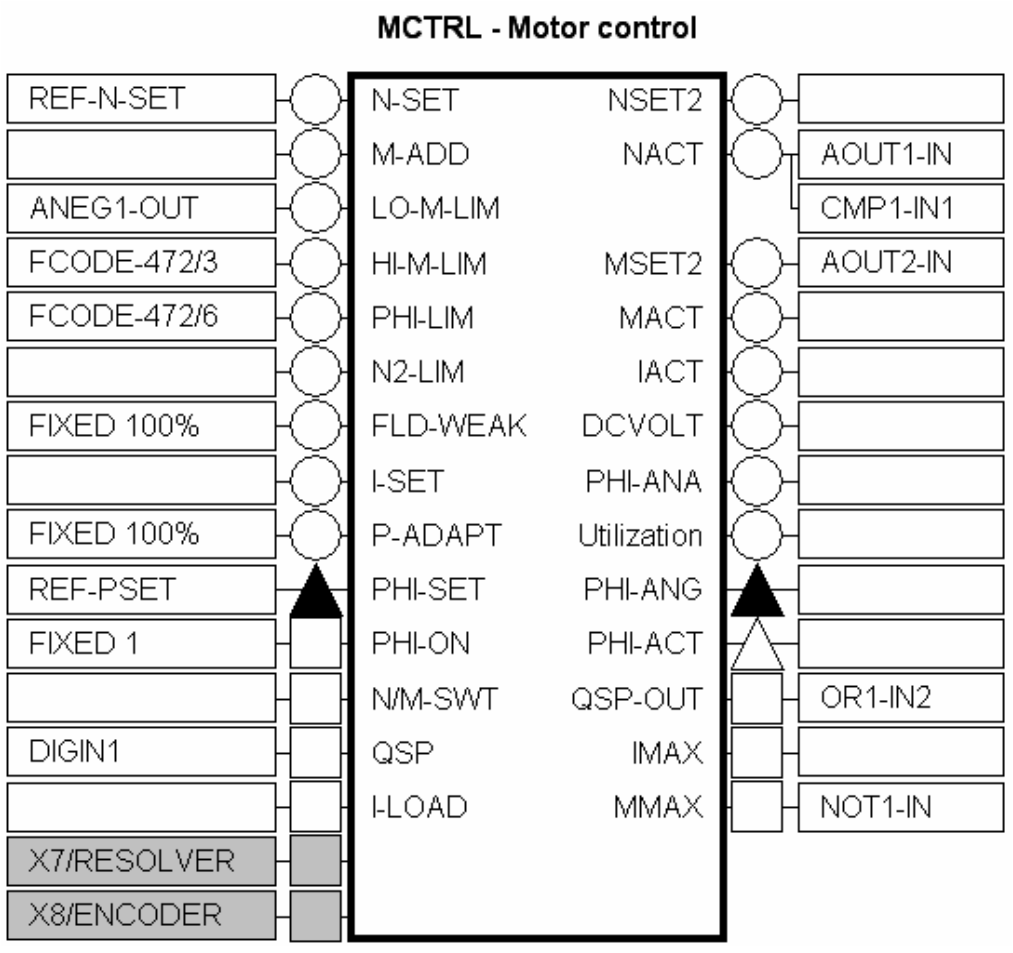
Kontrol kartı eksenlere, 1mm ilerlemeleri için eşit sayıda pulse üretir fakat motorların bir turda ilerledikleri uzunluk farklıdır. Gelen encoder sinyali, mekanizmadaki farklılıklar göz önüne alınarak her eksen için bir katsayı ile çarpıldıktan sonra motora referans sinyali olarak gönderilmesi gerekmektedir. Motorlar enkoder sinyalini takip ederken hata yapabilirler. Motorun hedeflenen hareketi ile, gerçekte gittiği hareket arasındaki hatanın belirlenmesi gerekmektedir. Sürücü içerisindeki DFSET fonksiyon bloğu hız referansını DFIN fonksiyon bloğundan alır, gelen değeri 32 numaralı kod içerinde yazılı olan değer ile çarpar, 33 numaralı kod içerisinde yazılı olan değer ile böler ve referans değeri olarak motora gönderir. Bunun yanında motorun gerçek hareketini, hedeflenen hareketten çıkartarak, açısal faz hatasının bulunmasını sağlar. Hata değeri sürücünün enerjisi kesildiği anda reset girişinin değeri bir yapılarak sıfırlanmalıdır. Sürücü içindeki DFSET fonksiyon bloğunun bağlantı şeması şekil 4.5'de gösterilmiştir.



Şekil 4.5. DFSET fonksiyon bloğunun bağlantı yapısı

Sistem çalıştırılmadan önce başlangıç noktasına gönderilebilir. Bu işlem kesme makinesinde eksen kontrol cihazı tarafından yapılacaktır. Servo sürücüler de içindeki REF fonksiyon bloğu ile home işlemini gerçekleştirebilirler. Motoru kontrol eden fonksiyon bloğu MCTRL'dir. Sürücü, motorun fonksiyon bloğu içerisindeki N-SET değişkenine gelen değer kadar ilerlemesini sağlar. Motorun sisteme uygulayacağı artı yöndeki tork değeri de HI-M-LIM, eksi yönde uygulayacağı tork değeri ise LO-M-LIM değişkenlerindeki değer ile belirlenir. Motora uygulanan referans değerine PHI-SET değişkeninin içerisindeki değer eklenir. Motorun faz hatası bu değişkene

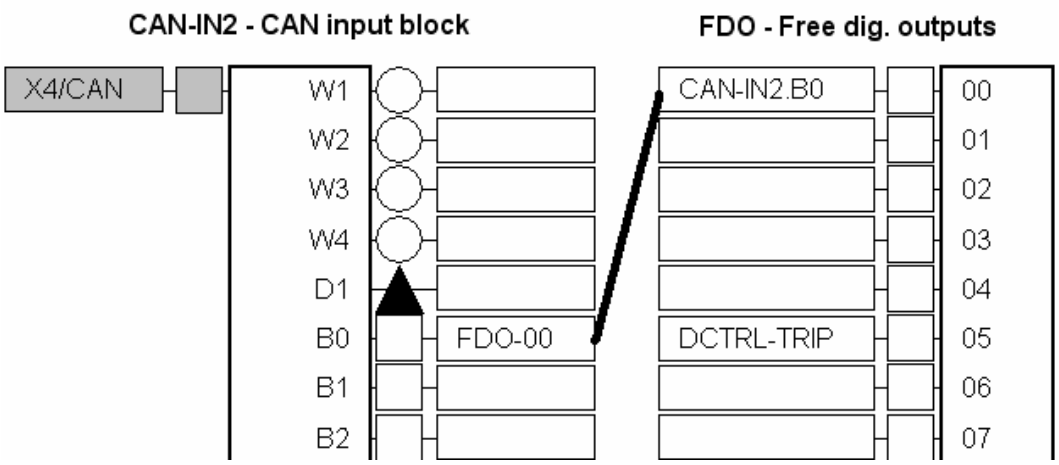
bağlanarak sürekli yapılan hata referanstan çıkartılır ve motorların olabilecek en hatasız dönüşü sağlanmış olur. Hata çıkartma algoritması PHI-ON değişkenine bir verildiği sürece aktif olacaktır. QSP girişine, birinci dijital giriş atanmıştır, böylece E1 klamensindeki birinci dijital giriş değerine voltaj geldiği anda motor acil duruşa geçecektir. NACT çıkışından motorun anlık hızı birinci analog çıkış değerine, MSET2 çıkışından motorun anlık momenti ikici analog çıkışa atanarak, sinyaller osiloskop gibi bir izleme cihazı tarafından incelenebilir ve motorun etkisine karşı sistemin tepkisi gözlenebilir. Bu fonksiyon bloğunda motorun çalışırken akım, DC voltaj, faz açısı gibi değerleri izlenebilir. MCTRL fonksiyon bloğunun bağlantı yapısı şekil 4.6'da gösterilmiştir.



Şekil 4.6. MCTRL fonksiyon bloğunun yapısı

Sürücülerin Can Bus üzerinden PDO kanalları ile haberleşmesi için özel fonksiyon blokları vardır. CAN-IN fonksiyon bloğu ile her PDO kanalından 4 word uzunluğunda veri alınabilir veya CAN-OUT fonksiyon bloğu ile her PDO kanalına 4 word uzunluğunda veri yollanabilir. PDO kanalındaki word'ler aracılığı ile bit uzunluğundaki sinyalleri göndermek için bit'leri word dizisine dönüştürmek gerekir. Bu dönüştürme işlemi için FDO fonksiyon bloğu kullanılır. Sürücü üzerinde 864 numaralı kod'a bir değeri atanarak, FDO fonksiyon bloğundaki bit dizisi, CAN2 PDO kanalının gönderilecek ilk word değerine atanır. Haberleşme kurabilmek için her sürücü ile PLC arasında bir PDO kanalı kullanılarak haberleşme sağlanabilir. Haberleşmede kullanılacak verilerin uzunluğu bir word'ü geçmiyor ise, sadece tek bir PDO kanalı kullanılarak da haberleşme hattı kurulabilir. Haberleşmenin PLC tarafından kontrolü için, PLC ikinci PDO kanalı olan CAN2 kanalının sıfırıncı bitinden sinyal gödermeketedir. Bu bit 5 saniye boyunca bir, ikinci beş saniye boyunca sıfır olmaktadır. Sinyal bütün cihazlar üzerinden geçerek tekrar PLC'ye geri dönmektedir. Bu döngünün sağlanması için sürücüye CAN2 kanalından gelen sıfırıncı bit, FDO fonksiyon bloğu aracılığı ile CAN2 kanalından bir sonraki sürücüye gönderilmelidir. Ayrıca sürücünün arızada olduğu bilgisi de PLC'ye CAN2 kanalı ile gönderilmiştir. Sürücünün adresi 5 olduğundan FDO fonksiyon bloğunun beşinci bit'i ile DCTRL fonsiyon bloğundaki "Trip" değeri gönderilmiştir. CAN2 ile gönderilecek verilerin adresi Y ekseninin sürücüsü olmalıdır. X ekseninin sürücüsü için fonksiyon bloklarının bağlantı şeması şekil 4.7'de gösterilmektedir.

Y sürücüsünün Can haberleşme bloklarının ayarlanması, X sürücüsünden farkı olmalıdır. Y sürücüsü, kendi hata değeri ile X sürücüsünün hata değerini Z sürücüsüne iletmektedir. CAN2 kanalından gelen beşinci biti, FDO fonksiyon bloğunun beşinci bit'ine, kendi hata sinyalini de altıncı bit'ine atamalı ve Z ekseninin sürücüsüne göndermelidir. Z ekseninin sürücüsü de CAN2 kanalından gelen beşinci ve altıncı bitleri, FDO fonksiyon bloğunun beşinci ve altıncı bit'lerine atamalı, kendi hata sinyalini de yedinci bit'e atamalı ve PLC'ye göndermelidir. Böylece bir PDO kanalı kullanılarak dört cihaz arasında haberleşme kurulabilir.

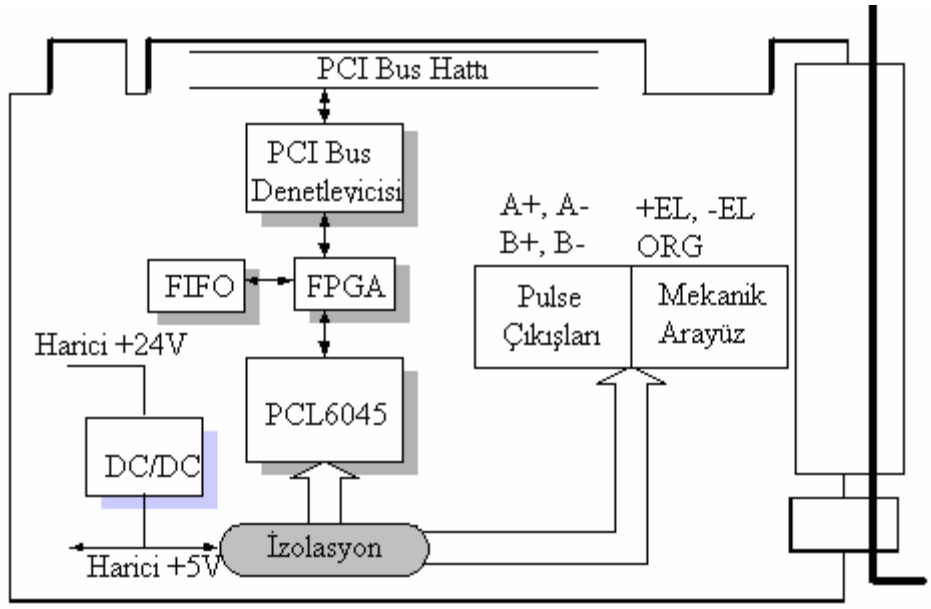


Şekil 4.7. Sürücü içerisinde Can Bus haberleşme fonksiyon bloklarının kullanım şeması

BÖLÜM 5. HAREKET VE EKSEN KONTROLÜ

5.1. Giriş

Sistemlerdeki en önemli ve hassas hareketler motorlar tarafından yapılır. Motorların hareketleri de sürücüler tarafından kontrol edilir. Gelişen sürücü teknolojisi ile en karmaşık uygulamalarda bile, sürücülerin içerisine motorları kontrol yöntemini belirleyen algoritmalar kurulabilmektedir. Bu algoritmalar, sadece motorun kendi ekseninin hareketi üzerine yoğunlaşmış, motorların grup hareketleri için yeterli düzeyde gelişmemiştir. Sürücüler hatasız pozisyon kontrol yapabilirler veya iki sürücü aynı anda farklı pozisyon profillerini harekete dönüştürmeye başlayabilir fakat iki hareketin bitiş süreleri aynı olmayabilir. Aynı hızda X ekseni 10 tur, Y ekseni 20 tur dönmesi istendiğinde, iki motorda harekete aynı andan başlayabilir fakat X ekseni hareketini, Y ekseninden önce bitirir. Ancak X ekseni verilen hızın yarısı ile ilerlemiş olsa bile, kalkış ve duruş rampalarındaki farklardan dolayı aynı anda X ve Y hareketini bitirememiş olabilir. Grup hareketlerinde sürücüler tarafından senkronizasyon yapılabilse de, interpolarizasyon yani eksenlerin hareketlerinin bitişi aynı sürede olaması için yapılan hız kontrolü, yapılamamaktadır. Eksenler arasında interpolarizasyon yapılabilmesi için bütün eksenlerin bulunduğu noktalar ve hedef pozisyonları mümkün olan en hızlı şekilde değerlendirerek, bütün hareketlerin aynı sürede tamamlanabilmesi için gerekli referansı üretebilecek bir kontrol devresine ihtiyaç vardır. Sistemdeki her motor bir eksen olarak değerlendirildiğinden, bu devre eksen kontrolü olarak isimlendirilir. Eksen kontrollü için yapılmış olan devrelerin kontrol edebileceği eksen sayısı sınırlıdır ve 2, 4 ve 8 olarak sıralanır. Eksen kontrol devrelerinin içerisinde dairesel interpolarizasyon ve teğet fonksiyonu gibi özellikler de bulunabilmekte, bu özellikler her eksen için yapılamayabilmektedir. 4 eksen için tasarlanmış bir kontrol devresi dairesel interpolarizasyonu sadece 2 eksen için yapabilir. PCI eksen kontrol kartının şeması şekil 5.1'deki gibi olabilir.



Şekil 5.1. PCI eksen kontrol kartının şeması

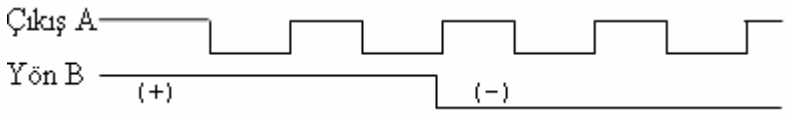
Eksen kontrol devreleri iki yöntem ile kullanılabilir. Birinci yöntem kontrol kartının bilgisayarın PCI portuna takılarak, port üzerinden haberleşmesidir. İkinci yöntem ise kapalı bir kutu içerisine koyulup, bilgisayardan harici bir noktaya konulması ve bilgisayardan haberleşme yollarından biri ile komut almasıdır. Eğer sistemde endüstriyel bir bilgisayar kullanılacak ise, PCI portu ile haberleşme yöntemi hem daha ucuzdur hemde daha hızlıdır, fakat kontrol kartının takılabileceği bilgisayar ile kullanılacak sürücülerin arasındaki uzaklık sınırlıdır. Eğer kontrol eden bilgisayar ile kullanılacak sürücülerin arasındaki mesafe uzun ise, ikinci yöntem seçilmelidir. Kesme makinesinde IPC ile sürücüler arasındaki mesafe 2 m olduğundan, PCI portuna takılabilen eksen kontrol kartı seçilmiştir.

PCI portuna takılabilen eksen kontrol kartları içerisinden Adlink firmasının ürettiği 8164 model, dört eksen kontrol kartı, Delphi ile kolay programlanabildiği için ve fiyat olarak uygun olduğu için seçilmiştir. PCI-8164 kartı, PCI bus ile 32 bit haberleşme hızına sahiptir. Dört ekseni encoder pulse üreterek sürebilir. Her hareket için hızlanma ve yavaşlama süresi verilebilir, rampalar S rampa veya lineer olarak seçilebilir. Dört eksenden iki tanesi üzerinde dairesel veya lineer interpolarizasyon yapabilir. Motor hareketli iken hızı değiştirilebilir. Eksen kontrol kartı home işlemini yapabilir. Kontol kartı, mekanizmanın limit noktaları veya home sensörleri gibi mekanik noktalarını, sensörler aracılığı ile elektriksel sinyaller olarak algılıyabilir.

Seçilen eksen kontrol kartının blok diagramı şekil 5.1'de gösterilmiştir. FIFO entegresinin içerisinde ardışık hareketleri kontrolü yapılır. FPGA entegtesi içerisinde ise komutlar toplanır. PCL6045 hareketin oluşturulduğu bölümdür.

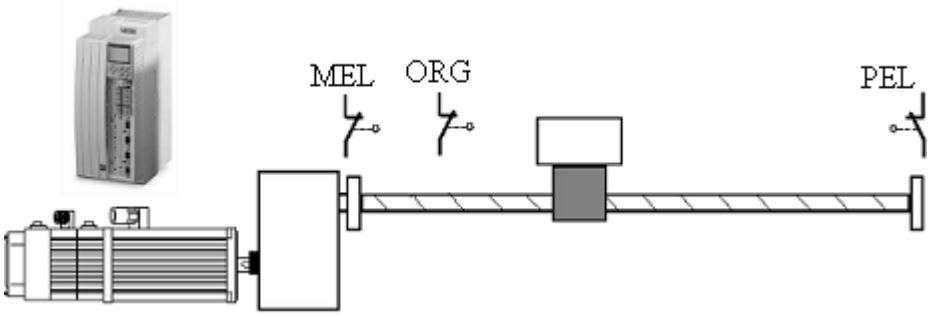
5.2. Hareket ve Limit Sinyalleri

Kontrol kartı motorun bir tur ilerlemesi için, motor sürücüsüne 8196 pulse kare dalgayı iki sinyal olarak, A ile sinyali, B ile de sinyalin yönünü belirleyerek gönderir. A ve B sinyalleri A+ ve B+ ile sinyalin kendisi, A- ve B- ile de sinyalin tersini diferansiyel olarak gönderilir. Sinyalin zamana bağlı integrali, yolu verir. Kontrol kartı bir dakika içerisinde sürücüye 81960 pulse gönderdiğinde, motor bir dakika boyunca 10 rpm hızında 10 tur döner. Bu süre içerisinde dönüş yönünü B sinyali belirler. Her eksenin sürücülerine ayrı ayrı bu sinyal gönderilmelidir. Eksen kontrol kartının üretmiş olduğu encoder sinyali şekil 5.2'de gösterilmiştir.



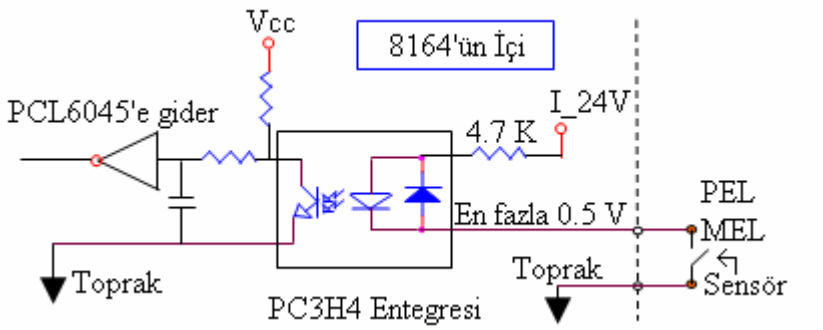
Şekil 5.2. Eksen kontrol kartının ürettiği encoder sinyali

Mekanizmanın herhangi bir hatadan ötürü, sistemin fiziksel olarak son noktasına ulaşabilme ihtimali vardır. Bu nokta yazılımdan oluşan bir hatadan dolayı yazılım içerisinde belirlenemeyebilir. Böyle bir durum oluştuğunda, motorlar mekanizmanın son noktasının ötesine ilerlemeye çalışır ise, mekanizma zarar görür. Böyle bir durum oluşma ihtimaline karşı mekanizmanın, mekanik olarak son noktalarına limit sensörleri konulur. Limit sensörlerinin herhangi birinden kontrol kartına sinyal geldiğinde sistem derhal durur. Böylece oluşan hatada mekanizmanın hasar görmemesi sağlanır. Sistemde mekanik limitlerden önce yazılım ile limit konulmalı, sistem çalışırken önce yazılım limitleri devreye girmelidir. Yazılım limitleri ile sistemin mekanik limitlere ulaşmaması sağlanmalıdır. Yazılım limitleri, mekanik limitlerden önce konulan bir güvenlik önlemidir.



Şekil 5.3. Eksen kontrol sisteminde PEL, MEL ve ORG sinyallerinin mekanizmadaki yerleri

Her eksenin artı ve eksi yönlerde olmak üzere iki limit noktası vardır. Artı yöndeki limit noktası PEL ve eksi yöndeki limit noktası MEL olarak tanımlanır. PEL sensöründen sinyal geldiğinde, sistem manuel olarak motorlar ile artı yönde hareket etmemeli, eksi yönde hareket edebilmelidir. MEL sensöründen sinyal geldiğinde ise sistem eksi yönde hareket etmemeli, sadece artı yönde hareket edebilmelidir. Böylece mekanizma manuel hareket ettirilerek tekrar mekanik limitlerin içerisine alınabilir. Home, PEL ve MEL sinyalleri için örnek sistem şekil 5.3'de gösterilmiştir. Kontrol kartının içerisinde PEL ve MEL için kullanılan devre şeması şekil 5.4'de gösterilmiştir.

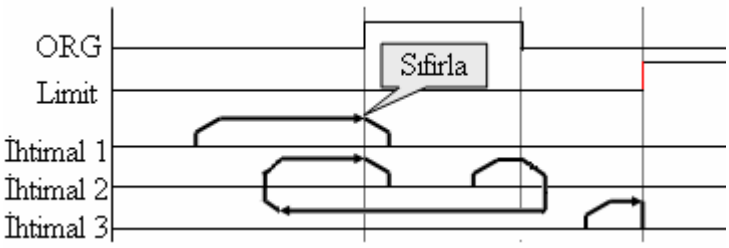


Şekil 5.4. Eksen kontrol kartının şeması [22]

5.3. Sıfırlama İşlemi

Kontrol kartı enerjisi kapatılıp açıldığında bulunduğu noktayı sıfır kabul eder. Eğer mekanizma sıfır noktasından farklı bir noktada enerji kapatılıp tekrar açılır ise, makinenin sıfır noktası ile kontrol kartının sıfır noktası birbirinden farklı olur. Bu fark ürünlerdeki kesimlerinin hatalı olmasına sebep olur. Hatayı önlemek için her

enerji açıldığında, mekanizma başlangıç noktasına gönderilmeli ve mekanizmanın başlangıç noktasında olduğu, bir sensör ile kontrol katına bildirilmeli ve kontrol kartının sayıcılarının sıfırlanması gerekmektedir. Bu işleme kısaca home adı verilir. Kontrol kartı üzerinde, mekanizmanın sıfır noktasında olduğunu gösteren sinyale ORG adı verilmektedir. Her eksenin başlangıç noktasının ayrı olmasından dolayı, kontrol kartında her eksen için ayrı bir ORG girişi vardır. Kontrol kartı home işlemini başlattıktan sonra, ORG sinyalinin yükselen kenar tetiklemesi gelene kadar belirlenmiş olan yönde yavaça ilerler. ORG sinyalinin yükselen kenarını algıladıktan sonra, kendi pozisyon sayıcılarını sıfırlar ve hareketi durdurur. Sistem home işlemini yaparken, birinci ihtimal ORG sinyalini görmesi ve durmasıdır. İkinci ihtimal ise home işlemine başlarken ORG sinyalinin daha önceden geliyor olmasıdır. Bu durumda mekanizma ORG sinyali kaybolana kadar ters yönde ilerlemeli ve sinyal kaybolduktan sonra tekrar home işlemine başlamalıdır. Üçüncü ihtimal ise ORG sinyalini üreten sensörde arıza olma ihtimalidir. Bu durumda mekanizma sıfır noktasına geldiği halde sinyal alamayacağından en son limit noktasına kadar ilerlemeye devam edecektir. Limit noktasına ulaştığında, yani limit sensöründen sinyal geldiğinde, motor mekanizmaya zarar vermemek için derhal durmalıdır. Home işleminin çalışma algoritması şekil 5.5'de gösterilmiştir.

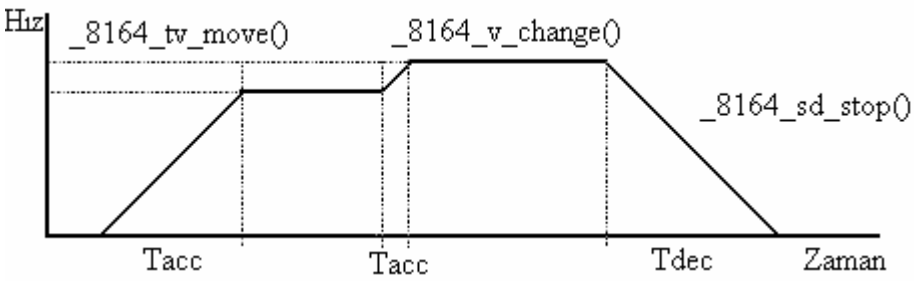


Şekil 5.5. Home işleminin çalışma algoritması

5.4. Hareketin Oluşturulması

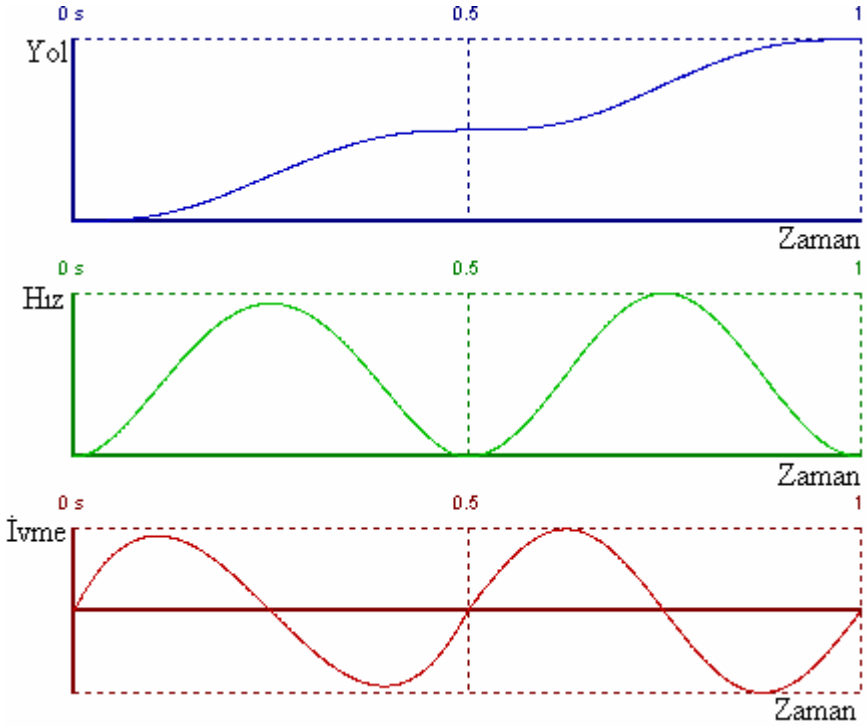
Eksen kontrol kartının sürücülere istenilen hareketi oluşturacak pulse çıkışlarını verebilmesi için Delphi programında karta özel komutlar kullanılır. Her hareket için farklı komutun yazılması gerekir. Aynı komutlar, içerisindeki eksen seçme değişkeni kullanılarak farkı eksenler içinde kullanılabilir. İstenen eksenin belirlenen rampa ile harekete başlayarak, istenilen hıza ulaşması ve belirli bir rampa ile durması için

kontrol kartında hız modu komutları vardır. Eksenin sabit bir hızla ve lineer bir ivme ile pozisyonunu önemsemeden hareket etmesi için _8164_tv_move(), bu hareketi S rampası ile yapması için _8164_sv_move(), motor hareket ederken hızının değişmesi için _8164_v_change() komutları kullanılabilir. Ekseni belirli bir rampa ile durdurmak için _8164_sd_stop(), rampasız acil duruş komutu için de _8164_emg_stop() komutları kullanılır. Bu komutların hız zaman grafiğine bağlı etkileri şekil 5.6'da gösterilmiştir.



Şekil 5.6. Komutların harekete olan etkisi

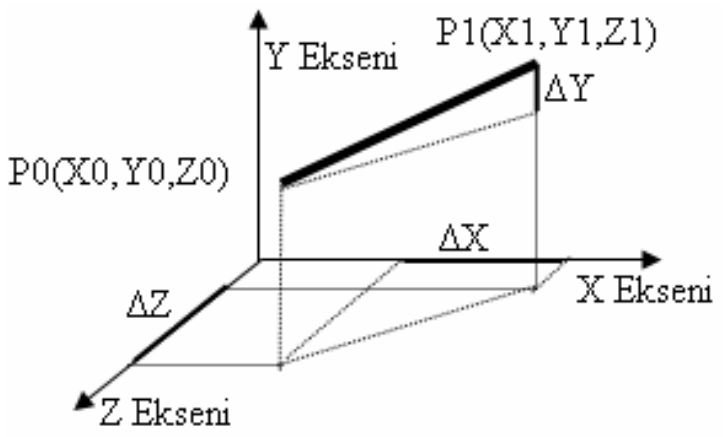
Rampalarda, lineer yerine S eğrisi kullanılması sistemin hareketin daha yumuşak olmasını sağlar. Duruş anından hızlanma rampasına geçişte veya hızlanma rampasından sabit hıza geçişte, yani ivme değişmlerinde daha yumuşak hareketin olmasını sağlar. Lineer ve dinamik hareketlerde, en hızlı kalkış rampasının elde edilebilmesi için ivmenin anlık olarak sonsuza çıkması gerekir. İvme sonsuza çıktığında motor mekanizmayı, sanki bir tokat atmış gibi bütün gücü ile hızlandırır. Duruş anında ise motor mekanizmayı, sanki ters yönde bir tokat atmış gibi bütün gücü ile durdurur. Bu anlık sert hareketler mekanizmada kötü seslere ve titreşimlere sebep olarak mekanizmayı oluşturan maddelerin yorulmasına sebep olur. Bunun yerine S rampa kullanarak motor hızlandırıldığında yani hız fonksiyonu beşinci dereceden bir fonksiyon olarak verildiğinde, ivme sinüzoidal olarak değişir. Böylece ivme değişimleri yumuşak olur, mekanizmanın daha az yorulması sağlanır ve motorun zorlanmasını azalacağından ömrü uzar. Hız beşinci dereceden bir fonksiyon olarak verildiğinde oluşan S rampası ile yapılan hareketin, yol-zaman, hız-zaman ve ivme zaman grafikleri şekil 5.7'de gösterilmiştir.



Şekil 5.7. S rampası ile hareketin zamana bağlı konum, hız ve ivme grafikleri

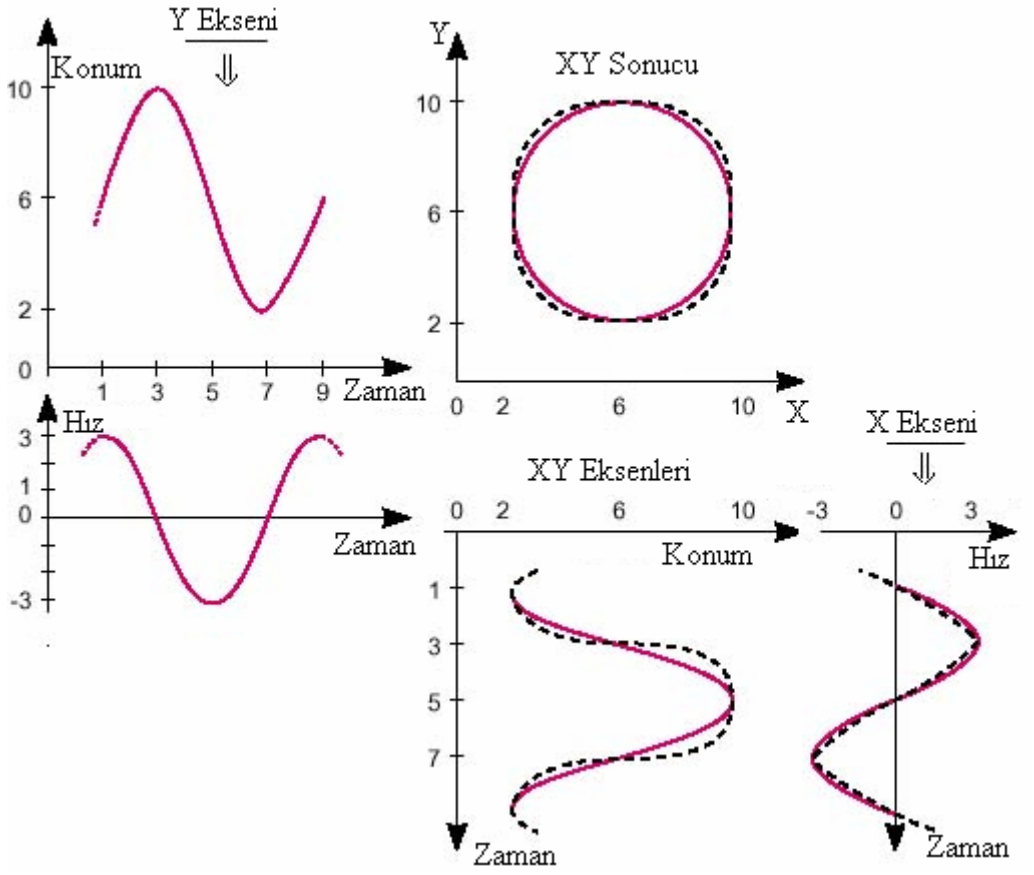
Eksenler teker teker bir noktadan diğer noktaya hareket edebilirler. Bu hareket sırasında motorlar istenilen rampa değerleri ile hızlanırlar. Motorlar istenilen pozisyona ulaştıklarında, rampa değeri ile durur. Hız yerine önemli olan pozisyondur ve kontrol kartı istenilen pozisyona gidebilmek için maksimum hız değerini belirler. Eğer kullanıcı bu hızdan yüksek bir hız istese bile, eksenler kontrol kartının belirlediği hızın üzerine çıkamaz. İki veya üç eksen, grup olarak bir noktadan diğer bir noktaya hareket edebilir. Bu hareket sırasında kullanıcı tarafından belirlenen rampalar ile eksenler hızlanır, eksenelerin kendi hızları yerine hareketin vektörel hızı kullanıcı tarafından belirlenir. Eksen kontrol kartıda hareket profiline göre eksenlerin vektörel hızlarını belirler ve sürücülere gönderir. Böylece bütün eksenler aynı anda harekete başlarlar, farklı hızlarla ilerlerler, aynı anda hareketi bitirirler. Eksenlerin bir grup olarak hareket etmesine interpolarizasyon denir. Hareketler absolute veya relative olarak iki yöntem ile gerçekleştirilebilirler. Absolute hareket yönteminde, başlangıç noktası referans alınarak hedef noktasına ulaşılır. Relative hareket yönteminde ise, bununan nokta referans alınarak hedef noktasına ulaşılır. Örnek olarak sıfır noktasından, 1,2 ve 3 hedefleri verildiğinde, absolute hareket yönteminde sistem 1,2 ve 3 noktasında olur. Relative hareket yönteminde ise sistem 1,3 ve 6 noktalarında olur. Kontrol kartına komut yazarken, _ta absolute hareketi, _tr relative

hareketi temsil eder. X ve Y eksenlerini bir noktadan diğer bir noktaya, absolute hareket modunda, S rampası ile hareket ettirmek için _8164_start_sa_move_xy() komutu kullanılabilir. X, Y ve Z eksenlerini, relative hareket modunda, lineer rampalar ile hareket ettirebilmek için ise _8164_start_tr_line3 komutu kullanılabilir. Üç eksen için interpolarizasyon, koordinat sistemi üzerinde şekil 5.8'de gösterilmiştir [22].



Şekil 5.8. Üç eksen için interpolarizasyonun gösterilmesi

Kullanılan kontrol kartı ile, üç eksenden ikisi üzerinde daire çizilebilmesi dairesel interpolarizasyon yapılmalıdır. Dairesel interpolarizasyonun olabilmesi için X ve Y eksenlerinden birinin sinüs, diğerinin ise cosinüs fonksiyonu ile hareket etmesi gerekir. Eğer eksen kontrol kartının dairesel interpolarizasyon yapabilme özelliği yok ise, lineer interpolarizasyon yöntemleri ile tam bir daire çizilemeyebilir. Şekil 5.9'da noktalı çizgiler ile, eksenlerin dairesel interpolarizasyon kullanılmadığında oluşacak sonuçlar ve daire çizebilmek için X ve Y eksenlerinin zamana bağlı hız zaman grafikleri, XY hareketinin konum zaman grafiği gösterilmektedir.

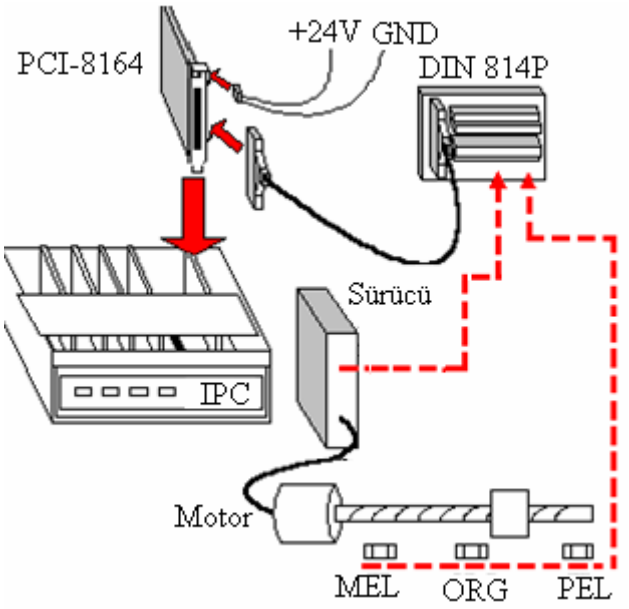


Şekil 5.9. Daire oluşturmak için gerekli hareketin gösterimi

5.5. Kablo bağlantıları

Konrol kartına gelebilecek ve gidebilecek sinyallerin toplam sayısı yüz'dür. Her bir sinyal için bir kablo kullanıldığından, yüz adet kablo kontrol kartına bağlanabilir. Kontrol kartının mekanik yapısının çok küçük olmasından dolayı, sistemin kablo montajınını kolaylaştırmak için, yüz kablo özel yüz uçlu sokete bağlanmıştır ve soketin karşılığı da eksen kontrol kartına yerleştirilmiştir. Soket yöntemi kontrol kartına yüz kablonun bağlanmasını kolaylaştırmıştır fakat sahadan gelen kabloların soket'e bağlanması zorlaşmıştır. Sahadan gelen kablolar bir klamens grubu üzerinde toplanarak, klamens grubundan da sokete sabit bir bağlantı ucu oluşturularak, sahadan gelen kabloların bağlantısı kolaylaştırılabilir. Kesme makinesinde DIN 814P klamens grubu kullanılmıştır. Klamens grubu sabit olarak yüz kablo için soket çıkışı verir. Klamens grubunun üzerindeki soket bağlantısı ile kontrol kartı üzerindeki soket bağlantısı arasında standart yüz damarlı soketli bir kablo kullanılarak, bağlantılar kolay bir şekilde yapılabilir. Sistem sökülmek istendiğinde ara kablo

sökülerek, sistem bağlanmak istendiğinde de ara kablo bağlanarak, istenilen işlem gerçekleştirilmiş olur. Sistemin basit bir diagramı şekil 5.10'da gösterilmiştir.



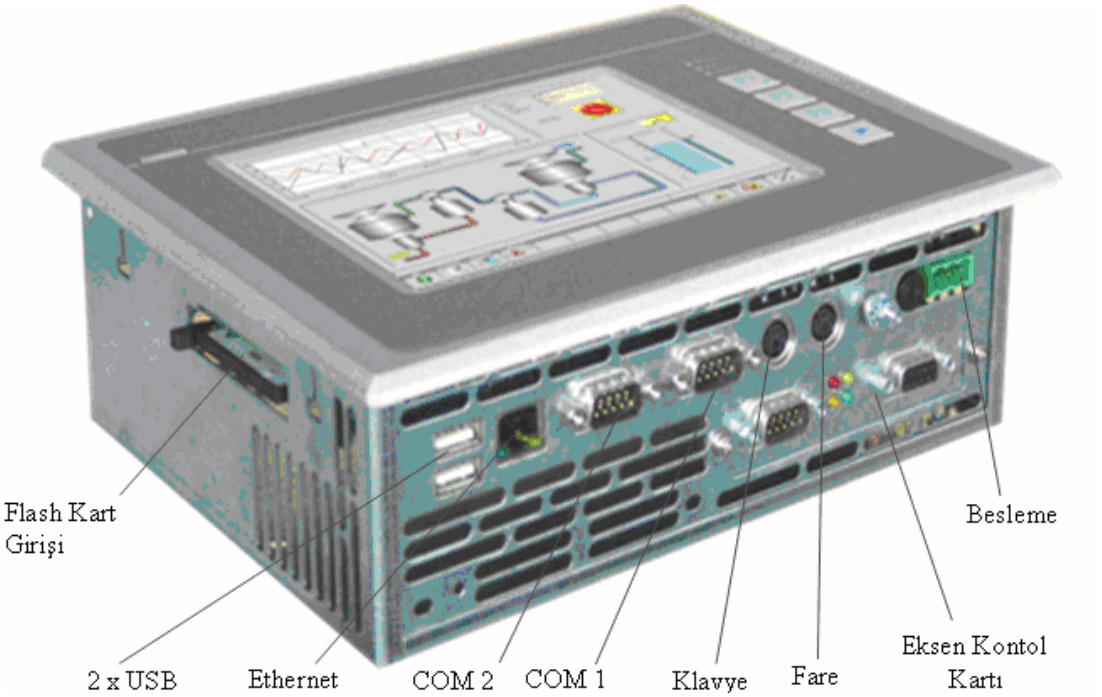
Şekil 5.10. Eksen kontrol sisteminin bağlantılarının şematik gösterimi [22]

BÖLÜM 6. ENDÜSTRİYEL BİLGİSAYARLAR VE SCADA

6.1. Giriş

Otomasyon sistemlerinin hızlı gelişimi karmaşık makinelerin kotrolünde, hareket kontrolü işlemlerinde, veri toplama ve izleme işlemlerinde, veri analizi işlemlerinde, PLC ve oparatör panelleri gibi elektronik cihazların yetersiz kalmasına sebep olmuştur. PLC'ler her ne kadar da gelişmiş olsa, fonksiyon kabiliyetleri bilgisayarlara erişememiştir. PLC'lerin lojik algoritmalardaki programlama kolaylıkları, bilgisayarlara karşı en üstün özellikleridir. Günümüzde gelişmiş otomasyon sistemlerinde, iki sisteminde üstün yanlarını, PLC'lerin lojik algoritmalardaki programlama hızı ile bilgisayarların fonksiyon üstünlüklerini kullanmak, en hızlı ve kolay çözümü getirir. Bilgisayarlar içinde yazılım oluşturarak, sistemelerin kontrolünün ve izlemesinin haberleşme kanalları kullanılarak yapılmasına kısaca SCADA denilmektedir. Sahada kullanılan bilgisayarların, normal bilgisayarlara göre birçok üstünlükleri vardır ve IPC kısa adını alırlar.

IPC'ler sistemlerde çok kritik noktalarda çalıştıklarından, bilgisayarın doğasından kaynaklanan kilitlenmeler veya bozulmalar önemli üretim kayıplarına neden olabilir. IPC'ler endüstriyel alanlarda elektronik devrelerin bozulmasını kolaylaştıran sıcaklık, nem, toz gibi kötü çevresel şartlarda çalışabilmektedirler. IPC'lerin zarar görmemeleri için içindeki elektronik devreler özel olarak, en kötü şartlarda çalışacak şekilde tasarlanmıştır. Ortamdaki toz ve nem gibi etmenlerden etkilenmemeleri için kartlar özel olarak yalıtılmıştır. Kartların üzerindeki komponentler, özel olarak yüksek ısılarda çalışacak şekilde seçilmiştir. Mekanik tasarımlarıda su ve toz geçirmeyecek şekilde tasarlanmıştır. İçlerinde özel havalandırma sistemleri vardır. IPC'lerin bütün iç bileşenleri titreşime karşı dayanıklıdır. İçerisindeki elektronik devreler elektromanyetik gürültülere karşı dayanıklıdır.



Şekil 6.1. Edüstriyel bilgisayar ve bağlantı noktaları [19]

Saha koşullarının klavye ve fare kullanmaya elverişli olmadığı durumlarda, IPC'lerin ekranları dokunmatik seçilebilir. Tuş takımı ekranın içerisine koyularak, fare kullanmak yerine ekranda seçilecek nesneye dokunularak kumanda edilebilirler. Dokunmatik ekranların en önemli özelliği, üzerindeki camın darbelere karşı dayanıklı olmasıdır. IPC'ler donanım özellikleri olarak izleyici ve kontrol edici olarak ikiye ayrılırlar. İzleyici IPC'ler içerisinde sadece izleme ve kumanda etmek için yeterli olan donanım vardır. Kontrol edici IPC'lerin içerisinde PLC gibi kumanda devresini kontrol edebilecek algoritma oluşturulabilir. Diğer sistemler ile haberleşebilirler ve veri işlemeyi içerisinde yapabilirler. Bir kontrol edici IPC şekil 6.1'de gösterilmektedir.

IPC'ler, standart bilgisayar ile aynı işletim sistemi ile çalışabilirler fakat genel olarak kullanılan işletim sistemlerinin bütün özelliklerini kullanmadıklarından, IPC'ler için windows CE gibi özel işletim sistemleri geliştirilmiştir. IPC ile standart bilgisayarlar arasındaki farklar genel olarak yazılımsal değil, donanımsaldır. IPC ile standart bilgisayarlar arasındaki farklar genel olarak tablo 6.1'de gösterilmiştir.

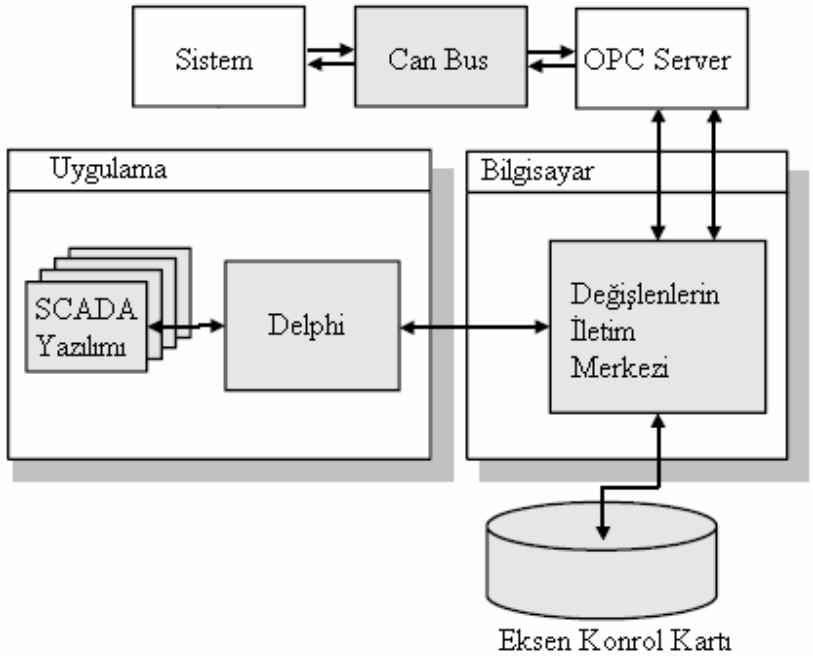
Tablo 6.1. Edüstriyel bilgisayar ile ofis bilgisayarının karşılaştırılması

Ofis Bilgisayarı ile IPC'nin Karşılaştırılması	
IPC	Ofis Bilgisayarı
Uzun zaman boyunca aynı ürün sağlanabilir veya yeni versiyonu ürünün yerine kusursuz çalışır.	Kullanılan ürünün yenisinin bulunabileceği garantisi verilmemektedir.
Cihazları oluşturan bileşenler nadiren değişir. Eğer gerekli ise, kullanıcı uyarılır veya değişimin sistem içerisinde çalışacağı garanti edilir	Aynı modellerde bile cihazların içeriği çok sık değişebilir.
EMC standarlarına uygundur	EMC standartlarına uygun değildir. Yakın giden enerji kablolarından parazit alabilir.
Yüksek performansa nadiren ihtiyaç duyulur	Yüksek performans bir standarttır.
Fanlar kullanıldığında, toza karşı filtrede kullanılır	Toza karşı filitre kullanılmaz
Mekanik yapı, kontrol panellerine göre yapılmıştır	Sadece zemin üzerine koyulmuştur
Kontrol kartları daha özenli üretilmiştir	Sadece belirli standartlarda kartlar üretilir
Titreşimlere karşı daha dayanıklıdır	Bileşenlerin bağlantıları titreşimde kopabilir
Çalışma sıcaklık aralığı genişletilmiştir	Çalışma sıcaklığı sabittir
Sorunlara karşı daha iyi teknik servis, uzmanlar tarafından verilir	Problem olduğunda üretici ile görüşmek gerekir
Daha az sayıda imal edilir	Yüksek sayıda imal edilir
Daha yüksek fiyatlıdır	Daha düşük fiyatlıdır

6.2. Kesme Makinesinde Endüstriyel Bilgisayarın Yeri

Kesme makinesinde, bilgisayarın PCI protunda çalışan endüstriyel eksen kontrol kartı, PLC ve sürücüler bir arada çalışmaktadır. Bütün bu donanımların ortak bir tabanda birleşmesi gerekir. En uygun ortak taban bir IPC'dir. Eksen kontrol kartı, IPC üzerine takılarak Delphi programı aracılığı ile programlanabilir. PLC ve sürüler OPC aracılığı ile Can Bus protokolünden IPC ile kumanda edilebilir. IPC ile bütün sistem tek bir noktadan kumanda edebilir ve kullanıcı tarafından izlenebilir. IPC üzerinden SCADA oluşturmak için iki yöntem vardır. Birinci yöntem birçok ara yazılımı daha önceden yapılmış olan bir SCADA programı kullanmaktır. Bu programların maliyetleri yüksektir ve izlenecek verilerin büyüklüğüne göre fiyatlandırılırlar. Gereki birçok fonksiyon üretici tarafından önceden yazılıp programın içine koyulmuş olduğundan, programcının işi kolaylaşır. İkinci yöntem ise Delphi gibi bir bilgisayar programlama dili ile bütün programı oluşturmaktır. Program baştan itibaren bütün fonksiyonları programcı tarafından oluşturulacağından

yazılımın maliyetini programcı oluşturur. Kesme makinesinde ikici yöntem kullanılarak bir SCADA programı yazılarak, Delphi programı aracılığı ile makine kumanda edilecektir. SCADA sistemlerinde haberleşme en temel faktördür. Ana bilgisayar verileri haberleşme kanalları ile sistemdeki cihazlardan okur. Sistemledeki veriler PLC gibi cihazlardan anlık ve doğru olarak okunabildiği için, sistemler en optimum şekilde kontrol edilebilir ve SCADA içermeyen otomasyon sistemlerine göre çalışma maliyetleri düşer. SCADA sistemleri en son teknoloji kullanılarak yapılmalıdır ve yeni eklere açık olmalıdır. SCADA'nın kullanılacağı sistemde, kullanılan cihazların SCADA sistemi ile uyumlu olmaları ve sisteme mümkün oluğunca kolay eklenebilmeleri gerekmektedir. Kullanılan cihazların donanım yeterliliği gün geçtikçe artmış olsa da, yazılımlarındaki karmaşıklık hala sürmektedir. Sistemdeki bütün cihazlar tam olarak SCADA yazılımı ile kontrol edilemez ise, sistemin kontrolü aksayabilir. Kesme makinesinde, Windows XP tabanlı bir IPC kullanılmıştır ve böylece son teknolojiler sisteme kolaylıkla entegre edilebilir. Kullanılan bütün cihazalar Can Bus ile haberleşebilmektedir. OPC programı kullanılarak, yazılan SCADA yazılımı cihazlardaki bütün parametreler değiştirilerek sistem tam olarak kontrol edilebilir. IPC'nin otomasyon sistemi içerisindeki yeri şekil 6.2'de gösterilmiştir.



Şekil 6.2. Endüstriyel bilgisayarın sistem içerisindeki yerinin şematik gösterimi

6.3. SCADA Sisteminin Genel Yapısı

SCADA sistemlerinin oluşumu beş temel seviyeye ayırılabilir. Birinci seviyede sahada kumanda edilecek cihazların veya ölçme yapacak sensörlerin kurulumu gerçekleştirilmelidir. İkinci aşamada sahadaki cihazların, PLC gibi kullanılacak kontrol sistemlerine kablolarının doğru olarak bağlanması ve kontrol sistemlerine sahadaki cihazlara komut verebilecek lojik programların yapılması gerekmektedir. PLC'ler sahadaki analog ve dijital sinyaller ile SCADA sistemi arasındaki en iyi arabirimlerdir. Üçüncü aşamada kontrol sistemlerinin SCADA sistemi ile haberleştirilmesi gerekmektedir. Dördüncü aşamada SCADA'yı kullanacak sistemlerin belirlenmesi gerekmektedir. Beşinci aşamada ise veri alışverişi yapılarak sistemin kontrolünün yapılması ve istenilen yerlere veri depolanması gerekmektedir. Kesme makinesinde sensörler ve valfler makineye yerleştirilmiş, bağlantı uçları PLC'ye bağlanmıştır. PLC içerisinde kumanda programı yazılmıştır. Can Bus ile cihazlar haberleştirilmiştir. Sistemde IPC kullanılmıştır ve son aşamada Delphi programı yazılarak sistemin kontrolü yapılmıştır. Karmaşık uygulamalarda, haberleşme için kullanıllan taşıyıcılar sinyaller radyo dalgaları , telefon hattı veya mikrodalga olabilir. Karmaşık sistemlerde, bütün cihazlar ve haberleşme sistemleri birbiri ile karıştırılarak en uygun çözüm bulunabilir [17].

SCADA sistemi oluşturmadan önce ayrıntılı kontrol gereksinimleri, lojik dizisi, kullanılması gereken analog ve dijital sinyal sayısı, veri alışverişi için gerekli olan hız, gerekli izleme ekranı, sistemin güvenilirliği ve erişilebilirliği, haberleşme hızı, sistemin fazlalıkları, yazılım uygulaması konuları değerlendirilmelidir. SCADA sistemi oluşturulurken, sistem mümkün olduğunca basit tasarlanmalıdır. Sistemin toplam tepki süresi hesaplanmalıdır. Sistemin bütünü içerisindeki cihazların, herhangi birinin bozulabileceği hesaplanmalı, bozulması halinde sisteme en az hasar vermesi sağlanmalıdır. Sistemdeki cihazların kullandıkları protokoller, açık standartlara sahip olmalıdır. Sahadan veri toplayan ve gönderen cihazların hepsinde ölçeklendirilebilme özelliği olmalıdır. Cihazların ve haberleşme sistemlerinin herzaman maksimum yoğunlukta çalışabilecekleri düşünülmelidir. Sistemdeki bütün hesaplar, en fazla cihaz kullanılmasına göre yapılmalıdır. Sistemden alınan veriler, gerçek ölçüler ile karşılaştırılmalı ve ölçme sisteminin doğruluğundan emin

olunmalıdır. Ölçme cihazlarının herhangi birisinin arızalanması durumunda, sistemin en az hasar alacağı yöntem belirlenmelidir. Sistemde kullanılacak cihazların patlayabilme ihtimalleri değerlendirilmeli ve kullanıcıların en az hasar görmesi sağlanmalıdır. Ayarlama, test sonuçları ve sistemin kullanım klavuzu belgelenmelidir. Sistemi kullanacak operatörler sistemi en iyi kullanacak şekilde eğitilmelidirler. Bütün sistemin göstergeleri çok karışık yapılmamalı, alarmlar ve çalışma değerleri için basit ve kolay anlaşılabilir arayüzler tasarlanmalıdır. SCADA sistemi oluşturulduktan sonra, sistemler optimize edilerek, sistemlerin çalışma maliyetleri düşürülebilir, çalışanların verimliliği yükseltilebilir, sistemin güvenliği arttırılabilir, çevresel ortamlardaki hatalardan kaynaklanan hasarlar azaltılabilir, sistemlerde enerji tasarrufu sağlanabilir, sistemlerdeki olayların hızlı izlenmesi sonucunda hızlı tepkiler verilebilir.

Sistemlerde başlangıç şartları, izleme ve sonuçların kontrolü SCADA yazılımı tarafından yapılmalıdır, prosesin kontrolünü yani verilere göre sistemin çalışma algoritmasını ise bağlı olan PLC gibi kontrol cihazları yapmalıdır. SCADA çalışırken sistemin kontrolü için PLC'ler de kendi aralarında haberleşebilirler. Sistemdeki PLC'lerin işlemcisi, hafızası, analog giriş ve çıkışları, sayıcı girişleri, dijital giriş ve çıkışları, haberleşme ara yüzleri, güç kaynakları olmalı ve bütün bunlar kapalı bir kutu içerisinde bulunmalıdır.

6.4. SCADA Yazılımının Oluşturulması

Eksen kontrol kartını kumanda edebilmek ve kart ile sistemi ilişkilendirebilmek için bir arabirim gerekmektedir. Yani pano üzerindeki "Start" tuşu ile makinenin çalışmaya başlaması veya "home" tuşu ile sistemin bütün eksenlerinin başlangıç noktalarına gitmesi gibi komutlar için, komut ile komutları uygulayan sistem arasında bir bağlantı kurulması gerekmektedir. Bu bağlantıya aynı zamanda "home" yapılmadan önce sistemin çalışmaması gibi, kısıtlamaları da eklemek gerekmektedir. Aynı zamanda sistem en uygun şekilde izlenmelidir. Sistemin çalışmasının engelleyen kısıtlamalar ve sistemde oluşan arızalar da kullanıcıya uyarı olarak bir monitörde bildirilmelidir. Bunlara benzer komutları içeren yazılımlar genellikle cihazları üreten firmalar tarafından müşteriye sunulmaktadır. Ara birim yazılımı

oluşturulmadan cihazların günümüz piyasa şartlarında satılması mümkün değildir. Her ne kadar bu yazılımlar üretici firmalara göre cihazların bütün fonksiyonlarını ortaya koysa da, kullanıcıların özel isteklerini karşılayamamaktadır. Üretici firmalar genel olarak cihazları belirli bir müşteriye göre değil, daha çok şatış hedeflediklerinden pazarın geneline hitap edecek şekilde üretirler. Bunun sonucunda hazırladıkları yazılımlar da kullanıcıların genel ihtiyacını karşılar. Özel kullanıcı ihtiyaçları için ya yazılımlara ek yapılması yada yazılımın tekrardan oluşturulması gerekir. Sistemde hareket kontrolü için kullanılan kontrol kartınında, üretici firma tarafından hazırlanmış yazılımı, kontrol kartının yazılımı kontrol fonksiyolarını yerine getirme konusunda yeterli görünse de, Can Bus ile haberleşemediğinden ve projedeki özel kısıtlamaları içermediğinden dolayı, kesme makinesinin ithiyaclarını karşılamamıştır. Sistemin kendi ihtiyaçlarına göre yeni bir yazılım oluşturulması gerekmektedir. Bu yazılım, standart bir SCADA programının içerdiği bütün fonksiyonları içermelidir.

6.4.1. Delphi yazılımı ile kontrol kartının programlanması

Sisteme özel geliştirilecek yazılım için programlama dilleri içerisinden Delphi 7.0 programını seçilmiştir. Delphi'yi programının seçilmesindeki en büyük sebepler "OPC Server" programı ile çok kolay haberleşmesi, haberleşme için gerekli component'in kullanımının çok kolay olması ve görsel açıdan program yazılımının çok rahat olması olarak sıralanabilir. Projenin yazılım aşamalarını geliştirilirken, Delphi programının genel özelliklerinden araştırmak yerine, yazılımı oluşturacak Delphi programında kullanılacak özellikleri araştırılacaktır. Bu projede araştırılmak istenen konu Delphi programı değil, onun aracılığı ile yapılabilecek yazılımlardır. Dolayısı ile sadece yazılımda kullanılacak fonksiyonlar incelenecektir. File-New-application sekmesini seçilerek yeni bir proje açılmalı ve programlamaya başlanılmalıdır. Proje açıldıktan sonra Form ve kod penceresi açılmalıdır. Kod bölümünü değişkenlerimi tanımlamak, prosedürleri oluşturmak ve fonksiyonları tanımlamak için kullanılmalıdır. Form'ları ise yazılmış olan kodların, kullanıcı ara birimleri olarak kullanılmalıdır. Delphi'de her form'un "unit" diye tanımlanan bir kod bölümü vardır. Kod bölümü ile form bölümü birlikte çalışır.

Projenin daha iyi anlatılabilmesi için, programın yazılma işlemine başlamadan önce Delphi hakkında genel birkaç nokta belirtilmelidir. Delphi, programın kod bölümünde kullanılacak tanımlamaları, "Public" yani genel ve "Private" yani özel olarak tanımlanabilir. "Genel" olarak tanımlanmış olan fonksiyonlara diğer formlardan da ulaşılabilir, fakat özel olarak tanımlanmış fonksiyonlara sadece ilgili formda ulaşılabilir. Bunun yanında yapılan form'u diğer forma tanıtıldığı takdirde, özel fonksiyonları diğer bir formun parçası olarak kullanılabilir. Delphi'deki "object inspector" menüsünde kullanılan nesnelerin özelliklerini, "properties" sekmesinden belirlenebildiği gibi, "events" sekmesinde de nesneye bağlı olayların ne zaman gerçekleşeceği belirlenebilir. "Component Palet" menüsünde ise kullanılacak nesneler seçilebilir ve form üzerine yerleştirilebilir. Bu konu başlıkları proje içerisinde birçok kez kullanılacaktır.

Nesnelere isim verilirken, belirli bir standart kullanılır ise projeyi oluşturan kodları izlemek kolaylaşır. Oluşturulacak formların isimleri frm, buttonların isimleri btn, timerların isimleri tmr, edit'lerin isimleri edt ile başlaması gibi programcıya bağlı bir standart belirlenmelidir. İlk olarak yeni açılan form'un adı, object inspector-properties-name bölümünden "frm_ana" olarak değiştirilmelidir. Otomatik olarak "caption" yani başlık bölümü de aynı ismi alır. Böylece formun üzerindeki isim de aynı olur. Daha sonra kontrol kartının fonksiyonlarının tanımlı olduğu pci_8164.pas dosyası programın kayıtlı olduğu klasöre de kopyalanmalıdır ve ismini de programın uses bölümüne de eklenmelidir. Bu dosyanın içinde eksen kontrol kartını Delphi arabirimi ile yönetebilmek için gerekli fonksiyonlar vardır. Bu fonksiyonlar kartın içindeki elektronik devreleri doğrudan çalıştırdığı için üretici firmadan alınması zorunludur. Bu fonksiyonları yeniden oluşturmak için üretici firmanın yaptığı çalışmanın tekrar harcanmasını gerekmektedir ve bu çalışma farklı bir konudur. Ayrıca verilmiş olan fonksiyonlar projedeki ihtiyaçları karşılamaktadır. Dolayısı ile fonksiyonlar üretici firmanın sunduğu dosya ile kullanılmalıdır. Bu fonksiyonların sayısı çok fazla olduğundan, sadece projede kullanılacak fonksiyonlar araştırılacaktır. Fonksiyonların kullanma mantığı anlaşıldıktan sonra, bilinmeyen fonksiyonlarda kolaylıkla kullanılabilir. İlk olarak kullanılması gereken fonksiyon, delphi programın başlangıcında, eksen kontrol kartının program tarafından komut çalıştırılmasına açılması fonksiyonudur. Bu başlangıç komutu _8164_initial(var

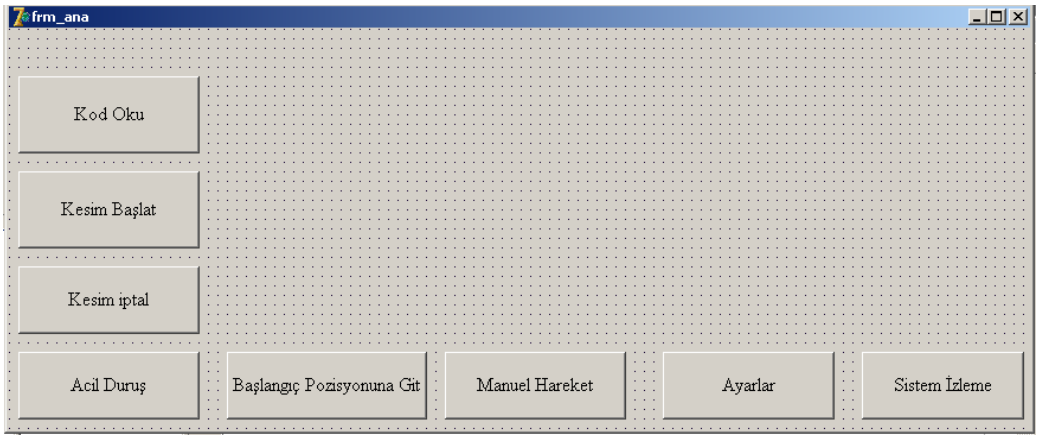
existcard:Smallint) çalıştırılmadan diğer komutlar da kontrol kartı üzerinde çalışmamaktadır (Fonksiyonun parantez içerisindeki değişkeni sıfır olarak tanımlanmış bir smallint değişkeni olmalıdır). Kodların programın açılışından sonra aktifleşmesi için, açılış fonksiyonu form'un açılışında yani Tfrm_ana.FormCreate prosedürünün içerisinde çalıştırılması gerekmektedir. Program kapatılırken de kontrol kartı program tarafından komut çalıştırılmasına kapatılmalıdır. Bu sonlandırma komutu _8164_close kullanılmadan program kapatılırsa, kartın enerjisi kesilmeden kontrol kartı tekrar kontrol edilemez. Bu yüzden bu özelligi formun kapatılırken yani Tfrm_ana.FormClose prosedürünün içerisinde çalıştırılmalıdır. Program açıldıktan sonra ilk olarak geri besleme fonsiyonunun aktifleştirilerek, kontrol kartının bulunduğu noktayı bulabilmesi ve kullanıcıya gösterebilmesi sağlanmalıdır. Geri besleme fonksiyonu _8164_set_feedback_src(eksen no:Smallint; Src:Smallint) fonksiyonu kullanılarak, kontrol kartının geri besleme özelliği açılabilir. Eksen no hangi eksenlerin geri besleme özelliğinin açılacağını belirler. Scr ise değiştirilecek komutu ifade eder, yani "(1,1)" yazıldığında ikinci eksenin geri besleme özelliğinin açılması anlamına gelir. Bu özelliği üç eksen için de açılması gerektiğinden "(0,1), (1,1), (2,1)" olarak ayrı ayrı tanımlanması gerekir. Bunlara bağlı olarak ana formun create ve close prosedürleri tablo 6.2.'deki gibi olmalıdır.

Tablo 6.2. Ana form'un create ve close prosedürleri

```
procedure Tfrm_ana.FormCreate(Sender: TObject);
var
  sifir: Smallint;
begin
  sifir:=0;
  _8164_initial(sifir);
  _8164_set_feedback_src(0,1) ;
  _8164_set_feedback_src(1,1);
  _8164_set_feedback_src(2,1);
end;
procedure Tfrm_ana.FormClose(Sender: TObject; var Action: TCloseAction);
begin
   _8164_close ;
end;
```

Ana form üzerinde component palet içerinden, kod oku, kesim başlat, kesim iptal, acil duruş, başlangıç pozisyonuna git, manuel hareket, ayarlar ve sistem izleme gibi fonksiyonları gerçekleştirmek için bu fonksiyonları içinde çalışacağı, buttonlar

oluşturulmalıdır. Bu butonlara isim olarak sırasıyla btn_oku, btn_basla_btn_iptal, btn_acil, btn_home, btn_manuel, btn_ayarlar, btn_izleme isimleri verilerek form, şekil 6.3.'deki gibi düzenlenmelidir.

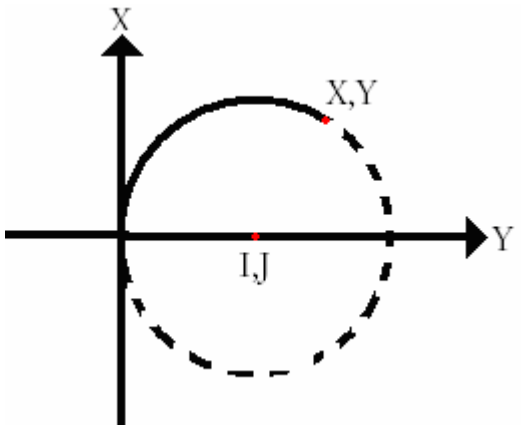


Şekil 6.3. Ana form'un gösterilişi

6.4.1.1. Gcode dizilerinin değişkenlere ayrılması

Makinenin SCADA yazılımında ilk yapılması gereken fonksiyon, Gcode algoritmasına göre hazırlanmış değişken dizisine göre, otomatik olarak çalışmasıdır. Otomatik çalışma içerisinde olacak bu fonksiyon kesme makinesinin yapacağı ana işlemdir. Kesilecek ölçüler daha önceden bir optimizasyon programına girilmekte ve optimizasyon programı en kısa yoldan ve en az kayıp ile ana parçanın küçük parçalara ayırılacağı yolu Gcode formatında vermektedir. Makinenin temel amacı da bu Gcode değişken listesini okuyup, değerlendirip, makinenin o yolu otomatik olarak takip etmesini sağlayarak, dolayısı ile büyük malzemeninde küçük parçalara ayılırmaktır. Bu fonksiyon dört ana temel fonksiyonu içermektedir. İlk olarak yazılı dosyanın Delphi programının içerisinde açılıp, okunması gerekmektedir. İkinci olarak okunan dosyada içerisindeki karakterlerden, gerekli olan nesneleri ayıklayarak Delphi içerisinde kullanılacak yazılım formatına dönüştürülmesi gerekmektedir. Üçüncü olarak bu ayıklanan nesnelerdeki değişkenleri, eksen kontrol kartının anlayacağı komutlara dönüştürülmesi gerekmektedir. Ve son olarak da oluşturulan komutların sıra ile kontrol kartının işlemesi için kontrol kartına gönderilmesi gerekmektedir.

İlk temel foksiyonu gerçekleştirmek için “Component Palet-dialogs” menüsününden “OpenDialog” nesnesinin programın ana formuna eklenmesi gerekmektedir. Bu nesnenin isminin daha sonradan kolay bulunabilmesi için opd olarak değiştirilmelidir. Bu fonksiyon ile istenen dosyayı açarak verilerin Delphi yazılımının içerisine girmesi sağlanır. Konuyu daha basit bir yöntem ile açıklamak için dört ana temel fonksiyon arasında kalın çizgiler çizmeden sonuca dayalı olarak programın akışı incelenecektir. Dosyadan okunması hedeflenen Gcode listesi üç temel formatta olabilir. İlk format G0X150Y100Z50F20000 dizisidir. Bu formatta G0 hareketinin çizgisel olduğunu belirtmek için, X harfi ve 150 sayısı gidilecek mesafenin X ekseninde 150 birim olduğunu, Y harfi ve 100 sayısı gidilecek mesafenin Y ekseninde 100 birim olduğunu, Z harfi ve 50 sayısı gidilecek mesafenin Z ekseninde 50 birim olduğunu, F ise hızın 20000 olduğunu gösterir. İkinci format ise G2X1800Y600I1000J0F5000 formatıdır. Bu formatta G2 hareketin dairesel olduğunu ve saat yönünde olduğunu belirtmek için, X harfi ve 1800 sayısı X ekseninde dairenin son noktasını, Y harfi ve 600 sayısı Y ekseninde dairenin son noktasını, I ve J harfleride dairenin sırasıyla X ve Y eksenlerindeki merkezlerini, F yine çizim hızının 5000 olduğunu belirtir. Bu formattaki örnek daire şekil 6.4’de gösterilmiştir.



Şekil 6.4. Dairenin gösterilmesi

Üçüncü format ise G3X1800Y600I1000J0F5000 formatıdır. Bu formatta bütün uzunluk ölçüleri ikinci format ile aynıdır fakat sadece dönüş yönü saat yönünün tersidir. Hızların değişken olmasının sebebi, sistemin mekanik olarak lineer hareketleri hızlı yapabilmesine karşılık, dairesel hareketlerde aynı kalitede kesim

elde edebilmek için hızın daha düşük olması gerekliliğidir. Lineer ve dairesel hareketler arasındaki hız oranı sistemin mekanik yapısı ve kontrol mekanizması tarafından belirlenir.

Üç formatta toplam 6 adet değişken mevcuttur. Bunlar sırası ile G, X, Y, Z, I, J, F'dir. Karakterlerden, gerekli olan nesneleri ayıklayarak Delphi içerisinde kullanılacak yazılım formatına dönüştürülmesi için kullanılabilecek ilk yöntem, kodları ve değerleri birbirinden ayırarak sıralı değişkenlere atama yöntemidir. Bu yöntemde Gcode satırı tamamen okunmalı, herzaman ikinci karakteri hareketin tipini belirlemek için kullanılmalı, X ve Y arasındaki değer X değişkeni olarak atanmalı, Y ve Z arasındaki değeri Y değişkeni olarak atanmalı ve diğerlerini de buna benzer şekilde atanmalıdır. Yöntemi kolaydır fakat değişkenlerin hep aynı sırada gelmesi gerekmektedir. Yani Y değişkeni ile X değişkeninin sırası değişir ise, programda hata oluşur. Piyasada kullanılan optimizasyon çıktılarında kısa hareketler için F hız değişkeni ilk olarak yazılmaktadır. Dolayısı ile harflerin tanımlanması konusunda bir standart oluşmuş olsada, yazım sırası bakımından bir standart yoktur. Yada bazı hareketler sadece tek bir ekseni içermektedir. Komut satırı farklı sırada geldiğinde fonksiyonun hata yapabilme ihtimali olduğundan sıralı değişkenlere atama yöntemi kullanılmamalıdır.

Kısıtlamaları ve hataları en aza indiren yöntem dizi yöntemidir. İlk olarak coor adında 6 değişkenli bir dizi tanımlanmalıdır. Daha sonra Gcode satırı okunmalıdır. Okunan satırda değişken aranmalıdır. Değişken bulduktan sonra bir sonraki değikene kadar aradaki sayı değeri bir iç değişkende tutulmalıdır. Örnek olarak, G0Y100X150Z50F20000 satırında, değişkenler okunmaya başlanmalıdır. İlk olarak G değişkeni bulunur, ikinci olarak Y değişkeni bulunur, üçüncü olarak X değişkeni bulunur, sonra Z ve F değişkenleri bulunur. Tanımlanmış olan coor adında altı değişkenli dizinin, G değişkeninin değeri için değişkeni dizinin sıfırıncı elemanına, Y değişkeninin değeri için dizinin ikinci elemanına, X değişkeninin değeri için dizinin birinci elemanına, Z değişkeninin değeri için dizinin üçüncü elemanına ve F değişkeninin değeri için de dizinin altıncı elemanı tahsis edilmelidir. Değişkenlerin dizide atanmış olduğu yerleri belirli olduğundan, değişkenlerin gelme sırası önemli olmayacaktır. Atama yapıldıktan sonra, satır istenildiği gibi ayıklanmış olacaktır.

Dizi yöntemini oluşturmak için ilk olarak coor değişkenini oluşturmak gerekmektedir. Değişkenin bütün formda tanımlı olması için global olarak tanımlanmalı ve "coor :array [0..6]of double;" kodu programın değişkenler bölümüne eklenmelidir. Bu dizide elemanların tahsis edildiği değişken anlamları, sırasıyla 0:G, 1:X, 2:Y, 3:Z, 4:I, 5:J, 6:F'tir. Gerekli dizi değişkeni oluşturulduktan sonra satırın karakterleri sırayla okumalı ve okunmuş olan değer aranan değişkenlerden biri değil ise yani bir sayı ise sabit bir değer atanmalı, eğer değişkenlerden biri ise değişkenin tipine göre eşlenigi olan değer atanmalıdır. Dizi yöntemine göre Gcode'un karakterleri için üretilmesi gereken sonuçlar tablo 6.3'de gösterilmiştir.

Tablo 6.3. Gcode satırına göre fonksiyonun vermesi gereken sonuçlar

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
Gcode	G	0	Y	1	0	0	X	1	5	0	Z	5	0	F	2	0	0	0	0
Sonuç	0	-1	2	-1	-1	-1	1	-1	-1	-1	3	-1	-1	6	-1	-1	-1	-1	-1

Bu sonuçlar her satır okunduğunda ayrı ayrı alınmalıdır. Ayrı ayrı her satırın değerlendirilmesi için iki yöntem vardır. Birinci yöntem, sonucu üreten formülün satır sayısı kadar yazılmasıdır. İkinci yöntem ise daha kısadır. Bir fonksiyon tanımlanmalıdır, her satır teker teker fonksiyona gönderilmelidir ve fonksiyon istenilen sonuçları üretmelidir. Böylece bir tek fonksiyon yazılarak bütün satırlar, aynı fonksiyonun içerisinde işlenebilir. Tablo 6.4'de yapılmış olan coor_yer fonksiyonu, char yani bir "karakter" alır, karakterin cinsine göre bir "integer" sonucu verir. Gcode satırları coor_yer fonksiyonuna karaktek karakter gönderilmelidir. Fonksiyonun başında sonuç "-1" yapılmalıdır. Fonksiyona gelen değer değişkenlerden biri ise sonuç değişkenin cinsine göre değiştirmelidir, değişkenlerden biri değil yani sayı ise sonuç "-1" olarak kalmalıdır. Sonuç gelen karakterin cinsine göre sürekli değişmektedir. Fonsiyona (ch:Char) değeri girmeli, integer değeri çıkmalıdır. Tablo 6.4'de istenilen çıkışı verebilecek bir fonsiyon tanımlanmıştır. Coor_yer fonksiyonun sonucuna göre result değerinin -1'den farklı olduğunda, coor dizisindeki indeks numarasını, -1 değerine eşit olduğunda değişkenler arasındaki sayılar ile de indeksin sayı değerini belirlemelidir.

Tablo 6.4. Gcode satırındaki karakterlere göre çıkış veren fonksiyon

```
Function Tfrm_ana.coor_yer(ch:Char):integer;
 begin
  Result := -1;
  if ch = 'G' then   Result := 0;
  if ch = 'X' then   Result := 1;
  if ch = 'Y' then   Result := 2;
  if ch = 'Z' then   Result := 3;
  if ch = 'I' then   Result := 4;
  if ch = 'J' then   Result := 5;
  if ch = 'F' then   Result := 6;
end;
```

Sayı ataması yapılmadan önce coor dizisinin içeriği sıfırlanmalıdır. Bunun için dizideki sıfırıncı ve beşinci elemanlar arasına sonsuz değeri atanmalıdır, altıncı değişkene de 20000 değeri atanmalıdır. Altıncı değişken hız değişkenidir. Komutlarda hız yazılmadığında, hızın sabit olarak 20000 olması için altıncı değişkene 20000 değeri atanmalıdır. Atanmış değişkenlere göre hareketin tipi, doğrultusu ve yönü için gerekli fonksiyonunda belirlenmesi gerekir. coor_yer içerisindeki işlem bir prosedür olarak tanımlanmalıdır. Fonksiyon ile prosedür arasında fark vardır. Fonsiyona veriler gönderildiğinde, bu verilere göre fonksiyon bir işlem yapar. Prosedüre veriler gönderildiğinde ise veriler sadece yapılmış olan algoritmaya göre analiz edilir. “camı kapat” komutu bir prosedürdür çünkü verilen komuta karşı bir dönüş yoktur, ama “bardağı ver” komutu bir fonksiyondur çünkü verilen komuta karşılığında bir nesne dönmüştür. Dizinin içinin sıfırlanma işleminde, sadece değişkelerin içeriği değiştirilmesi gerektiğinden, bunun karşılığında bir nesne beklenmediği için işlem bir prosedur olarak tanımlanmalıdır. Bu prosedürun isminin kolay hatırlanabilmesi için coor_reset olmalıdır. Değişkenleri sırayla yazılması yerine bir for döngüsü kullanılmalıdır. Bu döngüde i sırasıyla 0 ile 5 arasında değerler almalı, dizinin ilk altı elemanına sıra ile infinity yani sonsuz değeri atanmalıdır. Altıncı elemanına da 20000 değeri atanmalıdır. Resetleme algoritması tablo 6.5.’de gösterilmiştir.

Tablo 6.5. Coor dizisini resetleme prosedürü

```
procedure Tfrm_ana.coor_reset;
begin
   for i:=0 to 5 do
   coor[i] := Infinity;
   coor[6] := 20000;
end;
```

6.4.1.2. Satır ayıklama fonksiyonu

İkinci temel nokta olan satırları ayıklama proseduru için bir yöntem geliştirilmelidir. Bu prosedürü oluşturmak için prosedürün içinde 3 adet integer değişkeni ile bir adet string değişkeni oluşturulmalıdır. Değişken isimleri integer değişkenleri için sırayla index, tempIndex, ve i olabilir. String değişkeni için de tempStr ismi verilebilir. İşleme başlanmadan önce bu değişkenlerin tanımlanması gerekmektedir. Satır ayıkla prosedürü başlangıcında değişkenlerin ve memonun içerisinin sıfırlanması gerekir. Okunan satır boş ise veya M harfi ile başlıyor ise prosedurden çıkılmalıdır. Ara değişkenlerden index ve tempIndex değişkenlerine ilk değer olarak -1 başlangıç değerleri atanmalıdır. TempIndex değişkenine herhangi bir değer atanmadığında sonuç -1 olmalı, atandığında ise sonuç atanmış değişkenin değeri olmalıdır. Index değişkenine de, tempIndex'e değişken atanmış ise sadece atanan değişkenin değeri gelmelidir. Böylece TempIndex değişkeninde satırdaki karakterlere ait olan sayı değeri, Index değişkeninde ise satırdaki değişkenlere ait sayı değeri oluşturulabilir. Satır ayıklama prosedürünun algoritması tablo 6.6'da gösterilmiştir.

Tablo 6.6. Satır ayıkla prosedürü

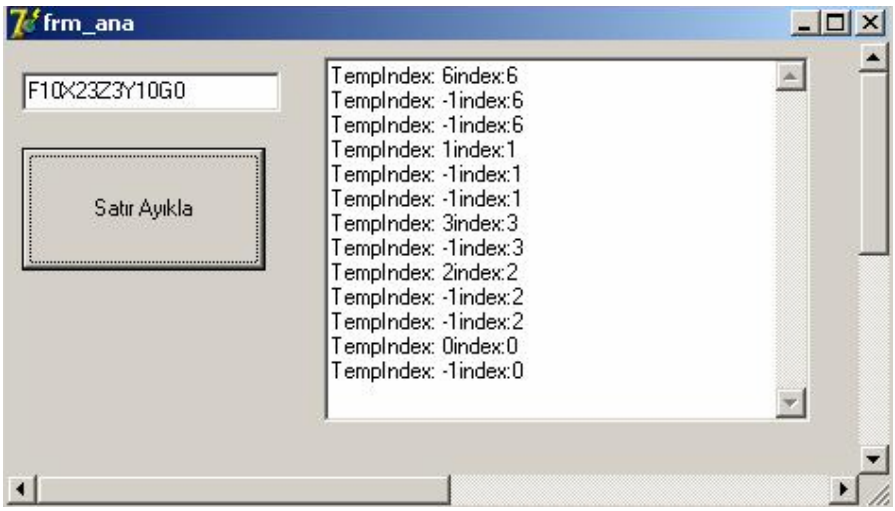
```
procedure Tfrm_ana.satir_ayikla(str:string);
var
  index,tempIndex ,i: integer;
  tempStr:string;

begin
  mmoDegisken.Lines.Clear;                    // memo'nu içini temizle
  coor_reset;                              // dizinin elemanlarını temizle
  if (trim(str)='') or (str[1]='M' )then exit;   // boşluk veya M varsa prosedürden çık
  index := -1;                                // başlangıç değerleri ata
  tempIndex := -1;                            // başlangıç değerleri ata
  tempStr := '';                              // başlangıç değerleri ata
 for i := 1 to length(str) do                 // 1'den satır sonuna kadar, bir döngü yap
  begin
    tempIndex := coor_yer(str[i]);            // str'nin i nolu elemanını numarasını bul
    if tempIndex <> -1 then                   // bu numara -1 den farklı ise
    index := tempIndex;                       // index'e yaz
    mmoDegisken.Lines.Add('TempIndex: '   // memo'da yazdır
     + IntToStr(tempIndex)+'index:' + IntToStr(index));
 end;
```

Satır ayıklama proseduru program içerisindeki en karmaşık bölümlerden biri olduğundan, satır ayıklama prosedurunun çalışma algoritmasını daha fazla

aydınlatmak için basit başka bir program kullanılabilir. Bu program için edit, buton ve memo kullanılması yeterlidir. Buton seçildiğinde tablo 6.7'daki gibi bir program oluşturulmalıdır. Programın edit bölümüne F10X23Z3Y10G0 gibi bir satır girildiğinde programın sonucu şekil 6.5'deki gibi olmalıdır.

Tablo 6.7. Satır ayıkla prosedurunu açıklayan algoritma

```
procedure Tfrm_ana.btn_satirClick(Sender: TObject);
begin
  satir_ayikla(edt_satir.Text);          // edit'ten gelen değeri, satir ayıklaya gönder
end;
```



Şekil 6.5. Satır ayıkla prosedurunu açıklayan programın sonucu

Tablo 6.6 porseduru ile diziyi parçalanabilir. Prosedurun sonucu -1 gelmediğinde, coor dizisinin hangi elemanının dolacağı belirlenmeli, -1 geldiğinde ise, tempStr iç değişkeni ilk olarak sıfırlandıktan sonra sayı değerleri arka arkaya içerisine yazılmasıdır.

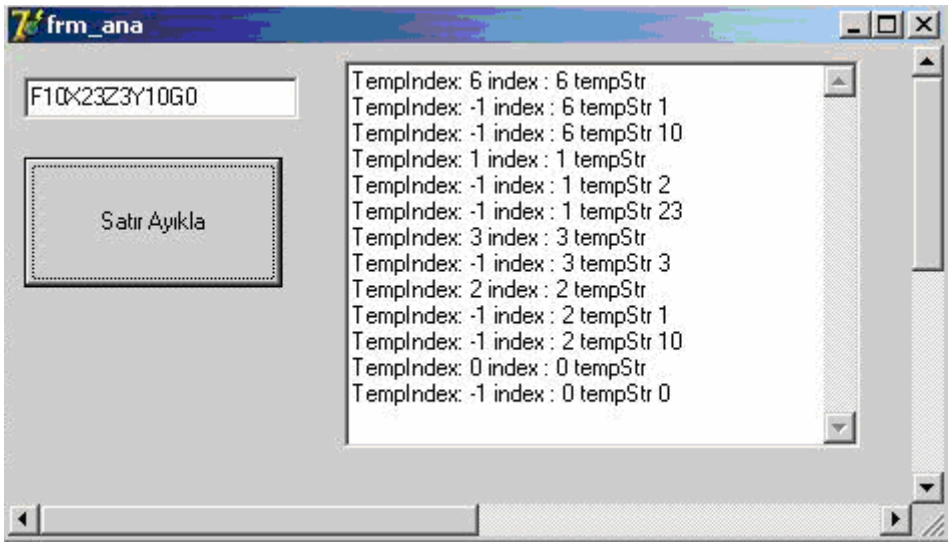
Tablo 6.8. Satır ayıkla fonksiyonunun geliştirilmiş algoritması

```
for i := 1 to length(str) do                //1'den satır sonuna kadar, bir döngü yap
 begin
  tempIndex := coor_yer(str[i]);            // kordinatlardan biri ise kordinat numarası gelecek,
                                            // değil ise -1 gelecek
    if tempIndex <> -1 then                 // konum bilgisi varsa, yada yenisi geldi ise
     begin
      if index <>-1 then                    // değişkenlerin kendisi geliyorsa
      coor[index] := StrToFloat(tempstr); // coor'un hangi elemanın dolacak
      tempStr := '';                        // ilk olarak, tempStr'yi boşalt
      index := tempIndex;                   // sadece değişlek adı değiştiğinde index'e yaz
```

Tablo 6.8. Devam

```
    end else                          // tempIndex = -1 ise, değişkenin sayı değeri ise
    tempStr := tempStr + str[i];      // sayı geldikçe, önceki sayı ile yenisini topla
end;
```

Satır ayıkla prosedurunun içine tablo 6.8'deki kodlar eklendiğinde programa Gcode satırı girildiğinde, tempStr'ye sırayla sayı değerleri atılır. Index'teki değişkeninin değeri değiştiğinde, tempStr'deki değer coor dizisine atılmalıdır. Sonra tempStr'nin içi boşaltılmalıdır. Düzenlenmiş programa F10X23Z3Y10G0 satırı girilip satır ayıkla tuşu seçildiğinde sonuç şekil 6.6'deki gibi olacaktır.



Şekil 6.6. Satır ayıkla fonksiyonunun geliştirilmiş sonucu

6.4.1.3. Dosyadan veri okuma fonksiyonu

Programın çalışma algoritmasının içerisinde, değişkenlerin bir satırdan değil, bir dosyadan okunması gerekir. Kullanıcı tarafından mekanizmanın sabit noktalara hareket edebilmesi için ayrıca bir manuel bölümü oluşturulmalıdır. Ana sayfada ise otomatik olarak çalışma fonksiyonları tanımlanmalıdır. Dosyayı Delphi programında açmak için form'a dialogs menüsünden "TopenDialog" eklenmelidir ve ismi opd yapılmalıdır. "kod oku" tuşuna basıldığında yapılması gerekli olan komutlar sırası ile, istenen dosya açılmalı, dosyanın ismi ve adresi başka bir fonksiyonun içerisindeki dosyaya atanmalı, atanmış olan dosyanın içi temizlenmeli, satırda eğer

boşluk varsa onlar silinmeli, okunmuş olan dosyadaki satırlar bir memo üzerinde kayıt edilmeli, satırlarda bir hata olma ihtimaline karşı satırlar satır ayıkla prosedurune gönderilmeli, eğer veriler eksik veya hatalı girilmiş ise hata mesajı üretilmeli ve değişkenler sıfırlanarak algoritmadan çıkılmalı, eğer hata yok ise dosya kapatılmalı ve coor değişkeninin içeriği sıfırlanmalıdır. Satırlardan G2 veya G3 komutlarından biri geldiğinde aynı satırda X, Y, I ve J değişkenlerine de sayı değeri bulunmalıdır, aksi takdirde komut çalışamaz. Programın algoritması içerisinde bir hata oluştuğunda algoritmanın durmaması, kullanıcıya bir hata mesajı gönderilmesi ve değişkenlerin içeriği sıfırlanması için try-execpt komutları kullanılmıştır. Algoritmanın içeriği tablo 6.9'daki gibi olabilir.

Tablo 6.9. Gcode'larının dosyadan okunması için üretilmiş algoritma

```
procedure Tfrm_ana.btn_okuClick(Sender: TObject);
var
 f:TextFile;                                  // yeni text dosyası değişkenim
 str:String;                                  // yeni string değişkenim
begin
 try                                          // hata oluşursa programı durdurma
  mmoGCodes.Lines.Clear ;                     // önce memo'nun içini boşalt
  mmoDegisken.Lines.Clear;                    // önce memo'nun içini boşalt
  if opd.Execute then                         // dosya aç
   begin
     AssignFile(f, opd.FileName);             // dosya ismini f'e gönder
     Reset(f);                                // f dosyasını içini temizle
    while not Eof(f) do                       // f'nin sonuna kadar yap
     begin
     Readln(f,str);                           // f dosyasını oku, satırını str ye gönder
     str:=Trim(str);                          // str'nin içindeki boşlukları sil
     mmoGCodes.Lines.Add(str);                // deneme amaçlı str'yi satır ayıklaya gönder
     satir_ayikla(str);
     if (coor[0] = 3) or (coor[0] = 2) then   // eğer daire fonksiyonu yazıldı ise
     if (coor[1] = Infinity)                  // X değeri girilmemiş ise
      or (coor[2] = Infinity)                 // Y değeri girilmemiş ise
      or (coor[4] = Infinity)                 // Z değeri girilmemiş ise
      or (coor[5] = Infinity) then            // I değeri girilmemiş ise
       begin
        mmoGCodes.Lines.Clear;                //memonu satırlarını temizle
        coor_reset;                           // coor değişkenini resetle
        CloseFile(f);                         // f dosyasını kapat
        ShowMessage('Dosya Okunamadı !');     // hata mesajı göster
        exit;                                 // procedure'den çık
       end;
     end;                                     // eğer hata yoksa
   CloseFile(f);                              // dosyayı kapat
   coor_reset;                                // coor'u resetle
   end;
 except
   begin                                      // okurken sorun oluştu ise
    mmoGCodes.Lines.Clear;                    // memonun içini temizle
```

Tablo 6.9. Devam

```
      coor_reset;                                  // coor değişkenini resetle
      CloseFile(f);                                // dosyayı kapat
      ShowMessage('Dosya Okunamadı !')     // hata mesajı göster
      end;
    end;
  end;
```

6.4.1.4. Verilerin hareket komutlarına dönüştürülmesi

Satır ayıkla ve dosyadan okuma algoritmaları ile veriler dosyada okunabilmekte ve istenilen formata dönüştürülebilmektedir. Okunan veriler komut haline getirilmeli ve sıra ile kontrol kartına gönderilerek ardışık hareketler oluşturulmalıdır. Ardışık hareketleri oluşturabilmek için procedür tanımlanmalıdır ve prosedürün içeriğinde, G değişkeninin tipine göre, kontrol kartına gönderilecek hareket komutu seçilmelidir. Gerçekleştirilmesi hedeflenen hareketler interpolarizasyon içeren lineer hareketler veya dairesel hareketlerdir. Tanımlanacak olan hareketlerin tipi, yazılıma yapılacak ek ile çoğaltılabilir. Eğer hedeflenen hareketler lineer ise, hareketin doğrultusununda bulunması gereklidir. XYZ düzleminde XYZ, XY, XZ, YZ, X, Y, Z olmak üzere 7 farklı doğrultu vardır. Hareketin doğrultusu, değişkenlerin değerlerine yapılmış atamaların incelenmesi ile bulunabilir. Değişkene değer atanmış ise yani coor dizisindeki değişenin değeri infinity'den farklı ise değişken hareket doğrultusunun içerisine alınmalıdır. Eğer coor dizisindeki değişkenin değeri infinity ise hareketin doğrultusunun içerisine alınmamalıdır. X100Y100 Gcode satırı için, X ve Y değişkenlerinin değerleri 100 olarak atanmalıdır, Z değişkeninin değerine ise infinity atanmalıdır. Kontrol kartına komut gönderilmeden önce, XYZ eksenlerinin değerleri araştırılmalıdır. Z ekseninin atanmış olan değeri infinity olduğundan hareketin doğrultusu XY ekseninde olmalıdır. Hareketin gideceği açı kontrol kartı tarafından belirlenmelidir. Motoru kontrol eden sürücülere kontrol kartı üzerinden gönderilen pulse sayısı ile, motorların yapılması istenilen hareket belirlenir. Kontrol kartı, hareket etmesi istenilen motorun sürücüsüne 8196 pulse gönderdiğinde, motor bir tur dönmelidir. Motorlar bir tur ilerlediğinde, X ve Y eksenlerinde mekanizma 15.94 mm ilerler, Z ekseni ise 72 derece döner. Pulse sayısı, motorun yapılması istenen hareket ile orantılı gönderilir ise, mekanizmada istenilen hareketler elde edilir. Mekaniksel değerler her mekanizma için, değişik mekanik tasarımlara göre farklı

olabilir. Mekanizmanın değişmesi halinde sadece motorun bir turundaki hareket miktarı değişecektir. Motorun istenilen hareketi yapabilmesi için pulse miktarı ile motorun bir turunda aldığı yol arasında tablo 6.10'daki gibi bir algoritma üretmek gereklidir. Böylece veriler programa mm cinsinden girilebilir ve algoritma da kontrol kartına gerekli pulse miktarını gönderebilir.

Tablo 6.10. İstenilen hareketi oluşturmak için mm değeri pulse sayısına dönüştüren algoritma

```
function Tfrm_ana.mmToPulseX(pulse:Double): Double;   // X katsayısı
 begin
  result:= (pulse * 8196)/ 15.94                      // X puls değerini bul
 end;

function Tfrm_ana.mmToPulseY(pulse:Double): Double;   // Y katsayısı
 begin
  result:= (pulse * 8196)/ 15.94                      // Y puls değerini bul
 end;

function Tfrm_ana.mmToPulseZ(pulse:Double): Double;   // Z katsayısı
 begin
  result:= (pulse * 8196)/ 72                        // Z puls değerini bul
 end;
```

Programdan girilen Gcode satırları değişkenlerine ayırıldıktan sonra, mekanizmanın istenilen noktaya hareket etmesi için kontrol kartına gerekli komutlar gönderilmelidir. Komut dizisini üretecek bir prosedür tanımlanmıştır ve prosedüre fonksiyon üret adı verilmiştir. Prosedur işlemine başlamadan önce X, Y, Z, I, J değişkenleri için programa girilmiş olan mm değerlerinin pulse sayılarına çevirilmesi için kullanılacak, prosedur içinde kullanılacak değişkenler tanımlanmıştır. Kontrol kartı komutlarında eksenler dizi formatında tanımlandığından, algoritma içinde kullanılacak bir dizi değişkeni tanımlanmıştır. Prosedurde yapılması gereken ilk işlem, programda girilmiş olan mm değerlerinin, pulse karşılığına çevirilmesidir. Fonksiyonların ana şeçim koşulu G değişkenine atanmış olan değerdir. G değişkenine sıfır veya bir atanması durumunda mekanizmanın lineer, iki veya üç değerleri atandığında mekanizmanın dairesel hareket etmesi gereklidir. G değişkeninin değerine göre komut seçimi için case of döngüsü kullanılabilir. G değeri sıfır veya bir ise, Gcode satırı üzerinde infinity değerine eşit olmayan X, Y, Z değişkenleri aranmalıdır. Oluşabilecek yedi farklı ihtimale karşılık, yedi farklı komut seçimi if döngüsü kullanılarak seçilebilir. Algoritmanın içerisinde if döngüsü ile seçilen hedef noktası, kontrol kartına komut olarak gönderilir. Eğer G değeri 2 ise

saat yönünde, üç ise saat yönünün tersine daire çizme komutu kontrol kartına gönderilmelidir. İstenilen komut kontrol kartına gönderildikten sonra case döngüsü bitirilmelidir. İstenilen fonksiyonu üretebilecek algoritma tablo 6.11'deki gibi olabilir. Kontrol kartına komutlar adışık olarak gönderildiğinde istenilen hareketler ardışık olarak tamamlanabilir.

Tablo 6.11. Kontrol katına komut oluşturma algoritması

```
procedure Tfrm_ana.fonksiyon_uret;

var
 X_hedef,Y_hedef,Z_hedef: Double;
 I_hedef,J_hedef: Double;
 ax_ar:array [1..3] of Smallint;

begin
  X_hedef:= mmToPulseX(coor[1]);          // coor'daki değeri pulse sayısına çevir
  Y_hedef:= mmToPulseX(coor[2]);
  Z_hedef:= mmToPulseX(coor[3]);
  I_hedef:= mmToPulseX(coor[4]);
  J_hedef:= mmToPulseX(coor[5]);

case Round(coor[0]) of                        // fonksiyonu seç
   0,1:begin                                  // G0,G1 tanımlandı ise
      if (coor[1] <> Infinity) and (coor[2] <> Infinity) and (coor[3] <> Infinity) then begin
                                              //  XYZ tanımlandı ise
        ax_ar[1] := 0;
        ax_ar[2] := 1;
        ax_ar[3] := 2;
        _8164_start_ta_line3(0,ax_ar[1],X_hedef,Y_hedef,Z_hedef,0,coor[6],0.5,0.5)
     end else                                 // hareket sadece XY ekseninde, Z tanımsız
  if (coor[1]<>Infinity) and (coor[2]<>Infinity) and (coor[3]=Infinity)  then
   _8164_start_ta_move_xy(0,X_hedef,Y_hedef,0,coor[6],0.5,0.5)
  else                                        // hareket sadece YZ ekseninde, X tanımsız
  if (coor[2] <> Infinity) and (coor[3] <> Infinity) and (coor[1]=Infinity) then
   begin
    ax_ar[1] := 1;
    ax_ar[2] := 2;
    _8164_start_ta_line2(0,ax_ar[1],Y_hedef,Z_hedef,0,coor[6],0.5,0.5);
   end else
  if (coor[1] <> Infinity) and (coor[3] <> Infinity) and (coor[2]=Infinity)then
   begin                                      // hareket sadece XZ ekseninde, Y tanımsız
    ax_ar[1] := 0;
    ax_ar[2] := 2;
    _8164_start_ta_line2(0,ax_ar[1],X_hedef,Z_hedef,0,coor[6],0.5,0.5);
   end else                                   // hareket sadece X ekseninde, YZ tanımsız
  if  (coor[1] <> Infinity) and (coor[2]=Infinity) and (coor[3]=Infinity) then
   _8164_start_ta_move(0,X_hedef,0,coor[6],0.5,0.5)
  else                                        // hareket sadece Y ekseninde, XZ tanımsız
  if  (coor[2] <> Infinity) and (coor[1]=Infinity) and (coor[3]=Infinity) then
   _8164_start_ta_move(1,Y_hedef,0,coor[6],0.5,0.5)
  else                                        // hareket sadece Z ekseninde, XY tanımsız
  if  (coor[3] <> Infinity) and (coor[1]=Infinity) and (coor[2]=Infinity) then
      _8164_start_ta_move(2,Z_hedef,0,coor[6],0.5,0.5)
```

Tablo 6.11. Devam

```
      else;
      end;
    2: _8164_start_ta_arc_xy(0,I_hedef,J_hedef,X_hedef,Y_hedef,0,0,coor[6],0.5,0.5) ;
    3: _8164_start_ta_arc_xy(0,I_hedef,J_hedef,X_hedef,Y_hedef,1,0,coor[6],0.5,0.5) ;
      end;                                    // case işlemini bitir
end;
```

Tablo 6.11'daki algoritma ile değişken değerlerine göre istenilen fonksiyonlar üretilebilir. Sistemin otomatik çalışma modunda ardışık hareketler yapmalıdır. Ardışık hareketlerin yapılabilmesi için bütün algoritmaların kullanılması gerekmektedir. Ardışık hareketlerin oluşturulabilmesi için bir timer tanımlanarak, belirli zaman aralıklarında kontrol kartının içindeki registerlerin boş olup olmadığı kontrol edilmelidir. Registerler boş olduğunda kontrol kartına yeni bir hareket komutu gönderilmelidir. Yeni satır gönderildiğinde iç değişkenin sayısı bir arttırılmalı ve böylece bir sonraki adımda hangi satırın gönderileceği belirlenmelidir. Gönderilecek satırın sayısını içeren değişken, başka fonksiyonlar tarafından değiştirilebilmesi için global olarak tanımlanmalıdır ve başlangıç değerine -1 değeri atanmaldır. Satır sayısını içeren değişken memodaki satır sayısına eşit olana kadar, sırası gelen satır, satır ayıkla prosedurune gönderilmeli, ekranda yazdırılmalı ve fonsiyon üret prosedürüne gönderilerek komut üretilmelidir, eşit olduğunda timer durdurulmalı ve değişkene başlangıç değeri atanmalıdır. Prosedurun başında dosyanın açılıp memoya gönderildiği kontrol edilmelidir. Satırların ardışık olarak gönderilmesi için tablo 6.12'deki gibi bir algoritma oluşturulabilir.

Tablo 6.12. Satırların ardışık çalışmasını için gerekli algoritma

```
procedure Tfrm_ana.tmrKesimTimer(Sender: TObject);

begin
  if (mmoGCodes.Lines.Count <=0 ) then     // memo'da birşeyler yazılı değil ise
   begin
    tmrKesim.Enabled := False;             // timer'ı durdur
    exit;                                  // algoritmadan çık
   end;
  if (_8164_check_continuous_buffer(0)=0)  // kontrol kartında X,Y,Z bölümleri boş ise
    and (_8164_check_continuous_buffer(1)=0)
    and (_8164_check_continuous_buffer(2)=0) then

    begin
     satir := satir +1;                    // satır değişkenine bir ekle
    if satir = mmoGCodes.Lines.Count then  // satır değişken sayısı memodaki satırdan fazla ise
     begin
```

Tablo 6.12. Devam

```
      satir := -1;                          // Satıların tümü işlendi, satır sayısına başlangıç değeri ata
     tmrKesim.Enabled := False;          // timer'ı kapat

    end;                                          // satır numarası gelen satırı, satır ayıklaya gönder
   satir_ayikla(mmoGCodes.Lines.Strings[satir]);
                                                  // anlik satiri Label'e yaz
   lbl_anlik_satir.Caption:= mmoGCodes.Lines.Strings[satir];
   if coor[0] <> Infinity then                    // komut göndermeyi G tanımlıysa yap
   fonksiyon_uret;                                // kontrol kartı için komut üret
  end;
end;
```

Sistemin otomatik olarak Gcode satırlarlarını işletebilmesi için kesim başla tuşu ile sadece timer aktifleştirilmiştir. Timeri aktifleştirmek için gerekli algoritma tablo 6.13.'deki gibidir.

Tablo 6.13. Timer'ın çalıştırma algoritması

```
procedure Tfrm_ana.btn_baslaClick(Sender: TObject);
begin
   tmrKesim.Enabled := True;
end;
```

Otomatik kesim işleminin durdurulması için ise sadece timer'ı durdurmak yeterlidir ve durdurma işlemi "kesim durdur" tuşu seçilince çalışmalıdır. Algoritma, ürün kalitesinin korunması için, istenilen hareketi tamamladıktan sonra durmalıdır. Sistem kesme işleminin ortasında durur ise, duruş ve kalkış zamanlarından oluşan hız farkı sebebi ile kesim işleminin kalitesi bozulabilir. Timerı durdurmak için gerekli algorima tablo 6.14.'deki gibi olabilir.

Tablo 6.14. Timer'ın durdurma algoritması

```
procedure Tfrm_ana.btn_durdurClick(Sender: TObject);
begin
        tmrKesim.Enabled := False;
end;
```

Otomatik kesme işleminde, kesim iptal tuşu seçildiğinde, otomatik kesme işleminin iptal edilmesi için bir algoritma oluşturulmalıdır. Gerekli algoritmada, timer ve motorların hareketi durmalı, memonun satırları temizlenmeli ve satır sayıcıya başlangıç değerleri atanmalıdır. Algoritma tablo 6.15'deki gibi olabilir.

Tablo 6.15. Kesme iptal algoritması

```
procedure Tfrm_ana.btn_iptalClick(Sender: TObject);
begin
  tmrKesim.Enabled:= false;
  _8164_sd_stop(0,0.5);
  _8164_sd_stop(1,0.5);
  _8164_sd_stop(2,0.5);
  mmoGCodes.Lines.Clear;
  satir := -1;
end;
```

Acil bir durumda, acil duruş tuşuna basıldığında, motorların herhangi bir koşula bakılmadan durdurulması gerekmektedir. Acil duruş algoritması içerisinde timer durmalı, satır sayısına bir önceki değeri atanmalı, XYZ eksenlerinden hareket eden eksenler bulunmalı ve hareket eden eksene dur komutu gönderilmelidir. Acil duruş algoritması tablo 6.16'daki gibi olabilir.

Tablo 6.16. Acil duruş algoritması

```
procedure Tfrm_ana.btn_acilClick(Sender: TObject);
begin
  tmrKesim.Enabled := False;                    // otomatik işlem timerını durdur
  satir := satir -1 ;                           // satir değişkenine bir önceki değeri ata
  if  _8164_check_continuous_buffer(0)<> 0 then
    _8164_sd_stop(0,0.5);                       // X ekseni çalışıyorsa, durdur
  if  (_8164_check_continuous_buffer(0)= 0) and
   (_8164_check_continuous_buffer(1)<> 0) then
   _8164_sd_stop(1,0.5);                        // Y ekseni çalışıyorsa, durdur
  if (_8164_check_continuous_buffer(0)= 0) and
   (_8164_check_continuous_buffer(1)= 0) and
   (_8164_check_continuous_buffer(2)<> 0) then
    _8164_sd_stop(2,0.5);                       // Z ekseni çalışıyorsa, durdur
end;
```

6.4.1.5. Anlık pozisyonların gösterilmesi ve home işlemi

Mekanizmanın durduğu pozisyon sistemin kontrolünün ve izlenmesinin kolaylaştırılması için kullanıcıya mm cinsinden bildirilmelidir. Anlık pozisyonlar ana sayfa üzerinde gösterilmelidir. Ana sayfa üzerinde üç adet label tanımlanabilir ve isimlerine sırası ile lbl_x_pos, lbl_y_pos ve lbl_z_pos verilebilir. Gerçek pozisyonlar bir algoritma ile bu labellere atanmalıdır. Kontrol kartı anlık pozisyonları pulse sayısı olarak tutmaktadır. Kontrol kartından okunacak pulse değeri, kullanıcı menüsünde mm cinsine çevirebilmesi için bir fonksiyon tanımlanmalıdır ismi kolay hatılanabilmesi için pulseToMm olmalıdır. Üç eksenin anlık değerleri ayrı ayrı

okunması gerektiğinden, bu fonksiyonlar X, Y, Z eksenleri için farklı olarak tanımlanmalıdır. 8196 pulse, X ve Y eksenlerindeki motorların bir tur dönüşüne ve mekanizmanın 15.94 mm ilerlemesine eşittir. Okunan pulse değerinden mm cinsinden anlık değeri bulabilmek için, okunan değer 15.94 ile çarpılamalı ve 8196 değerine bölünmelidir. Z ekseni ise 72 derece ilerlediğinden, okunan değer 72 ile çarpılmalı ve 8196 değerine bölünmelidir. Algoritma tablo 6.17'deki gibi olabilir.

Tablo 6.17. Okunan pulse değerinin mm cinsine çevirilmesi

```
function Tfrm_ana.pulseToMmX(pulse:Double): Double;  // X anlık değeri
 begin
 result:= (pulse * 15.94)/8196                          // X anlık değerini mm cinsinden bul
 end;

 function Tfrm_ana.pulseToMmY(pulse:Double): Double;// Y anlık değeri
 begin
 result:= (pulse * 15.94)/ 8196                         // Y anlık değerini mm cinsinden bul
 end;

 function Tfrm_ana.pulseToMmZ(pulse:Double): Double; // Z anlık değeri
 begin
 result:= (pulse * 72)/ 8196                            // Z anlık değerini derece cinsinden bul
 end;
```

Kontrol kartından okunan pulse değerleri mm cinsine çevirildikten sonra, bir timer tanımlanmalı ve bu timer her 100 ms'de bir kontrol kartı üzerindeki anlık değer sayısını okuma komutu üretmeli ve eksenlerin anlık pozisyonlarını göstermek için tanımlanmış olan labellere değerleri yazmalıdır. Kullanılabilecek prosedür tablo 6.18'deki gibi olabilir.

Tablo 6.18. Anlık pozisyon değerlerinin ekranda gösterilmesi

```
procedure Tfrm_ana.tmr_posTimer(Sender: TObject);
var
  anlik_deger,gercekDeger: Double;
begin
  _8164_get_position(0,anlik_deger);               // karttan anlık X değerini oku
  gercekDeger:= pulseToMmX(anlik_deger);      // pulse'i mm'ye çevir
  lbl_x_pos.Caption:= FloatToStr(gercekDeger); // label'de göster

  _8164_get_position(1,anlik_deger);               // karttan anlık Y değerini oku
  gercekDeger:= pulseToMmY(anlik_deger);       / pulse'i mm'ye çevir
  lbl_y_pos.Caption:= FloatToStr(gercekDeger); // label'de göster

  _8164_get_position(2,anlik_deger);               // karttan anlık Z değerini oku
  gercekDeger:= pulseToMmZ(anlik_deger);           // pulse'i dereceye çevir
  lbl_z_pos.Caption:= FloatToStr(gercekDeger);     // label'de göster
end;
```

Sistem çalıştırılmadan önce home işlemi yapılmalıdır. Home işlemi için mekanizmanın başlangıç noktası bir yöne hareket edilerek aranmalı, başlangıç noktası bir sinyal ile algılandığında, kontrol kartı içerisindeki pozisyon sayıcıları sıfırlanmalıdır. Home işlemi sonucunda, mekanizma ile kontrol kartının sayıcı değerleri arasıda fark olmayacağından, mekanizma aynı desenleri hatasız yapabilir. Home fonksiyonu, makineye enerji verildiğinde veya acil duşur devresi çalıştırıldıktan sonra, bir kez çalışmalı, işlem bittikten sonra birdaha enerji kesilmesine veya pano üzerindeki acil duruş tuşuna basılana kadar birdaha çalışmamalıdır. Mekanizmanın başlangıç noktası araştırılmadan önce, başlangıç noktasının hangi yönde aranacağı ve başlangıç noktası bulunduktan sonra mekanizmanın yapacağı hareketin belirlenmesi gerekmektedir. Form'a ayrıca trm_home adında bir timer daha eklenmeli ve bu timer, hareket başladıktan sonra belirli zaman aralıkları ile kontrol kartının bufferlarını kontrol etmeli, bufferlar boşaldığında ise home işleminin tamamlandığını belirten bir değişkenin değerini değiştirmelidir. Bufferları kontrol eden algoritma tablo 6.19'daki gibi olabilir.

Tablo 6.19. Bufferları kontrol eden algoritma

```
procedure Tfrm_ana.trm_homeTimer(Sender: TObject);
begin
  if (_8164_check_continuous_buffer(0)=0)       // kontrol kartında X,Y,Z bufferları boş ise
    and (_8164_check_continuous_buffer(1)=0)
    and (_8164_check_continuous_buffer(2)=0) then
    home_ok:= true;                             // home ok sinyali ver
end;
```

Tablo 6.20. Home işlemini gerçekleştirebilecek algoritma

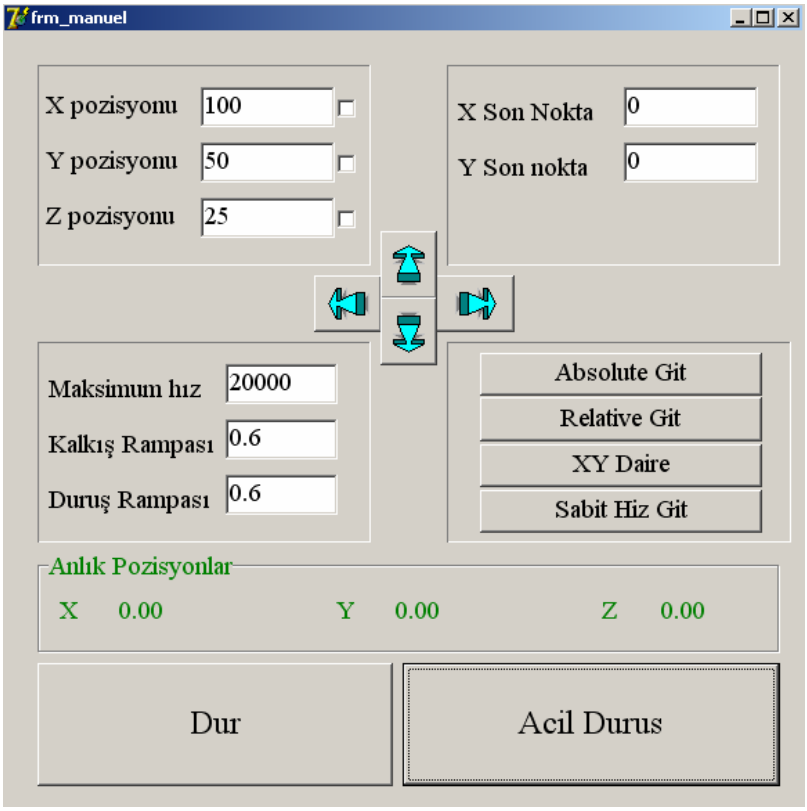
```
procedure Tfrm_ana.btn_homeClick(Sender: TObject);
begin
   _8164_set_home_config(0,1,0,0,0,0);       // X ekseni home tipini belirle
   _8164_set_home_config(1,1,0,0,0,0);       // Y ekseni home tipini belirle
   _8164_set_home_config(2,1,0,0,0,0);       // Z ekseni home tipini belirle
   _8164_home_move(0,0,-2000,0.1);           // X ekseni home yap
   _8164_home_move(1,0,-2000,0.1);           // Y ekseni home yap
   _8164_home_move(2,0,-2000,0.1);           // Z ekseni home yap
   trm_home.Enabled:=true;
end;
```

Home işleminin başlatılabilmesi için ana ekrana, başlangıç pozisyonuna git isimli bir tuş eklenmeli, bu tuş seçildiğinde home işleminin tipi belirlenmeli ve eksenlere başlangıç noktalarını araması komutu gönderilmelidir. Home işleminin

gerçekleştiğinin belirlenebilmesi için ise home işlemi için atanmış olan timerın çalıştırılması gerekmektedir. Home işleminin gerçekleştirebilecek algoritma tablo 6.20'deki gibi oluşturulabilir.

6.4.1.6. Manuel hareket sistemi

Mekanizma otomatik çalışma modunda ardışık hareketleri yapabildiği gibi, manuel çalışma modunda sabit hareketleri yapabilebilmelidir. Mekanizma istenilen noktalara hareket relative ve absolute hareket modlarında hareket ettirilebilmelidir. Sistemdeki X ve Y eksenleri ayrıca, belirli tuşlar seçildiğinde artı veya eksi yöndeki sabit hareketleri sağlanmalıdır. Bütün bu hareketler için, hareketin hızı ve rampaları seçilebilmelidir. Hareketler sonucunda mekanizmanın, pozisyonunun bulunduğu nokta kullanıcıya belirtilmelidir. Gerekli olan hareketler otomatik hareket menüsünden farklı menüde, manuel işlemler için özel hazırlanmış frm_manuel isimli bir formda tanımlanmalıdır. Manuel formunun tasarımı şekil 6.7'deki, içeriği tablo A.1.'deki gibi olabilir.

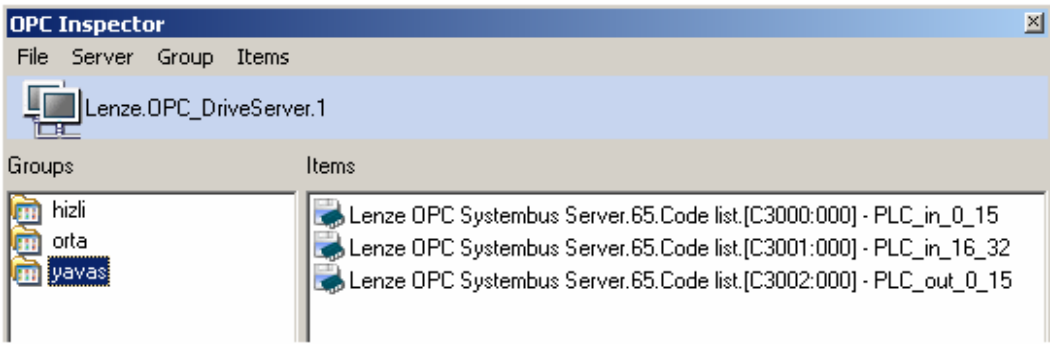


Şekil 6.7. Manuel formunun gösterilişi

6.4.1.7. OPC aracılığı ile haberleşme sistemi

Kontrol sisteminin hareket algoritmaları oluşturulduktan sonra, Delphi ile sistemi kontrol etmek için, Delphi programı, elektronik ortam ve hız kontrollörleri arasında bir haberleşme alt yapsı oluşturulması gerekmektedir. PLC kontrol ünitesi ve motor sürücülerinden Can bus aracılığı ile aldığı bilgileri, Delphi programı aracılığı ile bilgisayar ekranında gösterilmelidir. Delphi ile elektronik cihazlar arasında haberleşmenin gerçekleştirilebilmesi için Can Bus protokolünü tamamen Delphi içerisinde oluşturularak, Can Bus – Usb çeviricinin çalışması için gerekli algoritmanın yeniden yazılması, yöntemlerden biriridir. Sadece haberleşme protokolünün içeriğini oluşturmak bile çok fazla kod gerektirirken, protokolün tekrar oluşturulması Delphi programında aşırı derecede fazla kod yazmayı gerektirir. Bu nedenle sistemi oluşturuken bu işlemleri yapan aracı programların kullanılması sistemi ve Delphi yazılımını daha kolaylaştırır. Bu ara yazılımlardan PLC ve sürücülerin Can Bus Usb ile haberleşmesi için OPC programı, kullanılabilecek en basit programlardandır. Elektronik cihazları üretici firma olan Lenze'nin OPC Drive Server programı kullanılarak sistemdeki haberleşme sağlanabilir. Bu program kendi başına Can bus ile bilgisayarı haberleştirebilmektedir ve cihazların içerisindeki bütün kodların Delphi programından okunabilmesini ve yazılabilmesini sağlamaktadır. Bu projede, haberleşme sisteminin amacı, sadece belirlenmiş olan kodların okunması ve yazılması olduğundan, OPC programının kullanılması uygundur. Ayrıca cihazlar içerisindeki kod yığınlarından ihtiyaç olanların haberleşme sisteminde seçilebilmesi için Delphi yazılımında da bir OPC arabirimi kullanılmalıdır. Lenze'nin OPC yazılımı ile Delphi yazılımını haberleştirecek olan OPC programı component olarak Delphi'ye eklenebilir. Bu componentlerden en kolayı KASSL DOPC'dir ve bu componentin çalışması incelenecektir. Bu component Delphi'ye eklendikten sonra, diğer menülerin yanına, componentin oluşturduğu OPC menüsü, Delphi programına eklenir. Bu menünün içinde bulunan OPCServer nesnesini form'a eklenerek haberleşme sağlanabilir. Bu nesneye Lenze'nin OPC serverininde bulunan kodlar, grupların içinde, item olarak eklenir ve item'ların grup sıralarına göre Delphi programının içerisinde değerlendirilir. Oluşturulması hedeflenen Delphi programına da OPC menüsünden dOPCServer nesnesi eklenmiştir. Bu nesnenin üzerine çift tıklandığında OPC inspector menüsü açılır. Bu menünün sol tarafından, OPC'ye grup

eklenir. Burada grup oluşturmak okunacak nesnelerin okuma hızını belirlemek için gereklidir. Farklı gruplar ile farklı okuma hızları kullanılabilir. PLC üzerindeki dijital giriş ve çıkışların ekranda göstergerilmesi hızlı olması gereken bir fonksiyon değildir ama ekran üzerinden PLC'ye gönderilecek komutların programın çalışma hızını doğrudan etkilediğinden hızlı olması gerekmektedir. Farklı hızlarda haberleşecek kodların farklı gruplar içerisinde tanımlanarak, farklı haberleşme hızları ile haberleşmesi sağlanabilir. Sistem için hizli, orta ve yavaş grupları tanımlanabilir ve haberleşme zamanı olarak 100ms, 500ms ve 1000ms tanımlanabilir. Kesme makinesi için tanımlanmış gruplar ve yavaş grubunda tanımlanmış olan kodlar şekil 6.8'de gösterilmiştir.



Şekil 6.8. OPC inspector menüsünde tanımlama

Delphi ile yapılması hedeflenen SCADA yazılımında, PLC üzerindeki giriş ve çıkışlar bilgisayar üzerinde gösterilmelidir. Bu gösterge dijital sinyallerin çalışmasına uygun olarak, kontrol edilen noktada sinyal olduğunda bir yanan bir lamba resmi, sinyal olmadığında sönmüş bir lamba resmi ile gösterilebilir. PLC üzerinde 22 dijital giriş ve 16 dijital çıkış bulunmaktadır. Haberleşme için her bir giriş ve çıkış için bir item tanımlanması durumunda 38 item kullanılması gerekmektedir ve bu gereksiz yere haberleşmenin yoğunlaşmasına sebep olacaktır. Bu yöntemin yerine wordleri ayıklama methodunu kullanılması daha uygundur. Bu yöntem için PLC içerisinde üç tane word değişkeni tanımlanır, iki word dijital girişler için bir word ise dijital çıkışlar için kullanılabilir. Bu word değişkeni bir dizi olarak düşünülebilir. Her bir dijital giriş veya çıkış bu word'un elemanları olarak atanabilir böylece 48 bit, üç word içerisinde temsil edilebilir. Wordler üzerindeki bitlerin atanma yöntemi tablo 6.21'deki gibi olabilir.

Tablo 6.21. Wordler içerisindeki bitlerin gösterilmesi

Word 1																
Word Bit No	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Giriş numarası	I1	I2	I3	I4	I5	I6	I7	I8	I9	I10	I11	I12	I13	I14	I15	I16

Word 2																
Word Bit No	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Giriş numarası	I17	I18	I19	I20	I21	I22	I23	I24	I25	I26	I27	I28	I29	I30	I31	I32

Word 3																
Word Bit No	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Çıkış numarası	I1	I2	I3	I4	I5	I6	I7	I8	I9	I10	I11	I12	I13	I14	I15	I16

Word değişkenlerinden bit dizilerini oluşturabilmek için, PLC tarafından gelen word değişkenini Delphi tarafında bit'lere ayırmak, word değişkenini binary dizisine çevirmek gerekmektedir. Bu çevirici işlemi için Delphi'de yapılmış bir yazılım olmadığından, programa yeni bir Dec_To_Bin fonksiyonu eklenmeli ve gerekli işlem bu fonksiyonun içerisinde üretilmelidir. Bu fonksiyonun amacı gelen word değerini binary dizisine string formatında çevirmek, ve oluşan binary dizisi 16 haneden az ise sonuç 16 hane olacak şekilde sonucun önüne sıfır eklemek olmalıdır. İşemi gerçekleştirmek için sayının 2 tabanında modu alınmalı, sonuç bir ara değişkende saklanmalı, sayı ikiye bölünüp, tekrar 2 tabanında modu alınmalı ve önceki değerle toplanmalıdır. İşlem kalan sayının iki sayısından eşit veya küçük olana kadar devam etmelidir. Çıkan sonuçun kaç haneli olduğu belirlenmeli, 16 sayısından çıkartılmalı, çıkan fark kadar sıfır rakamı elde edilen sayıya sol tarafından eklenmelidir.

Sıfırların eklenmesi ile, sayının değeri degişmemekte, hane sayısı word için gerekli olan onaltı haneye çıkartılmaktadır. Eğer sonuç onaltı haneden küçük olursa, word dizisini parçalayan işlemde hata oluşacaktır. Bu fonksiyon oluşturulduktan sonra OPC serverdan okunan word değerleri bu fonksiyona gönderilerek sonuçları değerlendirilmelidir. İstenilen algoritmayı gerçekleştirecek fonksiyon tablo 6.22'deki gibi olmalıdır.

Tablo 6.22. Decimal sayıyı binary formata çeviren fonksiyon

```
function Tfrm_ana.DecToBin(sayi: integer): string;
var
 buffer: string;
 i, j: integer;
begin
 buffer := '';                                  // başlangıçta değişkelerin içeriğini temizle
 Result := '0';
  while (sayi > 0) do                           // sayı O'dan büyük olduğu sürece
   begin
    buffer := IntToStr(sayi mod 2) + buffer; // sayının mod 2 karşılığını al,önceki ile topla
    sayi := sayi div 2;                         // sayıyı ikiye böl
   end;
 j := 16 - Length(buffer);                      // oluşan binary dizisinin uzunluğunu 16'dan çıkart
  for i := 1 to j do                            // çıkan değer kadar
    buffer := '0' + buffer;                     // sayının onune sıfır ekle
  Result := buffer;                                 // sonuç sıfır eklenmiş sayıdır
end;
```

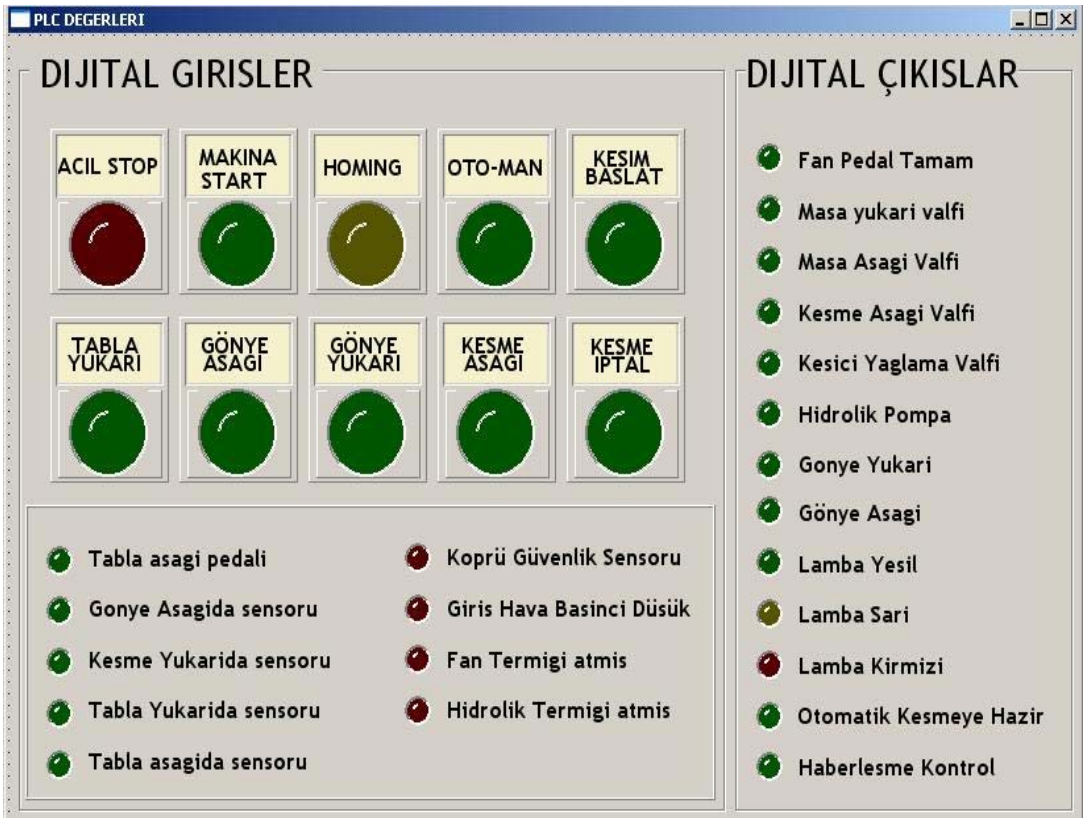
PLC'ye gönderilmek istenen bit uzunluğunda oluşan değerleri de, bit dizilerinde sıralanarak word oluşturulur ise, bir word içerisinde 16 komutu göndermek mümkün olur. Bitlerden word oluşturabilmek için yeni bir algoritma üretilmelidir ve bu fonksiyona BintoDec ismi verilebilir. Algoritma içerisinde, for do döngüsü tanımlanarak her bit değeri için, bit değeri bir sayısına eşit ise, sayının hane sayısı kadar ikinin katı alınıp sonuca eklenererek, istenilen decilmal sayı oluşturulabilir. Algoritmayı gerçekleştirebilecek işlem tablo 6.23'deki gibi olabilir.

Tablo 6.23. Binary diziyi decimal sayısına çeviren fonksiyon

```
function Tfrm_ana.BinToDec(str: string): Integer;
var
 i: integer;
begin
 Result := 0;                                    // başlangıçta değişkenin içeriğini temizle
 for i := 1 to Length(str) do                    // string'in sonuna kadar
  if str[i] = '1' then                           // stringdeki değer bir ise
   Result := Result + round(power(2, i - 1));    // 2'nin katını al, önceki sayı ile topla
end;
```

OPC programı, form'un açılış olayında "opc.Active:= True;" komutu ile haberleşmeye açılmalı, form'un kapanış olayında da opc.Active:= false ile haberleşmeye kapatılmalıdır. Haberleşme anını seçmek çok önemlidir. Eğer haberleşecek veriler sürekli okunur ise, haberleşme kanalı gereksiz yere meşgul edilmiş olur. Değerler sadece değiştiği anda okunması, haberleşme hattının optimum

kullanılma yöntemidir. Bunu yapılabilmesi için kodlar eklenmiş olan opc nesnesinin, properties sekmesinden OPCGroups özelliği seçilir. Buradan istenen grup seçilir, ve events sekmesinden OnDatachange olayının içerisine istenilen kodlar yazılmalıdır. PLC'nin dijital giriş ve çıkışlarının değerleri ekranda izlenmelidir. Bunun amacı PLC'nin dijital giriş ve çıkışlarında voltaj olup olmadığının kullanıcı tarafından anlaşılmasıdır. Bu fonksiyon arıza takibi açısından çok önemlidir. Herhangi bir fonksiyonun çalışmaması halinde kullanıcının kontrol edeceği ilk nokta sinyalin olup olmamasıdır. Bu hızlı okunması gereken bir değer olmadığından, haberleşme gruplarından yavaş grubunun içerisine alınmalıdır. Yavaş grubuna, PLC'nin 3000, 3001 ve 3002 kodları eklenmiştir. Yavaş grubunun, onDatachange olayının içerisine, değişen item'in değerini ve Id numarasını dijital giriş ve çıkışlara ait göstergelere gönderilir ise, ve bu göstergede değişen bitlere ait olan ledlerin ekrandaki rekleri değiştirilir ise, değerlerin değişimi izlenebilir. Kesme makinesinde kullanılan dijital giriş ve çıkışlar için şekil 6.9'daki gibi, bütün dijital giriş ve çıkışları içeren, bir form düzenlenebilir.



Şekil 6.9. Dijital giriş ve çıkışkar için düzenlenen form

Bu formda lamba görünümlü nesnelerin hepsi birer IledRound'dur ve isileri led ile başlamalı arkasından dijital giriş numaraları gelmelidir. led1 acil stop için, led2 otoman için atanmalıdır. Her bir led pozisyonu için bir fonksiyon tanımlanmamıştır. Yeni bir metot geliştirilerek bir fonksiyonda bütün led'lerin pozisyonu ayarlanmalıdır. Ledlerin numaralarının diziliminin kolaylaştırabilmek için bütün ledlerin yazı değeri aynı, yazıdan sonra gelen rakam değeri PLC'deki word değeri ile aynı sırada olmadır. Ayrıca gelen word değerlerine gelmiş olduğu item numaralarına göre bir sınıflandırma yapmak gerekir. Sıfırıncı ve birinci word'ler dijital girişlere, ikinci word ise dijital çıkışlara atanmalıdır. Yani 1 ile 32 arasındaki led'ler dijital girişlere, 33 ile 48 arasıdaki led'ler dijital çıkışlara atanmalıdır. Fonksiyonun led numarası, word dizisindeki numası ile, word değişkenin onaltı sayısı ile çarpımının toplamına eşit olmalıdır. Sinyellerden, "hidrolik termiği atmış" sinyali PLC'de birinci word'ün dördüncü bit'ine atanmıştır. Delphi yazılımında da led numarası 20 yapılmalıdır.

Delphi'de kod bölümünde, önce verinin değiştiği algılanmalı, bu değişim algılandığında led'lerin pozisyonları tekrar belirlenmelidir. Değişimin algılanması için ikinci grubun Datachange olayı içerisine tablo 6.24'deki gibi düzenlenir ise, bütün değişen değerleri, item numarası ile frm_PLC'deki, IO_gostergeye gönderilebilir.

Tablo 6.24. Verinin değişti anda, göstergeye yeni kodların gönderilmesi

```
 procedure Tfrm_ana.opcOPCGroups2Datachange(Sender: TObject;
  ItemList: TdOPCItemList);
var
  i: integer;
begin
 for i:=0 to ItemList.Count - 1 do  // 0'dan son item'ların sonuna kadar
  begin                             // değişen değerin değerini ve numarasını göstergeye gönder
    frm_PLC.IO_gosterge(ItemList[i].Value,ItemList[i].ID);
  end;
end;
```

Gelen veriler formüle uygun şekilde isimleri belirlenmiş olan led'lerin reklerini ayarlayarak, kullanıcının girişte sinyal olup olmadığını anlamasını sağlamak için, öncelik ile gelen değer binary formata dönüştürülerek bitlere ayırılmalı, 16 bit için teker teker formda led isimli componentler aranmalı, bulunan componentler

iledround olarak algılanmalı, bulunan component ile bit numarası eşlenmeli, eğer gelen değer bir sayısına eşit ise, bulunmuş olan ledin rengi değiştirilmelidir. Gelen word numarası 16 sayısının katı olarak led numrasına eklenerek bir algoritmada bütün word değerleri işlenebilir. Gerekli algoritma tablo 6.25'deki gibi oluşturulabilir.

Tablo 6.25. Ledlerin renklerinin değiştirilmesi

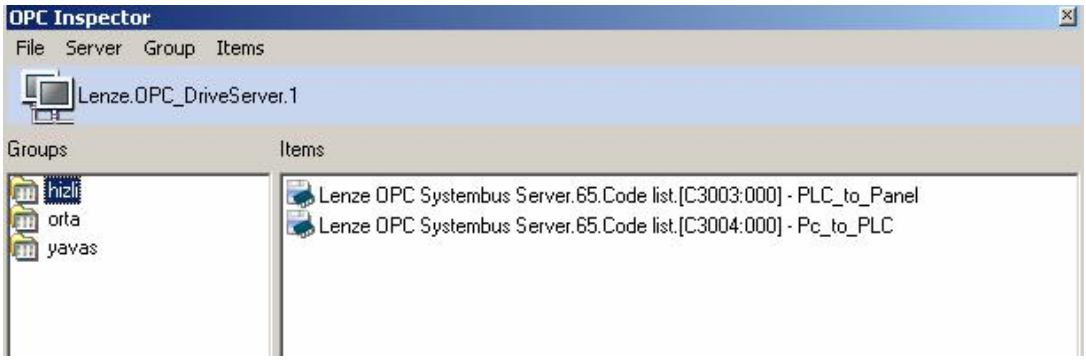
```
procedure Tfrm_PLC.IO_gosterge(deger: word; itemID: integer);
var
   Wordden_gelen_bin: String;
   i:integer;
begin
if frm_PLC = nil then Exit;                          // eğer form açılmış ise, işlemleri gerçekleştirme
 Wordden_gelen_bin:= frm_ana.DecToBin(deger); // girilen değeri binary formata dönüştür
 for i:=1 to 16 do                                   // 16 bit'lik binary'in her bir bit'i için
  begin                                              // formdaki led isimli komponentleri ara bulursan
                                                     // bulduğun komponent'i Lenround olarak algıla
                                                     // okunan değer 1 ise active=true yap
  if frm_PLC.FindComponent('led'+ IntToStr(i+(itemID*16))) <> nil then
   (frm_PLC.FindComponent('led' + IntToStr(i+(itemID*16))) as TiLedRound ) . Active : =
   (Wordden_gelen_bin[i]='1');
  end;
```

6.4.1.8. Sisteme kısıtlamalarının eklenmesi

PLC ile Delphi'nin OPC üzerinden oluşturulan haberleşme kanalı ile, sistemin çalışma ile ilgili komutları da gönderilmeli ve alınmalıdır. Makine üzerinde, sistem çalışmadan önce olması gereken birçok kısıtlama mevcuttur. Bunlara bir örnek olarak hava basıncı olmadan makine çalışmamalıdır yada pano üzerinden acil duruş tuşuna basıldığında Delphi yazılımı kontrol kartına pulse göndermemelidir. Bu kısıtlamları oluşturabilmek için yapılabilecek yöntemlerden biri PLC üzerinden Delphi programına aktarılarak, gerekli lojik algoritma Delphi içerisinde de yapılmasıdır. Diğer bir yöntem ise bütün kısıtlamaların PLC üzerinde toplanaması, Delphi programına sadece komut gönderilmesidir. İkinci metot Delphi'de daha az yazılım yükü getirdiğinden ve PLC üzerinde programlama yapmanın daha kolay olduğundan, tercih edilmelidir. Bütün kısıtlamalar PLC içindeki programda olmalı, Delphi PLC'ye, çalışma isteği göndermeli, PLC'de şartlar uygun ise çalış komutunu Delphi'ye göndermeli, Delphi'de sistemi çalıştırmalıdır. Çalışma şartlarında bir

değişiklik olduğunda PLC, Delphi yazılımına durma komutu göndermeli ve sistem durmalıdır. Aynı yöntem home fonksiyonu için de yapılmalıdır. Bu birçok fonksiyon ile çeşitlendirilebilir, makinenin çalışması için bu iki fonksiyonun yapılması yeterlidir. Diğer fonksiyonlar bu fonksiyonun bir kopyası olacaktır.

İstenilen fonksiyonların gerçekleştirilebilmesi için Delphi'de, OPC ile veri okuma ve yazma için bir algoritma oluşturmak gereklidir. Bu algoritma bir kez oluşturulduğunda bütün komutlar bu algoritma içinde değerlendirilebilir ve sonuçlandırılabilir. Dolayısı ile veri okuma ve yazma istendiğinde sadece bu algoritma çağırılmalıdır. Aynı zamanda veri göndermek için bir word, veri okumak için bir word tahsis edilmelidir. Bu Delphi yönünden, PLC'den komut okuma işlemi için PLC'de 3003 numaralı kod, komut yazmak için PLC'de 3004 numralı kodlar tahsis edilmiştir. Bunun karşılığı olan OPC programına da bu kodlar eklenmiştir. Eklenecek kodların sayısı yapılması gereken işlem yüküne göre artabilir. Bu veri alışverişinin hızı, makinanın daha hızlı çalışabilmesi için, hızlı olmalıdır ve kodlar hızlı grubunun içerisine koyulmuştur. Hızlı grubundaki kodlar şekil 6.10'daki gibi gösterilebilir.



Şekil 6.10. Hızlı grubunun gösterilişi

Kodlar OPC içerisine eklendikten sonra, PLC ile OPC arasındaki haberleşme sağlanmıştır. Word içirisindeki bit numaralarının anlamları alıcı ve verici tarafında bilinmelidir. Veri alışverişi yapılırken PLC içindeki bit numarası ile Delphi yazılımı içerisindeki bit numaraları ayrı olabilir. Verinin alındığı ve gönderildiği cihazlardaki haberleşmenin kolay yapılabilmesi için, kodlar içersindeki bit numaralarının anlamlarını içeren, tablo 6.26'deki gibi tablolar yapılmalıdır.

Tablo 6.26. PLC'den Delhi yazılımına giden komutların listesi

3003 PLC'den Delphi Yazılımına Giden Komutlar		
PLC'deki bit numarası	Fonksiyon	OPC'deki bit numarası
0	Kesme Başlat	1
1	Köprü Güvenlik Sensörü	2
2	Hava Basıcı Düşük	3
3	Fan Termiği	4
4	Hidrolik Termiği	5
5	Acil Stop	6
6	Sistem Hazır	7
7	Sistem Home Yap	8
8	Sürücü Hatada Değil	9

3004 Delphi yazılımından PLC'ye giden komutlar		
PLC'deki bit numarası	Fonksiyon	OPC'deki bit numarası
0	Sistem home yaptı	1
3	Kesim Başlatma İsteği	4
4	Home Yapma İsteği	5
5	Sürücü Arızasını Resetle	6

Bu bölümden sonra OPC ile Delphi arasındaki haberleşme sağlanmalıdır. İlk olarak OPC ile veri okumamak için bir metot geliştirilecektir. Bu metodun ismi OPC_Oku'dur. Bu fonksiyona okunacak verinin grup numarası, item numarası ve okunacak bit numarası girildiğinde, fonksiyon biti okuyup cevabını verecektir. Metot'u gerçekleştirebilecek fonksiyonun içeriği tablo 6.27'daki gibi oluşturulabilir.

Tablo 6.27. Delphi yazılımında, OPC aracılığı ile veri okunması

```
function Tfrm_ana.OPC_Oku(GroupID, itemID, index: Smallint): Smallint;
var
  s: string;
begin                                   // istenilen değeri oku, binary format'a çevir
  s := DecToBin(opc.OPCGroups[GroupID].OPCItems[itemID].Value);
  Result := StrToInt(s[index]);         // istenilen index numarasını sonuç olarak ver
end;
```

Buna karşılık olarak yazılacak değer içinde bir metot geliştirilmelidir. Bu metotun ismi de OPC_Yaz olacaktır. Sadece isenilen bit nımarasına değer yazılıp gönderilir ise, diğer komutlara 0 gideceğinden bu fonksiyon çalışmayacaktır. Opc'de veri yazmadan önce veri okunmalı, okunan değerde, sadece değiştirilmek istenen bit'in değeri değiştirilmeli daha sonra yazılmalıdır. Böylece diğer komutların değerleri değişmemiş olur. Metot'u gerçekleştirebilecek fonksiyonun içeriği tablo 6.28'deki gibi özetlenebilir.

Tablo 6.28. Delphi yazılımında, OPC aracılığı ile veri yazılması

```
procedure Tfrm_ana.OPC_Yaz(GroupID, itemID, index, Value: Smallint);
var                                  // Opc ile bit yazma fonksiyonu
 s, t: string;                       // dahili değişkenler
 ch:Char;
begin                                // öncelikle değeri oku ve binary format'a çevir
 s := DecToBin(opc.OPCGroups[GroupID].OPCItems[itemID].Value);
 if value = 0 then   ch := '0'       // yazılacak değer 0 ise ch'yi sıfır yap
 else
 if value = 1 then  ch := '1';       // yazılacak değer 1 ise ch'yi bir yap
 s[index] := ch;                     // s değişkeninin değişecek index'ini ch'ye eşitle
                                     // yeni oluşan değeri opc'de yaz
  opc.OPCGroups[GroupID].OPCItems[itemID].WriteAsync(BinToDec(s));
end;
```

OPC_Yaz ve OPC_Oku fonksiyonları ile yazma ve okuma işlemi için genel bir metot geliştirilebilir. Bu metot geliştirildikten sonra makinanın otomatik çalışma fonksiyonunun değiştirilmesi gerekmektedir. Kesim başlat tuşunun içeriği artık değiştirilmelidir. Delphi PLC'ye kesim başlat isteğini OPC_Yaz(0,1,4,1) komutu ile, kesim durdurma isteğini OPC_Yaz(0,1,4,0) komutu ile gönderebilir. Bu komutun anlamı sıfırıncı grubun, birinci item'inin, dördüncü bitinin değerini, bir'e eşitle demektir. Bu da 3004 numaralı kodun dördüncü biti'ni bir yap demektir. Bu bit PLC'de Delphi'den gelen kesim başlatma isteği olarak daha önceden tanımlanmıştır. Bu isteğin sonucunda PLC, eğer hiçbir kısıtlama yok ise, 3003 numarali kod'un birinci bit'ini bir sayısına eşitleyecektir. Bu Delphi açısından birinci grubun, sıfırıncı item'ının değişmesi anlamına gelmektedir. Bunun Delphi tarafından algılanması ve bunun çalışma komutuna çevirilmesi yani trmkesim isimli timer'ı çalıştırılması gerekmektedir. PLC tarafından herhangi bir sebep ile makinenin durdurulması gerektiğinde veya Deplhi'den kesim durdurma isteği geldiğinde, yani 3003 numaralı kod'un birinci bit'i sıfır değerini aldığında, tmrkesim isimli timer'ın da durdurulması

gerekmektedir. Ayrıca motorların durması için de acil duruş fonksiyonunun da çalıştırılması gerekmetedir. Bu olay sıfırıncı grubun data change olayının içerisinde olmalıdır. Bu olayın içerisinde bir de uyarı mesajlarının komutları koyulmalıdır. Örnek olarak hava basınç'ının düşük olmasından dolayı makinenin çalışmadığı kullanıcıya belirtilmelidir. Bu komutlar'da 3003 numaralı kodun içerisinden OPC programına gönderilmelidir. PLC'den gelen komutları algılayabilecek fonksiyonun içeriği tablo 6.29'deki gibi özetlenebilir.

Tablo 6.29. PLC'den gelen komutların Delphi programında algılanması

```
procedure Tfrm_ana.opcOPCGroups0Datachange(Sender: TObject;
  ItemList: TdOPCItemList);
var
 s: string;
 i: integer;
begin
 for i:=0 to ItemList.Count - 1 do              // item'ların sonuna kadar
  if ItemList[i].ID = 0 then                    // değişen item eğer PLC'den PC'ye gelen ise
  begin
   s := DectoBin(itemlist[i].Value);            // gelen değeri binary'e çevir ve iç değişkene ata
    if (s[1] = '1') then                        // eğer ilk bit bir ise
     tmrKesim.Enabled := True                   // timer'i başlatarak otomatik kesimi başlat
    else
    begin                                       // eğer ilk bit 0 ise
     btn_acilClick(nil);                        // acil duruş düğmesine bas
     tmrKesim.Enabled := False;                 // timer'i durdur
    end;
   if (s[8] = '1')then frm_ana.home_yap;         // PLC'den gelen home emri
                                                 // bit'ler aktifse led'leri aktifleştir
    led_kopru_guvenlik.Active:=(s[2] = '1');
    led_hava_basinc.Active:= (s[3] = '1');
    led_fan_termik.Active:= (s[4] = '1');
    led_hidrolik_termik.Active:= (s[5] = '1');
    led_acil_stop.Active:= (s[6] = '1');
    if led_acil_stop.Active then
     begin                                      // home yaparken panodan acil duruşa basılır ise,
      trm_home.Enabled:=false;                  // home işlemini durdur
      home_ok:= false;                          // home_ok değişkenini resetle
      end;
   led_sistem_hazir.Active:=(s[7] = '1');
   led_surucu_devrede.Active:=(s[9] = '1');
 end;
end;
```

6.4.1.9. Uyarı sinyallerinin gösterilmesi

OPC'den gelen uyarı bit'leri için form aşşağıdaki gibi düzenlenebilir. Led'lerin

rengi, gelen değerin bir veya sıfır olmasına göre değişmektedir. Buna göre kullanıcıda makinede herhangi bir sorun olup olmadığını gözleyebilir. Ledlerin rengi tablo 6.29'de ayarlanmalıdır. Formda uyarı sinyalleri için oluşturulabilecek, düzenleme şekil 6.10'daki gibi olabilir.

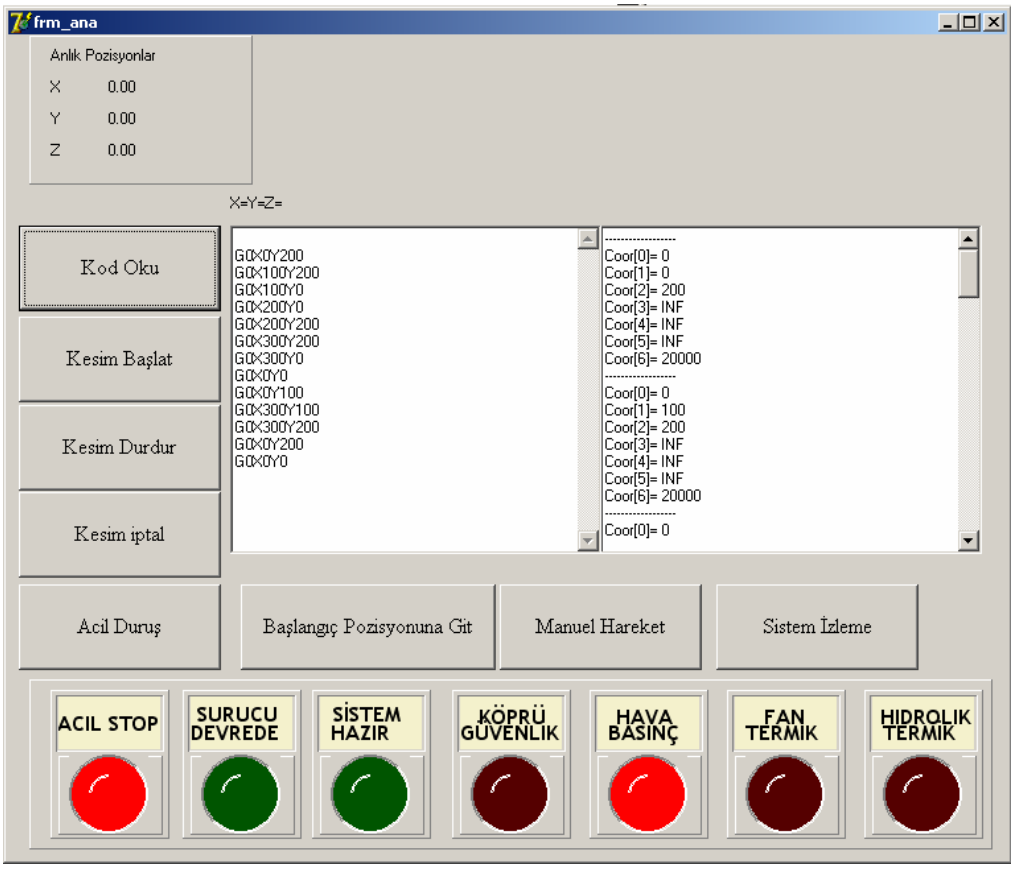
Şekil 6.11. Uyarı sinyallerinin Delphi yazılımında gösterilmesi

Sistem çalıştırılmadan önce mutlaka başlangıç noktasına gönderilme işlemi yani home işlemi yapılmasını sağlayacak bir metot geliştirilmelidir. Eğer ilk program açıldığında kontrol kartı mekanizmanın bulunduğu gerçek yeri bilemez ise, mekanizmayı limitlerin dışına gönderebilir. Her enerji kapatıldığında kontrol kartının sayıcısı kendini sıfırladığından, ilk enerji açıldığında bulunduğu noktayı (0,0) kabul eder. Örnek olarak mekanizma gerçekte (500,500) noktasında enerji kesilmiş olsun ve tekrardan enerji geldiğinde kontrol kartı gerçek pozisyonu (0,0) noktasında olduğunu düşünür. Bu da gerçek kesimlerde hataya sebep olur. Bu sebeple PLC'nin içerisindeki lojik işlemleri ile home işlemi yapılmadan, otomatik start işlemine başlamamalıdır. PLC, Delphi'den home tamamlandı, sinyalini beklemeli ve bu sinyali aldıktan sonra çalışmaya başlamalıdır. Home isteği PLC'ye iki yol ile gönderilebilir. İlk olarak dijital bir sinyal olarak gönderilebilir ve bu PLC'nin lojik fonksiyonları içerisinde yer alır. İkinci yol ise Delphi programından göndermektir. Delphi'de başlangıç pozisyonuna git tuşuna basıldığında, daha önceden belirlenen prosedur yerine, Delphi PLC'ye, başlangıç noktasına gitme isteğini OPC_Yaz(0,1,5,1) komutu ile gönderir. Bu komutu alan PLC, komutu kendi içinde değerlendirdikten sonra Delphi'ye 3003 kod'udaki sekizinci bit'i bir yaparak home yapılması emrini gönderir. Bit'in bir olduğunu algılayan Delphi home_yap prosedürünü işletir. Home_yap prosedürü daha önceden anlatılan home tuşu prosedürü ile aynıdır. Home işlemi başladıktan sonra trm_home timer'ının çalışmaya başlar, bu timer bufferların boşalmasının ardından home_ok sinyali üretir ve bu sinyali OPC_Yaz(0,1,1,1) komutu ile PLC'ye gönderir, son olarak da kendi timer'ını

durdurur. Komutu alan PLC bundan sonra sistemin çalışmasına izin verir. Home_ok sinyali ve trm_home timer'i, Delphi'deki acil stop tuşuna veya pano üzerindeki acil duruş düğmesine basıldığında da resetlenmelidir. Delphi'deki acil duruş tuşunun içerisine trm_home.Enabled:=false ve home_ok:= false komutları eklenmelidir. PLC'den acil stop'a basıldığı 3003 numaralı kod'un altıncı bit'inin değerinin sıfır olması ile anlaşılabilir. Home işlemi ile birlikte, standart bir makinede istenebilecek bütün komutlar tasarlanmıştır.

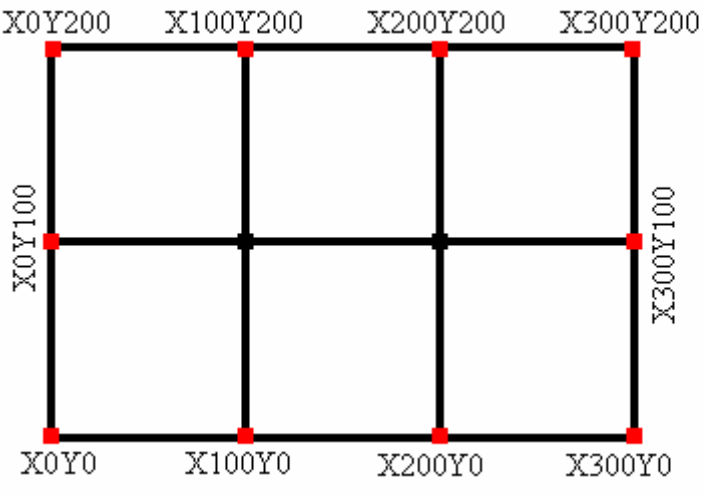
6.4.2. Sistemin çalıştırılması

Üretilmiş olan SCADA yazılımı ile temel olarak, bir kontrol kartı kullanılarak, PLC'nin, ve mekanik sistemi kontrol eden sistremin bir bilgisayar yazılımı tarafından izlenmesi, yönetilmesidir ve arızalarının izlenmesi gerçekleştirilmiştir. Delphi ile yapılmış olan SCADA yazılımında ana form şekil 6.12'deki gibi oluşturulmuştur.



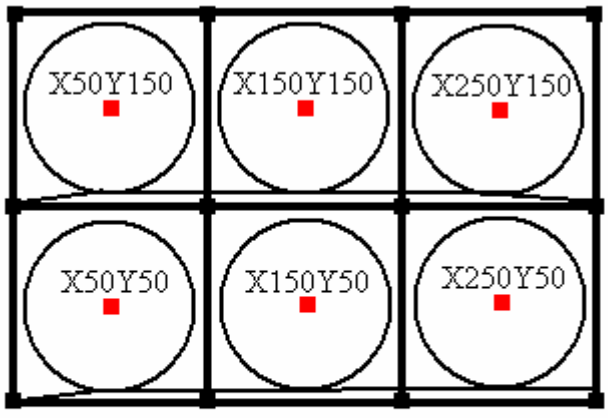
Şekil 6.12. SCADA yazılımının ana sayfası

300x200 cm boyutundaki bir plaka altı adet 100x100 cm parçalanmak istendiğinde ilk olarak kesimi yolunu oluşturan Gcode dosyası oluşturulmalıdır. Şekil 6.2'de kesilecek plaka üzerinde gidilecek hedef noktaları belirlenerek örnek bir Gcode dizisi oluşturulmuştur. Kod oku tuşu seçilerek istenilen Gcode formatındaki kesim dosyası SCADA yazılımının içerisine alınmıştır. Daha sonra Gcode satırları parçalanarak değişkenlere ayrılmıştır. Kesim başlat tuşu ile hareket başlatılarak kesim işlemi gerçekleştirilmiştir.




```
G0X0Y200
G0X100Y200
G0X100Y0
G0X200Y0
G0X200Y200
G0X300Y200
G0X300Y0
G0X0Y0
G0X0Y100
G0X300Y100
G0X300Y200
G0X0Y200
G0X0Y0
```

Şekil 6.13. Kesilecek plaka üzeride hedefler ve hareketi oluşturacak Gcode dizisi

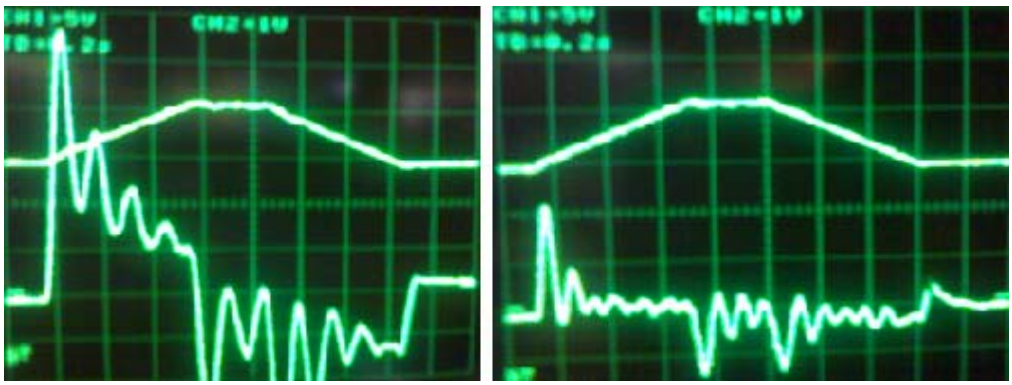



```
G0X0Y0Z0
G0X50Y5
G2X50Y5I50J50
G0X150Y5
G2X150Y5I150J50
G0X250Y5
G2X250Y5I250J50
G0X300Y5
G0X300Y100
G0X250Y105
G2X250Y105I250J150
G0X150Y105
G2X150Y105I150J150
G0X50Y105
G2X50Y105I50J150
G0X0Y100
G0X0Y0
```

Şekil 6.14. 90cm çapında 6 daire çizilmesi için daire merkezleri ve hareketi oluşturacak Gcode dizisi

Aynı plaka üzerinde 90 cm çapında altı adet daire, aralarında 10 cm boşluk kalıcak şekilde kesilebilmesi için gerekli Gcode dizisi ile dairelerin plaka üzerindeki merkez noktaları şekil 6.14'deki gibidir. Dairelerin ekrafında kareler oluşturulmak istendiğinde şekil 6.13'deki Gcode dizisi, şekil 6.14'deki Gcode dizisine eklenenebilir.

Hareket kontrolünde en önemli nokta motorların sisteme göre ayarlanmasıdır. Bu ayarlama motorların hızındaki değişimlere ait olan PID değerleri değiştirilerek yapılmaktadır. Motorların PID değerleri yanlış ayarlandığında motorların hareketlerinde hatalar oluşmakta ve sistemin hareketinde bozulmalar meydana gelmektedir. Kesme makinesinde motorların en iyi PID değerlerini bulabilmek için osiloskop kullanmılmıştır. Sürücülerin üzerindeki analog çıkışların birinden motorun anlık hızı, diğerinden motorun faz hatası sinyalleri bir osiloskop ile incelenerek, PID değerlerinin sisteme olan etkisi incelenmiştir. Kesme makinesinde Y ekseninin sürücüsünün üzerindeki analog çıkışlar osiloskop'a taşınarak şekil 6.15 elde edilmiştir. Şekilde yukarıda görülen grafik eksenin hız ve zaman, aşağıda görülen grafik hata ve zaman değerlerini göstermektedir. En büyük hata hız değişimlerinde oluşmaktadır. Osiloskop ile incelenen hareketlerin tümünde eksen aynı yönde 1200 mm hareket ettirilmiştir. Başlangıç değerlerinde PID değerleri sırası ile 1.5,200,0'dır. Bu PID değerlerinde motorun hareketinde aşırı hata oluşmaktadır. PID değerleri 3,50,0 yapıldığında hata değeri azalmıştır fakat sistemde aşırı bir titreşim oluşmuştur.



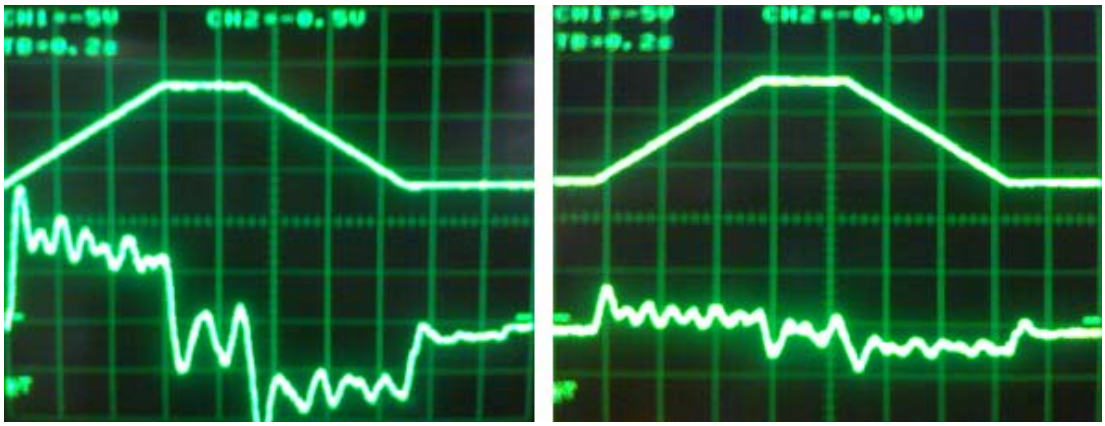
a) b)

Şekil 6.15. Y ekseninin hareketinin osiloskop ile izlenen sonuçları a) PID=1.5,200,0 b) PID=3,50,0

Optimum PID değerleri toleransın içerisindeki hatanın oluşmasına izin verilerek, sistemin en az titreşimde çalıştığı değerlerdir. Yapılan deneylerde optimum PID değerleri 2.5,200,0'dır. Bu değerde Y ekseninde oluşan hata ve titreşim toleransın içerisinde kalmaktadır. Yapılan hereket deneyinde osiloskop ile incelenen sinyaller şekil 6.16'deki gibi oluşmuştur.

Şekil 6.16. Y ekseninin hareketinin osiloskop ile izlenen sonuçları PID=2.5,200,0

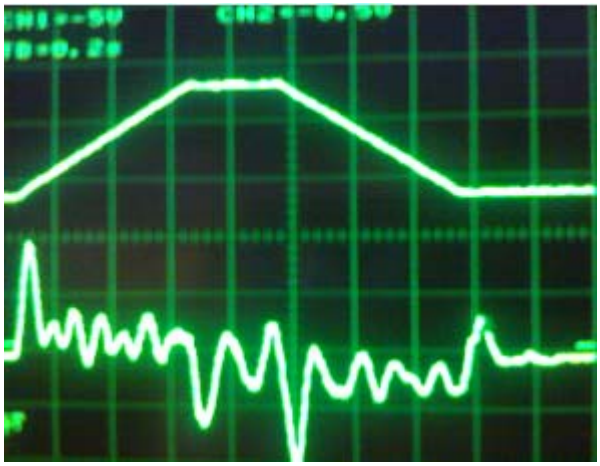
Aynı deneyler X ekseninde de yapılmıştır. X ekseninin sürücüsünün üzerindeki analog çıkışlar osiloskop'a taşınarak şekil 6.17 elde edilmiştir. Başlangıç değerlerinde PID değerleri sırası ile 10,200,0'dır. Bu PID değerlerinde motorun hareketinde aşırı hata oluşmaktadır. PID değerleri 26,50,0 yapıldığında hata değeri azalmıştır fakat sistemde aşırı bir titreşim oluşmuştur



a) b)

Şekil 6.17. X ekseninin hareketinin osiloskop ile izlenen sonuçları a) PID=10,200,0 b) PID=26,50,0

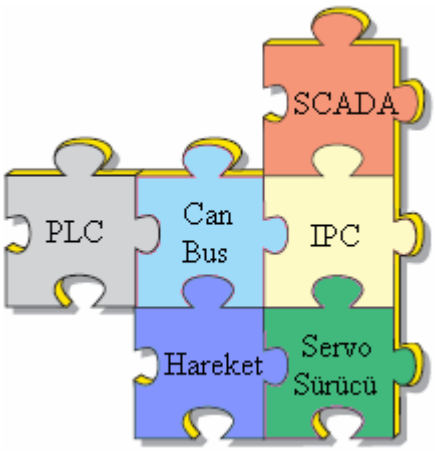
Yapılan deneylerde optimum PID değerleri 10,50,0'dır. Bu değerde X ekseninde oluşan hata ve titreşim toleransın içerisinde kalmaktadır. Yapılan hereket deneyinde osiloskop ile incelenen sinyaller şekil 6.18'deki gibi oluşmuştur. X ekseninde kullanılan motor gücü daha fazla olduğundan Y eksenine göre daha küçük hata değerleri oluşmuştur. Bu deneylerin sonucuna göre motor gücü büyüdükçe sistemde oluşan hata değeri küçülmektedir.



Şekil 6.18. X ekseninin hareketinin osiloskop ile izlenen sonuçları PID=10,50,0

BÖLÜM 7. SONUÇLAR VE ÖNERİLER

7.1. Sonuçlar



Şekil 7.1. Birleştirilmiş yap boz

Bu çalışma sonucunda, otomasyon sistemlerini oluşturan servo sürücüler, PLC'ler, endüstriyel bilgisayarlar, eksen kontrol kartları, SCADA yazılımları ve haberleşme sistemleri yap bozu oluşturmuşlardır. Yapboz oluşturan cihazlar ortak bir tabanda birleştirilerek karmaşık bir otomasyon sistemine çözüm oluşturulmuştur. Sistemdeki bütün cihazların fonksiyon kabiliyetleri bulunduktan sonra sistemin çalışması için gerekli fonksiyonlar cihazlar üzerinde gerçekleştirilmiştir. Sistemin optimum tasarımı yapılarak fonksiyonları gerçekleştirebilecek en uygun elektronik cihazlar seçilmiştir. Bu tezde bir cihazın üzerinde yoğunlaşılmamış, otomasyon sisteminin genel mantığına yer verilmiş ve cihazların sistemler içerindeki yerleri ile haberleşme sistemleri anlaşılmıştır. Bu tezde üretilen sistemde son teknolojiye sahip elektronik cihazlar kullanıldığından, sistem yeniliğe ve eklere açıktır.

Bu tezde araştırılmış Gcode formatı ile standartlaştırılmış eksen kontrollü sistem örnek alınarak CNC, robot, işleme, kesme makineleri gibi birçok uygulama gerçekleştirilebilir. Cam, metal, mermer gibi malzemeleri işleyebilecek bir

makinenin otomasyonu yapılabilir. Uluslararası standartlara uygun bir sistem üretildiğinden, işleme işlemleri için kullanılan optimizasyon yazılımların ürettiği çözümlerde kullanılabilir.

Bu çalışmada kullanılan cihazlar ve yazılımlar fiyat açışından pazarda rekabetçi, kullanım açışısında kolay, programlama ve fonksiyon açısından genişlemeye uygundur. Böylece sistemler üzerinde optimum çözümler üretilebilecek bir yöntem geliştirilmiştir.

Bu çalışma ile birlikte otomasyon sistemleri tasarımcılarına, cihazların ince detaylarına girilmeden, otomasyon sistemlerinin çalışmasını açıklayan bir kaynak sunulmuştur. Bu kaynakta tasarımcıya sunulan yöntemi, tasarımcı kendi sistemine uyarlayabilir ve kendi sistemini çalıştırabilir. Tasarıma yeni başlayanlara da bu kaynak ile otomasyon sistemlerinin çalışma algoritmasını inceleyebilir.

Sisteme enerji verildikten sonra hazırlanmış olan programlar cihazlara yüklenmiştir. Enerji kapatılıp tekrar açıldıktan sonra PLC ve ek IO modülleri devreye girmiştir. Bilgisayara yüklenmiş olan Windows tabanlı yazılım açıldıktan sonra Delphi programında hazırlanmış SCADA yazılımı devreye girmiştir. Pano üzerinde kontrol voltajının sisteme iletilmesini sağlayan tuşa basıldıktan sonra sürücüler de devreye girmiştir. Acil duruş tuşuna basılmadığı sürece sürücüler devrede olacaktır. Enerji açıldıktan 3 saniye sonra Can Bus haberleşme sistemi devreye girmiş ve bütün cihazlar birbileri ile haberleşmeye başlamıştır. Can Bus sisteminin devreye girmesinden sonra OPC sistemi de devreye girmiş ve bilgisayar üzerindeki SCADA yazılımı haberleşmeye katılmıştır. Haberleşme hattındaki bütün cihazlar devreye girdikten sonra sistem kullanılmaya hazırdır. Panoda Manuel tuşu seçilerek sistem üzerinde hazırlanmış olan manuel fonksiyonlar aktif hale gelmiştir. Manuel çalışma konumunda malzeme yüklemek için masa yukarı tuşuna basıldığında, valfe enerji verilerek masa yukarı kaldırılmıştır. Malzeme yükledikten sonra masa aşağı tuşuna basılmış ve masa aşağıya inmiştir. Masa aşağıya indikten sonra, gönye yukarı tuşuna basılarak kesilecek malzemenin masaya göre hizalanması için, gönye takozları yukarı çıkarılmıştır. Malzeme hizalandıktan sonra, gönye takozları aşağıya indirilerek makine malzemeyi kesmeye hazır hale gelmiştir. Bu durumda pano

üzerinden otomatik tuşu seçilerek masa ve gönye valfleri gibi manuel fonksiyonlar devre dışı bırakılmış, otomatik fonksiyonlar devreye girmiştir. Otomatik fonksiyonlar kullanılarak sistem istenilen noktalara hareket ettirilmiş veya yazılmış bir program ile sistem arka arkaya istenilen noktalara gönderilmiştir. Sistemde herhangi bir pozisyonlama hatası bulunmamıştır. SCADA yazılımı ile dijital giriş ve çıkışların değerleri izlenerek, sistemin çalışmasında herhangi bir hatanın olmadığı izlenmiştir.

7.2. Öneriler

Sistemde kullanılan eksenler arttırılmak istendiğinde eksenleri kontrol eden motor ve sürücülerin sayısı arttırılabilir. Can Bus haberleşme sistemi ile 63 cihaza kadar haberleşme kurulabilir. Sistemin genişletilmesi ile sistemde kullanılan dijital giriş ve çıkış sayısıda artabilir. Ek giriş çıkış modülleri kullanarak ve bu modülleri PLC ile haberleştirerek artan IO ihtiyaçı karşılanabilir. PLC içerisindeki program gelen dijital giriş ve çıkışlara göre geliştirilebilir. PLC içerisinde yeni algoritmalar kurulabilir. Farklı bir sistem için çözüm arandığında PLC içerisindeki program, sisteme çözüm olacak şekilde genişletilebilir.

Sistemlerde gerekli olan fonksiyon ihtiyaçlarına sürücüler içerisineki yazılım özellikleri kullanılarak da çözüm üretilebilir. Sürücülerin içerisindeki yazılım fonksiyonları, PLC kadar olmasada, çözüme katkıda bulunabilir. İnterpolarizasyonun ihtiyaç duyulmadığı sistemlerde, sürücü tabanlı bir kontrol yapılabilir. Pozisyonlama ve senkronizasyon gibi uygulamalarda, sürücü içerisindeki yazılım ile sistem kontrol edilebilir. Sürücü içerisine haberleşme sistemleri aracığı ile dijital sinyaller taşınarak, sürücü içerisinde birçok algoritma oluşturulabilir.

SCADA yazılımlarını çalıştırabilecek ve sistemin izlenmesini sağlayabilecek birçok endüstriyel bilgisayar alternatifi vardır. Bilgisayar teknolojisinin ilerlemesi ile endüstriyel bilgisayarların özellikleri de artmaktadır. Ortamdaki kirliliğin fazla olduğu sistemlerde dokunmatik ekran yerine tuş takımlı ekran ile endüstriyel klavye ve fare kullanılabilir. Sistemin büyüklüğüne göre farklı ekran boyutlarına sahip bilgisayarlar kullanılabilir. Bilgisayarların ana kartı ekranın arkasına

yerleştirilebileceği gibi bir kasa içerisine de monte edilebilir. Teknolojinin gelişmesine paralel olarak bilgisayarlar üzerinde daha hızlı işlemciler kullanılarak SCADA yazılımlarının çalışma performansı arttırılabilir.

İnterpolarizasyon hareketinin oluşturulması için gerekli kontrol kartları, sistemin ihtiyacına göre farklılaşır. Kontrol edilecek eksen sayısına göre kontrol kartı da genişlemektedir. Bu tezde geri beslemesi olmayan ve üç eksen dairesel interpolarizasyon yapamayan bir kart kullanılmıştır. Dolayısı ile daire kesme hareketi sadece iki eksende yapılabilmektedir. Sistemdeki ihtiyaçlar doğrultusunda üç veya daha fazla eksen için dairesel interpolarizasyon yapabilecek bir eksen kontrol kartı kullanılabilir. Eksen kontrol kartı içerisinde ayrıca endüstriyel robotlarda kullanılan teğet fonksiyonunu da içeren kontrol kartı da seçilebilir. Sistemlerde kullanılan eksen kontrol kartınınlj, sistemin ihtiyaçlarına göre fonksiyonları arttırılabilir.

Sistemlerin kontrolü ve izlemesi için kullanılan SCADA yazılımı Delphi programlama dilinde oluşturulduğunda, yazılım olarak programcının bütün taleplerini karşılamaktadır. Windows tabanlı olduğundan yeniliklere de açıktır. Yapılan yazılım istenildiği gibi formlar ve fonksiyonlar eklenerek genişletilebilir. Her sisteme özel yazılım aynı taban kullanılarak oluşturulabilir. Oluşturulmuş olan yazılıma ayrıca sistemin hareketinin ekranda izlenebilmesi için bir similasyon eklenebilir. Gcode satırları işlenirken bu similasyon ile sistemin hareketi ekrandan izlenebilir. Sisteme internet veya modem ile uzaktan erişim sağlanabilir. Programın Delphi dili ile yazılmasının yanı sıra, bu sistem için geliştirilmiş özel bir program da satın alınabilir. Böylece oluşabilecek sorunlar için hazır bir çözüm uygulanmış olur ve sistem içerisindeki programcının yükü azalır.

KAYNAKLAR

[1] FISCHER O., MACK M., SCHUMANN T., ZELTWANGER H., ZITZMANN R., Can Dictionary, Mart 2006

[2] http://www.can-cia.org/can/

[3] Lenze System Bus eğitim seminer notları, Şubat 2005

[4] DONGLİANG Z., XİNGCHENG T., Application of Can Bus technique in Digital AC Servo drives, 2002

[5] Fujitsu Product overview, Can Bus Microcontrollers,2005

[6] Can in automation, Can Specification 2.0

[7] KVASER AB Sweden, CAN

[8] TALAT M., Programmable logic control

[9] Lenze Drive PLC seminer notları, Mart 2006

[10] KATO E.R.R., MORANDİN O., POLİTANO P.R., CAMARGO H.A., A modular modeling approach for CNC machines control using Petri Nets, 2000

[11] IOANNIDES M. G., Desing and implementation of PLC based monitoring control system for induction motor, 2004

[12] YOUNİS B. M., FREY G., Visualization of PLC programs using XML, 2004

[13] GRABNER M., LEONHARTSBERGER G., LEUTGEB A., ALTMANN J., Java in industrial automation- a virtual PLC, 2001

[14] PLOMP J., HUUSKONEN PERTTİ, MALM E., PAANASALO J., An object oriented approach to PLC Explanation, 1996

[15] Lenze 9300 Servo seminer notları, Mart 2005

[16] Product manual, Global Drive 9300 servo inverters

[17] BAİLEY D., WRIGHT E., Practical SCADA for Industry. 2003

[18] http://www.inosoft.com

[19] Digitec endüstriyel bilgisayar seminer notları, Mayıs 2006

[20] http://www.pacontrol.com

[21] http://www.newport.com/servicesupport/

[22] PCI 8164 kartının kullanım klavuzu, Ağustos 2006

[23] CANTU M., Mastering Delphi 7, 2003

[24] Borland software corporation, Borland Deplhi 7.0 for Windows

EKLER

Ek. A

Tablo A.1. Manuel hareket için gerekli kodlar

```
Var
 frm_manuel: Tfrm_manuel;                          // değişkenleri gloabal olarak tanımla
 X_boyu,Y_boyu,Z_boyu,X_son,Y_son,hiz,k_rampa,d_rampa: Double;
implementation
{$R *.dfm}
procedure Tfrm_manuel.btn_durClick(Sender: TObject);
 begin
  d_rampa:= StrToFloat(edt_dur_rampa.Text);        // dahili değişkene formdaki değeri yaz
  if  _8164_check_continuous_buffer(0)<> 0 then     // X harekeli ise X'i durdur
    _8164_sd_stop(0,d_rampa);
  if  (_8164_check_continuous_buffer(0)= 0) and     // Y harekeli ise X'i durdur
  (_8164_check_continuous_buffer(1)<> 0) then
  _8164_sd_stop(1,d_rampa);
  if (_8164_check_continuous_buffer(0)= 0) and      // Z harekeli ise X'i durdur
  (_8164_check_continuous_buffer(1)= 0) and
  (_8164_check_continuous_buffer(2)<> 0) then
  _8164_sd_stop(2,d_rampa);
end;
procedure Tfrm_manuel.btn_acil_durClick(Sender: TObject);
begin
      _8164_emg_stop(0);                             // X'i acil durdur
      _8164_emg_stop(1);                             // Y'i acil durdur
      _8164_emg_stop(2);                             // Z'i acil durdur
end;
procedure Tfrm_manuel.btn_abso_gitClick(Sender: TObject);
var
  Hedef: array[0..2] of Smallint;
begin
  Hedef[0]:=0 ;                                      // XYZ için bir array oluştur
  Hedef[1]:=1 ;
  Hedef[2]:=2 ;                                      // XYZ nin mm değerlerini pulse çevir
    X_boyu:= frm_ana.mmToPulseX(StrToFloat(edt_x_hedef.Text));
  Y_boyu:=frm_ana.mmToPulseY(StrToFloat(edt_y_hedef.Text));
  Z_boyu:=frm_ana.mmToPulseZ(StrToFloat(edt_z_hedef.Text));
  hiz:=StrToFloat(edt_hiz_max.Text);                 // hız değerini formdan al
  d_rampa:=StrToFloat(edt_dur_rampa.Text);           // rampa değerini formdan al
  k_rampa:=StrToFloat(edt_kalkis.Text);              // rampa değerini formdan al
                                                     // absolute olarak hareket et komutu
_8164_start_ta_line3(0,hedef[0],X_boyu,Y_boyu,Z_boyu,0,hiz,k_rampa,d_rampa);
```

Tablo A.1. Devam

```
procedure Tfrm_manuel.btn_rela_gitClick(Sender: TObject);
var
  Hedef: array[0..2] of Smallint;
begin
  Hedef[0]:=0 ;                                    // XYZ için bir array oluştur
  Hedef[1]:=1 ;
  Hedef[2]:=2 ;
                                                   // XYZ nin mm değerlerini pulse çevir
  X_boyu:= frm_ana.mmToPulseX(StrToFloat(edt_x_hedef.Text));
  Y_boyu:=frm_ana.mmToPulseY(StrToFloat(edt_y_hedef.Text));
  Z_boyu:=frm_ana.mmToPulseZ(StrToFloat(edt_z_hedef.Text));
  hiz:=StrToFloat(edt_hiz_max.Text);               // hız değerini formdan al
  d_rampa:=StrToFloat(edt_dur_rampa.Text);   // rampa değerini formdan al
  k_rampa:=StrToFloat(edt_kalkis.Text);
                                                   // relative olarak hareket et komutu
  _8164_start_tr_line3(0,hedef[0],X_boyu,Y_boyu,Z_boyu,0,hiz,k_rampa,d_rampa);
end;

procedure Tfrm_manuel.btn_daire_cizClick(Sender: TObject);
begin
  X_boyu:= frm_ana.mmToPulseX(StrToFloat(edt_x_hedef.Text));// X merkez değerini formdan al
  Y_boyu:=frm_ana.mmToPulseY(StrToFloat(edt_y_hedef.Text)); // Y merkez değerini formdan al
  hiz:=StrToFloat(edt_hiz_max.Text);                                    // hız değerini formdan al
  k_rampa:=StrToFloat(edt_dur_rampa.Text);                              // rampa değerini formdan al
  X_son:= frm_ana.mmToPulseX(StrToFloat(edt_x_son_pos.Text)); // X son değerini formdan al
  Y_son:=frm_ana.mmToPulseY(StrToFloat(edt_y_son_pos.Text));  // Y son değerini formdan al
                                                                        // daireyi çiz komutu
  _8164_start_ta_arc_xyu(0,X_boyu,Y_boyu,X_son,Y_son,0,0,hiz,k_rampa);
end;

procedure Tfrm_manuel.FormCreate(Sender: TObject);
begin
 DecimalSeparator:= '.';                 // ayracı nokta olarak belirle
end;

procedure Tfrm_manuel.btn_sabit_gitClick(Sender: TObject);
begin
  hiz:=StrToFloat(edt_hiz_max.Text);                    // hızı değerini form'dan al
  k_rampa:=StrToFloat(edt_kalkis.Text);                 // rampa değerini form'dan al
                                                        // form'dan seçilenleri sabit hızla yürüt
  if chb_x.Checked then _8164_tv_move(0,0,hiz,k_rampa);
  if chb_y.Checked then _8164_tv_move(1,0,hiz,k_rampa);
  if chb_z.Checked then _8164_tv_move(2,0,hiz,k_rampa);
 end;


procedure  Tfrm_manuel.sbt_y_posMouseDown(Sender: TObject; Button: TMouseButton; Shift:
TShiftState; X, Y: Integer);
begin
  hiz:=StrToFloat(edt_hiz_max.Text);               // hızı değerini form'dan al
  k_rampa:=StrToFloat(edt_kalkis.Text);            // rampa değerini form'dan al
   _8164_tv_move(1,0,hiz,k_rampa);                 // Y ekseni sabit hızda artı yönde harekete başla
end;

 procedure Tfrm_manuel.sbt_y_posMouseUp(Sender: TObject;
  Button: TMouseButton; Shift: TShiftState; X, Y: Integer);
Begin
```

Tablo A.1. Devam

```
d_rampa:=StrToFloat(edt_dur_rampa.Text);     // rampa değerini form'dan al
 _8164_sd_stop(1,d_rampa)                     // Y eksenini durdur
end;

procedure Tfrm_manuel.sbt_y_minMouseDown(Sender: TObject;
  Button: TMouseButton; Shift: TShiftState; X, Y: Integer);
begin
   hiz:=StrToFloat(edt_hiz_max.Text);          // hızı değerini form'dan al
  k_rampa:=StrToFloat(edt_kalkis.Text);        // rampa değerini form'dan al
   _8164_tv_move(1,0,hiz*(-1),k_rampa);       // Y ekseni sabit hızda eksi yönde harekete başla
end;


procedure Tfrm_manuel.sbt_x_posMouseDown(Sender: TObject;
  Button: TMouseButton; Shift: TShiftState; X, Y: Integer);
begin
  hiz:=StrToFloat(edt_hiz_max.Text);           // hızı değerini form'dan al
  k_rampa:=StrToFloat(edt_kalkis.Text);        // rampa değerini form'dan al
  _8164_tv_move(0,0,hiz,k_rampa);             // X ekseni sabit hızda artı yönde harekete başla
end;


procedure Tfrm_manuel.sbt_x_minMouseDown(Sender: TObject;
  Button: TMouseButton; Shift: TShiftState; X, Y: Integer);
begin
  hiz:=StrToFloat(edt_hiz_max.Text);           // hızı değerini form'dan al
  k_rampa:=StrToFloat(edt_kalkis.Text);        // rampa değerini form'dan al
  _8164_tv_move(0,0,hiz*(-1),k_rampa);        // X ekseni sabit hızda eksi yönde harekete başla
end;


procedure Tfrm_manuel.sbt_x_posMouseUp(Sender: TObject;
  Button: TMouseButton; Shift: TShiftState; X, Y: Integer);
begin
 d_rampa:=StrToFloat(edt_dur_rampa.Text);     // rampa değerini form'dan al
 _8164_sd_stop(0,d_rampa)                     // X eksenini durdur
end;
```

ÖZGEÇMİŞ

Cihan KARAMAN, 1980 yılında İstanbul'da doğdu. İlk öğrenimini Malatya'da, orta ve lise öğrenimini İstanbul'da tamamladı. 1998 yılında kazandığı Sakarya Üniversitesi Elektrik-Elektronik Mühendisliği Bölümü'nden 2002 yılında mezun oldu. 2002 yılında LES sınavını kazanarak Sakarya Üniversitesi Elektrik Mühendisliği'nde Lisans üstü eğitime hak kazandı. Halen Yüksek Lisans öğrencisi olarak eğitimine devam eden Karaman, 2001 yılında başladığı LSE otomasyon firmasının proje bölümünde çalışmaktadır.