

**T.C.  
AKDENİZ ÜNİVERSİTESİ  
FEN BİLİMLERİ ENSTİTÜSÜ**

**ENDÜSTRİYEL AMAÇLI LİNEER MAKAS KONTROLÜ VE  
OPTİMİZASYONU**

**Ozan KOŞUN**

**YÜKSEK LİSANS TEZİ  
ELEKTRİK-ELEKTRONİK MÜHENDİSLİĞİ ANABİLİM DALI**

**2016**



## ÖZET

### ENDÜSTRİYEL AMAÇLI LİNEER MAKAS KONTROLÜ VE OPTİMİZASYONU

**Ozan KOŞUN**

**Yüksek Lisans Tezi, Elektrik-Elektronik Mühendisliği Anabilim Dalı**

**Danışman: Doç. Dr. Şükrü ÖZEN**

**Haziran 2016, 95 sayfa**

Günümüzde otomatik kontrole dayalı üretim, modern sanayinin temel yapıtaşıdır. Gelişen teknolojiyle beraber firmaların kaliteli, daha ekonomik ve seri bir şekilde imalat yapabilmeleri gerekmektedir. Bu bağlamda, endüstriyel otomasyon ve robotların kullanıldığı uzman sistemlere olan ihtiyaç artmaktadır.

Seri malzeme üreticilerinin çoğunda halen son ürün ebatlama için dayama yöntemiyle hizalanan ürün durdurularak kesme işlemi yapılmakta ve kesim bittikten sonra arkadan ürün akmaya devam etmektedir. Bu şekilde üretimin zaman ve maliyet açısından ne kadar zarar getireceği aşikar olmakla birlikte hali hazırda kullanılan lineer makas uygulamalarının hala teknik altyapıdan uzak, yazılım ve elektronik kontrol donanımı bakımından eksik olduğu gözlemlenmiştir.

Endüstride halen kullanılmakta olan lineer makas uygulamalarında öngörülen iki temel sorun bulunmaktadır. Bunlar; çıkan ürün boyunun hata oranının yüksek olması ve birim zamanda çıkan ürün sayısının az olmasıdır. Bu çalışmada; bu iki temel soruna yönelik araştırma yapılmış, hareket halindeki malzemelerin istenen boyutlarda en az hatayla otomatik olarak kontrollü bir biçimde kesilmesi için algoritma geliştirilmiştir. Programı yazılan makine uygulaması sanayide denenmiş ve +/- 0.2 mm boy hassasiyetle, emsallerine göre birim zamanda iki kat ürün çıkardığı görülmüştür.

**ANAHTAR KELİMELER:** Lineer Makas, Uçan Makas, Uçan Testere, Senkron Hareket Kontrolü, Somachine, PLCopen, PLCmotion

**JÜRİ:** Doç.Dr.Şükrü ÖZEN (Danışman)  
Yrd.Doç.Dr.Yavuz ÜSER  
Yrd.Doç.Dr.Mesud KAHRİMAN

## **ABSTRACT**

### **GENERAL PURPOSE LINEAR SHEAR APPLICATION AND OPTIMISATION**

**Ozan KOŞUN**

**MSc Thesis in Electric-Electronic Engineering**

**Supervisor: Assoc. Prof. Dr. Şükrü ÖZEN**

**June 2016, 95 pages**

Nowadays, the automatic control based on the production is the cornerstone of modern industry. With the developing technology, companies need to manufacture the higher quality and more economical. In this context, it is increasing the need for expert systems used in industrial automation and robots.

For the sizing of end product, product is stopping that aligned with using docking method and after the cutting process the products flow from behind in too many factories that mass produce still. In this way it is clear that the production brings how much harm from the point of time and cost. However observed that used linear scissors applications still do not have technical base and they have many deficiencies with regard to software and electronic control hardware.

There are two main convenient issues on currently used linear scissors applications in industry. These are, failure rate of produced product length being high and number of produced product in unit of time being low. In this study survey was done for these two main issues and the algorithm was improved to cut the moving stuffs automatically and controlledly in needed dimensions with least failure. A programmed machine application was tested in factory and seen that number of produced product in unit of time being double with +/- 0.2 mm accuracy.

**KEYWORDS:** Linera Shear, Flying Shear, Flying Saw, Synchronous Motion Control, Somachine, PLCopen, PLCmotion

**COMMITTEE:** Assoc. Prof. Dr. Şükrü ÖZEN (Supervisor)  
Asst.Prof.Dr.Yavuz ÜSER  
Asst.Prof.Dr.Mesud KAHRİMAN

## ÖNSÖZ

Tez çalışması boyunca çalışmalarına yön veren ve desteğini hiç esirgemeyen değerli tez danışmanım Doç.Dr.Şükrü ÖZEN'e, teknik destek ve çalışma alanı sağlayan ETCOMA Mühendislik A.Ş. şirketine, maddi manevi desteklerini esirgemeyen aileme ve arkadaşlarıma teşekkürü bir borç bilirim...

Ozan KOŞUN  
Antalya, Haziran 2016



## İÇİNDEKİLER

|                                                    |     |
|----------------------------------------------------|-----|
| ÖZET                                               | i   |
| ABSTRACT                                           | ii  |
| ÖNSÖZ                                              | iii |
| İÇİNDEKİLER                                        | iv  |
| SİMGELER VE KISALTMALAR DİZİNİ                     | vii |
| ŞEKİLLER DİZİNİ                                    | ix  |
| ÇİZELGELER DİZİNİ                                  | xii |
| 1. GİRİŞ                                           | 1   |
| 2. KURAMSAL BİLGİLER VE KAYNAK TARAMALARI          | 3   |
| 2.1. Kontrol Sistemi                               | 3   |
| 2.1.1. Kontrol sistemi kavramları                  | 3   |
| 2.1.2. Kontrol sistemi türleri                     | 3   |
| 2.1.2.1. Açık çevrimli kontrol sistemleri          | 4   |
| 2.1.2.2. Kapalı çevrimli kontrol sistemleri        | 5   |
| 2.2. Servomekanizma                                | 6   |
| 2.2.1. Hız kontrol                                 | 7   |
| 2.2.2. Pozisyon kontrol                            | 8   |
| 2.2.3. Moment kontrol                              | 8   |
| 2.2.4. Hibrit kontrol                              | 9   |
| 2.3. Servo Motorlar                                | 9   |
| 2.3.3. Servo motor terminolojisi                   | 10  |
| 2.3.4. Servo motor seçiminin önemi ve hesaplanması | 11  |
| 2.4. Geri Besleme Elemanları                       | 12  |
| 2.4.1. Takometre                                   | 13  |
| 2.4.2. Resolver                                    | 13  |
| 2.4.3. Mutlak tip (absolute) enkoder               | 13  |
| 2.4.4. Artımsal tip (incremental) enkoder          | 13  |
| 2.5. PLC (Programlanabilir Lojik Kontrolör)        | 14  |
| 2.5.1. PLC'nin kullanım amacı ve alanları          | 14  |
| 2.5.2. PLC'nin genel çalışma mantığı               | 16  |
| 2.6. CoDeSys (Controller Development System)       | 18  |
| 2.6.1. IEC-61131-3 standardı                       | 19  |
| 2.6.2. CoDeSys platformundaki veri tipleri         | 19  |
| 2.6.3. CoDeSys platformundaki programlama dilleri  | 20  |
| 2.6.3.1. Komut listesi (IL)                        | 20  |
| 2.6.3.2. Yapısal metin (ST)                        | 20  |
| 2.6.3.3. Merdiven dili (LD)                        | 21  |
| 2.6.3.4. Fonksiyon blok diyagram (FBD)             | 21  |
| 2.6.3.5. Grafik editör (CFC)                       | 22  |

|                                                                        |    |
|------------------------------------------------------------------------|----|
| 2.6.3.6. Ardışık fonksiyon gösterimi (SFC)-----                        | 22 |
| 2.7. CanBus Haberleşme Protokolü-----                                  | 23 |
| 2.8. CANopen Katman Protokolü-----                                     | 25 |
| 2.9. Lineer Makas (Uçan Makas, Uçan Testere)-----                      | 26 |
| 2.9.1. Senkronizasyon işlem sırası-----                                | 27 |
| 2.9.2. Pozisyona senkronizasyon-----                                   | 28 |
| 2.9.3. Parametrelendirilebilir sınır koşulları-----                    | 28 |
| 2.9.4. Senkronizasyon koşulları-----                                   | 28 |
| 2.10. CAM Profili (Kam)-----                                           | 29 |
| 2.10.1. Mekanik CAM-----                                               | 29 |
| 2.10.2. Elektronik CAM-----                                            | 31 |
| 2.10.3. CAM ile döner bıçak uygulaması-----                            | 31 |
| 2.11. PLCopen Blokları-----                                            | 34 |
| 2.11.1. Mc_Power fonksiyon bloğu-----                                  | 35 |
| 2.11.2. Mc_ReadAxisError fonksiyon bloğu-----                          | 35 |
| 2.11.3. Mc_Reset fonksiyon bloğu-----                                  | 36 |
| 2.11.4. Mc_ReadStatus fonksiyon bloğu-----                             | 37 |
| 2.11.5. Mc_ReadActualPosition fonksiyon bloğu-----                     | 39 |
| 2.11.6. Mc_ReadActualVelocity fonksiyon bloğu-----                     | 39 |
| 2.11.7. Mc_ReadActualTorque fonksiyon bloğu-----                       | 40 |
| 2.11.8. Mc_Home fonksiyon bloğu-----                                   | 41 |
| 2.11.9. Mc_Jog fonksiyon bloğu-----                                    | 42 |
| 2.11.10. Mc_MoveVelocity fonksiyon bloğu-----                          | 44 |
| 2.11.11. Mc_Stop fonksiyon bloğu-----                                  | 46 |
| 2.11.12. Mc_MoveAbsolute fonksiyon bloğu-----                          | 47 |
| 2.11.13. Mc_MoveRelative fonksiyon bloğu-----                          | 48 |
| 2.11.14. Mc_MoveSuperImposed fonksiyon bloğu-----                      | 50 |
| 2.11.15. Mc_SetPosition fonksiyon bloğu-----                           | 51 |
| 2.11.16. Mc_GearIn fonksiyon bloğu-----                                | 52 |
| 2.11.17. Mc_GearInPos fonksiyon bloğu-----                             | 54 |
| 2.11.18. Mc_GearOut fonksiyon bloğu-----                               | 55 |
| 2.11.19. Mc_TouchProbe fonksiyon bloğu-----                            | 56 |
| 2.11.20. Tüm fonksiyon bloklarda aynı anlama gelen giriş-çıkışlar----- | 59 |
| 2.11.21. PLCopen motion blokları durum diyagramı-----                  | 59 |
| 3. MATERYAL VE METOT-----                                              | 60 |
| 3.1. Uygulamada Kullanılan Donanımlar ve Mekanik Tasarım-----          | 60 |
| 3.2. PLC ve HMI Programının Yazıldığı SoMachine Platformu-----         | 64 |
| 3.2.1. Somachine program yapısı-----                                   | 64 |
| 3.2.2. Operatör paneli programı: Vijeo Designer-----                   | 65 |
| 3.3. PLC Programındaki İşleyiş ve Algoritma-----                       | 66 |
| 3.3.1. Pres grubu-----                                                 | 67 |
| 3.3.2. Rollform grubu-----                                             | 69 |

|                                                                             |    |
|-----------------------------------------------------------------------------|----|
| 3.3.3. Lineer makas grubu-----                                              | 69 |
| 3.3.4. Manuel çalışma-----                                                  | 75 |
| 3.4. Operatör Paneli Manuel ve Otomatik Kontrol Sayfaları-----              | 76 |
| 4. BULGULAR VE TARTIŞMA -----                                               | 80 |
| 5. SONUÇLAR -----                                                           | 82 |
| 6. KAYNAKLAR -----                                                          | 84 |
| 7. EKLER-----                                                               | 87 |
| EK-1: Pres grubu için geliştirilen fonksiyon bloğu program kodu -----       | 87 |
| EK-2: PLCopen motion blokları durum diyagramı-----                          | 94 |
| EK-3: Tüm Sistemin PLC ve HMI programının bulunduğu proje dosyası.---       | 95 |
| EK-4: Otomasyon panosunun elektrik projesi.-----                            | 95 |
| EK-5: Hattın kurulduğu yerde tam kapasite çalışırken çekilmiş videosu.----- | 95 |
| ÖZGEÇMİŞ                                                                    |    |

## SİMGELER VE KISALTMALAR DİZİNİ

### Simgeler

|           |                                         |
|-----------|-----------------------------------------|
| $a$       | : İvme (m/s <sup>2</sup> )              |
| $A$       | : Akım Şiddeti (Amper)                  |
| $c$       | : Sabit Sayı                            |
| $d$       | : Yoğunluk                              |
| $f$       | : Frekans (Hz)                          |
| $F$       | : Kuvvet                                |
| $G(s)$    | : Transfer Fonksiyonu                   |
| $J_L$     | : Yük Momenti                           |
| $J_r$     | : Rotor Eylemsizlik Momenti             |
| $l$       | : Uzunluk                               |
| $m$       | : Kütle                                 |
| $M_o$     | : Servo Kalkış Momenti (N.m)            |
| $N$       | : Newton (kg.m/s <sup>2</sup> )         |
| $P_{kW}$  | : Gerekli Servo Gücü (kW)               |
| $t$       | : Süre (s)                              |
| $T$       | : Sistem Zaman Sabitesi                 |
| $t_{acc}$ | : Hızlanma İvme Süresi (s)              |
| $t_{dec}$ | : Yavaşlama İvme Süresi (s)             |
| $t_{sum}$ | : Toplam Hareket Süresi (s)             |
| $t_{max}$ | : Maksimum Hızda Gitme Süresi (s)       |
| $V_{max}$ | : Maksimum Hız (m/s)                    |
| $x_{acc}$ | : Hızlanma Boyunca Katedilen Mesafe (m) |
| $Y(s)$    | : Sistem Çıkışı                         |

## **Kısaltmalar**

|      |                                        |
|------|----------------------------------------|
| AN   | : Analog                               |
| ASM  | : Asenkron Servo Motor                 |
| CFC  | : Devamlı Fonksiyon Çizelgesi          |
| CNC  | : Bilgisayar Sayımlı Yönetim           |
| CPU  | : Merkezi İşlem Birimi                 |
| CRC  | : Periyodik Fazlalık Kontrolü          |
| DCSM | : DC Servo Motor                       |
| FBD  | : Fonksiyon Blok Diyagramı             |
| HMI  | : İnsan Makine Arayüzü                 |
| IEC  | : Uluslararası Elektroteknik Komisyonu |
| IL   | : Komut Listesi                        |
| IN   | : Fiziksel Giriş                       |
| LD   | : Merdiven Diyagramı                   |
| OUT  | : Fiziksel Çıkış                       |
| PLC  | : Programlanabilir Lojik Kontrolör     |
| RAM  | : Rastgele Erişilebilir Bellek         |
| ROM  | : Sadece Okunabilir Bellek             |
| SFC  | : Sıralı Fonksiyon Çizelgesi           |
| SSM  | : Senkron Servo Motor                  |
| ST   | : Yapısal Metin                        |

## ŞEKİLLER DİZİNİ

|                                                                       |    |
|-----------------------------------------------------------------------|----|
| Şekil 1.1. Tambur Tipi Uçan Makas .....                               | 2  |
| Şekil 2.1. Kontrol Sistemi .....                                      | 3  |
| Şekil 2.2. Açık Çevrim Kontrol Sistemi .....                          | 4  |
| Şekil 2.3. Kapalı Çevrim Kontrol Sistemi .....                        | 5  |
| Şekil 2.4. Sistemin Zaman Sabitesinin Değiştirilmesi. ....            | 6  |
| Şekil 2.5. Servo Hız Kontrol Sistemi .....                            | 7  |
| Şekil 2.6. Servo Pozisyon Kontrol Sistemi .....                       | 8  |
| Şekil 2.7. Servo Tork Kontrol Sistemi .....                           | 8  |
| Şekil 2.8. Servo Motorların Sınıflandırılması .....                   | 10 |
| Şekil 2.9. Geri Besleme Elemanları .....                              | 12 |
| Şekil 2.10. Standart Enkoder ve Temel İç Yapısı .....                 | 13 |
| Şekil 2.11. PLC İç Organizasyon Şeması .....                          | 14 |
| Şekil 2.12. PLC Çalışma Şeması .....                                  | 16 |
| Şekil 2.13. Örnek Kumanda Devresi.....                                | 20 |
| Şekil 2.14. Örnek IL Programı .....                                   | 20 |
| Şekil 2.15. Örnek ST Programı .....                                   | 21 |
| Şekil 2.16. Örnek LD Programı .....                                   | 21 |
| Şekil 2.17. Örnek FBD Programı .....                                  | 22 |
| Şekil 2.18. Örnek CFC Programı .....                                  | 22 |
| Şekil 2.19. Örnek SFC Programı.....                                   | 22 |
| Şekil 2.20. CanBus Şeması .....                                       | 23 |
| Şekil 2.21. CanBus Lojik Seviyesi.....                                | 24 |
| Şekil 2.22. İkili Düğüm Veri Gönderirken Yol Durumu ve Lojik VE ..... | 24 |
| Şekil 2.23. CanBus Tahkimi .....                                      | 25 |
| Şekil 2.24. Örnek Uçan Makas Mekanikliği .....                        | 27 |
| Şekil 2.25. CAM ile Hareket Ettirilen Taşıyıcı.....                   | 30 |
| Şekil 2.26. CAM Profiline Göre Taşıyıcı Hareketi.....                 | 30 |
| Şekil 2.27. Boy Uzunluğuna Göre Bıçak Pozisyonu .....                 | 33 |
| Şekil 2.28. Mc_Power Fonksiyon Bloğu .....                            | 35 |
| Şekil 2.29. Mc_ReadAxisError Fonksiyon Bloğu .....                    | 36 |
| Şekil 2.30. Mc_Reset Fonksiyon Bloğu.....                             | 37 |
| Şekil 2.31. Mc_ReadStatus Fonksiyon Bloğu .....                       | 37 |
| Şekil 2.32. Mc_ReadActualPosition Fonksiyon Bloğu .....               | 39 |
| Şekil 2.33. Mc_ReadActualVelocity Fonksiyon Bloğu.....                | 39 |
| Şekil 2.34. Mc_ReadActualTorque Fonksiyon Bloğu .....                 | 40 |
| Şekil 2.35. Mc_Home Fonksiyon Bloğu.....                              | 41 |
| Şekil 2.36. Genel Servo Mekanizması .....                             | 41 |
| Şekil 2.37. Mc_Jog Fonksiyon Bloğu .....                              | 42 |
| Şekil 2.38. Jerk Etkisi Grafiği .....                                 | 43 |
| Şekil 2.39. Mc_MoveVelocity Fonksiyon Bloğu .....                     | 45 |

|                                                                       |    |
|-----------------------------------------------------------------------|----|
| Şekil 2.40. Mc_MoveVelocity Örnek Program .....                       | 45 |
| Şekil 2.41. Mc_MoveVelocity Örnek Grafik.....                         | 46 |
| Şekil 2.42. Mc_Stop Fonksiyon Bloğu .....                             | 46 |
| Şekil 2.43. Mc_MoveAbsolute Fonksiyon Bloğu.....                      | 47 |
| Şekil 2.44. Mc_MoveRelative Fonksiyon Bloğu .....                     | 48 |
| Şekil 2.45. Mc_MoveAbsolute ile Mc_MoveRelative Farkı .....           | 49 |
| Şekil 2.46. Konveyör Örneği .....                                     | 50 |
| Şekil 2.47. Mc_MoveSuperImposed Fonksiyon Bloğu .....                 | 51 |
| Şekil 2.48. Mc_MoveSetPosition Fonksiyon Bloğu .....                  | 52 |
| Şekil 2.49. Mc_GearIn Fonksiyon Bloğu .....                           | 53 |
| Şekil 2.50. Mc_GearInPos Fonksiyon Bloğu.....                         | 54 |
| Şekil 2.51. Mc_GearInPos Fonksiyon Bloğu Zamanlama Diagramı.....      | 55 |
| Şekil 2.52. Mc_GearOut Fonksiyon Bloğu.....                           | 56 |
| Şekil 2.53. Mc_TouchProbe Fonksiyon Bloğu .....                       | 56 |
| Şekil 2.54. Mc_TouchProbe Fonksiyon Bloğu ile WindowOnly Örneği ..... | 57 |
| Şekil 3.1. Hattın Modellemesi.....                                    | 61 |
| Şekil 3.2. Rulo Açıcı Önündeki Loop Mekanizması .....                 | 61 |
| Şekil 3.3. Rulo Açıcı .....                                           | 62 |
| Şekil 3.4. Pres Grubu .....                                           | 62 |
| Şekil 3.5. Piston Grubu Sonundaki Saç Süren Servo Mekanizması .....   | 63 |
| Şekil 3.6. Lineer Makas .....                                         | 63 |
| Şekil 3.7. Somachine Arayüzü .....                                    | 64 |
| Şekil 3.8. Vijeo Designer Arayüzü .....                               | 65 |
| Şekil 3.9. Otomatik Mod Algoritma Şeması.....                         | 66 |
| Şekil 3.10. Pres Grubu İçin Geliştirilen Fonksiyon Bloğu .....        | 67 |
| Şekil 3.11. Ekran-Reçete Seçim Sayfası .....                          | 68 |
| Şekil 3.12. Ekran-Pres Servoları Pozisyonlama Reçetesi .....          | 68 |
| Şekil 3.13. Ekran-Pres Seçim Reçetesi .....                           | 69 |
| Şekil 3.14. Ekran-Pres Grubu Adım Reçetesi.....                       | 69 |
| Şekil 3.15. Kenar Sensörü.....                                        | 70 |
| Şekil 3.16. Delik Sensörü.....                                        | 70 |
| Şekil 3.17. Rollform ve Encoder.....                                  | 71 |
| Şekil 3.18. Programda Kullanılan Mc_GearInPos Fonksiyon Bloğu .....   | 72 |
| Şekil 3.19. KG_Otomatik_LD Programı 1 .....                           | 73 |
| Şekil 3.20. KG_Otomatik_LD Programı 2 .....                           | 74 |
| Şekil 3.21. KG_Otomatik_LD Programı 3 .....                           | 75 |
| Şekil 3.22. Ana Sayfası .....                                         | 76 |
| Şekil 3.23. Manuel Sayfası .....                                      | 76 |
| Şekil 3.24. Rulo Açıcı – Pres Grubu Manuel Sayfası.....               | 77 |
| Şekil 3.25. Pres Grubu Servoları – Rollform Manuel Sayfası .....      | 77 |
| Şekil 3.26. Kesme Grubu Manuel Sayfası .....                          | 77 |
| Şekil 3.27. Alarm Sayfası .....                                       | 78 |

|                                                                                    |    |
|------------------------------------------------------------------------------------|----|
| Şekil 3.28. Parametre – Zamanlar Sayfası .....                                     | 78 |
| Şekil 3.29. Reçete Seçim – Pres Grubu Ölçü Reçete Sayfası .....                    | 78 |
| Şekil 3.30. Pres Grubu Pres Seçim Reçete – Pres Grubu Pres Pozisyon Reçete Sayfası | 79 |
| Şekil 5.1. Üretilen Ürün Yandan ve Üstten Görünümü .....                           | 82 |
| Şekil 5.2. Üretilen Ürün Kesildikten Sonra Üstten Görünümü.....                    | 82 |
| Şekil 5.3. Üretilen Ürün Son Hali .....                                            | 83 |



## ÇİZELGELER DİZİNİ

|                                                                              |    |
|------------------------------------------------------------------------------|----|
| Çizelge 2.1. Veri Tipleri Çizelgesi.....                                     | 19 |
| Çizelge 2.2. Mc_Power Fonksiyon Bloğu Giriş-Çıkış Çizelgesi.....             | 35 |
| Çizelge 2.3. Mc_ReadAxisError Fonksiyon Bloğu Giriş-Çıkış Çizelgesi.....     | 36 |
| Çizelge 2.4. Mc_Reset Fonksiyon Bloğu Giriş-Çıkış Çizelgesi.....             | 37 |
| Çizelge 2.5. Mc_ReadStatus Fonksiyon Bloğu Giriş-Çıkış Çizelgesi.....        | 38 |
| Çizelge 2.6. Mc_ReadActualPosition Fonksiyon Bloğu Giriş-Çıkış Çizelgesi.... | 39 |
| Çizelge 2.7. Mc_ReadActualVelocity Fonksiyon Bloğu Giriş-Çıkış Çizelgesi ..  | 40 |
| Çizelge 2.8. Mc_ReadActualTorque Fonksiyon Bloğu Giriş-Çıkış Çizelgesi ....  | 40 |
| Çizelge 2.9. Mc_Home Fonksiyon Bloğu Giriş-Çıkış Çizelgesi .....             | 42 |
| Çizelge 2.10. Mc_Jog Fonksiyon Bloğu Giriş-Çıkış Çizelgesi .....             | 43 |
| Çizelge 2.11. Mc_MoveVelocity Fonksiyon Bloğu Giriş-Çıkış Çizelgesi.....     | 44 |
| Çizelge 2.12. Mc_Stop Fonksiyon Bloğu Giriş-Çıkış Çizelgesi .....            | 47 |
| Çizelge 2.13. Mc_MoveAbsolute Fonksiyon Bloğu Giriş-Çıkış Çizelgesi .....    | 48 |
| Çizelge 2.14. Mc_MoveRelative Fonksiyon Bloğu Giriş-Çıkış Çizelgesi .....    | 49 |
| Çizelge 2.15. Mc_MoveSuperImposed Fonksiyon Bloğu Giriş-Çıkış Çizelgesi      | 51 |
| Çizelge 2.16. Mc_MoveSetPosition Fonksiyon Bloğu Giriş-Çıkış Çizelgesi ..... | 52 |
| Çizelge 2.17. Mc_GearIn Fonksiyon Bloğu Giriş-Çıkış Çizelgesi.....           | 53 |
| Çizelge 2.18. Mc_GearInPos Fonksiyon Bloğu Giriş-Çıkış Çizelgesi .....       | 54 |
| Çizelge 2.19. Mc_GearOut Fonksiyon Bloğu Giriş-Çıkış Çizelgesi .....         | 56 |
| Çizelge 2.20. Mc_TouchProbe Fonksiyon Bloğu Giriş-Çıkış Çizelgesi .....      | 58 |
| Çizelge 2.21. TriggerRef Register Çizelgesi.....                             | 58 |
| Çizelge 2.22. Fonksiyon Blokları Genel Giriş-Çıkış Çizelgesi .....           | 59 |

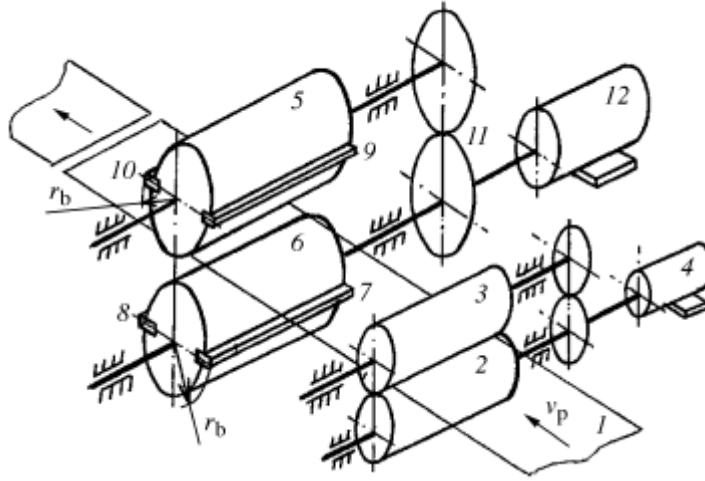
## 1. GİRİŞ

Son yıllarda yurt içi ve yurt dışı pazarda rekabet edebilmek için daha iyi, ucuz maliyetli ve aynı zamanda kaliteli üretim yapabilmek gereksiniminin hızla artması sebebi ile firmaların tasarım ve yeni imalat yöntemlerine ağırlık vermeleri kaçınılmaz bir hale gelmiştir. Geçmişte çizilen ekonomik sınırların aksine günümüz koşullarında, sektörel bazda rekabete katılan firma sayısı sürekli artmaktadır. Bu koşullar altında firmaların geleceği, günümüz rekabet koşullarına olan uyumuna bağlıdır. Bu aşamada, ülkemizde ve dünyada imalatı yapılan parçalarının daha seri bir şekilde ve kısa süre içerisinde üretilmesi ön plana çıkmaya başlamıştır. Firmaların; kontrol sistemleri ve teknolojiye kaydedilen gelişmelere paralel, kaliteli, ekonomik ve seri şekilde imalat yapabileceği makine ve teçhizatlar ile oluşturulan üretim hatlarını benimsemeleri modern çağın bir gereğidir. Seri üretimin temel gayesi birim zamanda üretilen ürün miktarını artırmaktır. Bu durum, uygulamada, üretim hattı boyunca her noktada ölü zamanları azaltma ve ürünün işleneceği makine ve diğer donanımların dinamik performanslarını artırma ihtiyacını doğurur.

Seri malzeme üreticilerinin çoğunda halen ürün ebatlamak için dayama yöntemiyle hizalanan ürün durdurularak kesme işlemi yapılmakta ve kesim bittikten sonra arkadan ürün akmaya devam etmektedir. Bu şekilde üretimin zaman ve maliyet açısından ne kadar zarar getireceği aşikar olmakla birlikte hali hazırda kullanılan lineer makas uygulamalarının hala teknik altyapıdan uzak, yazılım ve elektronik kontrol donanım bakımından eksik olduğu gözlemlenmiştir. Algoritma geliştirmeden önce halen kullanılmakta olan otomasyon sistemlerindeki sorunlar mühendislik seviyesinde incelenmiştir. Endüstride halen kullanılmakta olan lineer makas uygulamalarında öngörülen iki temel sorun bulunmaktadır. Bunlar; çıkan ürün boyunun hata oranının yüksek olması ve birim zamanda çıkan ürün sayısının az olmasıdır. Endüstride uygulamada karşılaşılan bu problemlerin mekanik tasarım problemlerinden mi, PLC-sürücü-motor yetersizliğinden mi yoksa algoritma zayıflığından mı kaynaklandığı incelenmiş, alanla ilgili çalışma yapıp alan dışı konularda uzman kişilerden görüşler alınmıştır.

Endüstride lineer makas ve benzeri otomasyon uygulamaları mevcut olmasına rağmen, bu tip makineler özel makine olarak geçmekte ve ihtiyaca uygun olarak özel üretilmekte olup tanımlanmış yerleşik herhangi bir standardı bulunmamaktadır. Bu yöneme genel olarak iki servo motor senkronizasyonu denebilir ve hakkında yapılan uygulamalar dışında bilimsel olarak çok az detay bulunmaktadır.

Bilimsel çalışmaların çoğu tambur tipi (drum-type) uçan makas (flying shear) üstünde yoğunlaşmıştır. Bu yöntemde Şekil1.1'de görüldüğü gibi ürün iki tambur arasından geçer. İki tamburun da üstünde ebatlama yapmak için bıçak mevcuttur ve karşılıklı geldikleri zaman ürünü kesmektedir. Üretilcek ürün boyu ise tamburun çevresel hızının programlanmasıyla ayarlanmaktadır (Peric ve Petrovic 1990, Kolesnikov 2004, Belobroy vd 2006). Bu yöntemin hız olarak avantajı olmasına rağmen sert ve kalın metal ürünler için işlevselliği bulunmamaktadır.



Şekil 1.1. Tambur Tipi Uçan Makas

Lineer makas (sanayide uçar/uçan makas/testere olarak da geçmektedir.) otomasyonu sanayide sıklıkla roll-form denen saca şekil veren mekanizmanın sonunda şekil almış metali ebatlamak için kullanılmaktadır. Lineer makas otomasyonu girişinde genelde düz saca delik ve belli boşluklar açmak için kullanılan pres grubu ve saca şekil vermek için kullanılan rollform grubu bulunmaktadır. Bu çalışma ile bu hatların da bulunduğu lineer makas otomasyonu uygulaması programı yazılmıştır. Plc programının yanında Hmi(operatör paneli) programı da yazılmıştır. Hmi’de yapılan reçete sistemi ile üretilen ürün için ayarların bir sefer kaydedilmesi yeterli olmaktadır. Daha sonra tekrar aynı ürün üretileninde reçeteden ürün seçilip hızlı şekilde otomatik üretime başlanabilmektedir.

Plc programında PLCmotion bloklarından faydalanılmıştır. PLCmotion blokları ise CanOpen haberleşme protokolü ile çalışmaktadır, bu yüzden tüm hareket eksenleri CanOpen haberleşme protokolü ile yönetilmiştir. Sistemde sanal eksen olarak tanımlanmış enkoder master kabul edilmiş ve lineer makas eksenini slave kabul edilmiştir. Slave eksen master eksene pozisyonel elektronik dişli fonksiyon bloğu olan Mc\_GearInPos ile bağlanmıştır. Bu blokların içinden paketleme uygulamaları için bir çözüm olan Mc\_TouchProbe bloğu amaca uygun olarak kullanılmıştır. Bu bloğun işlevsellik kazanması için pozisyon hedefi olarak kullanılan enkoder, sanal eksen olarak tanımlanmıştır. Bu yöntem ile delikli ürünlerde hata oranının azalması ve hatanın arkadan gelen ürünlerde toplanarak artmasının önüne geçilmiştir.

Sanayide genelde üretilen malzemenin hata toleransının +/- 1 cm olması beklenir. Hali hazırda en hassas parça işleme aracı olan CNC hassasiyeti +/- 0.05mm’dir. Çalışma sonucu geliştirilen algoritma ve uygun olarak tasarlanan mekanik ile tasarlanan lineer makas otomasyon hattının ürün boyu hassasiyetinin +/- 0.2 mm olduğu görülmüştür. Aynı zamanda uygun otomasyon ürünleri ve mekanik yapı seçilerek birim zamanda çıkan ürün sayısının da yaklaşık iki katına çıktığı görülmüştür.

## 2. KURAMSAL BİLGİLER VE KAYNAK TARAMALARI

### 2.1. Kontrol Sistemi

Sistem davranışının istenilen ve belirlenen iş akışı doğrultusunda çalışmasını sağlamak amacı ile yapılan çalışmalar kontrol şeklinde ifade edilir. Kontrol sistemi ise, kendisini ya da başka bir sistemi düzenlemek, kumanda etmek ya da yönetmek üzere uygun bir biçimde bağlanmış fiziksel elemanlar kümesidir.

Son yıllarda önemli bir bilim dalı haline gelen kontrol sistemleri, gerek literatürde gerekse de günlük uygulamalarda geniş bir kullanım alanına sahiptir. Kontrol sistemleri ile günlük yaşamda kullanılan birçok sistemi kontrol edebilmek mümkündür. Örneğin; evlerde kullanılan klimalar, buzdolapları, şofben vb. birçok elektronik cihaz bünyesinde bir kontrol sistemi barındırmakta ve belli bir ısı derecesinde çalışması için ayarlanabilmektedir. Bu cihazların bazıları tam otomatik sistemler olabildiği gibi bazıları ise aç-kapat sistem şeklinde çalışmaktadır (Megep 2007).

#### 2.1.1. Kontrol sistemi kavramları

Temelde bir kontrol sisteminin üç ögesi bulunmaktadır. Bu üç ögenin birbiriyle ilişkisi basitçe Şekil 2.1’de gösterilmiştir.



Şekil 2.1. Kontrol Sistemi

Sistem, belli bir işlev için bir araya getirilmiş, bir bütünü oluşturan ve birbiri ile bağlı olan elemanlar topluluğudur. Kontrolü istenilen sistemi kontrol etmek için uç elemanlardan sisteme alınan dijital, analog, haberleşme vb. veriler veya işaretler giriş şeklinde ifade edilmektedir. Çıkışlar ise, kontrol edilen sistemin girişlerinin, sistem dahilinde bulunan bir işlemcide sistemden istenilen fonksiyonu gerçekleştirecek şekilde hesaplanması ile elde edilen işaretlere verilen adlardır. Bir başka deyişle amaçlar (u girişleri) ya da sürücü işaretleri ile belirlenir, sonuçlar ise y çıkışları ya da kontrol edilen değişkenleri etkiler (Bozkurt 2006). Genel olarak kontrol sisteminin amacı, kontrol sisteminin elemanları aracılığı ile girişleri kullanarak, çıkışları önceden belirlenen senaryoya uygun şekilde kontrol etmeye çalışmaktır (Kuo 2005).

#### 2.1.2. Kontrol sistemi türleri

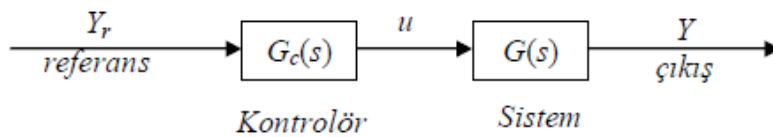
Kontrol sistemleri, çıkış işaretinin kontrolü ya da yönetilmesi açısından, geri besleme elemanının yapısında varlığına göre ikiye ayrılmaktadır. Geri besleme elemanı bulundurmayan sistemler Açık Çevrimli Kontrol Sistemi; bulunanlar ise Kapalı Çevrimli Kontrol Sistemleridir.

### 2.1.2.1. Açık çevrimli kontrol sistemleri

Sistemi kontrol eden düzeneğin, sistemin çıkışından etkilenmediği, sadece verilen referans değerine göre denetim işleminin yapıldığı sistemlerdir. Hassasiyet gerektirmeyen sistemlerde kullanılan bir denetim sistemi mekanizmasıdır. Sisteme etkiyen bozucu faktörlerin algılanması insan faktörüyle olabilmektedir. Verilen referans işareti kontrol elemanı tarafından alınır ve oransal bir kontrol işareti üretir. Bu işaret, kontrol edilen sisteme verildiğinde, sistem giriş değişkenini süreç içine alır ve istenilen çıkış işaretini verir. Açık çevrim denetim sistemi, genellikle kumanda edilen sistemin yapısının ve sisteme etkiyen diğer girişlerin önceden çok iyi bilindiği uygulamalarda kullanılır.

Bu sistemi daha iyi anlamak için trafik kontrolü yapan trafik ışıkları örneğini göz önüne alabiliriz. Bu sistemde belirli zamanlayıcılar bulunur ve ışıklar bu zaman ayarlayıcılara göre çalışırlar. Sisteme göre belli bir  $t_k$  zamanı boyunca kırmızı ışık yanar ve geçişleri durdurur. Diğer bir  $t_y$  zamanı boyunca da yeşil ışık yanarak trafik akışı sağlanır ve bu işlem  $t_k$  ve  $t_y$  için uygun görülen süreler aralığında sürekli olarak devam eder. Böyle bir kontrol sisteminin trafik yoğunluğunu ölçmeksizin aynı şekilde çalışması, verimli bir sistem olmadığına göstergesidir. Bu kontrol sisteminde girişler ( $t_k$ ,  $t_y$ ) kırmızı ve yeşil ışıkların yanma süreleri, çıkış ise trafik yoğunluğudur ve kontrol sisteminin girişleri, ışıkların yanma sürelerinden etkilenmemektedir. Farklı bir ifade ile belirtmek gerekirse kumanda işaretini sağlayan girişler, çıkışı oluşturan durumlardan bağımsızdırlar. Bu şekilde çalışan sistemler açık çevrim kontrol sistemini oluşturmaktadır. Bu sistemde, sistem çıkışı olan trafik yoğunluğu ölçülür, referans bir değerle karşılaştırılır ve trafik ışıklarının yanma süreleri bu işlemlere göre yeniden belirlenirse, bu yeni sistemin kapalı çevrim bir kontrol sistemi olduğu söylenilebilir.

Açık çevrim kontrol sistemleri kararlı olmalarına rağmen en önemli özellikleri sistem parametre değişimine karşı oldukça duyarlıdırlar. Örneğin Şekil 2.2’de verilen kontrolör sürekli- zamanlı ve transfer fonksiyonu  $G_c(s)=1$  ve sistem de sürekli-zamanlı ve transfer fonksiyonu  $G(s)$  olsun.



Şekil 2.2. Açık Çevrim Kontrol Sistemi

Eğer sistem parametrelerinde  $\Delta G(s)$  gibi bir artış olursa ve  $|G(s)| \gg |\Delta G(s)|$  ise sistemin çıkışı,

$$Y(s) + \Delta Y(s) = [G(s) + \Delta G(s)] Y_r(s) \quad (2.1)$$

dir ve sistem çıkışında  $\Delta Y(s) = \Delta G(s) Y_r(s)$  gibi bir fark oluşur. Yani sistemin nominal çalışma şartlarındaki çıkış  $Y(s)$  ise  $\Delta G(s)$  artışıyla sistem performansı  $\Delta G(s) U(s)$  kadar değişmiştir. Sistem girişteki performansı sağlamak için sistem giriş enerjisi bunu

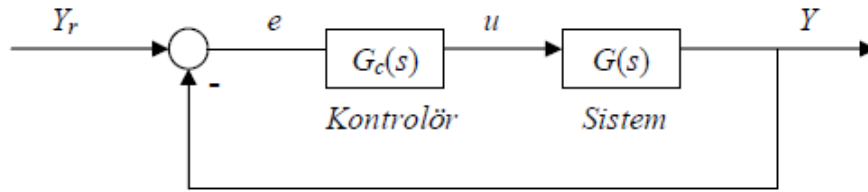
karşılama için nominal değerin üstünde giriş isteyecektir ancak bu sistemde zorlanmalara sebep olacak ve sonuç olarak hedeflenen girişten çok uzak bir çıkış elde edilebilir (Dumanay 2009).

### 2.1.2.2. Kapalı çevrimli kontrol sistemleri

Kontrol faaliyetinin, sistemin denetlenen çıkışına bağlı olduğu sistemdir. Kapalı çevrim kontrol sisteminin açık çevrim kontrol sisteminden en belirgin farkı “geri besleme” etkisidir.

Geri besleme denetlenen çıkış değişkeninin ölçülüp geri beslenerek istenen giriş değeri ile karşılaştırılmasını sağlar. Geri besleme negatif veya pozitif yönde etki edebilir. Negatif etkinin anlamı, çıkışın girişe ters yönde etki etmesidir. Pozitif etkinin anlamı ise çıkışın girişe aynı yönde etki etmesi demektir. Ancak kapalı çevrim denetim sisteminde kullanılan sistemler çoğunlukla negatif geri beslemelidirler (Sarioğlu 1985).

Kapalı çevrimli bir kontrol sistemi Şekil 2.3’de verilmiştir. Şekil 2.3’den de görüldüğü gibi kapalı çevrimli kontrol sistemlerinde çıkış girişle sürekli karşılaştırılmaktadır.



Şekil 2.3. Kapalı Çevrim Kontrol Sistemi

Sistemde  $\Delta G(s)$  gibi bir değişim olursa, Şekil 2.3’deki kontrol sisteminin çıkışı,

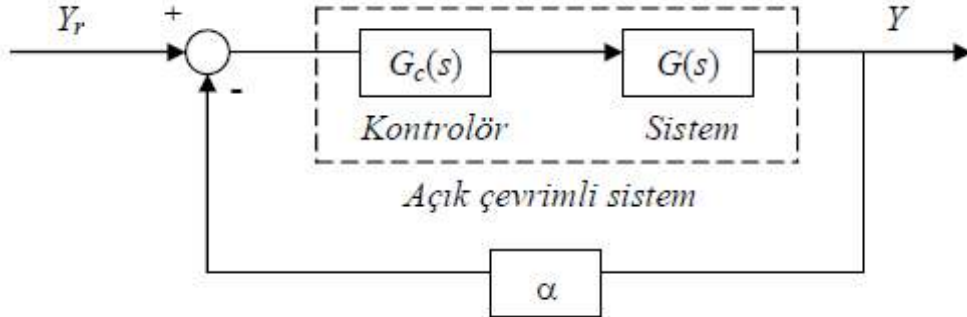
$$Y(s) + \Delta Y(s) = \frac{G(s) + \Delta G(s)}{1 + G(s) + \Delta G(s)} \quad (2.2)$$

ve  $|G(s)| \gg |\Delta G(s)|$  olduğundan çıkıştaki fark,

$$\Delta Y(s) = \frac{\Delta G(s)}{1 + G(s)} \quad (2.3)$$

$1 + G(s) \gg \Delta G(s)$  olduğundan çok küçük bir sayının çok büyük bir sayıya bölümünden meydana gelen  $\Delta Y(s)$  çok çok küçüktür. Dolayısıyla, sistem çıkışı  $\Delta G(s)$ ’e göre çok az duyarlıdır.

Kapalı çevrimli yada başka bir deyişle geri beslemeli kontrol sistemlerinin bir çok avantajları vardır. Bu avantajlardan biride Şekil 2.4'te gösterildiği gibi sistemin zaman sabitinin  $\alpha$  ile değiştirilmesidir.



Şekil 2.4. Sistemin Zaman Sabitesinin Değiştirilmesi.

Örneğin, kontrolör  $G_c(s) = K$  ve sistem birinci dereceden olsun. Bu durumda açık çevrimli sistemin transfer fonksiyonu,

$$G(s) = \frac{K}{Ts+1} \quad (2.4)$$

dir.  $\alpha$  katsayılı kapalı çevrim transfer fonksiyonu;

$$G(s) = \frac{\frac{K}{1+K\alpha}}{\left(\frac{T}{1+K\alpha}\right)s+1} \quad (2.5)$$

dir. İki denklem karşılaştırıldığında sistem zaman sabitesi  $T$ 'den  $T/(1+K\alpha)$  değerine ve sistem kazancı  $K$  değerinden  $K/(1+K\alpha)$  değerine düşmüştür.

## 2.2. Servomekanizma

Servo sözcüğü Latince "servus" sözcüğünden türetilmiştir; "hizmetçi", "köle", "yardımcı" anlamına gelmektedir. Servo motor sistemleri günümüzde pek çok uygulama alanında kullanılmaktadır. Bunun asıl nedeni, motor kontrol sistemlerinin kompakt yapıya sahip, kumandalarının basit ve verimli olmalarıdır. Servo sistemler aşağıda belirtilen özelliklerinden dolayı kontrol uygulamalarında çokça tercih edilmektedirler:

- Pozisyon doğruluğu,
- Hız doğruluğu,
- Tork kararlılığı,
- Yüklenebilme kapasitesi,
- Dinamik performans.

Servomekanizma olarak adlandırılan geri besleme kontrol sistemi endüstriyel uygulamalarda ve kontrol sistemi literatüründe yaygın olmasından dolayı dikkat çeken sistemdir (Ogata 1990).

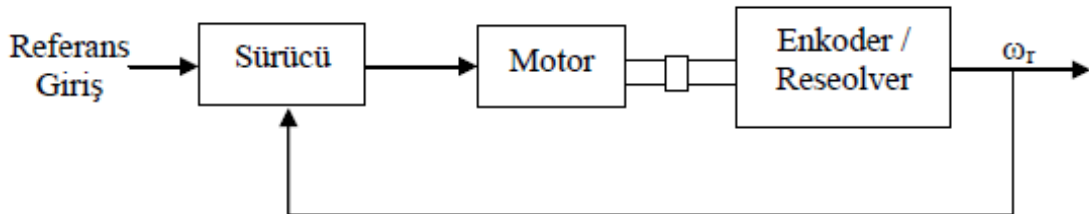
Servomekanizma, kapalı çevrimli geri beslemeye sahiptir. Bu sistem merkezi bir işlemci ya da denetleyici, işlemci ya da denetleyiciye komut gönderen bir arayüz, işlemciye gelen sinyalleri değerlendiren ve ileten bir arabirim, sinyalleri alan bir motor grubu ve motorun konum, hız ya da hata bilgisini tekrar denetleyiciye gönderen ve sistem çevrimini kapatan geri besleme elemanından oluşur (Bozkurt 2006). Servo sistem veya servomekanizma çıkışın, giriş tarafından, çalıştırılmasına neden olan bir hareket ve negatif geri besleme prensibi üzerine çalışan bir sistemdir (Younkin 1996). Servomekanizma otomatik geri beslemeli kontrol sistemidir. Servomekanizma tamamen otomatik olarak yapılabilir ve mekanik hareketleri çok küçük hata değerleri ile kontrol edilebilir. Servomekanizma kullanılan kontrol sinyaline bağlı olarak, dijital veya analog olabilir (Hall ve Clifford 2005). Bir servomekanizma şu parçalardan oluşur:

- 1) Motor (Servo, Asenkron)
- 2) Dişli kutusu (Bulunmayabilir.)
- 3) Motor Sürücüsü
- 4) Merkezi kontrol birimi (PLC, İşlemci vb.)
- 5) Güç ve Motor kablosu
- 6) Fren kablosu (Bulunmayabilir.)
- 7) Geri besleme (Resolver/Enkoder) kablosu
- 8) Kumanda sistemi ve kabloları

Servo kontrol sistemi veya servomekanizmada giriş genel olarak değişkendir ve sistem, çıkışı giriş sinyallerini ve bundaki değişimleri yakından takip edecek şekilde işler (Bozkurt 2006). Servo sistemler hız, pozisyon, moment (tork) ve hibrit (karma) kontrol olmak üzere dört farklı kontrol uygulama alanına sahiptir.

### 2.2.1. Hız kontrol

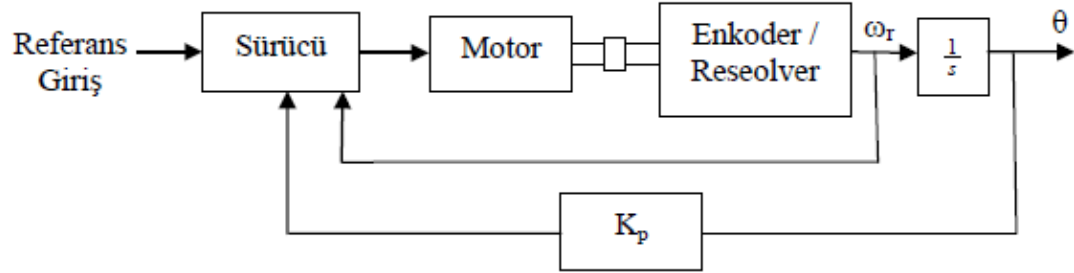
Hız kontrol sistemi; motor hızının ayarlandığı yapıdır ve servo sürücülerin hepsinde bulunan birimdir (Hancı 2007). Servo sistemde geri beslemeyi oluşturan elemandan alınan hız bilgisi sürücüye iletilerek referans giriş ile kıyaslanır ve oluşan fark motora iletilir. Servo hız kontrol sistemine ait şema Şekil 2.5’de verilmiştir.



Şekil 2.5. Servo Hız Kontrol Sistemi

### 2.2.2. Pozisyon kontrol

Pozisyon kontrolünde, motor milinden alınan dinamik hız bilgisi konum bilgisine çevrilir ve belirli bir kazanç sabiti ile çarpılır. Elde edilen negatif geri besleme sistem çıkışına tekrar yansıtılır. Pozisyon kontrol sistemi yapısında hız kontrol sistemini de barındırır. Pozisyon kontrol sistemi Şekil 2.6'da verilmiştir.

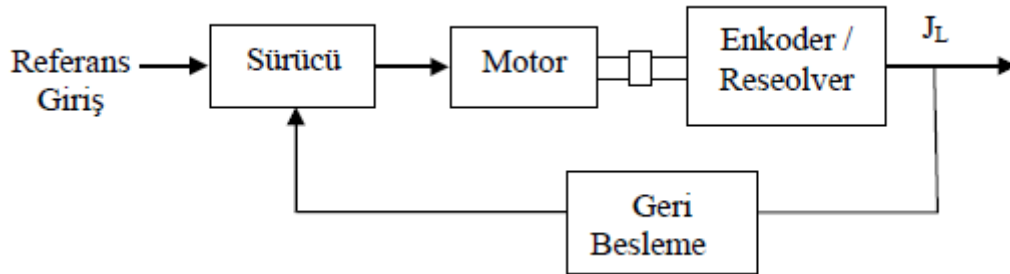


Şekil 2.6. Servo Pozisyon Kontrol Sistemi

Servo pozisyon kontrolörler; boy kesim, taşıma-aktarma hatları, uçan testere vb. uygulamalarda kullanılmaktadır (Hancı 2007).

### 2.2.3. Moment kontrol

Moment kontrol, motor milinden yüke uygulanan momentin istenen bir değerde sabit tutulmasını sağlar. Moment kontrol sistemi Şekil 2.7'de verilmiştir.



Şekil 2.7. Servo Tork Kontrol Sistemi

Motor akımı; motor momentiyle orantılı olduğundan; moment kontrolünü gerçekleştirmek için motor dinamik akımından ( $I_m$ ) bir geri bildirim alınır (Hancı 2007). Sürücü devresi istenen çalışma moment değerini ayarlamak için moment değeriyle oransal bir akım değerini, yapısında bulunan karşılaştırma elemanında, dinamik motor akımı ile kıyaslar. Bu işlemde farkın sıfır olması durumunda motorun moment kontrolü sağlanmış olur. Servo moment kontrol sistemleri özellikle sarıcıboşaltıcı uygulamalarında yaygın olarak kullanılmaktadırlar (Hancı 2007).

#### 2.2.4. Hibrit kontrol

Hibrit kontrol, çeşitli kontrol sistemlerinin bir arada kullanılmasından oluşur. Kullanılacak kontrol modu, parametre ayarları ve ilgili servo girişin anahtarlanması ile seçilir. Hız, pozisyon ve moment kontrol sistemlerinden herhangi ikisi ile bir hibrit kontrol yapı oluşturabilir.

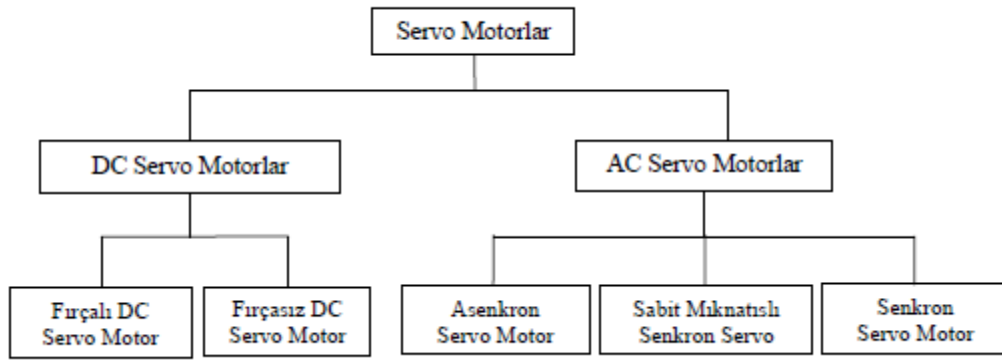
#### 2.3. Servo Motorlar

Aslında servo motor diye bir şey yoktur. Servo, bir sistemdir. Motor, herhangi bir enerjiyi harekete dönüştüren araçtır. Motor kelimesi motorious-move-motion kelimelerinden türemiştir ve hareket eden anlamındadır. Motor bir makinedir. Elektrik enerjisini mekanik enerjiye dönüştüren makinelere elektrik motorları denir. Her elektrik motoru biri sabit (stator) ve diğeri kendi çevresinde dönen (rotor ya da endüvi) iki ana parçadan oluşur. Bu ana parçalar; elektrik akımını ileten parçalar (örneğin; sargılar), manyetik akıyı ileten parçalar ve konstrüksiyon parçaları (örneğin; vidalar, yataklar) olmak üzere tekrar kısımlara ayrılırlar. Mekanik servo sistemlerin doğal olarak bir motoru olmak zorundadır ancak bu motorlar "servo motor" değil günümüzde en yaygın kullanılan şekilleriyle sıradan DC veya AC motorlardır (Dülger ve Kireççi 2001).

1 d/dk'lık hız bölgelerinin altında bile kararlı çalışan, hız-pozisyon-moment kontrolü yapan yardımcı motorlara servo motor denir. Hassas bir şekilde konum kontrolü yapabilen, değişken devirlere hızlı bir şekilde cevap verebilen, otomatik kontrol sistemlerinde çok kullanılan özel motorlardır. Servo motorlar "kontrol motorları" olarak da adlandırılır. Özellikle geri beslemeli kontrol sistemlerinde çıkış hareketini kontrol edici olarak kullanılırlar. Servo motorlar aşağıdaki isteklere cevap verebilirler (Akar 2005).

- Motor milindeki 40 N.m' ye kadar olan büyük dönme momenti,
- Dönme momentinin, iki katına kadar olan kısa aralıkta aşırı yük yüklenebilme,
- Yüksek devir kararlılığı, böylece çeşitli yüklerde hızın sabit kalması,
- Yaklaşık 1-10.000 arasındaki devir sayısının ayarlanabilmesi,
- Çok küçük yol adımları ile hareket edebilme,
- Küçük atalet momenti sayesinde, komutların gecikmeden yerine getirilmesinin sağlanması.

Servo motorlar günümüzde çok farklı tahrik çözümlerinde yaygın şekilde kullanılmaktadır. Sürücü devresiyle birlikte kullanılan motor veya motor grupları; PLC, endüstriyel PC vb. otomasyon elemanlarıyla beraber makinelerin kontrol ve tahrik sistemini oluştururlar (Hancı 2007). Servo motorlar; robotlar, radarlar, CNC tezgahlarda, otomatik kaynak makinelerinde, pres makinelerinde, paketleme makinelerinde, sargı yarı iletken üretim ünitelerinde, yüksek hızlı çip yerleştiricilerinde, tıbbi cihazlarda, anten sürücülerinde vb. yerlerde kullanılır (Megap 2007). Servo motorlar besleme gerilimine ve yapılarına göre Şekil 2.8'de görüldüğü gibi sınıflandırılabilir.



Şekil 2.8. Servo Motorların Sınıflandırılması

### 2.3.3. Servo motor terminolojisi

Kalkış momenti  $M_0$  [Nm];

Durma momenti  $n < 100$  1/min olmak üzere, belirsiz bir hızdaki ve nominal ortam koşullarında ölçülen momenttir.

Anma momenti  $M_n$  [Nm];

Nominal moment motor nominal hızında ve akımında olduğu zaman üretilen momenttir. Sürekli operasyonda motor nominal hızında nominal moment üretilir.

Kalkış akımı  $I_{0rms}$  [A];

Motorun  $n < 100$  devir olduğu hızda durağan momentü oluşturmak için gerekli olan efektif sinüzoidal akımdır.

Tepe akımı (darbe akımı)  $I_{0max}$  [A];

Tepe akımı nominal akımın yaklaşık olarak dört katına eşittir. Anlık değer kullanılan servo sürücüsü ile belirlenir.

Moment sabiti  $K_{Trms}$  [Nm/At];

Moment sabiti motor 1 A akımında iken Nm cinsinden üretilen momentü belirler. Bağıntı şu şekildedir:  $M = I \times K_T$

Gerilim sabiti  $K_{E_{rms}}$  [mV/min-1];

Gerilim sabiti, endüklenen motor EMF'sini, her 1000 devirde ve iki terminal arasında efektif değeri belirler.

Rotor eylemsizlik momentü  $J$  [kgcm<sup>2</sup>];

$J$  sabiti motorun ivmelenme kapasitesine bağlı bir sabittir. Buna göre, durağan halden 3000 devire hızlanan motorun eylemsizlik momentü aşağıdaki gibi hesaplanır.

$$t_b[s] = \frac{3000 \times 2\pi}{M_c \times 60s} \times \frac{m^2}{10^4 \times cm^2} \times J \quad (2.6)$$

Termal zaman sabiti  $t_{th}$  [min];

Termal zaman sabiti, motoru sabit bir yükün altındayken soğuk halden üst sıcaklık olan  $0.63 \times 10^5$  Kelvin'e kadar ısınmasını belirleyen sabittir. Motor nominal akımındaysa çok daha hızlı bir şekilde ısınabilir.

#### 2.3.4. Servo motor seçiminin önemi ve hesaplanması

Özellikle eksenel pozisyonlamanın hassas olduğu endüstriyel uygulamalarda konum, hız ve tork kontrolünü periyodik, hassas ve kararlı bir şekilde gerçekleştirmek servolarla mümkün olmaktadır. Yatay hareket eksenleri için gerekli olan servo hesaplamalarında makinenin konumlama hızı, hareket eden bölümlerin mekanik yükleri ve sürtünmeleri ile ataletleri dikkate alınmalıdır (Akbaş 2010). Aksi halde, servoların kısa süreli gereksiz yere yüklenmeleri hareket yeteneklerini kısıtlamaktadır.

Gerekli tahrik gücünü belirlemede kütlenin ( $m$ ;  $kg$ ) yanı sıra ivme de ( $a$ ;  $m/s^2$ ) önemlidir. Bu tarz makinalar seri üretimde kullanılacağı için ivme değerinin mümkün olduğunca yüksek seçilmesi avantajdır. Tabiki bu noktada servonun bu ivmeye karşı göstereceği tepkide önemlidir.

Lineer makas mekanizması için servo seçecek olursak;

Yatay bir eksende yük taşımak için gerekli tahrik motoru hesabında Newton' un II. Yasası geçerlidir (Akbaş 2010):

$$F = m \cdot a \text{ (Newton)} \quad (2.7)$$

Burada sistem için ivme,  $a = 2.50 \text{ m/s}^2$  alındığında ve lineer makas mekanizmasının ağırlığı yaklaşık 1000kg olduğu varsayılırsa;

$$F = 1000 \text{ kg} \cdot 2.50 \text{ m/s}^2 = 2500 \text{ N} \quad (2.8)$$

değerinde ataleti karşılayabilecek itme gücüne ihtiyaç vardır. Lineer makas mekanizmasının X ekseni hareket alanı 4.60 m' dir. Yatay hareket hız değeri  $V_{max} = 1.25 \text{ m/s}$  seçildiğinde; makinenin hızlanma süresi :

$$t_{acc} = V_{max} / a = 0,5 \text{ s} \quad (2.9)$$

'dir. Bu esnada kat edeceği yol miktarı ise;

$$x_{acc} = 0,5 \cdot a \cdot t^2 = 0.313 \text{ m} \quad (2.10)$$

'dir. Hızlanma ve yavaşlama değerleri eşit olduğundan, motorun tüm ekseni kat edebileceği süre;

$$t_{sum} = t_{acc} + t_{max\_V} + t_{dec} \quad (2.11)$$

formülünden 4.18 s gibi uygun bir değere tekabül etmektedir.

Motor gücü hesabında, motorun etiket devri 1500 d/dk alındığında;

$$M_0 \cdot 1.25 \text{ m/s} \cdot 2\pi \cdot 1500 / 60 = 2500 \text{ N} \cdot 1\text{m} \quad (2.12)$$

eşitliğinden motor momenti  $M_0=12.13\text{N.m}$  olarak hesaplanır. Momentten motor gücüne geçildiğinde :

$$P_{kW} = M_0 \cdot ( \text{Nominal Motor Devri}(d/dk) / 9550 ) \quad (2.13)$$

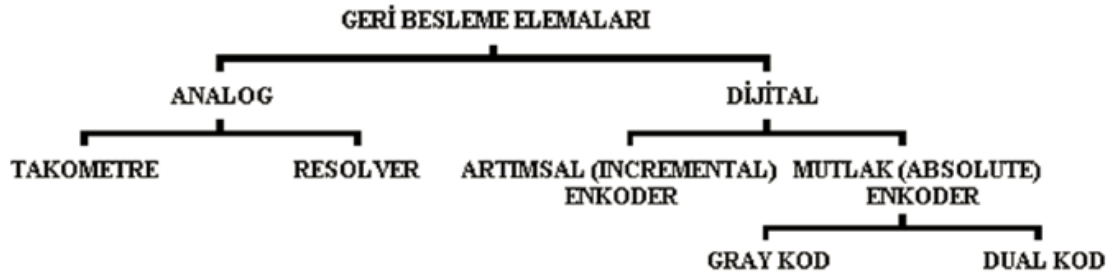
güç 1.97 kW bulunur. Bu değer 1.25 gibi bir emniyet kay sayısı ile çarpıldığında;

$$P_{kW} = 1.97 \cdot 1.25 = 2.46\text{kW} \quad (2.14)$$

değeri elde edilir.

#### 2.4. Geri Besleme Elemanları

Geri besleme elemanları bir servo sistemin, hızını, motor milinin bulunduğu konumu ve yükün bulunduğu konumu ölçmek ve belirlemek için kullanılır. Uygulamalarda kullanılan geri besleme elemanı türleri Şekil 2.9'daki gibi sınıflandırılabilir.



Şekil 2.9. Geri Besleme Elemanları

Geri besleme elemanları motora bağlı bir şekilde kullanılırlar ve genellikle motor ile aynı devirde çalışırlar. Bu nedenle yüksek hızlarda ısınır ve ürettikleri işaretler bulundukları ortamdaki manyetik alandan etkilenir. Geri besleme elemanlarının, sistemi olumsuz yönde etkileyecek bu etkilerden arındırılması gerekir. Pratik uygulamalarda yüksek hız nedeniyle ısınmanın önlenmesi için uygun yataklama sistemleri kullanılabilir. Elemanların elektromanyetik alandan korunması için de ekranlama yapılabilir.

### 2.4.1. Takometre

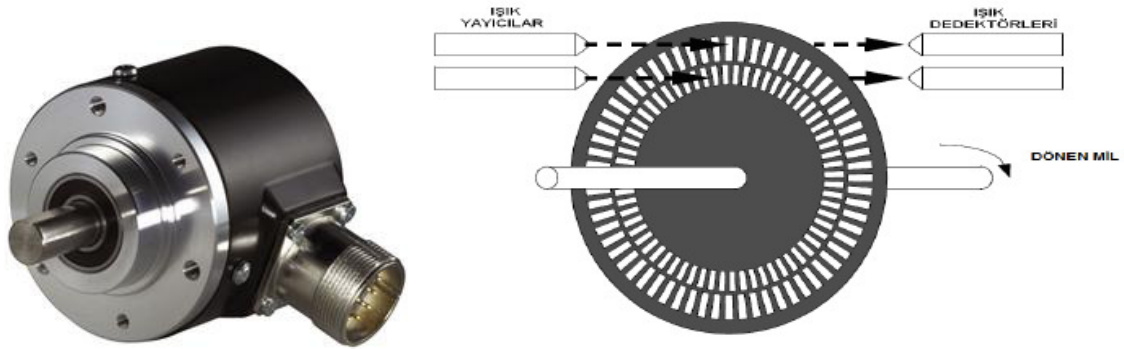
Eski sistemlerde genellikle doğru akım motorlarında en çok kullanılan hız geri besleme elemanıdır. Doğru akım stator sargıları rotor devrine göre belli bir seviyede ( $\pm 10V$  gibi) hızla orantılı bir doğru gerilim üretir. En önemli özelliği (Volt / Devir Sayısı) oranının sabit olmasıdır.

### 2.4.2. Resolver

Bu algılayıcılar sargılı bir rotor, birbirlerine göre  $90^\circ$  faz farkıyla yerleştirilmiş iki stator sargısı yanında, bir sargısı stator ve bir sargısı rotorda bulunan sabit hava aralıklı döner transformatörden oluşur. Statorda bulunan döner transformatörün uyarma sargısına yüksek frekanslı (2-10kHz) besleme gerilimi verildiğinde, stator sargılarında  $90^\circ$  faz farklı gerilimler indüklenir. Bu gerilimlerin genlikleri konuma bağlı olarak değişir. Statorda indüklenen gerilimlerin birbirine göre oranı alınarak rotor konumu yüksek kesinlik derecesinde elde edilir. Resolverin rotor ve statorunda mekanik bir bağlantı yoktur. Bu nedenle çok yüksek hızlarda sorunsuz olarak kullanılabilirler. Bu elemanlar mekanik olarak da oldukça dayanıklıdır.

### 2.4.3. Mutlak tip (absolute) enkoder

Bu Enkoderler, her pozisyonlarında farklı sayıdaki bitlerden oluşan dijital bit dizileri şeklinde birbirine benzemeyen çıkışlar üreterek, gerçek pozisyonlarını tam olarak gösterirler. Enerjisi kesilse bile mevcut durumunu korur. Şekil 2.10'da standart encoder ve temel iç yapısı görülmektedir.



Şekil 2.10. Standart Enkoder ve Temel İç Yapısı

### 2.4.4. Artımsal tip (incremental) enkoder

Bu enkoderler, her pozisyonunda benzer çıkış sinyalleri (Kare Dalga) üretirler. Bu sinyaller hız ölçümü (bir Takometre ile birlikte) yada sayma işlemi için (bir Sayıcı ile birlikte) kullanılabilirler. Bu tip enkoderler absolute enkoderlerden farklı olarak enerjilendikleri anda bulundukları yeri 0 kabul eder ve bunun üstüne işlem yapar. Absolute enkoderler, incremental enkoderlere göre daha pahalıdır.

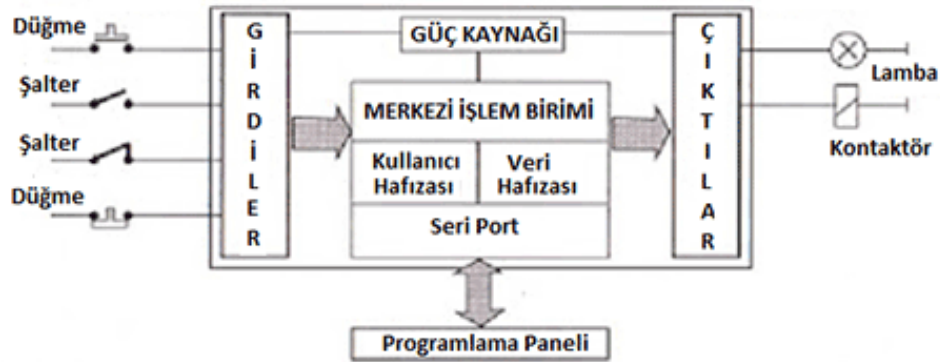
## 2.5. PLC (Programlanabilir Lojik Kontrolör)

Fabrikalardaki imalat hatları veya makinelerin kontrolü gibi işlemlerin denetiminde kullanılan, içinde mikroişlemcisi olan özel bilgisayarlardır. Genel kullanımlı bilgisayarların aksine PLC birçok girişi ve çıkışı olacak şekilde düzenlenir ve elektriksel gürültülere, sıcaklık farklarına, mekanik darbe ve titreşimlere karşı daha dayanıklı tasarlanırlar. PLC'lere denetleyeceği sistemin işleyişine uygun programlar yazılıp, yüklenir. PLC programları, giriş bilgilerini milisaniyeler mertebesinde hızla tarayarak buna uygun çıkış bilgilerini gerçek zamanlıya yakın, cevap verecek şekilde çalışırlar. PLC bir bakıma monitörü ve klavyesi bulunmayan bir bilgisayar gibidir. Bir diğer fark ise işletilecek verilerin gerçek ortamdan gelmesi ve sonuçların yine gerçek ortama analog veya sayısal (dijital) olarak gönderilmesidir.

Mikroişlemcilerin maliyeti daha düşük olmasına rağmen, PLC'lerin tercih edilmelerinin sebebi; elektronik tasarım için harcanacak zamanı en aza indirmesidir. Aynı zamanda endüstriyel ortamların sahip olduğu zor koşullardan (manyetik alan, büyük sıcaklık farkları, toz vb.) etkilenmeden çalışabilen hazır çözümler olmalarıdır. Bir fabrikanın tüm otomasyon işlerini yüklenilecek kadar Giriş/Çıkış sayısına sahip PLC'ler bulunmaktadır. Günümüzde geliştirilen modüler yapıdaki PLC'lere gerektiğinde ek giriş-çıkış modülleri, RS232, RS485, modem, Ethernet, CanOpen gibi haberleşme modülleri eklenebilmektedir. Bu gibi özelliklerle mevcut yapı geliştirilebilmektedir.

PLC'ler Şekil 2.11'de görüldüğü gibi 4 ana bölümden oluşmaktadır :

- \* Merkezi İşlem Birimi ( CPU )
- \* Bellek Birimi ( RAM,ROM, EEPROM vb.. )
- \* Giriş Birimi ( IN )
- \* Çıkış Birimi ( OUT )



Şekil 2.11. PLC İç Organizasyon Şeması

### 2.5.1. PLC'nin kullanım amacı ve alanları

PLC'ler endüstri alanında kullanılmak üzere tasarlanmış, sayısal prensiplere göre yazılan fonksiyonu gerçekleyen, bir sistemi ya da sistem gruplarını giriş/çıkış

birimleri ile denetleyen, içinde barındırdığı zamanlama, sayma, saklama ve aritmetik işlem fonksiyonları ile genel kontrol sağlayan elektronik bir cihazdır.

PLC sistemi sahada meydana gelen fiziksel olayları, değişimleri ve hareketleri çeşitli ölçüm cihazları (algılayıcılar) ile belirleyerek, gelen bilgileri, yazılan programa göre bir değerlendirmeye tabi tutar. Mantıksal işlemler sonucu ortaya çıkan sonuçları da kumanda ettiği elemanlar aracılığıyla sahaya yansıtır.

Sahadan gelen bilgiler ortamda meydana gelen aksiyonların elektriksel sinyallere dönüşmüş halidir. Bu bilgiler analog ya da sayısal olabilir. Bu sinyaller bir algılayıcıdan veya bir kontaktörün yardımcı kontağından gelebilir. Gelen bilgi analog ise gelen değerin belli bir aralığı için, dijital ise sinyalin lojik '0' veya lojik '1' olmasına göre sorgulama yapılabilir. Bu algılama olayları giriş birimleri ile müdahale olayları ise çıkış birimleri ile yapılır. PLC ile kontrolü yapılacak sistem büyüklük açısından farklılıklar gösterebilir.

PLC ile sadece bir makine kontrolü yapılabileceği gibi bir fabrikanın komple kumandası da gerçekleştirilebilir. Aradaki fark sadece, kullanılan kontrolörün (PLC'nin) kapasitesi ile ilgilidir. PLC'ler her türlü otomasyon işlerinde kullanılmaktadır.

Kimya sektöründen gıda sektörüne, üretim hatlarından depolama sistemlerine, marketlerden rafinerilere kadar çok geniş bir yelpazede PLC'ler kullanılmaktadır. Elektronik sektöründeki hızlı gelişmelere paralel olarak gelişen PLC teknolojisi, gün geçtikçe ilerlemekte otomasyon alanında mühendislere yeni ufuklar açmaktadır.

İmalat sanayi, tarım, enerji üretimi, kimya sanayi vb. endüstrinin tüm alanlarında kullanılan PLC'lerin genel uygulama alanları aşağıda sıralanmıştır;

#### i) Sıralı Kontrol

PLC'lerde en çok kullanılan kontrol yöntemidir. Bu kontrol yöntemi "sıralı çalışma" özelliği ile kumanda sistemlerine en benzer olan endüstriyel uygulamadır. Uygulama açısından, bağımsız makinelerde ya da makine hatlarında, konveyör ve paketleme makinelerinde ve hatta modern asansör denetim sistemlerinde kullanılmaktadır.

#### ii) Hareket Kontrolü

Doğrusal veya döner hareket denetim sistemlerinin PLC'ler ile ortak kullanılmasıdır. Örnek olarak servo kontrol veya hidrolik sürücülerde kullanılabilen tek ya da çok eksenli bir sistem denetimi verilebilir.

PLC hareket kontrolü uygulamaları, sonsuz bir makine çeşitliliği ve çoklu hareket eksenlerini kontrol edebilirler. Bunlara örnek olarak; kartezyen robotlar, film, kauçuk ve dokunmamış kumaş tekstil sistemleri gibi ilgili örnekler verilebilir.

#### iii) Süreç Denetimi

Bu uygulama PLC'nin birkaç fiziksel parametreyi (sıcaklık, basınç, debi, hız, ağırlık vb gibi) denetleme yeteneğiyle ilgilidir. Bu da bir kapalı çevrim denetim sistemi

oluşturmak için, analog giriş/çıkış gerektirir. PID fonksiyonunun kullanımıyla PLC, tek başına kapalı çevrim denetleme görevini yerine getirebilir. Buna tipik örnek olarak plastik enjeksiyon makineleri ve ısıtma fırınları verilebilir.

#### iv) Veri Yönetimi

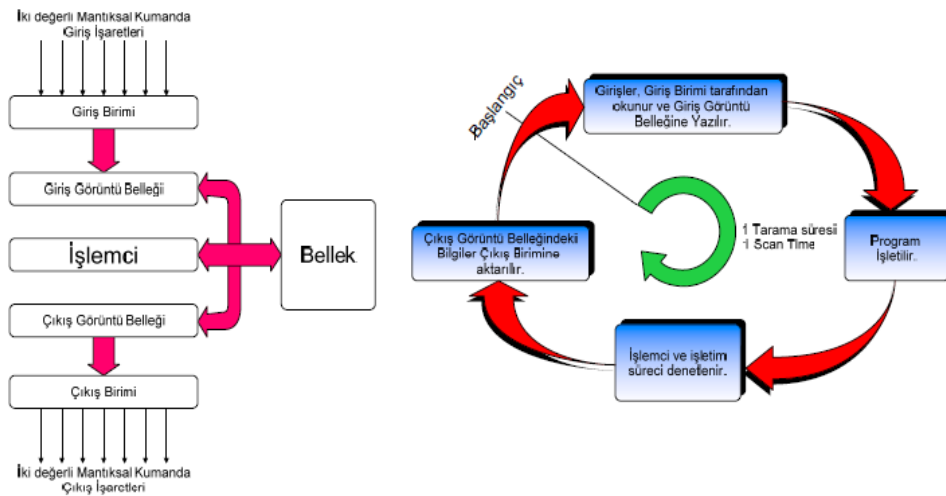
PLC ile verilerin toplanması, incelenmesi ve işlenmesi kolaylıkla yapılabilir. PLC'ler denetlediği proses hakkında veri toplayıcı olarak kullanılabilir. Bu veriler, denetleyicinin belleğindeki referans veri ile karşılaştırılır ve rapor alımı için başka cihazlara aktarılabilir. Veri yönetimi, endüstride, malzeme işleme tesislerinde, kağıt, metal ve yiyecek işleme gibi birçok proseste kullanılır.

#### Kullanım Alanlarına Örnekler:

- Havalandırma ve soğutma tesislerinde,
- Paketleme ve ambalajlama tesislerinde,
- Taşıma tesislerinde,
- Otomobil endüstrisi,
- Petrol dolum ve yıkama tesislerinde,
- Çimento sanayinde,
- Klima ve asansör tesislerinde,
- Aydınlatma ve vinç tesislerinde,
- İmalat, tarım, tekstil ve her türlü makinelerde,
- Elektro pnomatik–hidrolik sistemlerde,
- Robot tekniğinde kullanılmaktadır.

### 2.5.2. PLC'nin genel çalışma mantığı

PLC'de programın yürütülmesi, Şekil 2.12'de görüldüğü gibi belirli işlemlerin sürekli periyodik bir çevrim halinde yapılması ile gerçekleşir.



Şekil 2.12. PLC Çalışma Şeması

PLC'nin çalışmasını kısaca anlatacak olursak;

- 1) PLC'nin Giriş Birimine bağlı olan giriş elemanlarından gelen giriş bilgileri okunur ve PLC'nin Giriş Görüntü Belleğine yazılır. Bu bilgiler bir sonraki tarama çevrimine kadar değişmez.
- 2) Program belleğinde yer alan komutlar sırayla işlenir ve elde edilen çıkış değerleri Çıkış Görüntü Belleğine yazılır.
- 3) İşlemci ve işletim süreci denetlenir herhangi bir sorun yoksa Çıkış Görüntü Belleğindeki bilgiler Çıkış Birimine aktarılır. Çıkış Birimine aktarılan dijital bilgiler bir sonraki tarama çevrimine kadar değişmez. Böylece 1 tarama çevrimi tamamlanır ve bir sonraki çevrime geçilir. Bu çevrim PLC, STOP durumuna alınıncaya kadar devam eder.

PLC'lerde WATCHDOG Timer (Bekçi Köpeği) olarak isimlendirilen bir zamanlayıcı bulunmaktadır. Eğer PLC WATCHDOG Timer' da ayarlanan süre içerisinde tarama çevrimini tamamlayamaz ise PLC' de Hata (Fault) olduğu varsayılarak PLC durdurulur ve tüm çıkışlar sıfırlanır. Watchdog Timer' ın zaman süresini programcı PLC kataloğunda yer alan süreler göre ayarlar.

Normal çalışma durumunda PLC'nin girişleri tarama çevriminin başında okunur, PLC'nin çıkışları da tarama çevriminin sonunda güncellenir. Özel durumlarda, bazı girişlerin komutların işlendiği anda okunması, bazı çıkışların da komutların işlendiği anda yazılması (çıkışa yansıtılması) istenebilir. Başka bir deyişle girişlerin, giriş görüntü belleğinden değil de doğrudan giriş biriminden okunması ve/veya çıkışların çıkış görüntü belleğinden değil de doğrudan çıkış birimine yazılması istenir. Bu durumda İvedi Giriş/Çıkış komutları (Immediate I/O Instructions) kullanılarak PLC ile anında okuma yazma yapılabilir. Ancak bu komutlar ile sadece komutların işlendiği anda girişlerden okuma, çıkışlardan yazma yapılabilir.

Gerçek anlamda girişlerden gelen bilgileri kaçırmamak için (Hızlı değişen bilgileri algılayabilmek için) Kesme (Interrupt) kullanılır. Kesme kullanıldığı durumda programın bulunduğu yere bakılmaksızın Kesme Girişi geldiği anda PLC'nin o an yürüttüğü program durdurulur, Kesme alt programına geçilir ve kesme programı yürütülür. Kesme programı bitince PLC Kesmeye dallanmadan önce yürüttüğü programa kaldığı yerden devam eder. Üç farklı kesme durumu vardır. Bunlar; Zaman Kesmeli Çalışma, Olay Kesmeli Çalışma (Girişlerden gelir) ve Hızlı Sayıcı Kesmeli Çalışmadır.

Olay Kesmeli Çalışma; kesme kaynağı olay olan kesmeli çalışmadır. Olaylar PLC' nin belirli girişlerinden gelen yükselen veya düşen kenar sinyalleri ile algılanır. Zaman Kesmeli Çalışma; tarama süresinden bağımsız olarak tanımlanmış belirli sürelerde gerçekleşen kesmeli çalışmadır. Kesme süresi programcı tarafından belirlenmektedir. Hızlı sayıcılar, PLC' nin tarama süresine (scan time) göre çok kısa sürelerde gelen hızlı sinyalleri algılayıp, değerlendirip duruma göre kumanda sinyalleri üretmek için kullanılır.

## 2.6. CoDeSys (Controller Development System)

Hızla gelişen ve değişen endüstriyel uygulamaların taleplerine cevap verebilmek üzere, farklı PLC yapıları ile çalışmak ve bunları bütünüyle yönetebilmek gerekmektedir. Ancak günümüzde karşılaşılan en sık problemlerin başında farklı yapılardaki yazılım programlarının kullanıcıları zor durumda bırakmasıdır. CoDeSys, bu durumu ortadan kaldırarak donanımdan bağımsız yazılım yapısı ile endüstriyel PC'lerden PLC'lere kadar bu standardı kullanan çok sayıda kontrol cihazı için geliştirilmiş bir program yazma aracı sunuyor.

CoDeSys ile Fabrika Otomasyonu, Mobil Otomasyon, Enerji Otomasyonu, Gömülü Sistem Otomasyonu, Proses Otomasyonu, Bina Otomasyonu gibi endüstri alanlarında kullanılan kontrol sistemleri için yazılımlar yapılabilmektedir.

CoDeSys, açık kaynak kod çözümü olup yazılımı ücretsizdir. Yazılımda ayrıca ücretsiz fonksiyon blokları geliştirilebilmektedir. Dış kütüphaneler, library dosyaları hazırlanabilmektedir. Bu sayede ister üretici firmalar olsun, ister yazılımcılar olsun yeni kütüphane dosyaları hazırlayabilmektedir. Bu dosyalar kolaylıkla CoDeSys programına adapte edilebilmektedir. Yeni sistemlere adaptasyon için ek kütüphane yazılımları ücretsiz olarak temin edilebilmektedir. Kullanıcıya esneklik ve karşılıklı işletim imkanları sunar.

PLC programlamada çok seçenekli dil sunması ile rahat ve kolay yazılım yapılabilmektedir. Entegre derleyiciler, nesne yönelimli programlama, 3D CNC interpolasyon, alarm yönetimi, entegrasyon gibi bilişim dünyasından eklentiler CoDeSys'i güçlü, çığır açan bir yazılım yapmıştır.

CoDeSys ile yazılacak program kendi içinde mantıksal olarak farklı dillerde yazılabilir, örnek olarak programın ana yapısı SFC ile kurulup, her bir adımın içeriği FBD ile programlanabilir.

Eksen kontrolü için PLCopen standartlarında verilmiş tüm fonksiyon bloklarının kullanımı (CNC, cam, homing, vb.) mümkün olup, gerektiğinde tekrar çağırılabilir kullanıcıya özel yazılım kütüphaneleri oluşturulabilir.

Programda farklı görevler (task) tanımlanması ve belirlenen öncelik sırasına göre paralel programlama imkanı sağlar. Online simülasyon özelliği ile anlık olarak istenen parametrelerin çizdirilmesi ve izlenmesi sağlanabilir. Grafik özelliği ile her fonksiyon bloğu için önceden hazırlanmış görseller ve bu görsellere gerekli parametreler kolayca atanabilir. PLC döngü sürelerinin ve program içinde kullanılan değişkenlerin gerçek zamanlı olarak örneklenmesi ve izlenebilmesi imkanını sunar.

Sonuç olarak otomasyon mühendislerinin uygulama ihtiyaçlarına uyan, üreticiden bağımsız, üstün performans sunar. CoDeSys ile bir marka PLC için yazılan programı çok küçük donanım değişiklikleri ile diğer marka PLC'ye de uyarlamak mümkündür. Bugün onlarca güçlü firma otomasyon pazarına CoDeSys programlama arayüzü sunmaktadır.

CoDeSys'in ilk versiyonu olan 1.0, 1994 yılında 3s (Smart Software Solutions) tarafından piyasaya sürülmüştür. Güncel sürümü 3.5 olan otomasyon programı, IEC 61131-3 standartlarına uygun olarak 6 ayrıntılı program dilinde hizmete sunulmuştur. Bunlar; Instruction List(IL), Ladder Diagram(LD), Function Block Diagram(FBD), Structured Text(ST), Sequential Function Chart(SFC), Continuous Function Chart(CFC) olarak isimlendirilir(URL-2).

### 2.6.1. IEC-61131-3 standardı

IEC (International Electrotechnical Commission), 1906 yılında elektrik, elektronik ve ilgili teknolojiler konusunda uluslararası standartları hazırlama çalışmalarına başladı. Şu anda 51 üyesi bulunan IEC'ye Türkiye de 1956 yılında üye olmuştur. IEC'nin hedefleri global pazar gerekliliklerini karşılamak, ürünlerin ve hizmetlerin kalitesini arttırmak, insan sağlığı ve güvenliğine katkıda bulunmak, çevrenin korunmasına katkı sağlamaktır. IEC 61131-3 standartlarına uyan PLC'ler yazılım olarak 6 ana dili ile programlanmayı desteklemelidir ve ayrıca bu standarda göre tanımlanmış veri tiplerine bağlı kalmak zorundadır.

### 2.6.2. CoDeSys platformundaki veri tipleri

Program içinde kullanılabilecek ve tanımlanabilecek değişken tipleri Çizelge 2.1 ile verilmiştir. Bu çizelgeden data tiplerinin tanımlı olduğu aralıklarda görülebilmektedir. Program yazılırken bu tanımlı aralıklar dikkate alınmaz ise değişken bir sonraki hafıza adresine taşma yapar ve bu program akışı için sorun doğurabilir.

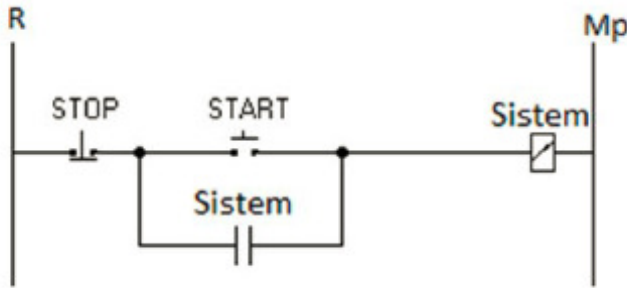
Çizelge 2.1. Veri Tipleri Çizelgesi

| DATA TİPİ | DATA UZUNLUĞU (BİT) | TANIMLI OLDUĞU ARALIK                                 |
|-----------|---------------------|-------------------------------------------------------|
| BOOL      | 1                   | [0,1]                                                 |
| BYTE      | 8                   | [0,...,+255]                                          |
| WORD      | 16                  | [0,...,+65.535]                                       |
| DWORD     | 32                  | [0,...,+4.294.967.295]                                |
| LWORD     | 64                  | [0,...,+2 <sup>64</sup> -1]                           |
| SINT      | 8                   | [-128,...,+127]                                       |
| INT       | 16                  | [-32768,...,+32.767]                                  |
| DINT      | 32                  | [-2.147.483.648,...,+2.147.483.647]                   |
| LINT      | 64                  | [-2 <sup>63</sup> ,...,+2 <sup>63</sup> -1]           |
| USINT     | 8                   | [0,...,+255]                                          |
| UINT      | 16                  | [0,...,+65.535]                                       |
| UDINT     | 32                  | [0,...,+4.294.967.295]                                |
| ULINT     | 64                  | [0,...,+2 <sup>64</sup> -1]                           |
| REAL      | 32                  | [1.175494351e-38,...,3.402823466e+38]                 |
| LREAL     | 64                  | [2.2250738585072014e-308,...,1.7976931348623158e+308] |

### 2.6.3. CoDeSys platformundaki programlama dilleri

Bir sistemin istenen biçimde çalışması için kullanılan sözel yada matematiksel kuralların PLC program belleğine aktarılacak üzere uygun bir dil kullanılarak düzenlenmesine program yazım dili denir. Genel olarak PLC'lerde yaygın olarak kullanılan 6 türlü program yazım dili tanımlanmıştır.

Program yazım dillerinin anlaşılması açısından Şekil 2.13'de verilen gibi basit bir kumanda devresini 6 farklı program yazım dili ile yazmak gerekirse;

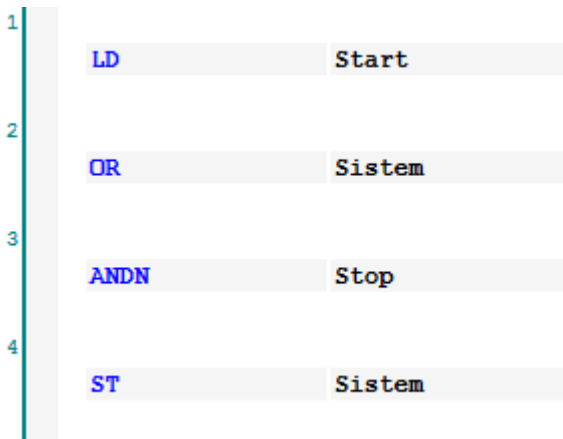


Şekil 2.13. Örnek Kumanda Devresi

#### 2.6.3.1. Komut listesi (IL)

Belirli bir komut kümesi kullanılarak program yazılır. En düşük seviyeli program yazım dilidir. Şekil 2.14'te Şekil 2.13'deki devrenin IL dilindeki programı verilmiştir.

**Komut Örnekleri:** LD, LDN, ANDN, OR, ORN, ST v.b.



Şekil 2.14. Örnek IL Programı

#### 2.6.3.2. Yapısal metin (ST)

Yüksek seviyeli dillere (Pascal, C, Basic v.b.) benzer program yazım dilidir.

**Komut Örnekleri:** IF, ELSE, END\_IF, VAR, AND, FOR v.b.

Şekil 2.15’de Şekil 2.13’deki devrenin ST dilindeki programı verilmiştir.

```

1 IF (Start=TRUE OR Sistem=TRUE) AND (Stop=FALSE) THEN
2   Sistem:=TRUE;
3 ELSE
4   Sistem:=FALSE;
5 END_IF;

```

Şekil 2.15. Örnek ST Programı

IF deyimi bir koşul deyimidir. Koşullar sağlandığı zaman IF deyimi içerisindeki komutlar işletilir.

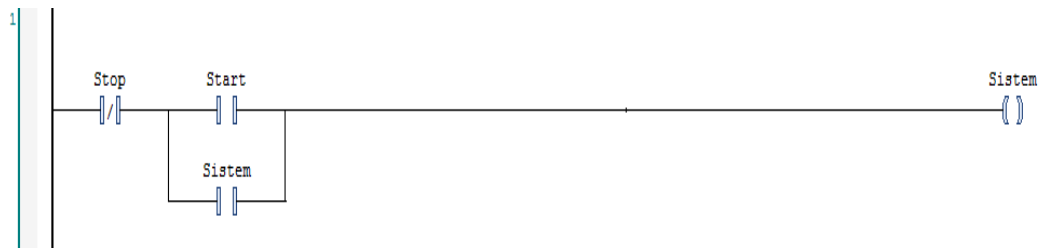
Çok sayıda iç içe IF blokları kullanıldığında ve şartların hep aynı kritere bağlı olduğu durumlarda IF deyimi yerine CASE deyimi kullanılabilir.

FOR döngüsü sayaç döngüsüdür. Verilen alt ve üst limitler arasında istenilen artım değeri kadar döngü içerisinde bulunan komutları tekrarlar. Diğer programlama dillerindeki gibi FOR döngüsü bittiği anda program döngüden çıkar. Ancak PLC her tarama süresinin (ScanTime) sonunda sürekli komutları yenilediğinden dolayı FOR döngüsünün başında herhangi bir koşul yok ise döngüyü tekrar çalıştıracaktır. Bu sebepten dolayı FOR döngüsü kullanılırken bu duruma dikkat etmek gerekir.

WHILE deyimi bir koşullu döngüdür. Koşullar sağlandığı sürece döngü içerisindeki komutlar işletilir.

### 2.6.3.3. Merdiven dili (LD)

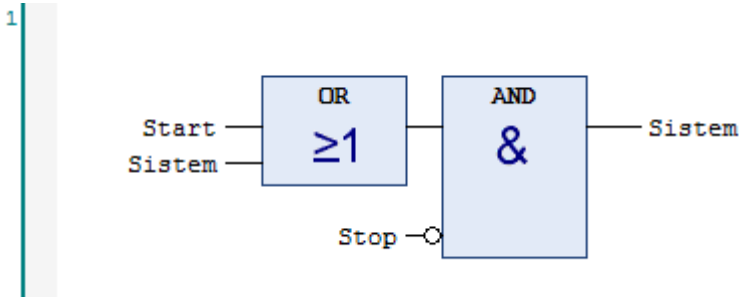
Açık kontak, kapalı kontak simgeleri ile lojik ilişkilerin gösterildiği program yazım dilidir. Şekil 2.16’da Şekil 2.13’deki devrenin LD dilindeki programı verilmiştir.



Şekil 2.16. Örnek LD Programı

### 2.6.3.4. Fonksiyon blok diyagram (FBD)

Lojik kapılar ve blok diyagram mantığı ile yazılan program yazım dilidir. Şekil 2.17’de Şekil 2.13’deki devrenin FBD dilindeki programı verilmiştir.



Şekil 2.17. Örnek FBD Programı

### 2.6.3.5. Grafik editör (CFC)

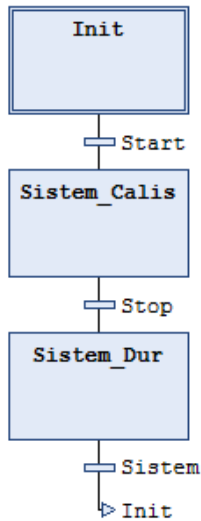
Grafiksel ara yüz kullanılarak yapılan program yazım dilidir. Şekil 2.18’de Şekil 2.13’deki devrenin CFC dilindeki programı verilmiştir.



Şekil 2.18. Örnek CFC Programı

### 2.6.3.6. Ardışık fonksiyon gösterimi (SFC)

Ardışık sıralı işlemlerin yapıldığı yazılımlar için uygun program yazım dilidir. SFC programlama, gerçekleştirilecek işlemlerin ardışık, karmaşık ve çok dallı olması durumunda tercih edilen bir programlama dilidir. Bu programlama dilinde işlemler basamaklar halinde sırayla algoritma olarak hazırlanır. Her basamağın başlama ve bitişini kontrol eden şartlar vardır. Bu şartların oluşmasıyla program bir sonraki basamağa geçer. Şekil 2.19’da Şekil 2.13’deki devrenin SFC dilindeki programı verilmiştir.



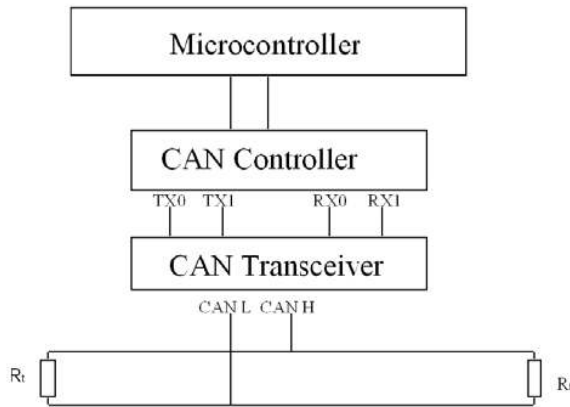
Şekil 2.19. Örnek SFC Programı

## 2.7. CanBus Haberleşme Protokolü

Controller Area Network (CAN) protokolü 1983 yılında otomotiv sektöründe kullanılmak üzere Robert Bosch tarafından geliştirilmeye başlanmıştır ve resmi olarak 1986 yılında kullanılmaya başlanmıştır. Araçlarda bulunan kablo ağının azaltılması amaçlanmıştır. Bu protokol sayesinde motor, frenler, klima, çeşitli sensörler gibi birimler arasında onlarca kablo yerine sadece 2 kablo kullanılmaktadır. CAN protokolü başta otomotiv alanında kullanılsa da veri iletim hızı, hata oranının düşüklüğü ve uygulama kolaylığı gibi sebeplerden dolayı mikroişlemcili sistemlerde de çoğu kişi tarafından bireysel uygulamalarda kullanılmaktadır. İletişim hızı 40m de 1Mbit/sn iken 1km uzaklıklarda 40Kbit/sn'ye düşmektedir. Yapılan olasılık hesaplarına göre sağlıklı çalışan bir CAN sisteminin tespit edilemeyen hata yapma olasılığı 1 asırda 1'dir (URL-3).

CAN diğer protokollerden farklı olarak adres temelli değil mesaj temelli çalışmaktadır. Her mesaja özgü bir ID numarası vardır. Mesajlar çerçeveler ile iletilirler ve veri ve istek mesajları olarak ikiye ayrılırlar. İstek mesajlarında veri bulunmaz. Veri mesajlarında ise en fazla 8 byte'lık veri aktarılır. CAN2.0A ve CAN2.0B olmak üzere 2 tane standardı vardır. Bunlar arasındaki fark mesaj uzunluklarının farklı olmasıdır (URL-3).

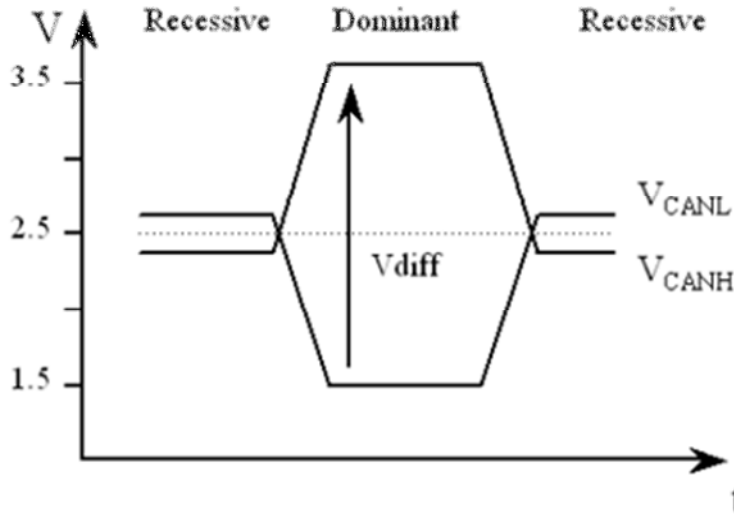
Genel olarak bir CAN bağlantı noktası Şekil 2.20'de gösterilmiştir. Bağlantı noktaları düğüm olarak isimlendirilirler ve mikrodnetleyici, CAN kontrolcüsünden oluşmaktadırlar. Bizim işlemcilerimizde bu CAN kontrolcüsü çipe dahil edilmiştir fakat harici entegre olarak da kullanılabilir. CAN kontrolcüsü CAN veri yoluna direk bağlanır. Bu veri yolu iki telden oluşmuş, iki tarafı 120 ohm dirençlerle sonlandırılmış bir veri yoludur.



Şekil 2.20. CanBus Şeması

CAN yapısında alıcı ve verici birbirinden fiziksel olarak bağımsızdır. Fakat düğümlerin yapısı gereği gönderilen mesaj alıcıdan dinlenebilmektedir. Bu sayede veriyi gönderen işlemci gönderdiği veri ile okuduğu veriyi karşılaştırarak hata ve mesajların öncelik seviyelerine göre iletilmelerine olanak sağlanır. Multimaster yapıda çalışabilen bu sistem için bu özellik önemlidir (URL-3).

Şekil 2.21’de görüldüğü gibi hattın lojik seviyesi 2 farklı değer alabilmektedir ve bu değerlerin seviyesi gözükmemektedir.



Şekil 2.21. CanBus Lojik Seviyesi

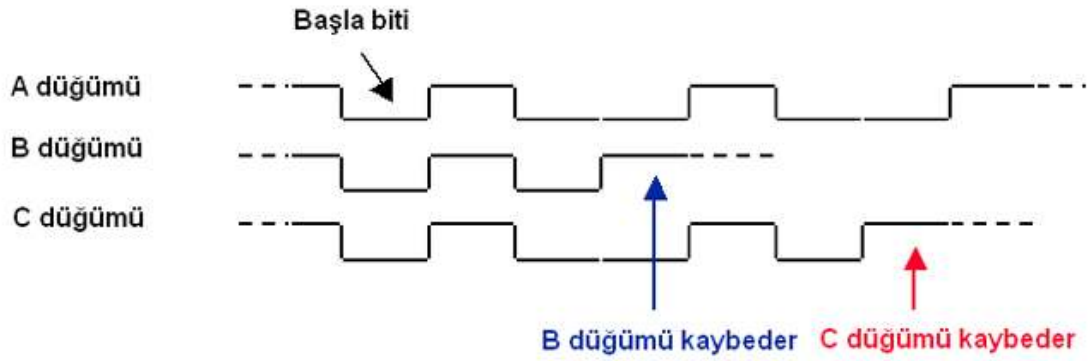
Lojik 1 recessive (çekingen), lojik 0 dominant (baskın) olarak adlandırılmaktadır. Bunun sebebi hatta farklı düğümlerden aynı anda 0 ve 1 yazılması durumunda 0'ın 1'e karşı baskın gelmesidir. Bu durumun doğruluk tablosu Şekil 2.22’de verilmiştir.

|           | dominant | recessive |
|-----------|----------|-----------|
| dominant  | dominant | dominant  |
| recessive | dominant | recessive |

|   | 0 | 1 |
|---|---|---|
| 0 | 0 | 0 |
| 1 | 0 | 1 |

Şekil 2.22. İkili Düğüm Veri Gönderirken Yol Durumu ve Lojik VE

Lojik 0'ın lojik 1'e baskın gelmesi sonucu küçük mesaj ID sine sahip mesajlar öncelik kazanırlar. Bir düğüm tarafından mesaj gönderilmesi kararlaştırıldığında mesaj yol boşalana kadar bekletilir. Her düğüm yolu devamlı izlemektedir. Yol boşaldıktan sonra düğüm yola başla işaretini vererek mesajı yollamaya başlar. Mesaj her düğüme ulaşmaktadır ve ilişkisi olan düğümler mesajı okuyup işlemektedirler. Eğer yol boşaldığında birden fazla düğüm yola mesaj yazmaya başlarsa düşük ID'li mesajı yazan düğüm yolu ele geçirir ve diğer düğümler aradan çekilerek tekrar göndermek üzere yolun boşalmasını beklerler. Bu mekanizma şu şekilde çalışır. Yazılan her bitin aynı anda okunur. Bir düğüm veri yoluna mesaj yazarken 1 yazdığında 0 okuyorsa eğer, başka bir düğümünde yola mesaj yazdığını anlar ve onun önceliği yüksek olduğundan veri yolunu ona bırakır. Yol boşaldığında tekrar göndermeye çalışır. Örneğin yola aynı anda veri yazmaya çalışan A, B ve C adında 3 düğümümüz olsun. A düğümü yola 36 (100100), B düğümü 47(101111) ve C düğümü 37(100101) yazsın. Şekil 2.23’de bu durum gösterilmiştir (URL-5).



Şekil 2.23. CanBus Tahkimi

A düğümü mesajını gönderdikten sonra daha önemli bir mesaj yoksa C düğümü sonrada B düğümü mesajını gönderir.

Veri yolunda mesajlar çerçevelere bölünerek iletilmektedir. İki farklı mesaj türü vardır. Bunlar veri çerçevesi (Message Frame) ve istek çerçeveleridir (Remote Transmit Request Frame). Farkları ise veri çerçevelerinin en fazla 8 byte uzunluğa kadar veri taşıyabilmesi, istek çerçevesinde ise belli bir mesaja ait verinin istenmesidir. 2 adet CAN2.0A ve CAN2.0B olmak üzere iki çeşit CAN standardı vardır (URL-5).

## 2.8. CANopen Katman Protokolü

CANopen, otomasyonda kullanılan gömülü sistemler için bir aygıt profili spesifikasyonu ve uluslararası standartlaştırılmış (CENELEC EN 50325-4, 2002'den beri) CAN tabanlı bir yüksek katman protokolüdür. OSI modeliyle ilişkilendirilirse, CANopen ağ katmanını ve üzerindeki katmanları yürütür. CANopen standardı adresleme şemasından, küçük iletişim protokollerinde ve bir aygıt profiliyle tanımlanan uygulama katmanından oluşur. İletişim protokolleri ağ yönetimi, aygıt izleme (device monitoring) ve düğümler arası iletişimden sorumludur. Bu iletişime basit bir mesaj bölümlendirme/birleştirme (message segmentation/desegmentation) de dahildir. Veri bağı katmanı ve fiziksel katman gibi daha düşük düzeydeki katmanlarda ise genelde CAN (Controller Area Network) kullanılır. Ayrıca Ethernet Powerlink, EtherCAT, SafetyNet, ve Varan gibi diğer iletişim yöntemleri de CANopen'ın aygıt profilini uygulayabilir.

CANopen protokolü, 1992 yılında kurulan CAN in Automation (CiA) adlı kâr amacı gütmeyen bir kuruluş tarafından geliştirilmiştir. Birkaç şirket tarafından kurulan kurum, bugün kullanıcı ve üreticileri dahil 540 üyeye sahiptir. Protokolün uluslararası standartlaştırılması, tanıtımı ve geliştirilmesinden sorumludur ve Nuremberg (Almanya) kentinde kayıtlıdır. Kurumun kardeş şirketi sayılan CiA GmbH ise sertifikasyon verme ve talimatlarını üretme işini üstlenmiştir. Bunların dışında CiA, CAN Newsletter adlı bir teknik yayın ve CAN ile ilgili ürün ve hizmetlerin klavuzlarını da yayımlamaktadır. CiA ayrıca CANopen güvenlik protokolünü gerçekleyen CSC02 entegre devresini ve (CiA 304) eğitim kitini de sağlamaktadır.

CANopen protokolünün kullanıldığı alanlar; laboratuvar, fabrika, bina ve makina otomasyonu, makine denetimi, tıbbi araçlar, arazi ve demiryolu araçları, denizcilik elektroniği ve enerji üretimi gibi geniş bir yelpazede kullanım alanına sahiptir.

- CAN in Automation (CiA) taslak standartları
- CANopen aygıt profilleri
- DS 401 - Genel giriş çıkış (I/O) modülleri
- DSP 402 - Sürücüler ve hareket denetimi (motion control)
- DS 405 - IEC 61131-3 Programlanabilir aygıtlar

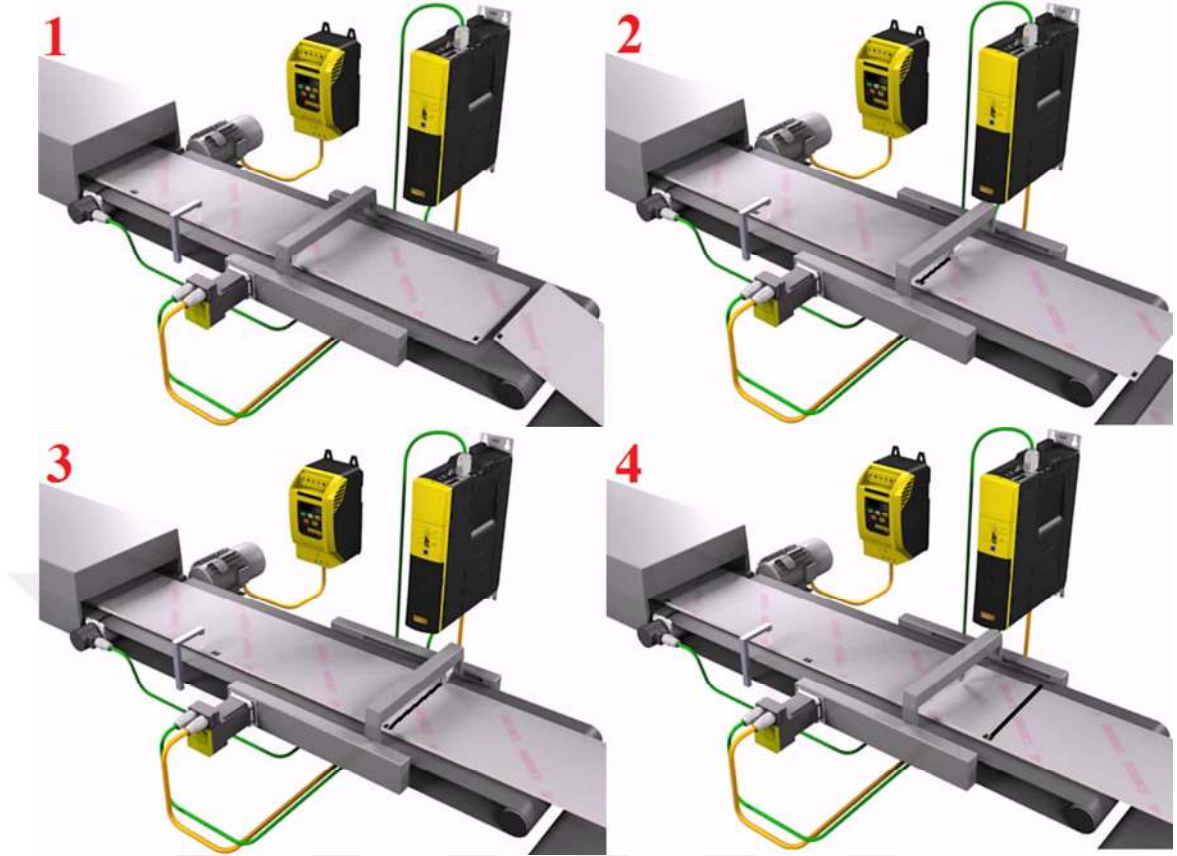
Yapılan çalışmada tüm servo sürücüler sisteme CanBus haberleşme ağıyla bağlanmıştır. Ve tüm yönetim CANopen Motion protokolüyle yapılmıştır. CANopen protokolü hem hızlı olması hem de PLCopen ve PLCmotion servo kontrol bloklarını desteklediği için sistemin programlanmasında hem kolaylık sağlamış hem de servolara ait tüm verilerin haberleşmeyle geri besleme (feedback) olarak alınabilmesi sayesinde avantaj sağlamıştır.

## 2.9. Lineer Makas (Uçan Makas, Uçan Testere)

Lineer makas prensip olarak bir eksenin ikinci bir eksene senkronize olma işlemidir. Birinci ve ikinci eksen birbirleriyle aynı hızda hareket ederken, bir kesme işlemi ya da herhangi başka bir işlem yapılabilir. Kesme işlemi bir kere bittikten sonra testere bir sonraki harekete hazır olmak için başlangıç pozisyonuna gider. Kontrolör tarafından, senkron olması istenilen eksenin uzunluğu ve hızına bağlı olarak başlama anı hesaplanır ve optimize edilir. Eğer bir sonraki pozisyona kadar yardımcı eksen hazır olamıyorsa buna sebep olan 2 etmen olabilir; bunlardan birincisi ana eksenin çok hızlı olmasıdır, ikincisi ise çok kısa mesafede bir senkronizasyon istenmesidir. Bu durumda ikinci eksen ana eksene yetişemediği için kesim kaybı olarak çıkış verilir.

Malzemeye göre yardımcı eksen konfigüre edildikten sonra kontrolör tarafından ayarlanır. Senkronizasyon için eksen her çevrimde pozisyona göre hesaplanan değer ile beslenir. Bir sonraki başlangıç noktası için ulaşma zamanı ayarlanır.

Lineer makas; yardımcı eksen ana eksene operasyonel bir işlemi gerçekleştirmek için senkron olur. Bu şekilde bir ana eksene senkron olma işlemi, bir makina iletim halindeyken o işlemin yapılabilmesi anlamına gelmektedir. İki farklı lineer makas vardır, universal lineer makas ve normal lineer makas arasındaki fark yardımcı eksenin senkron olmasındaki ilk durumların farklı olabilmesidir. Universal lineer makas, normal lineer makasın aksine, yardımcı eksen harekete başladıktan sonra da senkronizasyonu sağlayabilir ki bu yüzden eksenin durağan olması zorunlu değildir. Universal lineer makas, geliştirilen ayar değer profillerini hesaplayabilir, bu koşulların kullanıcı tarafından geniş sınırlar içinde değiştirilebilme imkanı vardır. Ana eksen ile yardımcı eksen arasındaki senkron durum için hız oranı kuplör faktörü parametresi ile belirtilir. Kuplör oranının 1 olduğu durumda yardımcı eksen ana eksene senkron hareket ederken paralel bir şekilde aynı yönde ve aynı hızda eşlik eder. Şekil 2.24’de örnek bir Lineer Makas mekanizması görülmektedir.



Şekil 2.24. Örnek Uçan Makas Mekanîği

Universal lineer makas uygulaması en genel şekilde iki tip senkronizasyonu destekler. Hıza senkronizasyon uygulamasında yardımcı eksen ana eksene mümkün olan en hızlı şekilde senkron olur. Buna göre ana eksen ile yardımcı eksenin senkron olacağı pozisyon eksen dinamiklerinin izin verdiği sınırlar içinde en hızlı şekilde hedef olarak belirlenir. Tam tersinde ise, ana eksen ile yardımcı eksenin senkron olma pozisyonu kullanıcı tarafından senkronizasyon pozisyonu parametresinden ayarlanabilir. Bu durumda ana ve yardımcı eksen hareketleri son pozisyonlarından senkron olma pozisyonu doğrultusunda olur. Bütün senkronizasyon yöntemlerinde, senkronizasyon fazını belirleyici sınır koşullarında izin verir. Hıza Senkronizasyonda hız oranı aşağıdaki gibidir:

$$V_{yardimci\_eksen} = F_{kuplor\_orani} \times V_{ana\_eksen} \quad (2.14)$$

### 2.9.1. Senkronizasyon işlem sırası

Ana Eksene, yardımcı eksenin senkronizasyon işlemi aşağıdaki sırayla gerçekleşmektedir:

- 1- Universal lineer makas başlatılır. Bu lojik olarak yardımcı eksenin ana eksene bağlanma işlemidir. Yükselen kenar tetiklemesi başlatma için yeterlidir. Bu zaman bağlama zamanı olarak geçmektedir.

- 2- Senkronizasyon fazı: Yardımcı eksenin ilk koşullarında belirlenen sınır koşullarına göre hızlanmaya başladığı durumdur. Senkronizasyon fazından senkron faza geçiş süresine senkronizasyon zamanı denir.
- 3- Senkron faz: Yardımcı eksen ana eksenle senkron hareket eder.
- 4- Universal lineer makasden ayrılma durumu. Ana eksene bağlanmış yardımcı eksen senkron durumdan ayrılır ve bağımsız bir ana eksen olurken hızında bir değişiklik olmadan devam eder.
- 5- Yardımcı eksen ya durur ya da işleme yeniden başlar.

### 2.9.2. Pozisyona senkronizasyon

Bu modda yardımcı eksenin ana eksene kullanıcı tarafından belirlenen pozisyonda senkron olma işlemidir. Bunun anlamı verilen senkronizasyon pozisyonunda ana eksenle yardımcı eksen aynı hızda olur ve bundan sonra ana eksene senkron devam eder. Bu modda da senkronizasyon prosedürü aynı adımları izler.

### 2.9.3. Parametrelendirilebilir sınır koşulları

Prensipite, ana eksen ile yardımcı eksenin birbirine senkronizasyonu için herhangi bir başlangıç koşulu için uygulanabilir yapıdadır. Yardımcı eksenin ilk koşulundan senkron hale geçişi kullanıcı tarafından tanımlanabilen sınır koşulları ile belirlenir ve daha sonra hep bu şekilde bir geçiş sürekli hale getirilir. Sınır koşullarının uygulanmasındaki amaç ise yardımcı eksenin maksimum hızını sınırlamak ya da pozisyonda aşım yapmasını engellemektir. Sınırlar koşullarının parametrelendirilebilir olması senkronizasyon fazı için karakteristik değerlerin belirlenmesinde önemli rol oynar. Karakteristik değerlerin belirlenmesinde, ideal çıkarım ana eksenin senkronize olma zamanından sonra ivmesiz bir şekilde yani sabit hızda ilerlediği durumdur. Tam hesaplama ve kontrolün yapılması için bu durumun oluşması gerekmektedir. Ana eksenin senkrona girme zamanından sonrası için yapılan başka bir çıkarım içinde olması mümkün değildir. Ana eksenin hızlanma eğilimde olması durumunda bu yardımcı eksen de etkileyecektir ve bu da senkronizasyona doğrudan etki etmektedir. Ana eksende meydana gelebilecek ivmelenme, ivmelenmenin şiddetine göre bazı durumlarda hesaplamalarda değişikliğe neden olur ki bu da bazen aşım bazen düşüm şeklinde meydana gelir.

### 2.9.4. Senkronizasyon koşulları

Lineer makas için eksen senkronizasyonu en genel şekilde yardımcı eksenin ivme parametresine bağlıdır. Senkronizasyon koşulları aşağıdaki eksen değişkenlerine bağlıdır:

- 1- Yardımcı eksen senkronizasyon pozisyonu
- 2- Yardımcı eksen temel pozisyonu, lojik olarak bağlanma pozisyonu
- 3- Ana eksenin senkronizasyon pozisyonu

- 4- Lojik olarak ana eksenin bağlanma pozisyonu (ana ve yardımcı eksenin bağlanma pozisyonu)
- 5- Dinamik ana eksen bağlanma pozisyonu (yardımcı eksenin dinamik başlamasına bağlı ana eksen pozisyonu)

Yardımcı eksen durağan halden senkron hale gelebilmesi için belli bir zaman geçmesi gerekir. Bu zaman süresince ana eksen dinamik bağlanma pozisyonundan senkron pozisyonuna kadar olan yolu alır. Bunun anlamı yardımcı eksen senkronizasyon pozisyonu, yardımcı eksen bağlanma pozisyonu, ana eksen senkronizasyon pozisyonu, dinamik ana eksen bağlanma pozisyonundan 3"ü birbirinden bağımsızdır dinamik ana eksen bağlanma pozisyonu buradan çıkarılır. Bu pozisyonlar arasındaki temel ifade aşağıda verilmiştir:

$$| \text{Dinamik ana eksen bağlanma pozisyonu} - \text{Ana eksenin senkronizasyon pozisyonu} | = 2 \times | \text{Yardımcı eksen bağlanma pozisyonu} - \text{Yardımcı eksen senkronizasyon pozisyonu} |$$

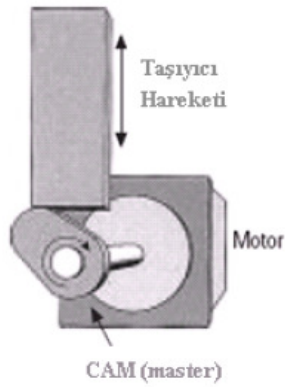
Bu denklemden de anlaşılacağı üzere dinamik ana eksen bağlanma pozisyonu ile senkronizasyon pozisyonu arasındaki mesafe, yardımcı eksen bağlanma pozisyonu ile senkronizasyon pozisyonu arasındaki mesafenin 2 katından fazla olması gerekmektedir. Başlangıçta, iki eksenin de senkronizasyon pozisyonu ile yardımcı eksenin bağlanma pozisyonu bilindiği için ana eksen dinamik bağlanma pozisyonu bu değerlerden hesaplanabilir. Eğer ana eksen, bağlanma pozisyonuna yardımcı eksen yetişemeyecek kadar yaklaştığı için senkronizasyon gerçekleşmez ise, bağlanma bir hata mesajı ile iptal edilir.

## 2.10. CAM Profili (Kam)

CAM, makine dinamiğinde sırası ile kendisine temas eden düzenekleri harekete geçirmeye yarayan bir mekanizmadır. 18. ve 19. Yüzyıllarda metal bir silindirin üzerine yerleştirilmiş çok sayıda kam sayesinde ardışık veya döngüsel hareketler elde edilebilmekteydi. Aynı düzenek müzik kutularında ve laterna mekanizmalarında da kullanılmaktaydı. Mekanik robotlar ve otomatlar için bir tür programlayıcı sayılabilecek kam düzenekleri bugün modern otomobil motorlarında da eksantrik milinin üzerinde konumlanarak yakıt ve egzoz sübaplarının açılmasını ve kapanmasını kontrol etmektedir.

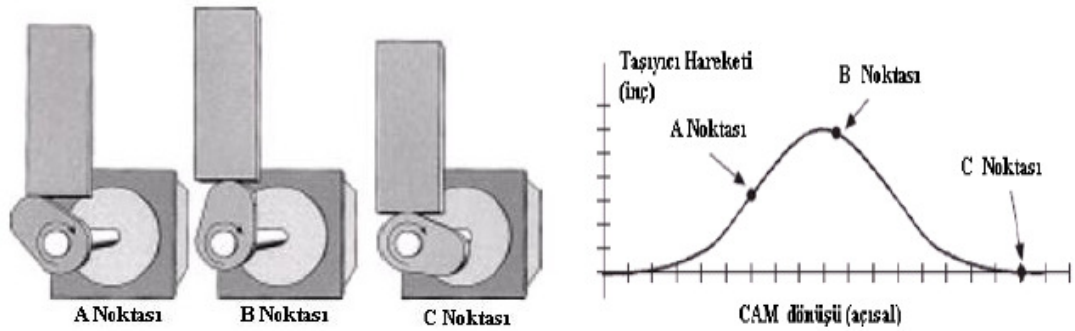
### 2.10.1. Mekanik CAM

CAM sistemleri otomasyonda yüzyıllardır kullanılmaktadır. Leonarda Da Vinci CAM ile başarılabilecek potansiyel hareketleri yapacak bir birçok CAM mekanizması çizmiştir (Çengelci ve Çimen 2005). Şekil 2.25'de mekanik CAM ile taşıyıcı örneği verilmiştir. CAM sabit hızda dönen bir motor miline bağlanmıştır. CAM tam bir daire olmadığı için taşıyıcıya CAM profiline bağlı olarak lineer bir hareket verilmektedir. İstenilen harekete bağlı olarak CAM'ın şeklinin değişmesi gerekmektedir. CAM kullanmanın bir avantajı sürekli tekrar edilebilir ve geçici ara verilebilen bir hareket elde edilmesidir.



Şekil 2.25. CAM ile Hareket Ettirilen Taşıyıcı

Makina üreticileri CAM üretme konusunda çok iyi adapte oldular ve standart haline gelen CAM'lar geliştirilmiştir. CAM profilleri ana eksen ile takip edici eksen (taşıyıcı) arasındaki pozisyonel ilişkiyi meydana getirmektedir bunu Şekil 2.26'da resmetmektedir. CAM sistemlerinin vazgeçilmez özelliği ana ve yardımcı eksen arasındaki ilişkinin devamlı tekrar etmesidir. Ana ve yardımcı eksen pozisyonları senkron olduktan sonra CAM noktasının her yerinde bu ilişki devam eder (Woelfel 1999).



Şekil 2.26. CAM Profiline Göre Taşıyıcı Hareketi

CAM'lar mekanik cihazlarda istenilen hareketi istenilen bir yüzeye göre şekillendirme olup, çok yaygın kullanılmaktadır. En çok kullanılan CAM profili dairesel ark CAM tipidir, bu şekilde adlandırılmasındaki neden dairesel arkların birleşimi olmasındandır. Bu tipte CAM'lerin üretimi kolaydır ve ölçeklendirmek de kolay olduğu için küçük olması gereken yerlerde de tercih edilir. Üretim olarak da basit bir yapısı vardır ve ekonomik şekilde imal edilebilir (Hsieh 2009). Literatürde dairesel ark CAM dizaynı ve sentezi için sistematik metod eksikliği bulunmaktadır. Metodu dört adımda anlatılmıştır:

- 1- CAM profiline uyan geometrik model inşa edilir.

- 2- Homojen koordinatlar geçişi ile CAM profili denkleminin türevleri elde edilir.
- 3- Dizayn edilen CAM'ın kinematik karakteristiği analiz edilir.
- 4- CAM profilinin karakteristik çözümlemelerine bağlı olarak makineye işlemesi için NC bilgisi elde edilir.

Bu metodoloji kullanılarak elde edilen CAM'ları mekanik cihazlarda kullanmak mümkündür; fakat çok zahmetli bir işlem olmakla beraber hesap hatalarından doğacak yanlışlıklar, yanlış hareket profili ve mekanik sonuçlar getireceğinden yapım aşamasında dikkatli olunması gerekmektedir. Bunun yerine günümüzde elektronik CAM'ların kullanımı yayılmaktadır. Elektronik CAM'larda değişiklik PLC çevriminde bile yapılabilmektedir. CAM profilini tanımlayan noktalar tanımlanır ve bu noktalar arasında matematiksel bir bağıntı tanımlanarak bu bağıntıya göre yardımcı eksen hesaplanan noktalara hareket eder.

### 2.10.2. Elektronik CAM

Hareket kontrol sistemleri arttıkça, mekanik sistemlerin fonksiyonlarını kopyalanması mümkün hale gelmiştir. Mekanik CAM'daki temel eksiklik kullanılacak CAM işlendikten sonra eğer değiştirilmek istenirse yeniden yapılması gerekmesidir. Bu prototip yapılacak makinalar için önemli bir sorun teşkil etmektedir; çünkü hareket profilleri deneysel olarak bulunur. Mekanik CAM yerine PLC kullanıldığında değişmesi gereken eğriler kolay bir şekilde hızlıca modifiye edilebilir. Elektronik CAM servo motor ve diğer ekipmanlara dayalı mekanik CAM'ın yaptığını simule eden elektronik bir ekipmandır (Jiazhong ve Lei 2009). Ayrıca hafızada bulunan profil tiplerine göre optimize edilerek değiştirilebilir. Dahası mühendisler, daha önceden yapılamayan makinaları elektronik CAM ile aşına olduklarında yapılabilir hale getirebilir. İlk başta verilen taşıyıcı örneğini elektronik CAM ile yapılmasını göz önüne alınacak olursa: Elektronik bir sistem aracılığı ile motorun dönmesi enkoder ile algılanır. Mekanik olarak dizayn edilen parça yerine hareket kontrolöründe bu şekle karşılık eğriler saklanır. Hareket kontrolörü orijinal motor hareketine bağlı olarak servo motora gitmesi gereken yeri hesaplayarak belirler. Hareket kontrolörü birçok hareket profilini hafızasında bulundurduğu için, CAM-taşıyıcı arasındaki bağıntı bu profilin değişmesiyle kolaylıkla yeni şeklini alır. Bu uygulama için elektronik CAM kullanmak daha hızlı çevrim süreleri elde etmeye olanak sağlar.

### 2.10.3. CAM ile döner bıçak uygulaması

Döner bıçak uygulaması dönüştürme ve baskı endüstrilerinde sıkça kullanılmaktadır. Döner bıçak mekanizması, bir eksene bağlanmış bıçak ağzından

oluşmaktadır. Silindir döndükçe bıçak altından geçen malzemeyi kesmektedir (Woelfel 1999). Bıçağın tanjant hızı altından geçen malzemeye eşit olduğunda, kesilen malzeme boyu bıçak çevresine eşit olur. Eğer tanjant hızı akan malzemedan daha büyük olursa daha kısa boyutlu kesimler elde edilir.

En iyi kalitede ve malzemeye zarar vermeden kesebilmesi için, kesim esnasında altından geçen malzeme ile bıçağın lineer hızı eşit olmalıdır. Eğer kesilmesi gereken boyut bıçak çevresinden kısa olması isteniyorsa, bıçak kesme esnası dışında daha hızlı hareket eder ve daha sonra yavaşlayarak bir sonraki kesime yetişebilmesi gerekir. Tam tersi durumu da geçerlidir, daha uzun kesim boyu için bıçak kesim dışında daha yavaş gider ve keserken gene aynı hıza ulaşması gerekir.

Birçok elektronik hareket kontrol sistemi bu yapıya kolayca adapte olurken elektronik dişli ve dişli üzerine eklemeli hareket potansiyeli kullanır. Fakat döner bıçak uygulamasında bu denemeyi yaparken zorlukla karşılaşılır çünkü bıçak bütün bir hareket boyunca direkt olarak dişli oranı ile çalışmaz. Bu uygulama için elektronik CAM en ideal seçimdir.

Elektronik CAM'dan önce tasarım:

Elektronik CAM'in özelliklerini daha iyi anlamak için, CAM olmayan bir hareket kontrolörü kullanıldığında dönel bıçağın kontrolü için nasıl bir yöntem izlenildiğine bakmak gerekir:

- 1- İlk önce elektronik dişli oranı tanımlanır ve aktif edilir (bıçağın altından akan malzemeye senkron hızda dönmesi sağlanır).
- 2- Ana enkoderden kesiciye gelecek olan ürün boyuna göre hızlanma ya da yavaşlama talimatı beklenir.
- 3- Kesim noktasından senkron hız üzerinde ekstra bir hareket yaparak kesim boyuna ulaşılır.
- 4- Ekstra hareket yapıldıktan sonra kesicinin bir sonraki hızlanma yavaşlanma noktası hesaplanır.
- 5- 2.adıma dönülüp ana enkoderden gelecek olan talimata göre davranılır.

Bu yöntem aslında çok basit durmaktadır, fakat bazı ince noktaları da bulunmaktadır. Hızlanma ve yavaşlama yerlerini iyi analiz etmek gerekir ve sistem dinamiklerini göz önünde bulundurmak lazımdır. Pozisyon düzeltmek için hız değişim noktalarında ani ivme değişimleriyle karşılaşılabilir. Bunun nedeni de zaten yüksek

hızda olan bir motora hızlanma veya yavaşlama yapıldığında motor maksimum hızına ulaşmadan izin verilen değerler içinde kalacak şekilde ulaşılması gereken hıza gelme ihtiyacından doğmaktadır. Bu özellik daha düşük hat hızlarını için daha tolere edilebilir; fakat en kötü senaryo düşünülerek yüksek hızlar için hızlanma profili belirlenmelidir. Bazen yavaşlama yapılması istenildiğinde hat hızının da altına inilerek ters yönde bir hareket görmek mümkündür ki, bu en kötü senaryodur çünkü kesici kesme yüzeyi ile tekrar buluşabilir ve kesme yönünün tersindeyken temas eder. Bu operasyon malzemenin ya da kesicinin hasar alması ile sonuçlanır, bu yüzden eskiden böyle durumları da göz önüne alarak tasarım yapılması gerekmektedir.

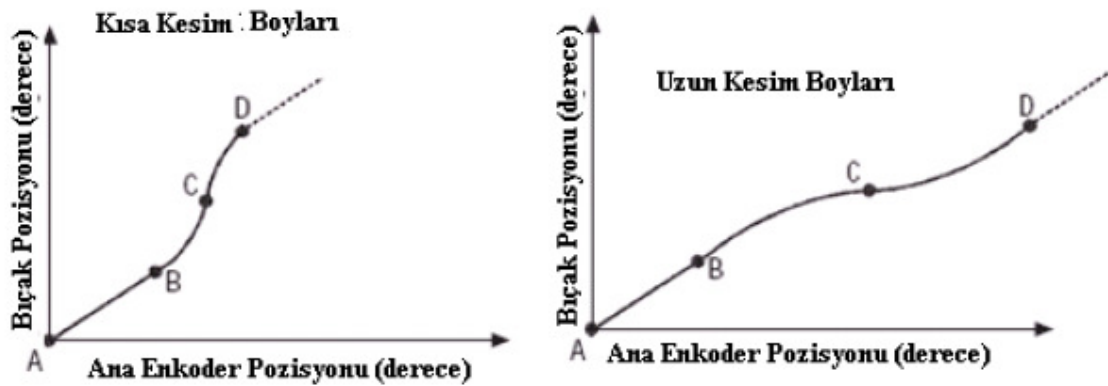
Elektronik CAM kullanarak tasarım:

Aynı uygulama için Elektronik CAM kullanmak oluşabilecek hataları düzelttiği gibi hareket programını da basitleştirmektedir. Hareket yapısı şu şekildedir:

- 1- Referans aldırılan kesici kesmesi gereken noktaya gelir ve CAM ile ana eksene bağlanır.
- 2- Elektronik CAM'a göre hareket edilir.

Bu yöntemle göre kesici belirlenen hız ve ivme profillerine göre hareket eder, ani hız değişimlerinden öte istenen şekilde yumuşatılmış hareketler görmek mümkündür (Şekil 2.27). Yöntemin ters gitmeyi engelleyici bir artısı bulunmaktadır. Ayrıca değişen kesim boylarında yapılması tek gereken oluşturulan tabloyu ölçeklendirmektir. CAM profili şu bölümlerden oluşmaktadır:

- 1- Sabit hız bölümü (kesim anındaki hız)
- 2- Hızlanma bölümü
- 3- Yavaşlama bölümü



Şekil 2.27. Boy Uzunluğuna Göre Bıçak Pozisyonu

Yardımcı eksen hareketi hesaplanan pozisyonlara göre davranır. Bu hesaplama kübik B-eğrilerine göre interpolate edilip, optimizasyon algoritmasının birleşimi ile elde edilir. Optimizasyon hız, ivme, ivmenin türevi gibi değişkenlerin limitlerine göre en iyi sonucu verme şeklindedir. B-eğrileri CAM eğrilerine göre ana pozisyonlara en iyi uyan eğriyle hesaplanır. Bu hesaplama için lagrange interpolasyonu kullanılarak lagrange eğrisi üretilir. Bu yöntem ile ölçüm hatası, tahmin edilen yörünge gürültüleri gibi büyüklükler ortaya çıkmaktadır. Ana eksenin hızı yük ve dış etmenlerin eklenmesi sonucu değişebilir. Sonuç olarak ana eksen her zaman sabit hızda devam etmez ve ortaya harmonikler çıkar. Hız tahmini metoduyla harmonikler ne kadar azaltılsa da, hızda değişimler görmek mümkündür. N.dereceden polinom uydurma yöntemi kullanarak ana eksen hızı zamana göre şu şekilde ifade edilir (Liao ve Jeng 2003).

$$\omega = \sum_{i=0}^N c_i t^i \quad (2.15)$$

Bu denklemdeki sabit C katsayılarını bulmak için iki prosedür vardır. Birinci prosedürde, inci dereceden polinom ekstrapolasyon yapılır. k gerçek zamanlı sayıcının sembolüdür. T de PC-tabanlı programın örnekleme zamanıdır, kT geçen zamanı ifade etmektedir. Ana prosedürde  $t = kT$ ,  $1 \leq k \leq N+1$  sabit N.dereceden ekstrapolasyondur:

$$\sum_{i=0}^N c_i (j.t)^i = w_l, l=(k-N-1) \text{ to } (k-1), j=l-(k-N-1) \quad (2.16)$$

## 2.11. PLCopen Blokları

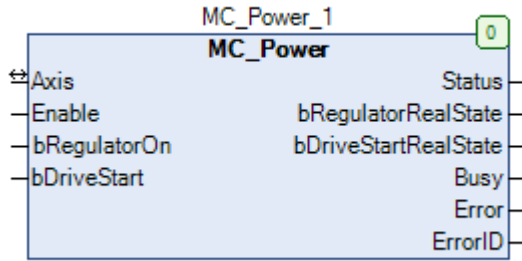
PLCopen 1992 yılında IEC 61131-3 standardının yayınlanmasından sonra kurulmuştur. PLCopen kullanıcıların ihtiyaçlarına göre endüstriyel otomasyon alanında verimlilik sağlayan bağımsız bir kuruluştur .

Otomasyon sektörü çok heterojen bir pazar ve bir çok çeşit PLC programlamak için bir çok çeşit yöntem bulunmakta. PLCopen üyeleri, IEC 61131-3 standardı üzerinde yoğunlaşarak endüstride maliyet azaltmak amacıyla teknik özellikleri ve uygulamaları yaratmaktadır. PLCopen uzmanları farklı uygulamalara yönelik standart kütüphaneler oluşturarak son kullanıcılara kolaylık sağlamaktadırlar (URL-4).

PLCopen blokları, PLC ile motor (servo, asenkron vb.) yönetimini kolaylaştırmaktadır ve farklı PLC'ler için bile aynı standartta programlama sağlamaktadır. Unutulmaması gerekir ki PLCopen blokları yalnızca haberleşme ile yönetilecek sürücüler için geçerlidir. Haberleşmeden kasıt, CANopen, EtherCat, ProfiNet tarzı makine kontrolüne yönelik endüstriyel haberleşmelerdir. Bu haberleşmelerden herhangi birine sahip bir sürücüyü, IEC 61131-3 standardına sahip bir PLC ile programlamak istersek bu sürücülerini yönetmek için PLCopen bloklarını kullanabiliriz (URL-4).

### 2.11.1. Mc\_Power fonksiyon bloğu

Kontrol edilecek eksenin hazır konuma geçmesini yani “rdy” konumundan “run” konuma geçmesini sağlayan motor kontrol bloğudur. Unutulmaması gerekir ki bu blok “Enable” değilken Mc\_Reset, Mc\_Stop gibi önemli kontrol blokları dahil hiçbir PLCopen bloğu çalışmaz. Mc\_Power fonksiyon bloğu Şekil 2.28’deki gibidir. Girişlerinin ve çıkışlarının açıklamaları Çizelge 2.2’de görüldüğü gibidir.



Şekil 2.28. Mc\_Power Fonksiyon Bloğu

Çizelge 2.2. Mc\_Power Fonksiyon Bloğu Giriş-Çıkış Çizelgesi

|                      | Data Tipi | Başlangıç Değeri | Tanım                                                          |
|----------------------|-----------|------------------|----------------------------------------------------------------|
| <b>VAR_INPUT</b>     |           |                  |                                                                |
| Enable               | BOOL      | FALSE            | Fonksiyon bloğunun çalışması için “TRUE” olmalı.               |
| bRegulatorOn         | BOOL      | FALSE            | Güç kademesini(Power Stage) etkinleştirmek için “TRUE” olmalı. |
| bDriveStart          | BOOL      | FALSE            | AcilDur(QuickStop) durumu yoksa “TRUE” olmalı.                 |
| <b>VAR_OUTPUT</b>    |           |                  |                                                                |
| Status               | BOOL      | FALSE            | Eksen harekete hazır ise “TRUE”.                               |
| bRegulatorRealState  | BOOL      | FALSE            | Güç kademesinin etkin durumu.                                  |
| bDriveStartRealState | BOOL      | FALSE            | Sürücü AcilDur ile bloke edilmemiş ise “TRUE”.                 |

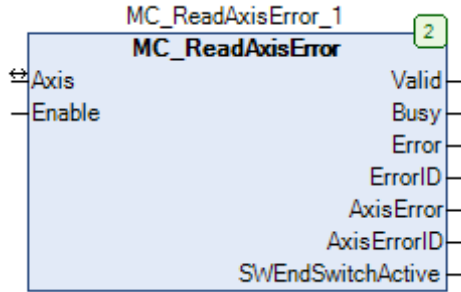
### 2.11.2. Mc\_ReadAxisError fonksiyon bloğu

Kontrol edilecek eksenin herhangi bir hata durumuna düşüp düşmediği bu blok ile kontrol edilir. Aynı zamanda hatanın ne olduğunu da blok çıkışlarında bulunan “AxisErrorID” çıkışı ile öğrenebiliriz. “AxisErrorID” listesini üretici kendi belirler ve

“help” kısmına koyar. Mc\_ReadAxisError fonksiyon bloğu Şekil 2.29’daki gibidir. Girişlerinin ve çıkışlarının açıklamaları Çizelge 2.3’de görüldüğü gibidir.

Çizelge 2.3. Mc\_ReadAxisError Fonksiyon Bloğu Giriş-Çıkış Çizelgesi

|                   | Data Tipi | Başlangıç Değeri | Tanım                                                                |
|-------------------|-----------|------------------|----------------------------------------------------------------------|
| <b>VAR_INPUT</b>  |           |                  |                                                                      |
| Enable            | BOOL      | FALSE            | “TRUE” olduğu sürece hata durumu sürekli kontrol edilir.             |
| <b>VAR_OUTPUT</b> |           |                  |                                                                      |
| Valid             | BOOL      | FALSE            | Çıkışlar düzgün şekilde çalışıyor ise “TRUE”.                        |
| AxisError         | BOOL      | FALSE            | Eksende hata olduğunu bildirir bayrak.                               |
| AxisErrorID       | DWORD     | 0                | Üretici kaynaklı eksen hata kodu.                                    |
| SWEndSwitchActive | BOOL      | FALSE            | Eksen için tanımlanmış yazılımsal limit değerleri aşıldı ise “TRUE”. |



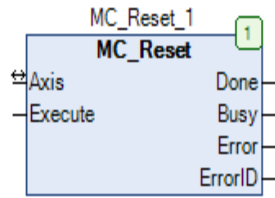
Şekil 2.29. Mc\_ReadAxisError Fonksiyon Bloğu

### 2.11.3. Mc\_Reset fonksiyon bloğu

Kontrol edilecek eksen herhangi bir hata durumuna düştüğünde, bu hata durumunun resetlenmesi gerekir. Bunun için Mc\_Reset fonksiyon bloğu kullanılır. Aynı zamanda; bazı durumlarda PLCopen blokları sürücülerden bağımsız olarak hataya düşebilmektedir. Örneğin bir PLCopen bloğu işlem yaparken yani “busy” konumdayken başka bir PLCopen bloğu işlem yaptırılmak istendiğinde hata durumu söz konusu olabilir. Hangi durumlarda blokların hataya geçip, hangi durumlarda bir önceki yürütmenin durdurulup yeni işleme geçilebileceği ile ilgili şema Şekil 2.80’de anlatılmıştır. Mc\_Reset fonksiyon bloğu Şekil 2.30’daki gibidir. Girişlerinin ve çıkışlarının açıklamaları Çizelge 2.4’de görüldüğü gibidir.

Çizelge 2.4. Mc\_Reset Fonksiyon Bloğu Giriş-Çıkış Çizelgesi

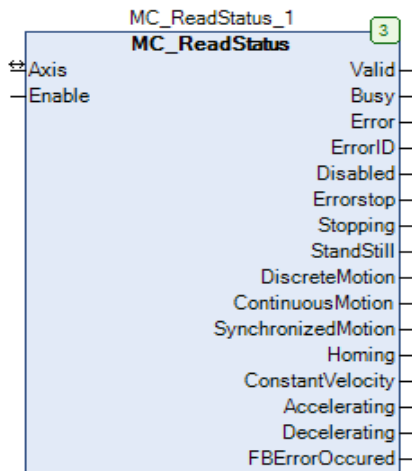
|                   | Data Tipi | Başlangıç Değeri | Tanım                                                               |
|-------------------|-----------|------------------|---------------------------------------------------------------------|
| <b>VAR_INPUT</b>  |           |                  |                                                                     |
| Execute           | BOOL      | FALSE            | Giriş sinyalinin yükselen kenarı ile fonksiyon bloğu işleme başlar. |
| <b>VAR_OUTPUT</b> |           |                  |                                                                     |
| Done              | BOOL      | FALSE            | Resetleme işlemi gerçekleşti ise "TRUE".                            |



Şekil 2.30. Mc\_Reset Fonksiyon Bloğu

#### 2.11.4. Mc\_ReadStatus fonksiyon bloğu

Kontrol edilecek eksenle ilgili bir çok diagnostik bilgiyi öğrenebileceğimiz fonksiyon bloğudur. Bunların arasında, eksenin hata durumunda olup olmadığı, durağan (standstill) konumda olup olmadığı, ev (home) pozisyonunda olup olmadığına kadar bir çok bilgi mevcuttur. Mc\_ReadStatus fonksiyon bloğu Şekil 2.31'deki gibidir. Girişlerinin ve çıkışlarının açıklamaları Çizelge 2.5'de görüldüğü gibidir.



Şekil 2.31. Mc\_ReadStatus Fonksiyon Bloğu

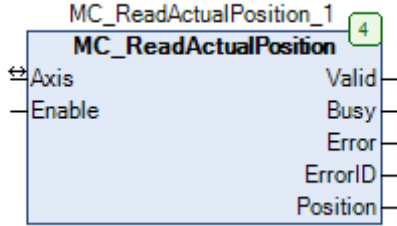
Çizelge 2.5. Mc\_ReadStatus Fonksiyon Bloğu Giriş-Çıkış Çizelgesi

|                    | Data Tipi | Başlangıç Değeri | Tanım                                                                    |
|--------------------|-----------|------------------|--------------------------------------------------------------------------|
| <b>VAR_INPUT</b>   |           |                  |                                                                          |
| Enable             | BOOL      | FALSE            | “TRUE” olduğu sürece eksen durumu sürekli kontrol edilir.                |
| <b>VAR_OUTPUT</b>  |           |                  |                                                                          |
| Valid              | BOOL      | FALSE            | Çıkışlar düzgün şekilde çalışıyor ise “TRUE”.                            |
| Disabled           | BOOL      | FALSE            | Eksen devre dışı iken “TRUE”.                                            |
| Errorstop          | BOOL      | FALSE            | Eksen “AcilDur” konumunda iken “TRUE”.                                   |
| Stopping           | BOOL      | FALSE            | Eksen duruyor ise “TRUE”.                                                |
| StandStill         | BOOL      | FALSE            | Eksen durağan konumda ise “TRUE”.                                        |
| DiscreteMotion     | BOOL      | FALSE            | Eksen ayrık-hareket yapıyor ise “TRUE”.                                  |
| ContinuousMotion   | BOOL      | FALSE            | Eksen sürekli-hareket yapıyor ise “TRUE”.                                |
| SynchronizedMotion | BOOL      | FALSE            | Eksen senkronize-hareket yapıyor ise “TRUE”.                             |
| Homing             | BOOL      | FALSE            | Eksen başlangıç(home) konumuna dönüyorsa veya daha önce yaptıysa “TRUE”. |
| ConstantVelocity   | BOOL      | FALSE            | Motor sabit hızda dönüyor ise “TRUE”.                                    |
| Accelerating       | BOOL      | FALSE            | Motor hızlanma aşamasında ise “TRUE”.                                    |
| Decelerating       | BOOL      | FALSE            | Motor yavaşlama aşamasında ise “TRUE”.                                   |
| FBErrorOccured     | BOOL      | FALSE            | Fonksiyon blok hataya geçti ve resetlenmedi ise “TRUE”.                  |

### 2.11.5. Mc\_ReadActualPosition fonksiyon bloğu

Kontrol edilen eksenin anlık pozisyonunu okumamızı sağlayan fonksiyon bloğudur.

Mc\_ReadActualPosition fonksiyon bloğu Şekil 2.32'deki gibidir. Girişlerinin ve çıkışlarının açıklamaları Çizelge 2.6'da görüldüğü gibidir.



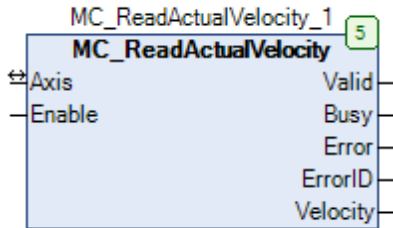
Şekil 2.32. Mc\_ReadActualPosition Fonksiyon Bloğu

Çizelge 2.6. Mc\_ReadActualPosition Fonksiyon Bloğu Giriş-Çıkış Çizelgesi

|                   | Data Tipi | Başlangıç Değeri | Tanım                                         |
|-------------------|-----------|------------------|-----------------------------------------------|
| <b>VAR_INPUT</b>  |           |                  |                                               |
| Enable            | BOOL      | FALSE            | “TRUE” olduğu sürece pozisyon sürekli okunur. |
| <b>VAR_OUTPUT</b> |           |                  |                                               |
| Valid             | BOOL      | FALSE            | Çıkışlar düzgün şekilde çalışıyor ise “TRUE”. |
| Position          | LREAL     | 0                | Eksenin anlık pozisyon bilgisi (br).          |

### 2.11.6. Mc\_ReadActualVelocity fonksiyon bloğu

Kontrol edilen eksenin anlık hızını okumamızı sağlayan fonksiyon bloğudur. Mc\_ReadActualVelocity fonksiyon bloğu Şekil 2.33'deki gibidir. Girişlerinin ve çıkışlarının açıklamaları Çizelge 2.7'de görüldüğü gibidir.



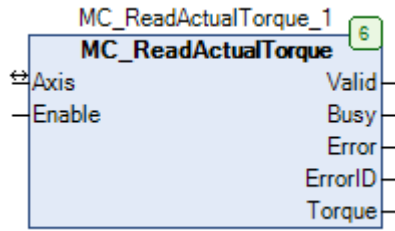
Şekil 2.33. Mc\_ReadActualVelocity Fonksiyon Bloğu

Çizelge 2.7. Mc\_ReadActualVelocity Fonksiyon Bloğu Giriş-Çıkış Çizelgesi

|                   | Data Tipi | Başlangıç Değeri | Tanım                                         |
|-------------------|-----------|------------------|-----------------------------------------------|
| <b>VAR_INPUT</b>  |           |                  |                                               |
| Enable            | BOOL      | FALSE            | “TRUE” olduğu sürece hız sürekli okunur.      |
| <b>VAR_OUTPUT</b> |           |                  |                                               |
| Valid             | BOOL      | FALSE            | Çıkışlar düzgün şekilde çalışıyor ise “TRUE”. |
| Velocity          | LREAL     | 0                | Eksenin anlık hız bilgisi (br/s).             |

### 2.11.7. Mc\_ReadActualTorque fonksiyon bloğu

Kontrol edilen eksenin anlık hızını okumamızı sağlayan fonksiyon bloğudur. Mc\_ReadActualTorque fonksiyon bloğu Şekil 2.34’deki gibidir. Girişlerinin ve çıkışlarının açıklamaları Çizelge 2.8’de görüldüğü gibidir.



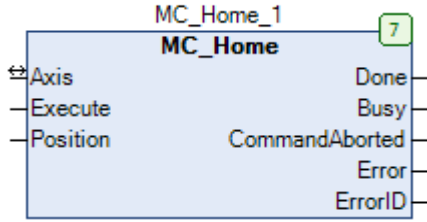
Şekil 2.34. Mc\_ReadActualTorque Fonksiyon Bloğu

Çizelge 2.8. Mc\_ReadActualTorque Fonksiyon Bloğu Giriş-Çıkış Çizelgesi

|                   | Data Tipi | Başlangıç Değeri | Tanım                                         |
|-------------------|-----------|------------------|-----------------------------------------------|
| <b>VAR_INPUT</b>  |           |                  |                                               |
| Enable            | BOOL      | FALSE            | “TRUE” olduğu sürece tork sürekli okunur.     |
| <b>VAR_OUTPUT</b> |           |                  |                                               |
| Valid             | BOOL      | FALSE            | Çıkışlar düzgün şekilde çalışıyor ise “TRUE”. |
| Torque            | LREAL     | 0                | Eksenin anlık tork bilgisi. (%)               |

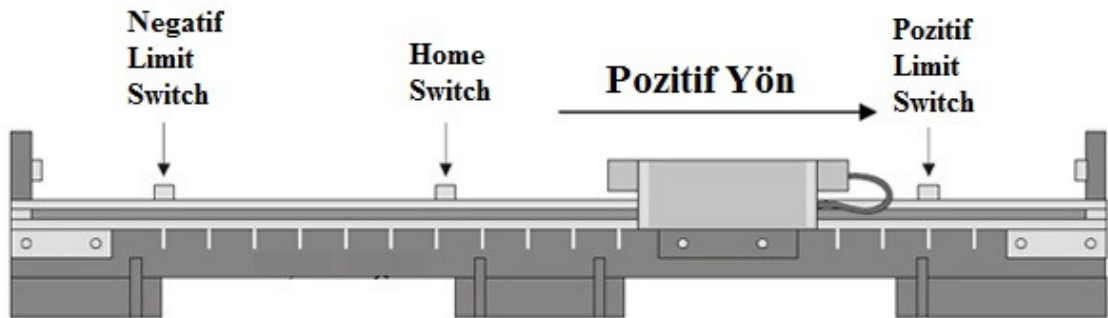
### 2.11.8. Mc\_Home fonksiyon bloğu

“Home” terimi, mekanizmanın başlangıç pozisyonu anlamına gelmektedir. Yani makinenin başlangıçta hangi pozisyonda olmasını istiyorsak makine çalıştırılmadan önce “homing” yaptırılır. Bu fonksiyon bloğunu tetikleyerek bu işlemi yaptırabiliriz. Mc\_Home fonksiyon bloğu Şekil 2.35’deki gibidir. Girişlerinin ve çıkışlarının açıklamaları Çizelge 2.9’da görüldüğü gibidir.



Şekil 2.35. Mc\_Home Fonksiyon Bloğu

Genelde lineer makas mekanizması ve switch pozisyonları Şekil 2.36’deki gibi olur. Servo’ya “Home” yap komutu geldiğinde servo geri yönde tanımlanan hızda dönerek “Home sensörü”nü arar. Home sensörü, sürücü üstündeki girişlere girilir ve giriş geldiğinde servo durur. Sürücü ve PLC bu durumda servonun aktüel pozisyonunu 0 (sıfır) kabul eder. Çeşitli home metodları olmakla beraber bazı PLC’lerin Mc\_Home blok girişinde bu metodu seçebilirken genel olarak sürücü içinden sabit tanımlı home metoduna göre işlem yapılır.



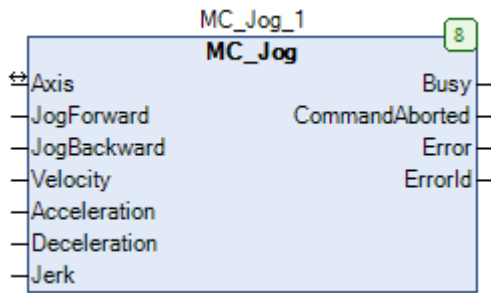
Şekil 2.36. Genel Servo Mekanizması

Çizelge 2.9. Mc\_Home Fonksiyon Bloğu Giriş-Çıkış Çizelgesi

|                   | Data Tipi | Başlangıç Değeri | Tanım                                                                           |
|-------------------|-----------|------------------|---------------------------------------------------------------------------------|
| <b>VAR_INPUT</b>  |           |                  |                                                                                 |
| Execute           | BOOL      | FALSE            | Giriş sinyalinin yükselen kenarı ile fonksiyon bloğu işleme başlar.             |
| Position          | LREAL     | 0                | Home sensörünü görüp, home pozisyonunda iken servonun kabul edileceği pozisyon. |
| <b>VAR_OUTPUT</b> |           |                  |                                                                                 |
| Done              | BOOL      | FALSE            | “Homing” işlemi gerçekleşti ise “TRUE”.                                         |

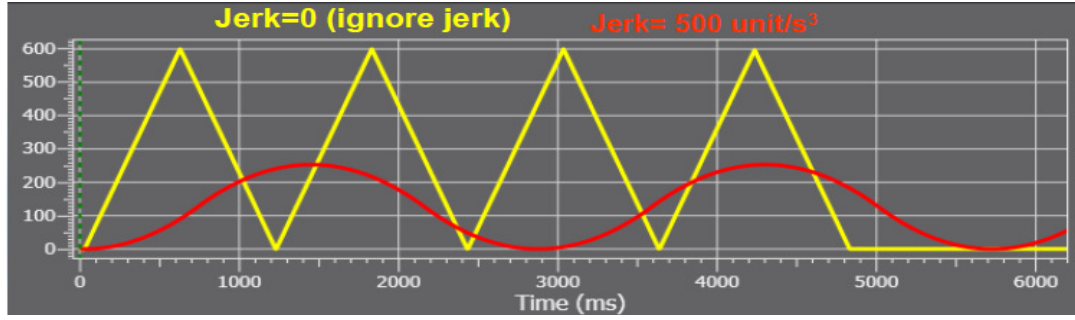
### 2.11.9. Mc\_Jog fonksiyon bloğu

Jog, motor kontrol için yaygın bir terimdir. Jog ile sürmek, haberleşmeli veya sürücü kontaklarından olsun şu anlama gelir; ileri veya geri komutu aktif (True-1-24Vdc) olduğu süre boyunca motor ileri veya geri döner. İleri komutu varken geri komutu, geri komutu varken ileri komutunun aktif olması durumunda motor mevcut durumunu korur. Komut pasif hale geldiği an motor yavaşlama (deceleration) moduna geçer. Jog modun dezavantajlarından birisi ileri veya geri komutu aktif iken hızının değiştirilememesidir. Motor ancak ileri-geri komutu kesilip, durduktan sonra hız değişimi gerçekleşebilir. Mc\_Jog fonksiyon bloğu Şekil 2.37'deki gibidir. Girişlerinin ve çıkışlarının açıklamaları Çizelge 2.10'da görüldüğü gibidir.



Şekil 2.37. Mc\_Jog Fonksiyon Bloğu

Jerk değeri sıfırdan farklı girildiğinde kalkış ve duruşlar daha yumuşak olur, bir nevi S rampası gibi çalışır. Jerk etkisini anlamak için Şekil 2.38 incelenebilir.



Şekil 2.38. Jerk Etkisi Grafiği

Çizelge 2.10. Mc\_Jog Fonksiyon Bloğu Giriş-Çıkış Çizelgesi

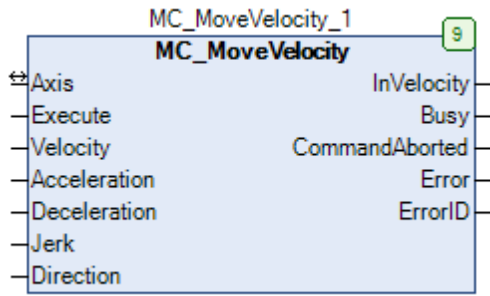
|                   | Data Tipi | Başlangıç Değeri | Tanım                                                                                                                                                 |
|-------------------|-----------|------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>VAR_INPUT</b>  |           |                  |                                                                                                                                                       |
| JogForward        | BOOL      |                  | “TRUE” olduğu sürece motor ileri döner.                                                                                                               |
| JogBackward       | BOOL      |                  | “TRUE” olduğu sürece motor geri döner.                                                                                                                |
| Velocity          | LREAL     |                  | Jog sürme için istenen hız.                                                                                                                           |
| Acceleration      | LREAL     |                  | Jog sürme için istenen hızlanma rampası.                                                                                                              |
| Deceleration      | LREAL     |                  | Jog sürme için istenen yavaşlama rampası.                                                                                                             |
| Jerk              | LREAL     |                  | Jog sürme için istenen yumuşatma katsayısı (Jerk).                                                                                                    |
| <b>VAR_OUTPUT</b> |           |                  |                                                                                                                                                       |
| Busy              | BOOL      | FALSE            | Motor hızı 0(sıfır) olmadığı sürece “TRUE”.                                                                                                           |
| CommandAborted    | BOOL      | FALSE            | İşlem başka bir PLCopen bloğu komutu ile yarıda kesilir ise “TRUE”. Bu komut ileri-geri komutu aktif iken “TRUE” kalır, pasif olduğu an “FALSE” olur. |
| Error             | BOOL      | FALSE            | Fonksiyon bloğunda hata var ise “TRUE”. Bu komut ileri-geri komutu aktif iken “TRUE” kalır, pasif olduğu an “FALSE” olur.                             |
| ErrorID           | SMC_ERROR | 0                | Fonksiyon bloğunda hata var ise hata kodu.                                                                                                            |

### 2.11.10. Mc\_MoveVelocity fonksiyon bloğu

Kontrol edilen eksenin hız modunda kontrolünü sağlayan fonksiyon bloğudur. Bu bloğun “execute” girişi aktif edildiğinde, motor blok girişinde tanımlanan hız ve hız profiliyle çalışmaya başlar. Unutulmaması gereken husus, bloğun “execute” girişinin yükselen kenar ile çalışması ve komut bir kere verildiğinde motora “Mc\_Stop” fonksiyon bloğu ile dur komutu gelene kadar çalışmaya devam etmesidir. Bu bloğun kullanım avantajlarından birisi de motorun durması beklenmeden yeni hız değerini geçerli kılabilmemizdir. Bunun için “execute” girişine tetik vermemiz yeterlidir. Mc\_MoveVelocity fonksiyon bloğu Şekil 2.39’daki gibidir. Girişlerinin ve çıkışlarının açıklamaları Çizelge 2.11’de görüldüğü gibidir.

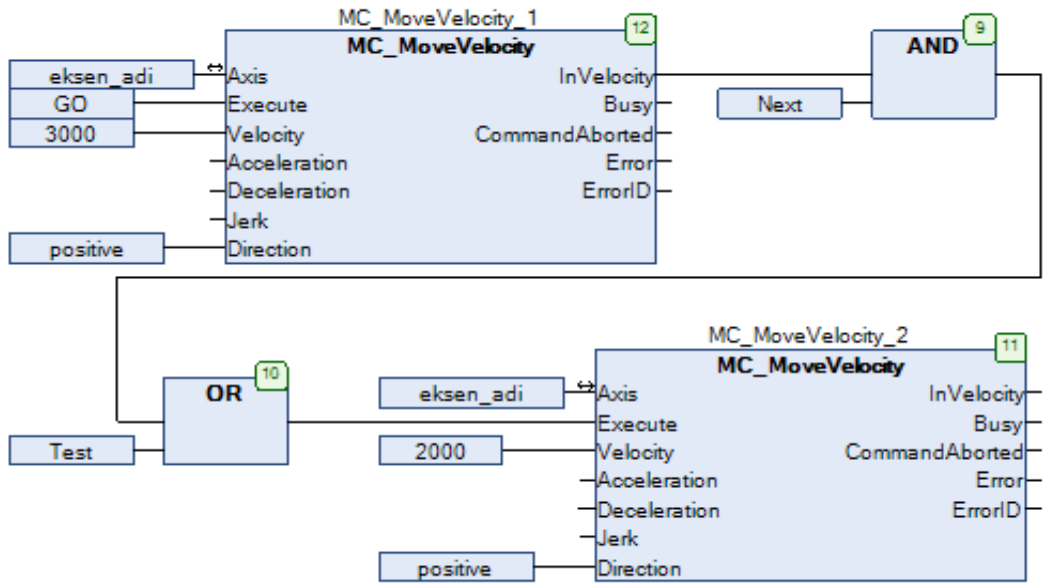
Çizelge 2.11. Mc\_MoveVelocity Fonksiyon Bloğu Giriş-Çıkış Çizelgesi

|                   | Data Tipi    | Başlangıç Değeri | Tanım                                                                                                                                          |
|-------------------|--------------|------------------|------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>VAR_INPUT</b>  |              |                  |                                                                                                                                                |
| Execute           | BOOL         | FALSE            | Giriş sinyalinin yükselen kenarı ile fonksiyon bloğu işleme başlar.                                                                            |
| Velocity          | LREAL        | 0                | Hız modunda sürme için istenen hız.                                                                                                            |
| Acceleration      | LREAL        | 0                | Hız modunda Jog sürme için istenen hızlanma rampası.                                                                                           |
| Deceleration      | LREAL        | 0                | Hız modunda sürme için istenen yavaşlama rampası.                                                                                              |
| Direction         | MC_Direction | current          | MC_Direction;<br>geçerli değerler:<br>positive : pozitif<br>negative : negatif<br>current : mevcut<br>shortest : en kısa<br>fastest : en hızlı |
| <b>VAR_OUTPUT</b> |              |                  |                                                                                                                                                |
| In Velocity       | BOOL         | FALSE            | İstenen hız değerine ilk ulaştığında “TRUE”.                                                                                                   |
| Busy              | BOOL         | FALSE            | Hızlanırken “TRUE”.                                                                                                                            |

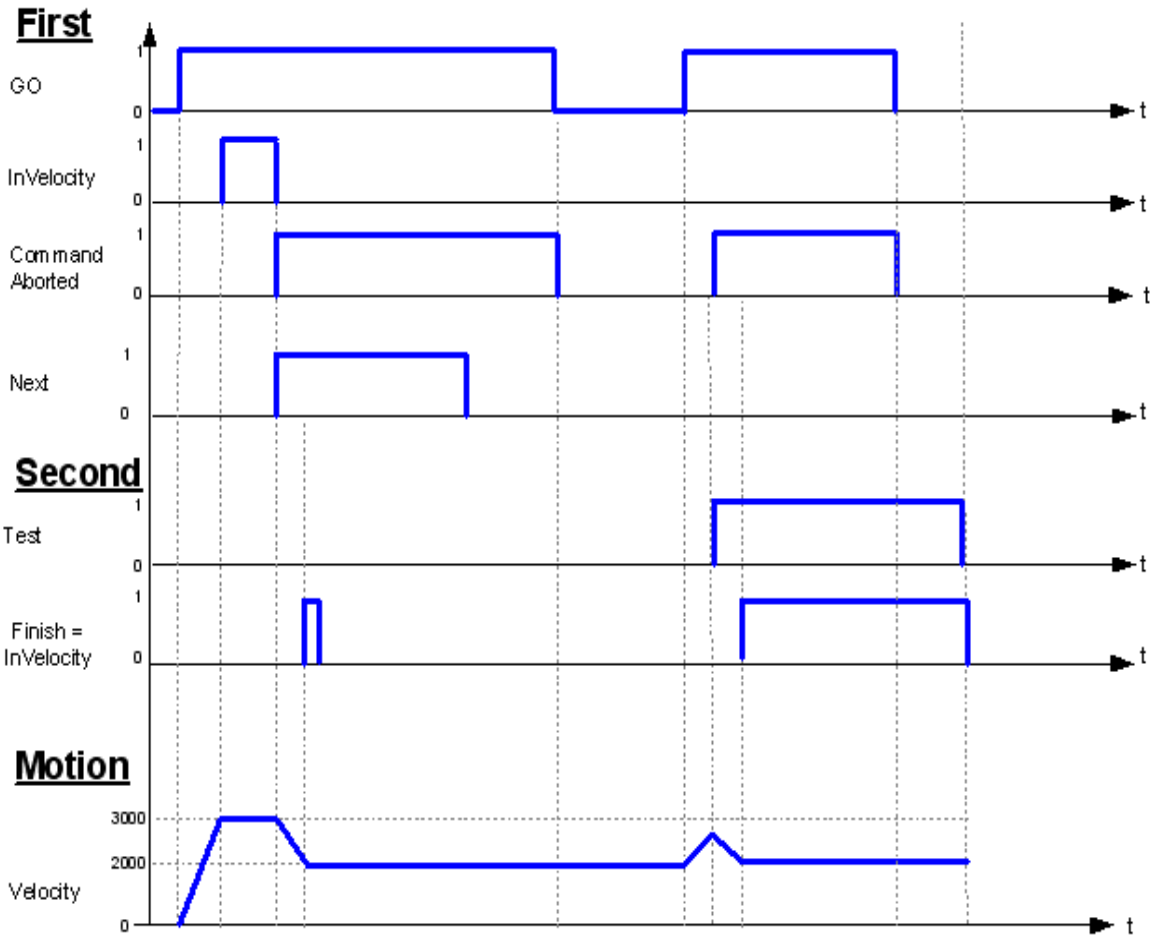


Şekil 2.39. Mc\_MoveVelocity Fonksiyon Bloğu

Motorun durması beklenmeden yeni hız değerini motora iletmemiz için gerekli olan blok şeması ve kullanım şeklini Şekil 2.40'daki örnek ve Şekil 2.41'deki grafikleri inceleyerek anlayabiliriz. Fonksiyon bloğunun “Execute” girişi yükselen kenar algılamalı olduğu için 1 sefer tetik verdiğimizde o anki hız değeri ile çalışmaya başlar, Mc\_Stop fonksiyon bloğu ile motora stop basana kadar motor çalışmaya devam eder. “Execute” girişine her tetik verdiğimizde “Velocity” girişini yenileyerek motorun yeni değerde çalışmasını sağlar.



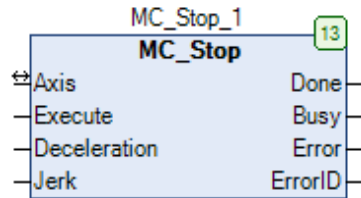
Şekil 2.40. Mc\_MoveVelocity Örnek Program



Şekil 2.41. Mc\_MoveVelocity Örnek Grafik

### 2.11.11. Mc\_Stop fonksiyon bloğu

Kontrol ettiğimiz eksenin herhangi bir zamanda durmasını sağlayan fonksiyon bloğudur. Mc\_stop fonksiyon bloğu Şekil 2.42'deki gibidir. Girişlerinin ve çıkışlarının açıklamaları Çizelge 2.12'de görüldüğü gibidir.



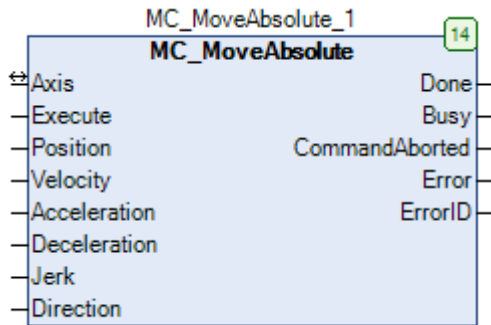
Şekil 2.42. Mc\_Stop Fonksiyon Bloğu

Çizelge 2.12. Mc\_Stop Fonksiyon Bloğu Giriş-Çıkış Çizelgesi

|                   | Data Tipi | Başlangıç Değeri | Tanım                                                                   |
|-------------------|-----------|------------------|-------------------------------------------------------------------------|
| <b>VAR_INPUT</b>  |           |                  |                                                                         |
| Execute           | BOOL      | FALSE            | Giriş sinyalinin yükselen kenarı ile fonksiyon bloğu işleme başlar.     |
| Deceleration      | LREAL     | 0                | Dururken istenen yavaşlama rampası.                                     |
| <b>VAR_OUTPUT</b> |           |                  |                                                                         |
| Done              | BOOL      | FALSE            | Execute girişi aktif edildikten sonra hız 0(sıfır)'a düştüğünde "TRUE". |
| Busy              | BOOL      | FALSE            | Fonksiyon bloğunun icrası bitmedi ise "TRUE".                           |
| Error             | BOOL      | FALSE            | Fonksiyon bloğunda hata var ise "TRUE".                                 |
| ErrorID           | SMC_ERROR | 0                | Fonksiyon bloğunda hata var ise hata kodu.                              |

### 2.11.12. Mc\_MoveAbsolute fonksiyon bloğu

Kontrol ettiğimiz eksenin mutlak şekilde pozisyonlamamızı sağlayan fonksiyon bloğudur. Mc\_MoveAbsolute fonksiyon bloğu Şekil 2.43'deki gibidir. Girişlerinin ve çıkışlarının açıklamaları Çizelge 2.13'de görüldüğü gibidir.



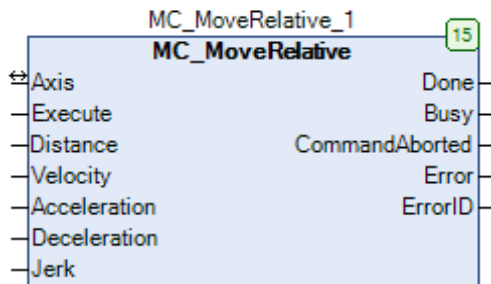
Şekil 2.43. Mc\_MoveAbsolute Fonksiyon Bloğu

Çizelge 2.13. Mc\_MoveAbsolute Fonksiyon Bloğu Giriş-Çıkış Çizelgesi

|                   | Data Tipi    | Başlangıç Değeri | Tanım                                                                                                                                          |
|-------------------|--------------|------------------|------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>VAR_INPUT</b>  |              |                  |                                                                                                                                                |
| Execute           | BOOL         | FALSE            | Giriş sinyalinin yükselen kenarı ile fonksiyon bloğu işleme başlar.                                                                            |
| Position          | LREAL        | 0                | Hedef pozisyon.                                                                                                                                |
| Velocity          | LREAL        | 0                | Hedef pozisyona ulaşmak için max. hız.                                                                                                         |
| Acceleration      | LREAL        | 0                | Hedef pozisyona ulaşmak için hızlanma rampası.                                                                                                 |
| Deceleration      | LREAL        | 0                | Hedef pozisyona ulaşmak için yavaşlama rampası.                                                                                                |
| Direction         | MC_DIRECTION | shortest         | MC_Direction;<br>geçerli değerler:<br>positive : pozitif<br>negative : negatif<br>current : mevcut<br>shortest : en kısa<br>fastest : en hızlı |
| <b>VAR_OUTPUT</b> |              |                  |                                                                                                                                                |
| Done              | BOOL         | FALSE            | Hedef pozisyona ulaşıldığında "TRUE".                                                                                                          |
| Busy              | BOOL         | FALSE            | Hedef pozisyona hareket halindeyken "TRUE".                                                                                                    |

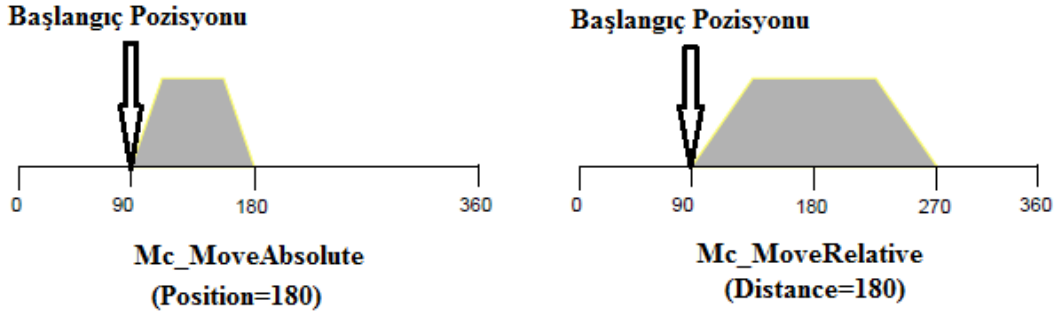
### 2.11.13. Mc\_MoveRelative fonksiyon bloğu

Kontrol ettiğimiz eksenin rölatif şekilde pozisyonlamamızı sağlayan fonksiyon bloğudur. Mc\_MoveRelative fonksiyon bloğu Şekil 2.44'deki gibidir. Girişlerinin ve çıkışlarının açıklamaları Çizelge 2.14'de görüldüğü gibidir.



Şekil 2.44. Mc\_MoveRelative Fonksiyon Bloğu

Mc\_MoveAbsolute ile Mc\_MoveRelative arasındaki fark Şekil 2.45’de görüldüğü gibidir. Mc\_MoveAbsolute mutlak pozisyona gider, yani kendi enkoder değerine göre pozisyona gider. Mc\_MoveRelative ise bulunduğu konumdan itibaren girilen mesafe kadar gider.



Şekil 2.45. Mc\_MoveAbsolute ile Mc\_MoveRelative Farkı

Çizelge 2.14. Mc\_MoveRelative Fonksiyon Bloğu Giriş-Çıkış Çizelgesi

|                   | Data Tipi    | Başlangıç Değeri | Tanım                                                                                                                                          |
|-------------------|--------------|------------------|------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>VAR_INPUT</b>  |              |                  |                                                                                                                                                |
| Execute           | BOOL         | FALSE            | Giriş sinyalinin yükselen kenarı ile fonksiyon bloğu işleme başlar.                                                                            |
| Distance          | LREAL        | 0                | Hedef mesafe.                                                                                                                                  |
| Velocity          | LREAL        | 0                | Hedef mesafeye ulaşmak için max. hız.                                                                                                          |
| Acceleration      | LREAL        | 0                | Hedef mesafeye ulaşmak için hızlanma rampası.                                                                                                  |
| Deceleration      | LREAL        | 0                | Hedef mesafeye ulaşmak için yavaşlama rampası.                                                                                                 |
| Direction         | MC_DIRECTION | shortest         | MC_Direction;<br>geçerli değerler:<br>positive : pozitif<br>negative : negatif<br>current : mevcut<br>shortest : en kısa<br>fastest : en hızlı |
| <b>VAR_OUTPUT</b> |              |                  |                                                                                                                                                |
| Done              | BOOL         | FALSE            | Hedef mesafeye ulaşıldığında "TRUE".                                                                                                           |
| Busy              | BOOL         | FALSE            | Hedef mesafeye hareket halindeyken "TRUE".                                                                                                     |

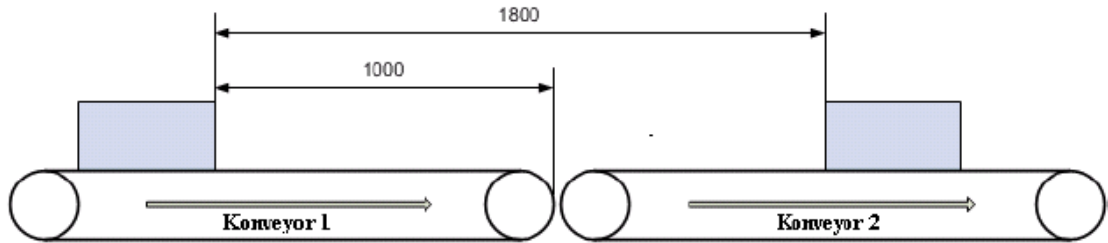
#### 2.11.14. Mc\_MoveSuperImposed fonksiyon bloğu

Bu fonksiyon bloğu hareket halinde olan bir eksene bir üst hareket vermek için kullanılır. Dişli oranı ile bağlı olan bir eksene süper pozisyon yaptırmak mümkündür. kullanılan yerlerden birisi konveyör sistemleridir. Uzun konveyörler normalde her biri sürücü ile sürülen birden fazla bölüme ayrılırlar. Konveyörler bir ürün taşınırken, ürünler arası mesafe düzeltilmeleri gerekir. Bu sebepten biten bir bölümün arkasından başlayan diğer bölümün hızı ilk tarafa göre farklı olabilir ya da hızlanma yavaşlamalar gerçekleşebilir. İki bölümden oluşan bir konveyör sistemini ele alalım: Şekil 2.46’da verilen sistemde iki ürün arasında hesaplanan uzaklığın 1800 mm olduğu ve bu uzaklığın 1500 mm’ye düşürülmekte olduğu görülmektedir. 1 numaralı konveyör mesafeyi düşürmek için hızlanması gerekmektedir. Hızlanma işleminin ürün 2 numaralı konveyöre geldiğinde bitmesi gerekir ve eski hızına dönerek ürün konveyörlerin birinden diğerine geçerken sarsılma engellenir. Bu işlemde 1 numaralı konveyörün hızlanması 500 mm/s ile sınırlandırılmıştır. Bu hız değişimi kullanıcı tarafından belirlenir. Fonksiyon bloğu parametreleri şu şekilde girilir:

$$\text{Mesafe (Distance)} = 1800 - 1500 = 300 \text{ mm (düzeltilen mesafe)} \quad (2.17)$$

$$\text{Length} = 1000 \text{ mm (1 numaralı konveyörde hızlanma yapılan yol)} \quad (2.18)$$

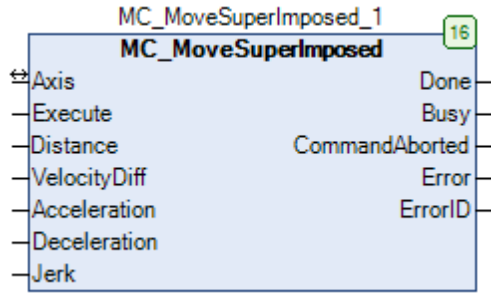
$$\text{VeloDiff} = 500 \text{ mm/s} \quad (2.19)$$



Şekil 2.46. Konveyör Örneği

Seçilen moda göre 1000 mm’lik yolun sonuna kadar maksimum 500 mm/s’lik hız farkı oluşturarak 300 mm’lik pozisyon düzeltmesi yapılır. Hareketin maksimum hızı hesaptan bulunur ve bu hızın üst limit değişimini 500 mm/s olarak ayarlanır.

Bu blok normalde elektronik CAM kullanılmadan önceki zamanlarda elektronik dişli oranı ile bağlı eksenlerde hız değişimi vermek için kullanılmaktadır. Mc\_MoveSuperImposed fonksiyon bloğu Şekil 2.47’deki gibidir. Girişlerinin ve çıkışlarının açıklamaları Çizelge 2.15’de görüldüğü gibidir.



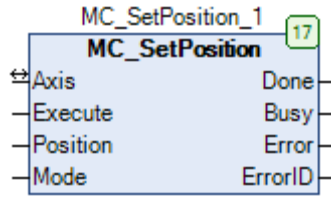
Şekil 2.47. Mc\_MoveSuperImposed Fonksiyon Bloğu

Çizelge 2.15. Mc\_MoveSuperImposed Fonksiyon Bloğu Giriş-Çıkış Çizelgesi

|                   | Data Tipi | Başlangıç Değeri | Tanım                                                               |
|-------------------|-----------|------------------|---------------------------------------------------------------------|
| <b>VAR_INPUT</b>  |           |                  |                                                                     |
| Execute           | BOOL      | FALSE            | Giriş sinyalinin yükselen kenarı ile fonksiyon bloğu işleme başlar. |
| Distance          | LREAL     | 0                | Hedef pozisyona eklenmek istenen mesafe.                            |
| VelocityDiff      | LREAL     | 0                | Hedef hıza eklenmek istenen hız farkı.                              |
| Acceleration      | LREAL     | 0                | Hedef mesafeye ulaşmak için hızlanma rampası.                       |
| Deceleration      | LREAL     | 0                | Hedef mesafeye ulaşmak için yavaşlama rampası.                      |
| <b>VAR_OUTPUT</b> |           |                  |                                                                     |
| Done              | BOOL      | FALSE            | Hedefe ulaşıldığında “TRUE”.                                        |
| Busy              | BOOL      | FALSE            | Hedef hareket halindeyken “TRUE”.                                   |

### 2.11.15. Mc\_SetPosition fonksiyon bloğu

Kontrol edilen eksenin aktüel pozisyonunu istediğimiz değer olarak ayarlamak için kullanılan fonksiyon bloğudur. Mc\_SetPosition fonksiyon bloğu Şekil 2.48’deki gibidir. Girişlerinin ve çıkışlarının açıklamaları Çizelge 2.16’de görüldüğü gibidir.



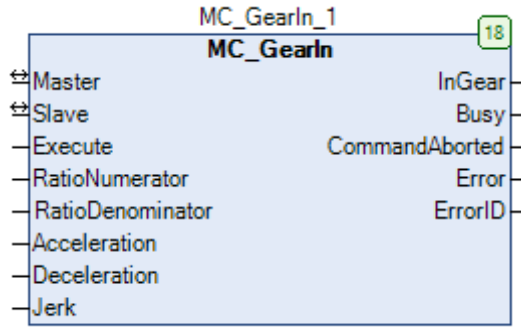
Şekil 2.48. Mc\_MoveSetPosition Fonksiyon Bloğu

Çizelge 2.16. Mc\_MoveSetPosition Fonksiyon Bloğu Giriş-Çıkış Çizelgesi

|                   | Data Tipi | Başlangıç Değeri | Tanım                                                               |
|-------------------|-----------|------------------|---------------------------------------------------------------------|
| <b>VAR_INPUT</b>  |           |                  |                                                                     |
| Execute           | BOOL      | FALSE            | Giriş sinyalinin yükselen kenarı ile fonksiyon bloğu işleme başlar. |
| Position          | LREAL     |                  | İstenen Pozisyon değeri.                                            |
| Mode              | BOOL      | FALSE            | “TRUE” = RELATIVE<br>“FALSE” = ABSOLUTE                             |
| <b>VAR_OUTPUT</b> |           |                  |                                                                     |
| Done              | BOOL      | FALSE            | Pozisyon servoya gömüldü ise “TRUE”                                 |

### 2.11.16. Mc\_GearIn fonksiyon bloğu

Elektronik dişli bloğunun özelliği ana eksen ile yardımcı eksenin sanki arada mekanik dişli bloğu varmış gibi elektronik olarak redüksiyon etmesidir. Hareket halindeki bir eksene başka bir eksen bağlanması bu blok ile mümkün değildir. Kullanılan bloğun bir diğer özelliği de dinamik olarak değiştirilebilen dişli oranıdır. Bloğun aktif etme girişine bağlı büyüklüğün lojik aktif olduğu sürece dişli oranı her PLC çevriminde değişebilir. Dişli oranı ile bağlanan eksenin bağını koparmak için de başka bir fonksiyon bloğu kullanılmaktadır. Bir hata olması durumunda sırası ile önce dişli oranından çıkarılır, dişli oranından çıkınca eksenin durup durmadığına bakılır eğer durmuyorsa durdurulur. Eksen durduktan sonra eksenin bir hata varsa bu hata resetlenir, eğer sürücüde bir hata varsa sürücü de resetlenir. Mc\_GearIn fonksiyon bloğu Şekil 2.49'deki gibidir. Girişlerinin ve çıkışlarının açıklamaları Çizelge 2.17'de görüldüğü gibidir.



Şekil 2.49. Mc\_GearIn Fonksiyon Bloğu

Çizelge 2.17. Mc\_GearIn Fonksiyon Bloğu Giriş-Çıkış Çizelgesi

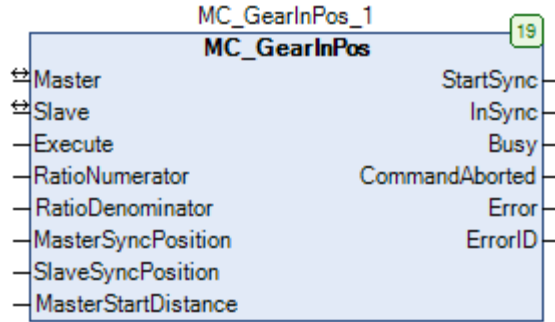
|                   | Data Tipi | Başlangıç Değeri | Tanım                                                               |
|-------------------|-----------|------------------|---------------------------------------------------------------------|
| <b>VAR_IN_OUT</b> |           |                  |                                                                     |
| Master            | AXIS_REF  |                  | Ana eksen.                                                          |
| Slave             | AXIS_REF  |                  | Ana eksene senkronize olacak yardımcı eksen.                        |
| <b>VAR_INPUT</b>  |           |                  |                                                                     |
| Execute           | BOOL      | FALSE            | Giriş sinyalinin yükselen kenarı ile fonksiyon bloğu işleme başlar. |
| RatioNumerator    | INT       | 1                | Dişli nümeratörü.                                                   |
| RatioDenominator  | UINT      | 1                | Dişli denümaratörü.                                                 |
| Acceleration      | LREAL     | 0                | Yardımcı eksen senkronize olurken kullanacağı hızlanma rampası.     |
| Deceleration      | LREAL     | 0                | Yardımcı eksen senkronize olurken kullanacağı yavaşlama rampası.    |
| <b>VAR_OUTPUT</b> |           |                  |                                                                     |
| InGear            | BOOL      | FALSE            | Senkronizasyon gerçekleşti ise "TRUE".                              |

Elektronik dişli oranı aşağıdaki gibi hesaplanır;

$$\text{Dişli Oranı} = \text{Dişli nümeratörü} / \text{Dişli Denümaratörü} \quad (2.20)$$

### 2.11.17. Mc\_GearInPos fonksiyon bloğu

Elektronik dişli bloğunun gelişmiş versiyonudur. Bu blok ile yardımcı eksenin hem hangi dişli oranı ile ana eksene bağlanacağını belirleyebiliyoruz hem de yardımcı eksenin hangi pozisyonda senkronizasyona başlayıp hangi pozisyonda tam senkronizasyonda olacağını ve bu süre zarfında ana eksenin ne kadar mesafe kat edeceğine kadar parametrik olarak ayarlayabiliyoruz. Mc\_GearInPos fonksiyon bloğu Şekil 2.50'deki gibidir. Girişlerinin ve çıkışlarının açıklamaları Çizelge 2.18'de görüldüğü gibidir.

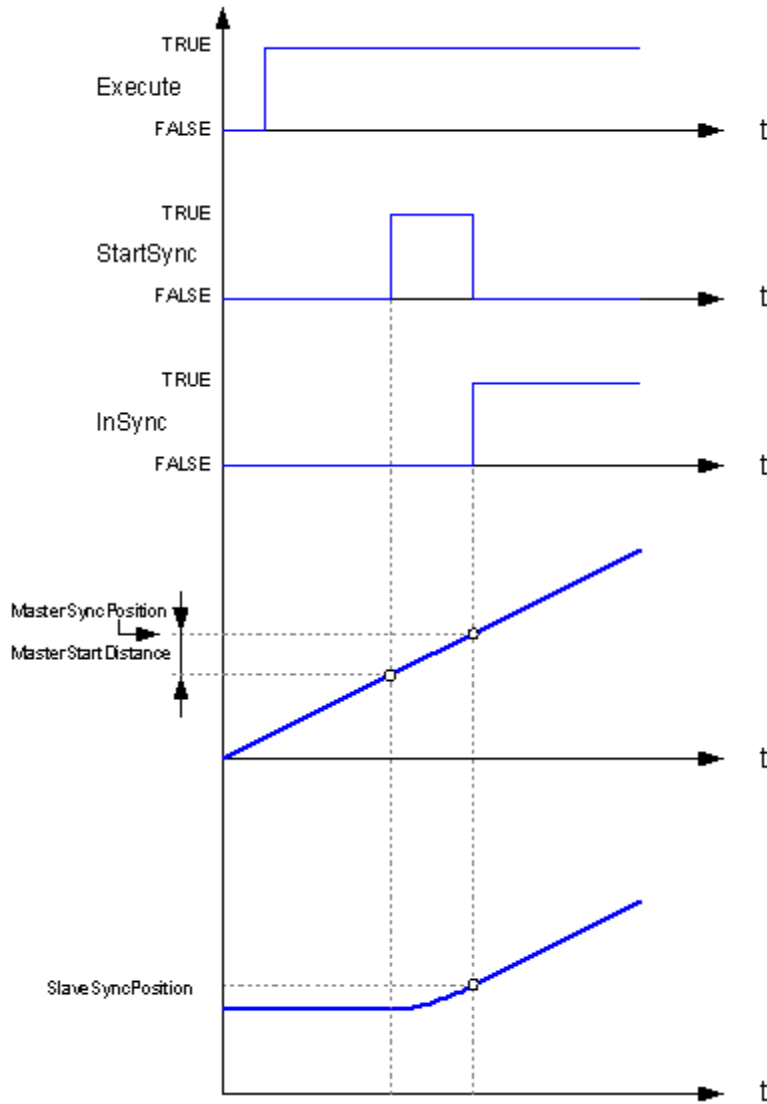


Şekil 2.50. Mc\_GearInPos Fonksiyon Bloğu

Çizelge 2.18. Mc\_GearInPos Fonksiyon Bloğu Giriş-Çıkış Çizelgesi

|                     | Data Tipi | Başlangıç Değeri | Tanım                                                               |
|---------------------|-----------|------------------|---------------------------------------------------------------------|
| <b>VAR_INPUT</b>    |           |                  |                                                                     |
| Execute             | BOOL      | FALSE            | Giriş sinyalinin yükselen kenarı ile fonksiyon bloğu işleme başlar. |
| RatioNumerator      | INT       | 1                | Dişli nümeratörü.                                                   |
| RatioDenominator    | INT       | 1                | Dişli denümaratörü.                                                 |
| MasterSyncPosition  | REAL      | 0                | Senkronizasyon süresince ana eksenin alması istenilen mesafe.       |
| SlaveSyncPosition   | REAL      | 0                | Senkronizasyon süresince yardımcı eksenin alması istenilen mesade.  |
| MasterStartDistance | REAL      | 0                | Senkronizasyona başlanması istenen ana eksen pozisyonu              |
| <b>VAR_OUTPUT</b>   |           |                  |                                                                     |
| StartSync           | BOOL      |                  | Senkronizasyona başlandığında "TRUE".                               |
| InSync              | BOOL      |                  | Senkronizasyon gerçekleştiğinde "TRUE".                             |

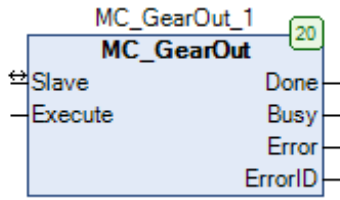
Giriş parametrelerinin ne anlamlara geldiği Şekil 2.51'deki grafikler ile daha net anlaşılabilir.



Şekil 2.51. Mc\_GearInPos Fonksiyon Bloğu Zamanlama Diagramı

#### 2.11.18. Mc\_GearOut fonksiyon bloğu

Mc\_GearIn ve Mc\_GearInPos fonksiyon blokları ile elektronik dişliye bağlanmış yardımcı eksen senkronizasyondan çıkarmak için kullanılan bloktur. Mc\_GearOut fonksiyon bloğu Şekil 2.52'deki gibidir. Girişlerinin ve çıkışlarının açıklamaları Çizelge 2.19'da görüldüğü gibidir.



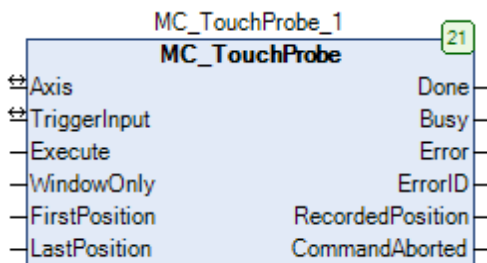
Şekil 2.52. Mc\_GearOut Fonksiyon Bloğu

Çizelge 2.19. Mc\_GearOut Fonksiyon Bloğu Giriş-Çıkış Çizelgesi

|                   | Data Tipi | Başlangıç Değeri | Tanım                                                               |
|-------------------|-----------|------------------|---------------------------------------------------------------------|
| <b>VAR_IN_OUT</b> |           |                  |                                                                     |
| Slave             | AXIS_REF  |                  | Ana eksene senkronize olan yardımcı eksen.                          |
| <b>VAR_INPUT</b>  |           |                  |                                                                     |
| Execute           | BOOL      | FALSE            | Giriş sinyalinin yükselen kenarı ile fonksiyon bloğu işleme başlar. |
| <b>VAR_OUTPUT</b> |           |                  |                                                                     |
| Done              | BOOL      | FALSE            | Senkronizasyondan çıkmış ise "TRUE".                                |

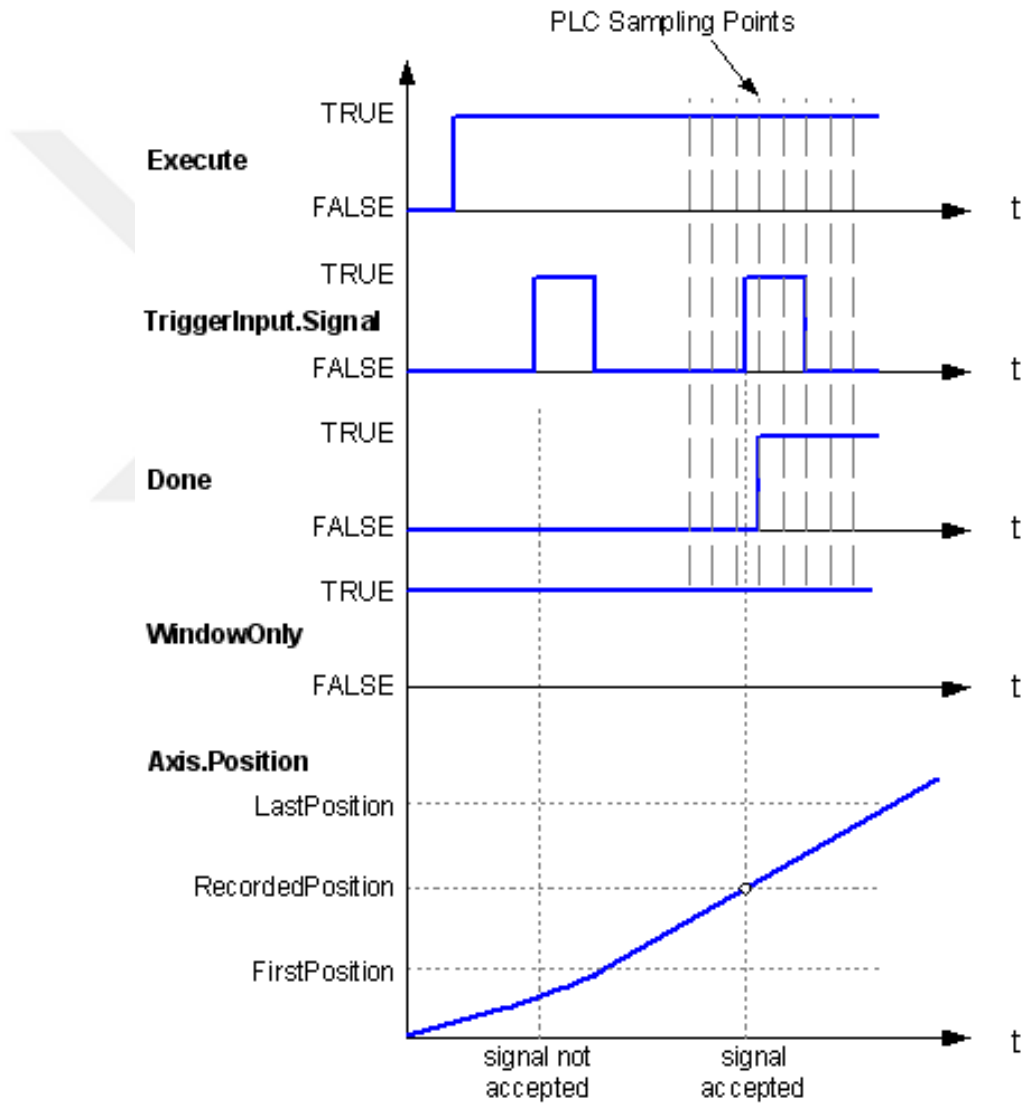
#### 2.11.19. Mc\_TouchProbe fonksiyon bloğu

Sürücüye veya PLC'ye tanımlanmış özel girişlere, benek/delik yakalayan bir sensörden "TRUE" geldiğinde PLC çevriminden bağımsız bir şekilde yükselen kenar algılama ile eksenin o anki pozisyonu kaydeden fonksiyon bloğudur. Mc\_TouchProbe fonksiyon bloğu Şekil 2.53'deki gibidir. Girişlerinin ve çıkışlarının açıklamaları Çizelge 2.20'de görüldüğü gibidir.



Şekil 2.53. Mc\_TouchProbe Fonksiyon Bloğu

Bu bloğun kullanımında bir pencere tanımlanabilir ve bu sayede pozisyonu yaklaşık olarak bilinen bir yere işaret edilebilir. Örnek olarak 300 mm ile 500 mm arasında gelmesi gereken bir işaret için “WindowOnly” girişine “TRUE” verilir, ilk pozisyona (FirstPosition) 300, son pozisyona (LastPosition) 500 girilir ve “Execute” girişine “TRUE” verilir. Daha iyi anlaşılması için Şekil 2.54’deki grafikler incelenabilir. “TriggerInput”a ilgili eksenin bağlı olan enkoder numarası girilir ve daha önceden konfigüre edilen servo sürücünün veya PLC’nin girişine gelen dijital işarete göre eksen pozisyonu kaydedilir.



Şekil 2.54. Mc\_TouchProbe Fonksiyon Bloğu ile WindowOnly Örneği

Çizelge 2.20. Mc\_TouchProbe Fonksiyon Bloğu Giriş-Çıkış Çizelgesi

|                   | Data Tipi   | Başlangıç Değeri | Tanım                                                               |
|-------------------|-------------|------------------|---------------------------------------------------------------------|
| <b>VAR_IN_OUT</b> |             |                  |                                                                     |
| TriggerInput      | TRIGGER_REF |                  | TriggerInput register                                               |
| <b>VAR_INPUT</b>  |             |                  |                                                                     |
| Execute           | BOOL        | FALSE            | Giriş sinyalinin yükselen kenarı ile fonksiyon bloğu işleme başlar. |
| WindowOnly        | BOOL        | FALSE            | Pencere aralığında tetik beklemek için "TRUE"                       |
| FirstPosition     | LREAL       | 0                | Pencere aralığında tetik için ilk pozisyon.                         |
| LastPosition      | LREAL       | 0                | Pencere aralığında tetik için son pozisyon.                         |
| <b>VAR_OUTPUT</b> |             |                  |                                                                     |
| Done              | BOOL        | FALSE            | Tetik alınıp, pozisyon kaydı yapılıncaya "TRUE".                    |
| RecordedPosition  | LREAL       | 0                | Tetik ile kayıt edilen pozisyon.                                    |

"Trigger\_Ref" register'ı içindeki ayarlanması gereken önemli parametreler Çizelge 2.21'de verilmiştir.

Çizelge 2.21. TriggerRef Register Çizelgesi

| Component      | Datentyp | Initialization | Description                                                                                                                                                       |
|----------------|----------|----------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| iTriggerNumber | INT      | -1             | Tetik sensörünün bağlandığı özel giriş numarası (Sadece "bFastLatching"="TRUE" iken kullanılır.).                                                                 |
| bFastLatching  | BOOL     | TRUE           | TRUE: Pozisyon yakalama sürücü üstünden yapılacak ise (önerilir);<br>FALSE: Pozisyon yakalama tetiği olarak motion task'da "bInput" kullanılacak ise (önerilmez). |
| bInput         | BOOL     |                | "bFastLatching"="FALSE" ise tetik sinyali girişi.                                                                                                                 |

### 2.11.20. Tüm fonksiyon bloklarda aynı anlama gelen giriş-çıkışlar

Tüm fonksiyon blokların ortak olan girişlerin ve çıkışların açıklamaları Çizelge 2.22’de görüldüğü gibidir.

Çizelge 2.22. Fonksiyon Blokları Genel Giriş-Çıkış Çizelgesi

|                   | Data Tipi | Başlangıç Değeri | Tanım                                                               |
|-------------------|-----------|------------------|---------------------------------------------------------------------|
| <b>VAR_IN_OUT</b> |           |                  |                                                                     |
| Axis              | AXIS_REF  |                  | Tanımlanan eksen adı                                                |
| <b>VAR_OUTPUT</b> |           |                  |                                                                     |
| Busy              | BOOL      | FALSE            | Fonksiyon bloğunun icrası bitmedi ise “TRUE”.                       |
| CommandAborted    | BOOL      | FALSE            | İşlem başka bir PLCopen bloğu komutu ile yarıda kesilir ise “TRUE”. |
| Error             | BOOL      | FALSE            | Fonksiyon bloğunda hata var ise “TRUE”.                             |
| ErrorID           | SMC_ERROR | 0                | Fonksiyon bloğunda hata var ise hata kodu.                          |

### 2.11.21. PLCopen motion blokları durum diyagramı

EK-2’deki şemada PLCopen motor kontrol bloklarının birbirleri ile olan ilişkilerini görebiliriz. PLCopen, motor kontrol bloklarını Senkronize hareket, Devamlı hareket, Ayrık hareket gibi bölümlere ayırmış ve buna göre bu grupların kontrol önceliklerini bu şema ile belirlemiştir. Örneğin devamlı hareket komutu olan Mc\_MoveVelocity bloğu işletilirken senkronize hareket komutu olan Mc\_GearIn bloğuna geçiş yapılabilirken, Mc\_GearIn fonksiyon bloğu işletilirken Mc\_MoveVelocity fonksiyon bloğu işletilmeye çalışılırsa sistem hata durumuna düşecektir ve motor “ErrorStop” yani hatadan durma gerçekleştirecektir.

### 3. MATERYAL VE METOT

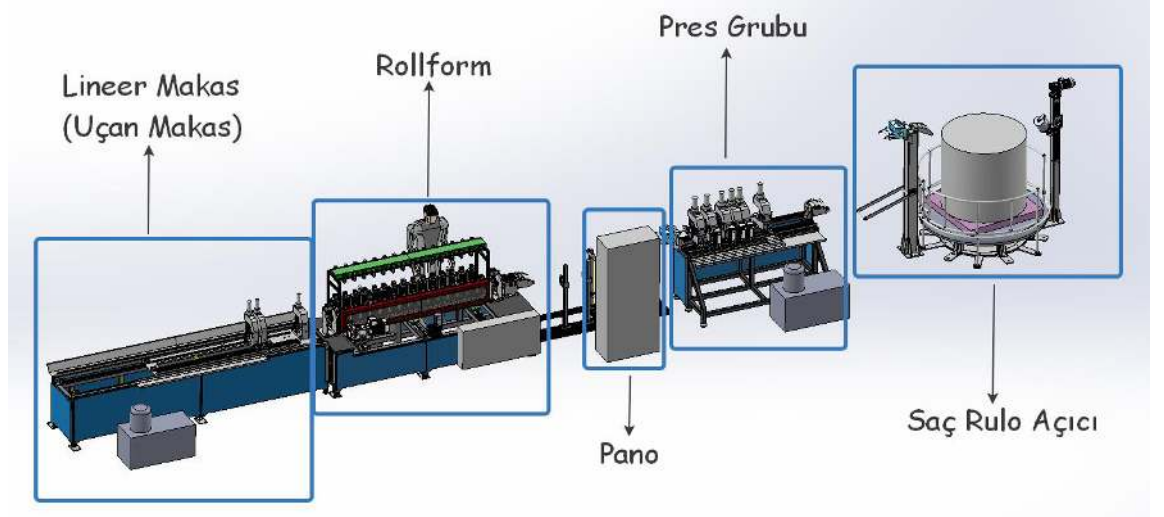
#### 3.1. Uygulamada Kullanılan Donanımlar ve Mekanik Tasarım

Otomasyon programı yazılan makine hattında; pres grubu, rollform grubu ve lineer makas grubu bulunmaktadır. Uygulamada otomasyon ve şalt pano ürünleri olarak Schneider Electric markasının ürünleri kullanılmıştır. Otomasyon ürünü olarak; PLC, PLC için ek giriş-çıkış-analog modülleri, hmi(insan makine arabirimi-kontrol paneli), servo motor, servo motor sürücü, encoder, ve karşılıklı sensörler kullanılmıştır.

Ürün Listesi Aşağıdaki gibidir;

|                                         |                                     |
|-----------------------------------------|-------------------------------------|
| 1-) PLC                                 | : Schneider Electric LMC058LF42     |
| 2-) PLC ek modül                        | : Schneider Electric TM5C24D18T     |
| 3-) PLC ek modül                        | : Schneider Electric TM5C12D8T      |
| 4-) Hmi                                 | : Schneider Electric HMIGXO5502     |
| 5-) Servo Motor(Pres 2-3-4-5)           | : Schneider Electric BMH1003P16A1A  |
| 6-) Servo Motor(Pres çıkış saç sürücü)  | : Schneider Electric BMH1002P16A1A  |
| 7-) Servo Motor(Rollform)               | : Schneider Electric BMH1002P16A1A  |
| 8-) Servo Motor(Lineer Makas)           | : Schneider Electric BMH1402P16A1A  |
| 9-) Servo Motor Sürücü(Tüm servolar)    | : Schneider Electric LXM32AD30N4    |
| 10-) Encoder (5000 puls)                | : Schneider Electric XCC1510SPA50Y  |
| 11-) Lazer sensör(Delik/Kenar algılama) | : Schneider Electric XUK-2LAPSM12RT |

Hattın girişinde loop sensörü ile gerginliği ölçüp saç rulosunu açan açıcı bulunmaktadır ve bu sistem dışarıdan hazır alınmıştır. Daha sonrasında 6'lı piston grubu bulunmaktadır. Bunlardan 4'ü servo motora bağlı pozisyon değiştirebilmektedirler. Baştaki ve sondaki pistonlar sabit, aradaki 4 piston üretilecek ürüne göre pozisyonlanmaktadır. Pozisyon değerleri operatör panelindeki reçeten ürün olarak seçilip otomatik atılmaktadır. Her ürüne özel reçete değerleri daha önceden operatör tarafından kaydedilip saklanabilir. Aynı zamanda her ürün için 6 pistonun da çalışması gerekmediğinden reçetede o ürün için hangi pistonların çalışacağı da belirleniyor. Piston grubunun hemen sonunda saç ürün boyu kadar sürüp tekrar piston reçetesini çalıştırmak için bekleyen servo sistemi bulunmaktadır. Ürün boyunu yani servonun sacı her defasında ne kadar boy süreceği de reçete içinde belirlenmektedir. Şekil 3.1'de programı yazılan sistemin modellenmesi görülmektedir.



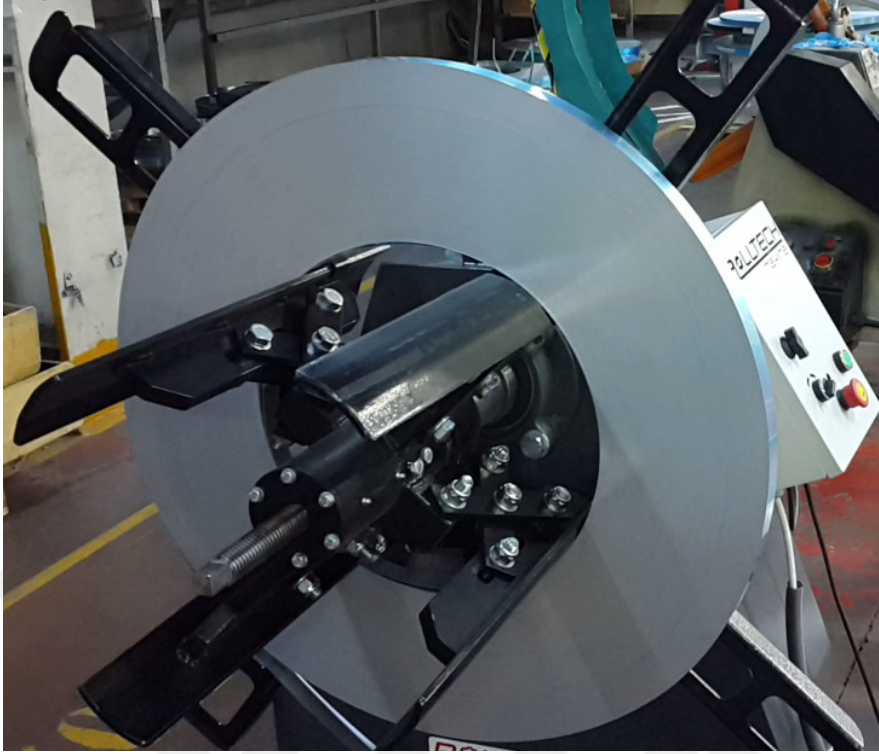
Şekil 3.1. Hattın Modellemesi

Piston gurubundan sonra 1/20 redüksiyon oranı ile servo ile dönen rollform grubu bulunmaktadır. Rollform, basit şekilde tanımlamak gerekirse düz şekilde gelen saca form veren demir ve çelik baskı toplarından oluşur. Döndüren servo hariç tamamen mekanik bir tasarımdır ve otomasyon olarak bir yükü yoktur.

Şekil 3.2’de ve Şekil 3.3’de sistemde bulunan rulo açıcı grup mekanizmalarına ait fotoğraflar görülmektedir.



Şekil 3.2. Rulo Açıcı Önündeki Loop Mekanizması



Şekil 3.3. Rulo Açıcı

Rollform grubundan sonra lineer makas kısmı gelmektedir. Bu kısım delinmiş, şekillendirilmiş ve son halini almış sonsuz ürünün ebatlanmasını sağlar. Ve bu işlemi arkadan akan ürün durmadan yapar. Şekil 3.4, Şekil 3.5 ve Şekil 3.6’da makineye ait fotoğraflar görünmektedir.



Şekil 3.4. Pres Grubu



Şekil 3.5. Piston Grubu Sonundaki Sağ Süren Servo Mekanizması



Şekil 3.6. Lineer Makas

### 3.2. PLC ve HMI Programının Yazıldığı SoMachine Platformu

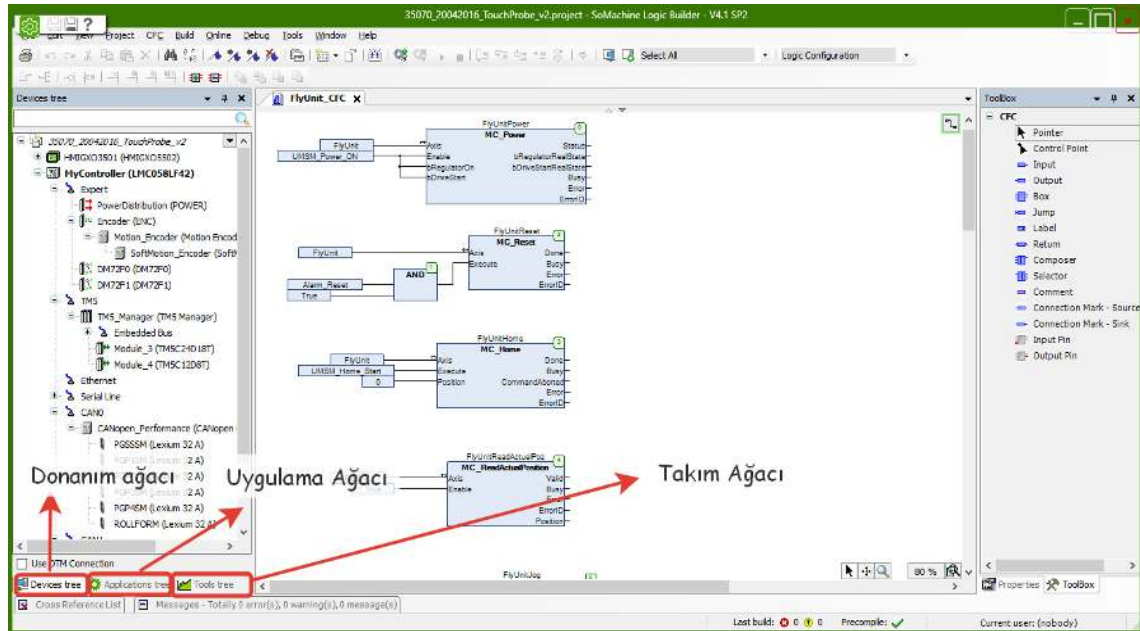
SoMachine, tüm makineyi; lojik, hareket kontrol, HMI ve ilgili tüm otomasyon işlevlerini de içerecek şekilde, tek bir yazılım ortamında geliştirmek, yapılandırmak ve devreye almak için Schneider Elecetric'in sunduğu bir çözümdür. SoMachine'de yer alan önceden tanımlanmış birçok şablon, uygulama ve fonksiyon kütüphaneleri sayesinde mühendislik süresinden tasarruf edilir.

Hareket kontrol cihazları için yapılandırma ve devreye alma araçları, IEC 61131-3 dilleri, entegre fieldbus konfigüratörleri, uzman diyagnostik ve hata ayıklama, bakım ve görselleştirme için web görselleştirme içerip, sayısız uygulamaya özel onaylı, test edilmiş, belgelenmiş ve uzman uygulama kütüphaneleri ile entegrasyon sağlamaktadır.

Somachine platformu CoDeSys altyapısı üzerine kurulmuştur. En büyük avantajlarından birisi PLC kapasitesi yetmediğinde yeniden programlama yapmadan PLC tipini anında çevrilebilinmesi ve programlayıcıya herhangi bir külfet getirmemesidir. Diğer avantajı ise hmi programlamayı da bütünleşik olarak sunması ve programda kullanılan değişken taglarını kolayca PLC ile HMI arasında herhangi bir adresleme yapmaya gerek duymadan birbirine entegre etmesidir.

#### 3.2.1. Somachine program yapısı

Somachine arayüzü Şekil 3.7'deki gibidir. Arayüzün sol tarafında porgramlama için gerekli ağaç yapıları mevcuttur. Bu ağaçlar; Donanım Ağacı (Device Tree), Uygulama Ağacı (Applications Tree) ve Takım Ağacı (Tools Tree) olmak üzere 3 tanedir.



Şekil 3.7. Somachine Arayüzü

Donanım ağacında; PLC'nin digital giriş ve çıkışları, analog giriş ve çıkışları, Ethernet ayarları, Seri haberleşme portlarının ayarları, CanOpen ve CanMotion haberleşme ayarları yapılır.

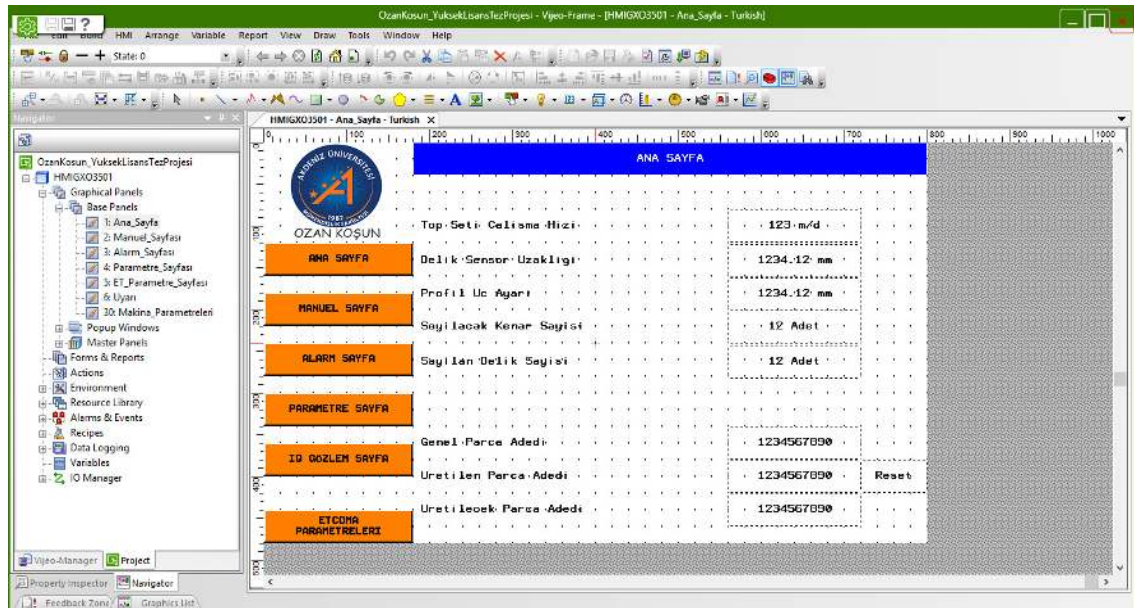
Uygulama ağacında; program parçaları yazılır, task yapıları belirlenir, programlarda kullanılacak değişken tanımlamaları yapılır ve global, persistent değişken tanımlamaları yapılır.

Takım ağacında, programda kullanılan ve kullanılmak istenen kütüphane ayarları, çevrimiçi PLC grafik takibi ve operatör paneli ile değişken haberleşmesini sağlayan "Symbol Configuration" ayarları yapılır.

### 3.2.2. Operatör paneli programı: Vjeco Designer

Vjeco Designer programı Somachine içinden adapte şekilde açılır ve farklı proje dosyası oluşturmaz. Yani PLC programı da HMI programı da tek proje dosyasındadır. Bunun en güzel tarafı da PLC programında oluşturulan ve kullanılan değişkenleri ekran tarafına aktarmanın çok basit olması.

Normalde seri iletişim protokolü olan modbus veya Ethernet protokolü olan modbus tcp/ip gibi bir haberleşme ile PLC ve HMI haberleştirilecek olsa aktarmak istediğimiz her değişkene adresleme yapmalıyız. Fakat Schneider'in geliştirdiği, kendi cihazları arası haberleşmeyi sağlayan "Somachine Network" protokolü sayesinde bu işlem çok basittir.

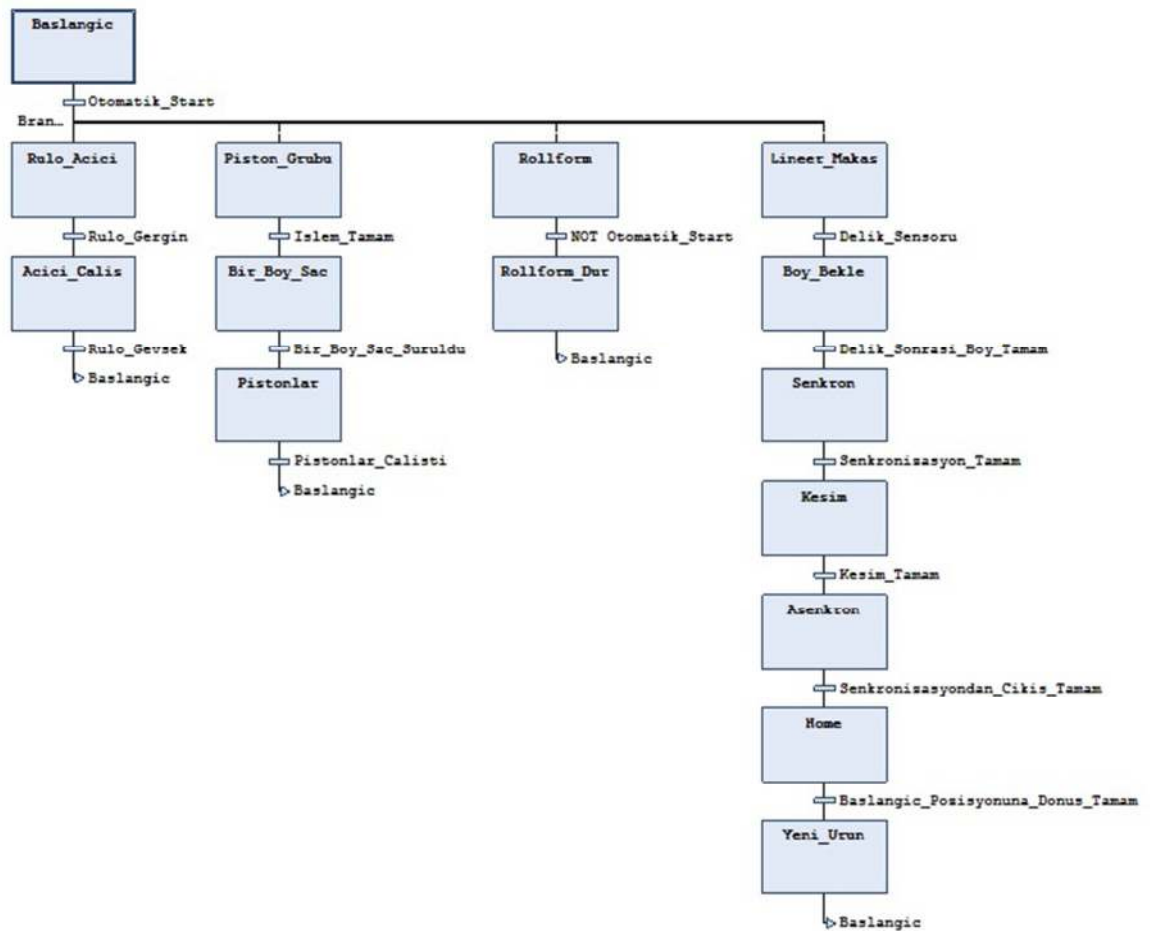


Şekil 3.8. Vjeco Designer Arayüzü

### 3.3. PLC Programındaki İşleyiş ve Algoritma

PLC ve HMI programı projesinin tamamı **Ek-1**'dedir. Tezin amacı; çıkan ürün boyundaki hatanın  $\pm 0,5$  mm olmasıydı. Fakat geliştirilen algoritma ile önemli gelişme kaydedilmiş ve hata  $\pm 0,2$  mm seviyelerine düşmüştür. Aynı zamanda fabrikada bulunan daha önce alınmış eşdeğer makine saatte 300 ürün çıkartırken bizim makinamızın aynı üründen 600 adet ürün çıkarma kapasitesinde olduğu görülmüştür.

Yazılan programın manuel kontrol kısmında tüm motorların, pistonların, giriş ve çıkışların ve tüm donanımın hem test hem gerekli durumlarda kontrol edilebilmesi mümkündür. Otomatik kontrol kısmında ise operatöre nerdeyse hiçbir sorumluluk bırakılmamış ve tüm kontrolün PLC ile yapıldığı seri üretim sağlanmıştır. Herhangi bir acil durumda, acil stop butonuna basıldığında, sensörlerle kontrol edilen saç gerilip koptuğunda veya herhangi bir sıkışma durumunda mümkün olduğunca farklı noktalardan sistem denetlenerek anında duruş yapacak şekilde ve iş güvenliği kurallarına uyularak program yazılmıştır. Otomatik mod için algoritma şeması Şekil 3.9'da verilmiştir.

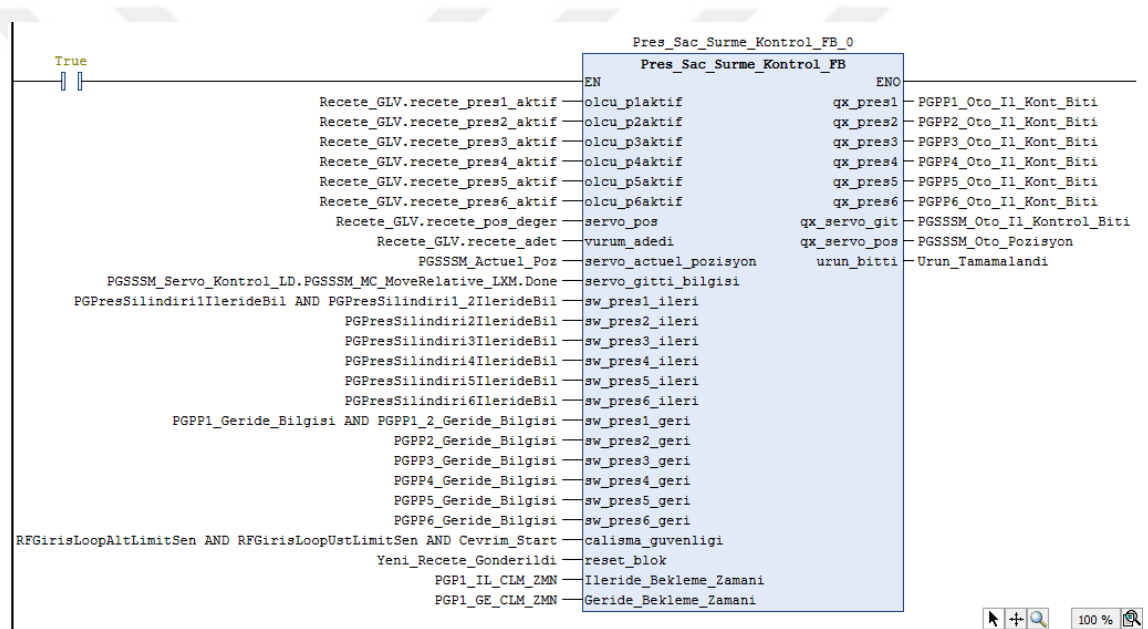


Şekil 3.9. Otomatik Mod Algoritma Şeması

### 3.3.1. Pres grubu

Otomatik çalışmaya başlamadan önce, ekrandaki reçete menüsünden operatöre ürün seçimi yaptırılarak pres grubunun üretilecek ürüne göre pozisyonlanması sağlanmıştır. Bu pozisyonlama işlemi çıkacak ürünün delik yerlerine ve ürün boyuna göre operatör tarafından reçeteye daha önceden kayıt edilmektedir. Ürün değişmediği sürece pozisyonlama işlemi bir sefere mahsus manuel olarak operatöre yaptırılır.

Pres grubu pozisyonlanması yaptırdıktan sonra otomatik çalışmaya hazırdır. Bu işlemten sonra otomatik modda ürünü bir ürün boyu kadar çekip presleri çalıştırıp sürekli bu döngüyü sağlayacak şekilde özel bir fonksiyon bloğu geliştirilmiştir. Fonksiyon bloğu ST dilinde geliştirilmiş ve EK-1’de verilmiştir. Fonksiyon bloğunun programdaki kullanımı Şekil 3.10’da verilmiştir.



Şekil 3.10. Pres Grubu İçin Geliştirilen Fonksiyon Bloğu

Bloğun çalışma mantığı adım adım aşağıdaki gibidir:

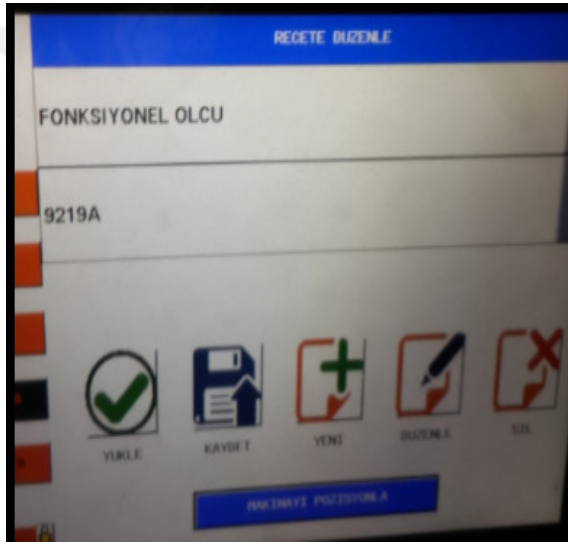
- 1- Ekrandan üretilecek ürün reçetesi seçilir.

Bu reçete içinde 20 adım vardır. Operatör isterse 1 isterse 20 adımlı reçete oluşturabilir. Reçete içinde; her adım için kaç mm ürünün servo ile çekileceği, servo hızı, o adım için hangi pistonların çalışacağı mevcuttur.

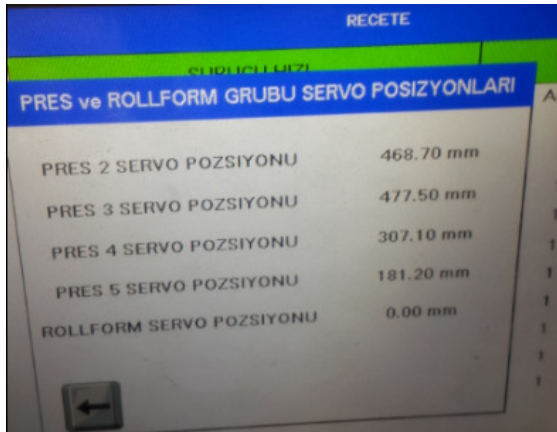
- 2- Sistem mandal buton ile otomatik moda geçirildiğinde blok loop sensörü sayesinde kontrol ettiği gerginliğe göre ve acil stop koşuluna da bakarak direk aktif duruma gelir.

- 3- İlk adım işletilmeye başlar ve bir ürün boyu ürün servo ile çekilir.
- 4- Servo'nun çektim bilgisinin ardından, o adım için çalışması gereken pistonlar(1,2,3,4,5,6) inip, geri çıkar.
- 5- Pistonların geri çıktığı bilgisinin ardından, blok bir sonraki adımın olup olmadığına bakar, varsa bir sonraki adımı işletir.
- 6- Yoksa 1. Adıma dönüp bu işlemi otomatik moddan çıkana kadar devam eder.

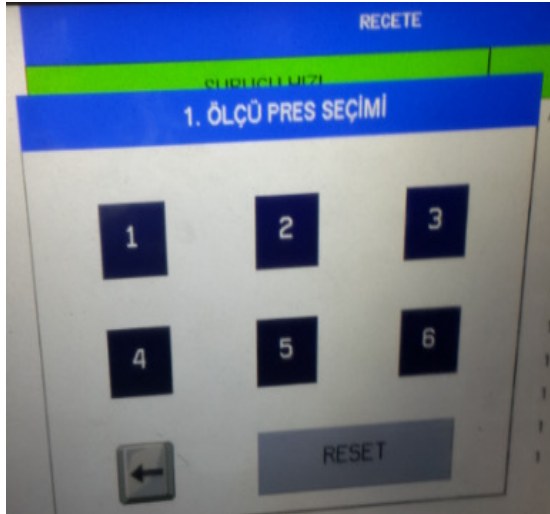
Piston grubu reçetesi ile ilgili panel sayfaları Şekil 3.11, Şekil 3.12, Şekil 3.13 ve Şekil 3.14'te verilmiştir.



Şekil 3.11. Ekran-Reçete Seçim Sayfası



Şekil 3.12. Ekran-Pres Servoları Pozisyonlama Reçetesi



Şekil 3.13. Ekran-Pres Seçim Reçetesi

| SURUCU HIZI |         |          | 1800 mm/s |      |          |
|-------------|---------|----------|-----------|------|----------|
| OLCULER     | ADET    | CIKISLAR | OLCULER   | ADET | CIKISLAR |
| 1           | 1096.60 | 0        | 11        | 0.00 | 0        |
| 2           | 0.00    | 0        | 12        | 0.00 | 0        |
| 3           | 0.00    | 0        | 13        | 0.00 | 0        |
| 4           | 0.00    | 0        | 14        | 0.00 | 0        |
| 5           | 0.00    | 0        | 15        | 0.00 | 0        |
| 6           | 0.00    | 0        | 16        | 0.00 | 0        |
| 7           | 0.00    | 0        | 17        | 0.00 | 0        |
| 8           | 0.00    | 0        | 18        | 0.00 | 0        |
| 9           | 0.00    | 0        | 19        | 0.00 | 0        |
| 10          | 0.00    | 0        | 20        | 0.00 | 0        |

Şekil 3.14. Ekran-Pres Grubu Adım Reçetesi

### 3.3.2. Rollform grubu

Normalde rollform setini döndürmek için asenkron motor kullanılır. Fakat bu hatta üretilecek ürün ince alüminyum olduğu için asenkron motorun devrinin stabil olmayacağı düşünülerek ve duruş kalkışlardaki performansı da göz önüne alınarak döndürmek için Servo motor kullanılmıştır. Servo motorun otomatik modda çalıştırılması hariç yazılımsal olarak bu kısımda önemli pek bir şey yoktur.

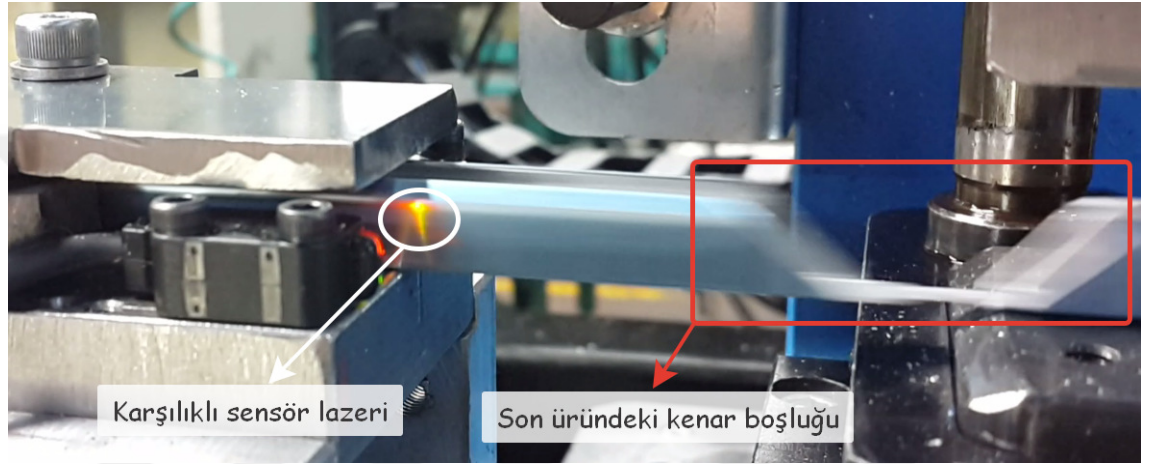
### 3.3.3. Lineer makas grubu

Lineer makaslar iki mantıkla çalışır. İlki direk girilen ürün boyuna göre ebatlayan, ikincisi ise ürün üzerinde daha önceden açılmış olan deliği algılayıp bunu referans alarak ebatlayandır. Direk boy ebatlamada boy doğruluğu offset ile

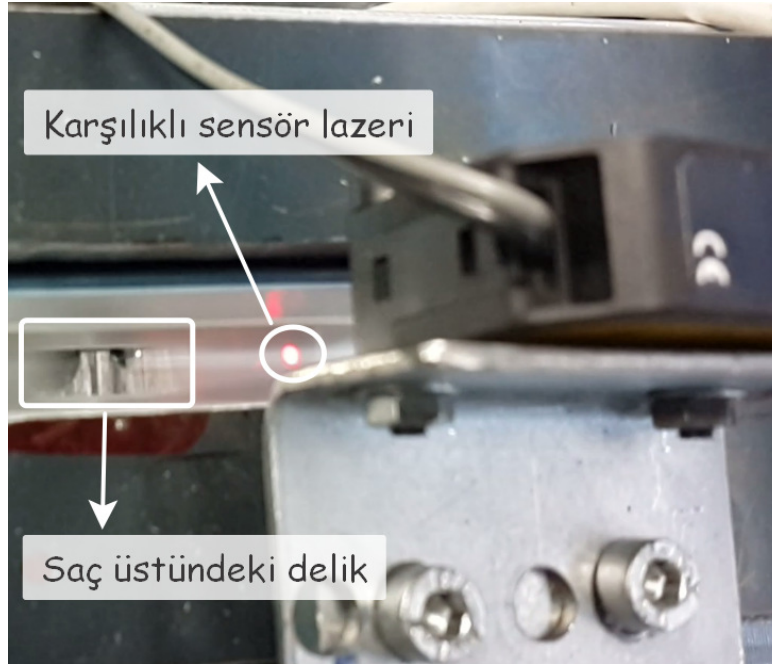
giderilebilirken, delikli ürün ebatlama için offset bir işe yaramamaktadır. Nedeni ise delikli çıkan ürünün o deliklerden başka bir ürüne veya başka bir yere montelenmesidir.

Bu yüzden ebatlama delik referans alınarak yapılmalıdır. Benim programını yazdığım lineer makas uygulaması ikinci versiyona uygun ve ürün boyu doğruluğunu artırmak için iki kontrol yapılmıştır.

İlk olarak Şekil 3.15’de görüldüğü gibi ürün şekil aldıktan sonra oluşan kenardaki boşluklar yan taraftan saydırılmaktadır. Değişken olarak ekrandan girilebilen kenar sayısı programda saydırıldıktan sonra Şekil 3.16’de görüldüğü gibi üstten ilk delik sensör tarafından algılanır.



Şekil 3.15. Kenar Sensörü



Şekil 3.16. Delik Sensörü

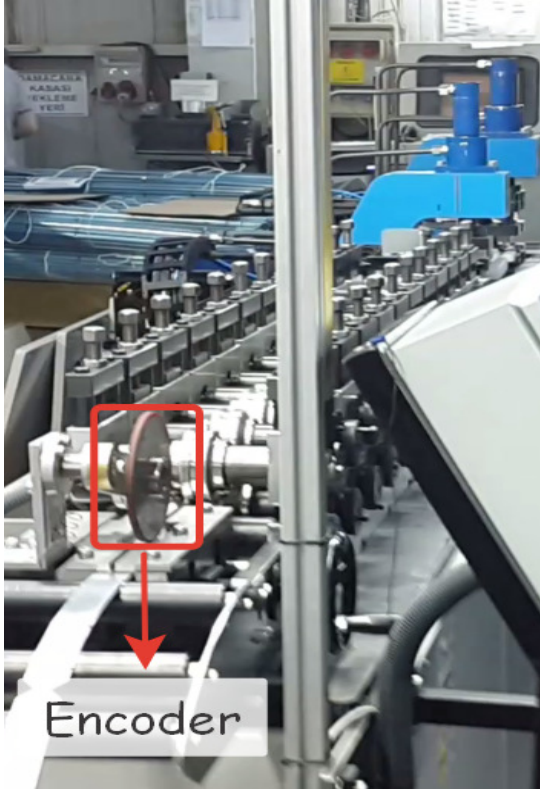
Delik özel Mc\_TouchProbe fonksiyon bloğu ile algılanıp, PLC işlem zamanına (cycle time) takılmadan enkoder pozisyonu tampon (buffer) değişkene yazılır. Burada bu özel bloğu amaca uygun kullanarak boy kaçırma sorunlarından en büyüğünü ekarte etmiş oldum. Yani PLC işlem mantığına göre delik algılandı sinyali PLC'nin 1 cycle time'ında işlendikten sonra enkoder pozisyonunu bir sonraki cycle time da tampon değişkene atacakken 2 cycle time kazanılmış oldu.

Cycle time PLC'nin içine yazdığımız programı baştan sonra 1 sefer işletebildiği süredir. Yazılan programın 1 cycle time'ının ortalama 4ms ve rollformun saç sürme hızının yani akan ürünün hızının 20 m/dk olduğunu bildiğimize göre kabaca 2 cycle time süre geçtiğinde ne kadar boy kaçırabileceğimizi aşağıdaki gibi hesaplayabiliriz:

$$\text{Kaçan boy} = (20000/60000)*4 = 1,33 \text{ mm} \quad (3.1)$$

Önemli noktalardan biri de ürün boyunu ölçmek için kullanılan enkoderin pozisyonunu Mc\_TouchProbe bloğu ile alabilmek için programda enkoderi sanal sürücü olarak tanıtmış olmamdır. Çünkü bloğun asıl amacı servo sürücünün tuttuğu pozisyon değerini PLC cycle time'a takılmadan tampona yazmaktır.

Ürün boyunu sacı şekillendirip aynı zamanda da süren rollform servosundan almamamın nedeni rollform toparlarının arasından geçirdiği saç üstünde arada bir patinaj çekmesidir. Böyle bir durumda servo döndüğü için patinaj çekmiş olsa ve ürün aslında patinaj boyunca çekilmemesine rağmen program sacın boyunu yanlış algılayabilir ve yanlış ebatlama yapabilirdi. Encoderin bağlantısı Şekil 3.17'de görüldüğü gibidir.



Şekil 3.17. Rollform ve Encoder

Encoder, Şekil 3.16'daki gibi rollform girişinde saca değerek kaymadan dönen plastik bir tekerleğin miline akupledir. Encoderin 1 tur döndüğünde ne kadar ürün akacağı, akuple olduğu tekerleğin çevresine bağlıdır. Yani encoderin 1 turu 5000 puls, tekerlek çapı 132 mm ise;

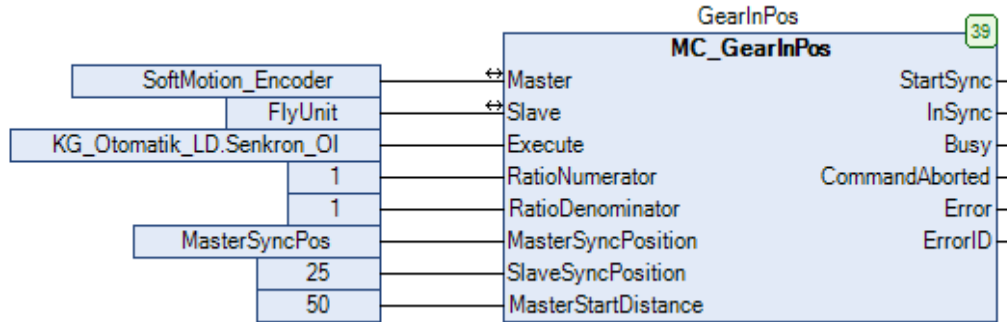
$$5000 \text{ puls} = 132 \times \pi = 414,82 \text{ mm} \quad (3.2)$$

Lineer Makas çalışma algoritması:

1- Senkron olunacak boy hesaplaması yapılır.

Ekrandan girilen kenar sayısı kadar kenar sensörü ile sayılır. Daha sonra Mc\_TouchProbe fonksiyon bloğu aktif edilip ilk gelen delik, delik sensörü ile yakalanıp enkoder pozisyon değeri tampon değişkene atılır. Ekrandan girilecek değer bu pozisyon üstüne eklenir. Aynı zamanda yine ekrandan girilen sensör ile bıçak arası mesafe de bu toplama eklenir. Çünkü sensörün deliği yakaladığı yer fiziksel olarak sensörün olduğu yeri işaret eder. Yani istenen pozisyon bıçak altına gelene kadar bu aradaki fiziksel mesafeyi de katetmesi gerekir.

$$\text{MasterSyncPos} = \text{TouchPos} + \text{Sensor\_Uzakligi} + \text{Offset\_Degeri} \quad (3.3)$$



Şekil 3.18. Programda Kullanılan Mc\_GearInPos Fonksiyon Bloğu

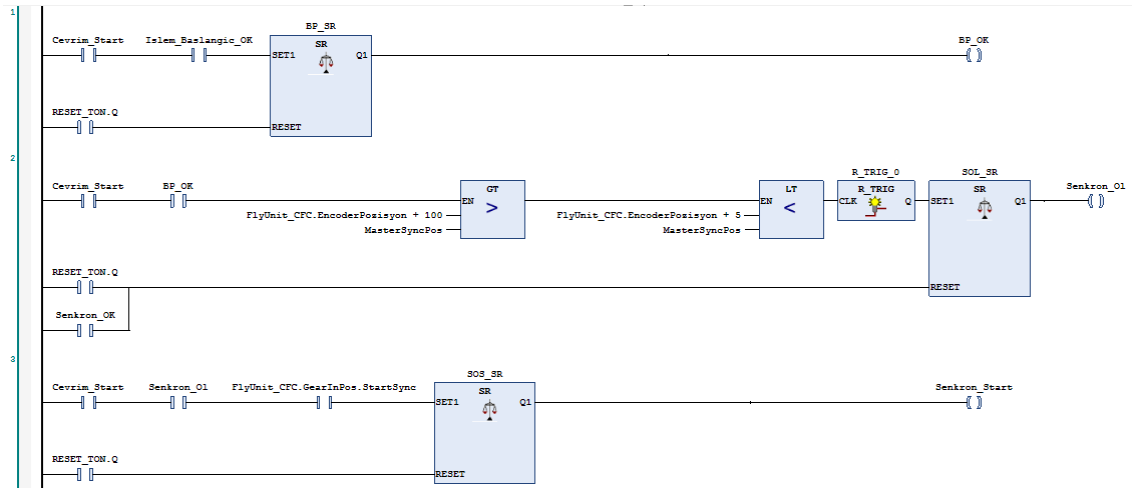
Yakalanan bu pozisyon artık bizim kesim yapacağımız pozisyonudur. Toplam değer olan “MasterSyncPos” yani akan saca tam senkronizasyonun olmasını istediğimiz pozisyon Şekil 3.17'deki Mc\_GearInPos fonksiyon bloğunun “MasterSyncPosition” girişine atanır.

2- Lineer makas ekseninin senkronizasyona başlaması için ilk döngü için direk senkron ol komutu gelir. Daha sonraki döngüler için bir önceki döngüyü tamamlayıp başlangıç pozisyonuna dönmüş olması arkasından “Senkron\_OI” komutu gelir.

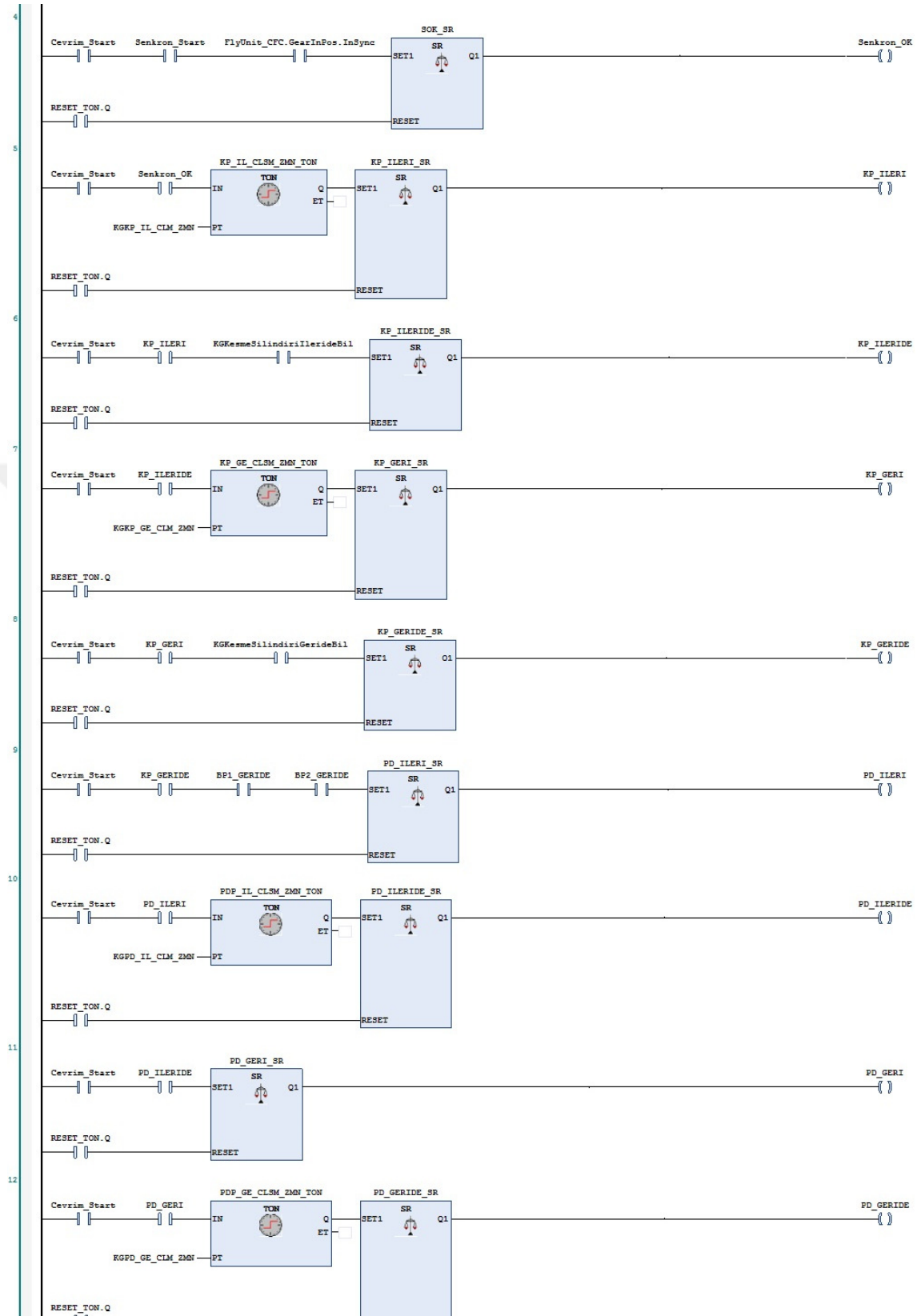
“Senkron\_OI” komutu başka bir alt program olan “KG\_Otomatik\_LD” programından belli şartlara göre gelir.

- 3- “MasterSyncPos” döngüsünün tamamlanması ve “Senkron\_Ol” komutunun gelmesi ile Mc\_GearInPos bloğu kendi içinde hesaplama yaparak senkron olacağı pozisyonu bekler ve rampalı şekilde harekete başlayıp senkron olma işlemine başlar.
- 4- Mc\_GearInPos bloğunun “InSync” çıkışı ile lineer makas ekseninin tam senkron olduğu bilgisi “KG\_Otomatik\_LD” programındaki şartı aktif edip kesme işlemini başlatır.
- 5- Makas mekanizmasının ileri konum bilgisini veren ilerde sensörü ile kestiği anlaşılır ve makas geri çekilir.
- 6- Yine geride olduğu bilgisini veren sensör sayesinde geride olduğu anlaşılır. Kesme işleminin bittiği bu geride sensörüyle anlaşıldıktan sonra lineer eksene Mc\_GearOut fonksiyonu ile senkronizasyondan çık komutu verilir.
- 7- Senkronizasyondan çıktıktan sonra lineer eksene başlangıç pozisyonuna dön komutu verilir.
- 8- Başlangıç pozisyonuna dönen lineer eksen döngüyü bu şekilde sürekli tekrarlar.

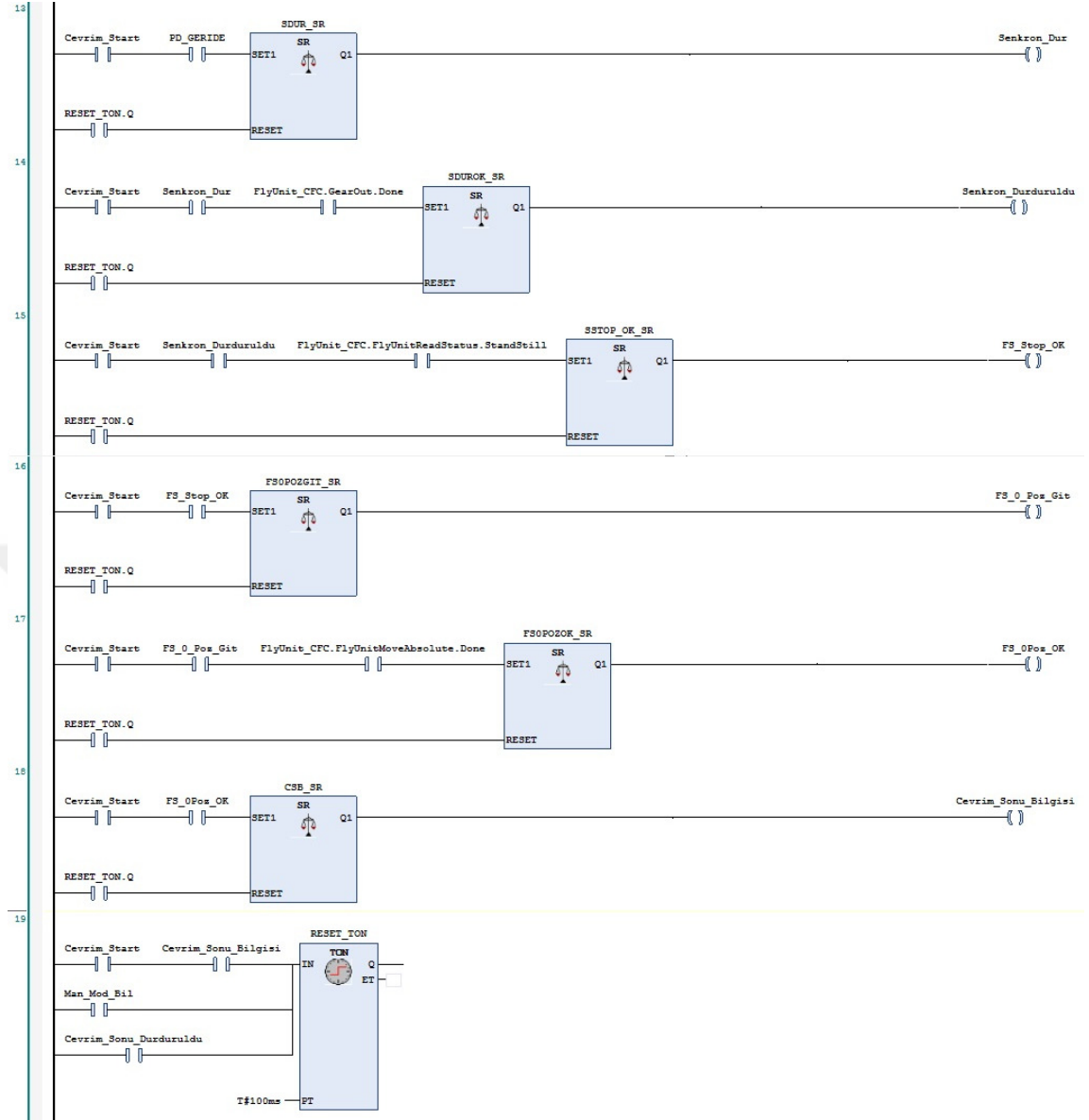
Tüm bu kesme grubuna ait otomatik mod için algoritma sırasıyla Şekil 3.18, Şekil 3.19 ve Şekil 3.20’deki gibidir.



Şekil 3.19. KG\_Otomatik\_LD Programı 1



Şekil 3.20. KG\_Otomatik\_LD Programı 2



Şekil 3.21. KG\_Otomatik\_LD Programı 3

### 3.3.4. Manuel çalışma

Tüm servo eksenlerinin, pistonların ve makasın manuel olarak hem test edilebilmesi hem de gerekli durumlarda çalıştırabilmek için manuel olarak kontrol için de program yazılmıştır. Çok fazla eksen, piston ve makas olduğu için her birine özel ayrı ayrı operatör panelinde manuel kontrol sayfası oluşturulmuştur.

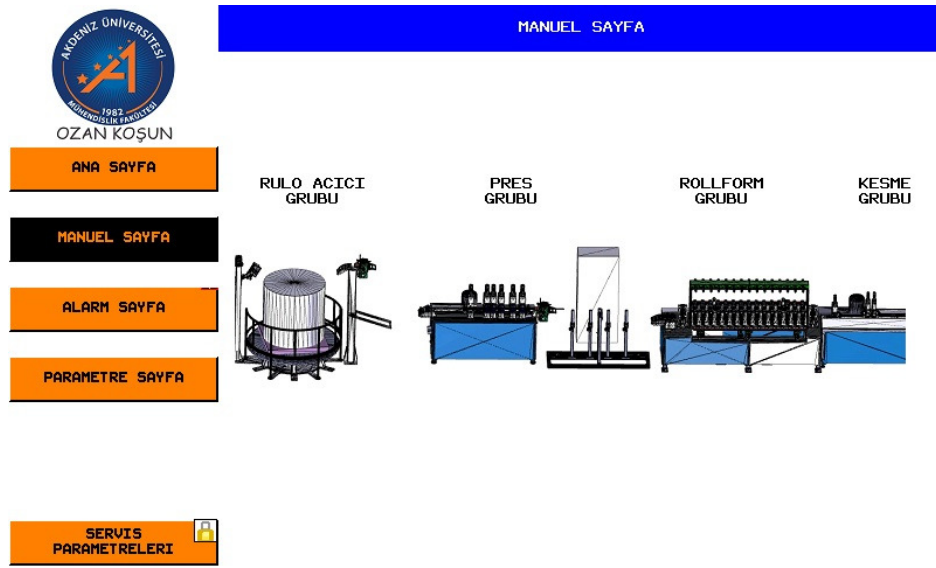
### 3.4. Operatör Paneli Manuel ve Otomatik Kontrol Sayfaları

Şekil 3.22'deki Ana Sayfa ekranından otomatik modda rollform çalışma hızını, delik sensörü ile bıçak arası mesafeyi, delikten sonra kesilecek mesafeyi, sayılacak kenar sayısını ve üretmek istediğimiz parça adedini yazabiliriz. Aynı zamanda diyagnostik bilgi olarak, Sayılan anlık kenar ve delik sayısını, anlık üretilen parça adedi, ve makine çalışmaya başladığından beri üretilen parça adedi bilgilerini görebiliriz.



Şekil 3.22. Ana Sayfası

Şekil 3.23'deki Manuel sayfasından Rulo Açıcı Grubu, Pres Grubu, Rollform Grubu ve Kesme Grubu manuel kontrol sayfalarına geçiş yapılabilir. Bu sayfaların da görüntüleri Şekil 3.24, Şekil 3.25 ve Şekil 3.26'da verilmiştir.



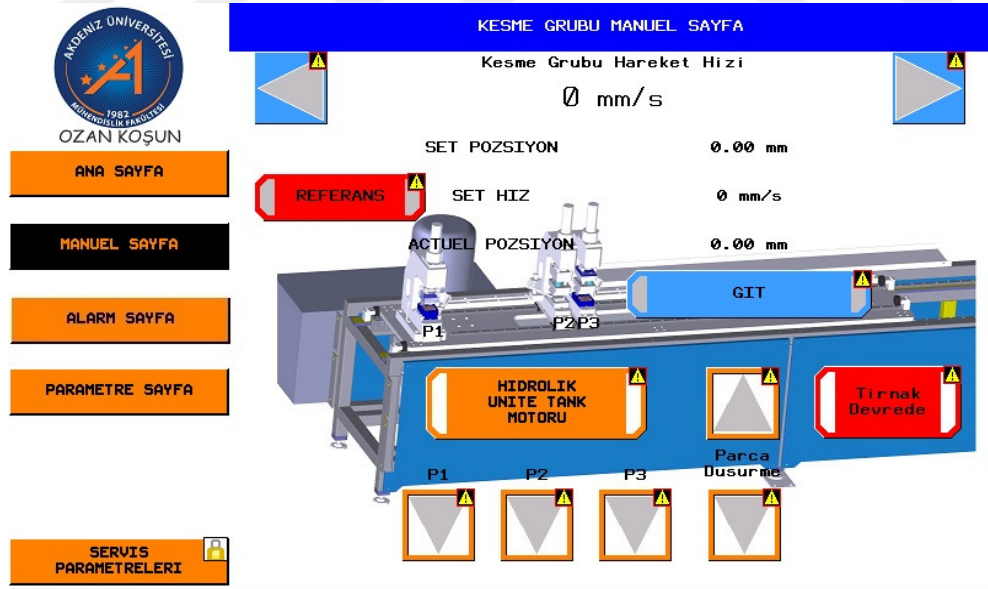
Şekil 3.23. Manuel Sayfası



Şekil 3.24. Rulo Açıcı – Pres Grubu Manuel Sayfası

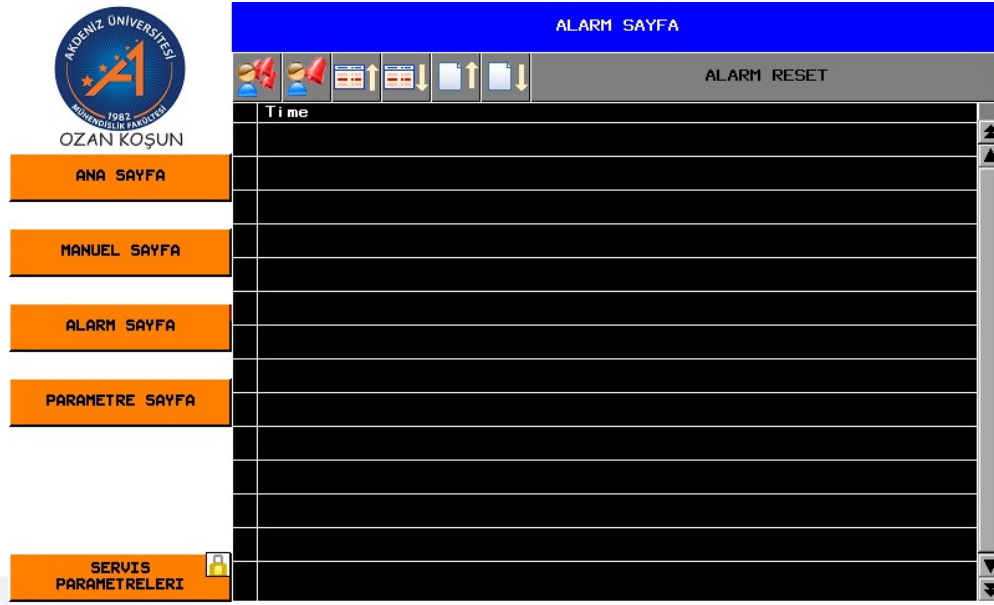


Şekil 3.25. Pres Grubu Servoları – Rollform Manuel Sayfası



Şekil 3.26. Kesme Grubu Manuel Sayfası

Şekil 3.27'deki Alarm Sayfası makinede herhangi bir hata olduğunda otomatik olarak açılır ve hatanın veya hataların ne olduğunu yazar.



Şekil 3.27. Alarm Sayfası

Şekil 3.28, Şekil 3.29 ve Şekil 3.30'da sırasıyla makine parametrelerinin girilebildiği, hassas zaman ayarlarının girilebildiği, reçete seçiminin yapıldığı sayfalar görülmektedir.



Şekil 3.28. Parametre – Zamanlar Sayfası



Şekil 3.29. Reçete Seçim – Pres Grubu Ölçü Reçete Sayfası



Şekil 3.30. Pres Grubu Pres Seçim Reçete – Pres Grubu Pres Pozisyon Reçete Sayfası

#### 4. BULGULAR VE TARTIŞMA

Bir CNC tezgahının ortalama pozisyonlama hassasiyeti 0.01 mm'dir. CNC'de işlenecek malzemenin sabitlendiğini ve işlemin sabit malzeme üstünde yapıldığı göz önünde bulundurulursa saatte 20 m/dk hızla akan ürüne servolu lineer makas mekanizması ile senkron olup işlem gerçekleştirmek ve bunu 0,2 mm hassasiyetle yapmak seri üretici firmalar için makul seviyede yeterlidir. Bunun yanında saç sürme hızı 10 m/dk hızlarının altına düşürdükçe hassasiyet oranının 0.01-0.1 mm arasında gezdiği görülmektedir.

Hat devreye alındıktan sonra makinenin kabulü için 2000 parça üretilmiş ve parçalar uzmanlar tarafından incelenmiştir. 20 m/dk saç hızıyla üretilen 2000 parça ürün hassasiyeti 0.2 mm'dir. Fakat saç hızı 25 m/dk'ya çıkarılıp 100 parça üretilmiş ve alınan hassasiyet sonucu 0.3 mm'ye çıkmıştır. Bunu gözlemlediğimizde sorunun senkron kaçırmasından değil, piston grubunun çıkışında sacı bir ürün boyu çeken servo motorun baskısının az olmasından kaynaklandığı, sacı çekerken motorun patinaj çektiği görülmüştür. Baskı artırıldığında ise saç ince olduğu için şekli değişmiş ve parça istenen formdan çıktığı için baskı artırma çözümünden vazgeçilmiştir. Bunun yerine servo motorun kalkış ve duruş rampaları uzatılmış, yumuşak duruş ve kalkış ile patinajın önüne geçilmeye çalışılmıştır.

Karşımıza çıkan ikinci sorun ise rampaları uzattığımızda, pistonlar ile delinip bir ürün boyu sürme döngüsünün uzaması ve hızın maksimum 20 m/dk ile sınırlı kalmış olmasıdır. Kısaca anlatmak gerekirse; sacın bir ürün boyu sürülüp piston grubunun işlemini yapıp bir sonraki ürün boyu kadar sürme süresi geçene kadar, piston grubu ile şekillendirme için sacı çeken rollform grubunun sacı girmesi, gerginlik sensörünü aktif edip motorun acil duruş gerçekleştirmesine neden olmasıdır. Bunun için pistonların çalışma süresi yani basıp tekrar geri çıkması arasındaki süre kısaltılabilir, akü takılan pistonların yaklaşık 50 ms'de işlemi tamamladığı bilinmektedir. Hali hazırdaki pistonların 3-4 saniye sürelerde işlemi tamamladığı görülmüştür.

Piston grubunun çıkışındaki servo motoru ne kadar patinaj yaparsa yani sacı ne kadar kaçırsa lineer makas kısmındaki delik sensörü bunu telafi etmeye çalışmaktadır. Bunun sonucu olarak gözlemlenen durum ise 10-15 parçada 1 tekrarlayan şekilde ürün boyundaki hatanın 0.01 mm'den yavaş yavaş artarak 0.2 mm'lere kadar çıkıp sensör ile tekrar durumun toparlanıp hatanın minimize edilmesi şeklinde olmuştur.

Hassasiyet için en önemli noktalardan birisinin delik sensörü ile PLC cyle time'dan bağımsız olarak pozisyon bilgisi kaydı olduğu gözlemlenmiştir. Bunun en büyük faydası hattın başında yani piston grubundan kaynaklanan yanlış boy sürme, patinaj gibi hataların üründe olması gereken sabit bir delik takibi yapılarak sürekli hatanın toplanmasının önüne geçmektir.

Yazılımsal ve donanımsal olarak delik sensörünün kullanımı iki farklı şekilde incelenmiştir. Yazılımsal yönden; PLC’deki sensörün bağlandığı girişin filtresinin 0.012 ms ve altında ayarlanmış olması ve sensörün de buna tepki verebilir şekilde olması gerekir yani uygulanan sistemdeki gibi 100 kHz üstü frekanstaki bir sensör seçildiği varsayılırsa bu sensörün yanıt süresi 0.1 ms olacaktır. Bu demektir ki sensör 0.1 ms’de bir delik kontrolü yapacak ve hıza göre boy kaçırma yok denecek kadar az olacaktır. Donanımsal yönden; her PLC için hem standart girişler hem de yüksek hızlı girişler bulunmaktadır. Hatta bazı özel motion PLC’lerde “TouchProbe” için özel hızlı girişler bulunmaktadır.

Sensörün bu hızlı girişlerden birine bağlantısının önemini anlatmak gerekirse, normal girişin tepki süresinin 4 ms, hızlı girişin tepki süresinin 0.012 ms olduğunu düşünürsek, normal girişe bağlı bir sensör ile hızlı girişe bağlı bir sensörün deliği algılayıp PLC’ye iletmesi arasındaki zaman farkı;

$$4 - 0.012 = 3.988 \text{ ms} \quad (4.1)$$

Olacaktır. 20 m/dk hızla akan saç için burada sensörden kaynaklanacak hata ise;

$$(20000 / 60000) \times 3.988 = 1.329 \text{ mm} \quad (4.2)$$

Olacaktır.

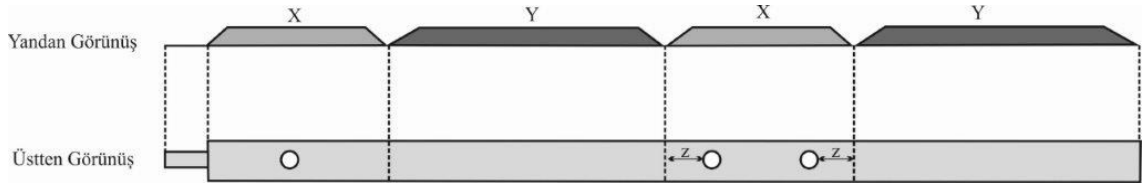
## 5. SONUÇLAR

Gelişen teknoloji ile birlikte fabrikalarda artık insan gücü en minimuma indirgenmiştir. Üreticiler artık insana değil tam otomatik makinalara yatırım yapmaktadır. Sebebi ise insan üretimi ürünlerde hata oranının makinelere oranla çok daha yüksek olması ve üretilen ürünün daha az olmasıdır. Bu yüzden piyasada gelişen teknolojiyi kullanıp otomasyona önem veren firmalar büyüyüp gelişmişlerdir.

Çıkan parti ürünün red yemesi, geri dönüştürülebilir bir ürün olsa bile zamandan ve maddi açıdan üreticiye büyük zarar getirir. Bu yüzden makine başındaki operatör sürekli aralıklarla çıkan üründen numune alıp gerekli kontrollerini yapar. Hata oranı belli bir yüzdeyi geçtiğinde o parti ürünün tamamı kontrolden geçirilir. Bu durum üreticiye başlı başına bir külfet getirir.

Çalışma sonucunda ortaya çıkan otomasyon sistemi ile seri üretim yapan bir fabrika için birim zamanda çıkan ürünün artması ve çıkan ürünlerin hata toleranslarının azalması, hatalı ürün sayısının azalması gibi iyileştirmelerle hem zamandan hem de maliyetten kazanç sağlanmıştır.

Makine kurulduğu fabrikada test edilme şansı bulunmuştur. Yapılan test özellikle hassasiyetin çok önemli olduğu bir ürün üzerinde yapılmıştır. Üretilen ürünün yandan ve üstten görünüşü Şekil 5.1’de verilmiştir. Burada ürün sonsuzdur ve ebatlanarak sürekli bu tipte ürün meydana gelmektedir.



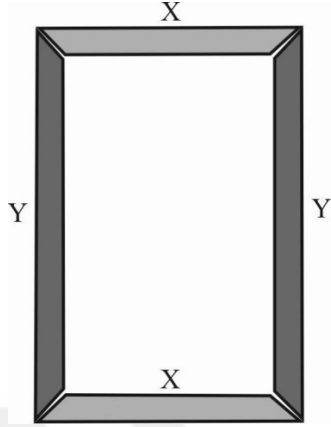
Şekil 5.1. Üretilen Ürün Yandan ve Üstten Görünümü

Şekil 5.1’deki gibi lineer makas altından bu ürün sağa doğru akarken lineer makas eksenine ürüne senkron oluyor ve sol tarafta ince şerit sonundan kesim gerçekleşmektedir. Senkronizasyon işlemi için en soldaki delik referans alınmaktadır. Bu delik yakalandıktan belli bir mesafe sonra kesim yapılacak şekilde algoritma geliştirilmiştir. Üretilen ürünün hassas noktası ise orta kısımdaki iki delik arasına kesim ile birlikte aynı anda Şekil 5.2’deki gibi bir dikdörtgen boşluk açılmasıdır.



Şekil 5.2. Üretilen Ürün Kesildikten Sonra Üstten Görünümü

Bu açılan dikdörtgen boşluk konumu iki yuvarlak deliğe de eşit uzaklıkta olmak zorundadır. Çünkü ürün Şekil 5.3'teki gibi son halini aldığı anda bu 3 boşluğa oturacak bir aparat takılması gerekmektedir.



Şekil 5.3. Üretilen Ürün Son Hali

Burada en büyük sorun ürün boylamada yapılan 1 br hatanın dikdörtgen boşluğun iki yuvarlak delik ortasına gelmesindeki 2 br olarak katlanmasıdır. Yani ürün boyu 1br fazla kesildiğinde, dikdörtgen boşluk soldaki yuvarlak deliğe (a-1) br uzaklıkta olurken sağdaki yuvarlak deliğe ise (a+1) br uzaklıkta olur. Sonuç olarak dikdörtgen boşluğun iki yuvarlak deliğe uzaklıkları farkı 2 br olmaktadır ve bu 3 boşluğa oturtulacak aparatın takılıp takılamamasını büyük oranda etkilemektedir. Bu yüzden bu ürün için istenen boy hatası hassasiyeti  $\pm 0.25$  mm istenmiştir, çünkü aparat  $\pm 0.5$  mm hatadan fazlasında son ürüne takılamamaktadır.

Makine kurulduğu fabrikada seri üretime başlamış ve her parti üründen ara ara örnek ürün alınarak incelenmiştir. Boyut olarak hatasının  $\pm 0.2$  mm ve altında olduğu görülmüştür. Aynı zamanda aynı fabrikada çalışan eşdeğer makinalar saatte 300 adet ürün çıkartırken yeni kurulan makine 600 adet ürün çıkartmaya başlamıştır. Yani tam olarak çıkan ürün adedinde %100 artış olduğu görülmüştür. Bunun sebebinin otomatik kontrol sırasında bu çalışma ile geliştirilen sistemdeki duruş kalkış zamanlarının diğer makinalara göre çok az olmasından kaynaklandığı görülmüştür.

Tek bir amaca yönelik düşünülen lineer makas sistemi aslında bir motorun diğer bir motora tam senkron olmasıyla gerçekleştirilen bir uygulamadır ve geliştirilen algoritmadaki küçük revizyonlarla akan bant üstündeki ürünlere barkodlama, etiketleme, hattı durdurmadan şişeleme vb. gibi süreçler için de adaptasyon yapılabilme olanağı bulunmaktadır. Çalışmanın pratikte ve endüstride üretim süreçlerine doğrudan olumlu katkısı olacağı, bu alanda yapılacak çalışmalara önemli bilgi birikimi, deneyim ve kaynak teşkil edeceği düşünülmektedir.

## 6. KAYNAKLAR

- AKAR, M. 2005. Bulanık Mantık Yöntemiyle Bir Servo Motorun Kontrolü ve Geleneksel Yöntemlerle Karşılaştırılması, Yüksek Lisans Tezi, *Marmara Üniversitesi Fen Bilimleri Enstitüsü*, İstanbul, 1-15.
- AKBAŞ, H.G. 2010. Servo Kontrollü CNC Cam Delme Makinesi, Yüksek Lisans Tezi, *Erciyes Üniversitesi Fen Bilimleri Enstitüsü*.
- AYÇİÇEK, E. 2005. Servo Tahrik Sistemlerinin Endüstriyel Uygulamaları ve Bir Servo Sistemin Tasarım Aşamaları, Yüksek Lisans Tezi, *Yıldız Teknik Üniversitesi Fen Bilimleri Enstitüsü*, İstanbul, 1-67.
- BAL, G. 2004. Özel Elektrik Makinaları, Seçkin Yayınevi, Ankara, 179-195.
- BELOBROY, N., ELETSEKH, V.I. and GRITSENKO, S.A. 2006. Ways of improving the speed parameters of drum-type flying shears for cutting flat-rolled products to measured lengths, *Metallurgist*, Vol. 50. No. 11-12, 597-601.
- BOLTON, W. 1999. Mechatronics-Electronic Control Systems in Mechanical Engineering, Longman.
- BOSE, B.K. 2002. Modern Power Electronics and ac Drives. Prentice Hall, Upper Saddle River, USA.
- BOZKURT, N. 2006. Mikrodenetleyici Kontrollü Servo Gerilim Regülatörünün Tasarım ve Uygulaması, Yüksek Lisans Tezi, *Gazi Üniversitesi Fen Bilimleri Enstitüsü*, Ankara.
- BROSCH, F.P. 1999. Variable Speed Drives for Automation, ISBN:3E8259E1904E8, Vogel.
- ÇENGELCİ, B. ve Çimen, H. 2005. Endüstriyel Robotlar, *Makine Teknolojileri Elektronik Dergisi*, ISSN:1304-4141, 69-78.
- ÇETİN, İ. 2001. Temel Elektrik Kullanım El Kitabı, İTO Yayınları.
- DOĞAN, Z. 2009. Alternatif Akım Servo Motorun Farklı Kontrol Yöntemleri ile Hız Kontrolü, *Marmara Üniversitesi Fen Bilimleri Enstitüsü*, İstanbul.
- DUMANAY, A.B. 2009. PID, Bulanık Mantık ve Kapan Kip Kontrol Yöntemleri ile İnternet Üzerinden DC Motor Hız Kontrolü, Yüksek Lisans Tezi, *Balıkesir Üniversitesi Fen Bilimleri Enstitüsü*.
- DÜLGER, L.C. ve KİREÇCİ, A. 2001. AC Servo Motorların Modellenmesi Simülasyonu ve Hareket Denetiminde Kullanılması. 10. Ulusal Mak. Teo. Sempozyumu Bil. Kit., Cilt 1, 181-189.

- HALL, P. and CLIFFORD, M. 2005. Modern Electric/Electronic Motors, Prentice Hall, Englewood Cliffs, 195.
- HAMEYER, K. 2004. Moderne Servomotoren, Institut für Elektrische Maschinen der RWTH Aachen, Deutschland.
- HANCI, O. 2007. Servo Motorlar ve Örnek Bir Uygulama Tasarımı, Yüksek Lisans Tezi, *Marmara Üniversitesi Fen Bilimleri Enstitüsü*, İstanbul.
- HSIEH, J.F. 2009. Design and analysis of cams with three circular-arc profiles, *Mechanism and Machine Theory*, 45 , 955–965.
- JIAZHONG, X. and LEI, Z. 2009. Research on Electronic Cam Based on Nurbs Interpolation Algorithm. *The Ninth International Conference on Electronic Measurement & Instruments*.
- KOLESNIKOV, N. 2004. Improving high-productivity drum-type flying shears for cutting flat-rolled products to measured lengths, *Metallurgist*, No. 11, 48–49.
- KUO, B.C. 2005. Otomatik Kontrol Sistemleri, 3. Baskı, Literatür Yayıncılık, İstanbul, 9-19.
- LIAO, C.S. and JENG, S.L. 2003. Electronic cam motion generation with special reference to constrained velocity, acceleration, and jerk, *ISA Transactions*, 43 , 427–443.
- MEGEP, 2007. Servo Motor ve Sürücüler, *Elektrik Elektronik Teknolojisi*, Ankara.
- OGATA, K. 1990. Modern Control Engineering, Second Edition, Prentice Hall, Inc., USA, 7.
- PERIC, N. and PETROVIC, I. 1990. Flying shear control system, *IEEE Transactions On Industry Applications*. Vol. 26. No. 6, 1049-1056.
- POLAT, A. 2009. X-Y Eksenli Kayış-Kasnak Servo Sistemin Modellenmesi Ve Kontrolü, Yüksek Lisans Tezi, *İTÜ Fen Bilimleri Enstitüsü*, İstanbul.
- SARIOĞLU, M.K. 1985. Otomatik Kontrol I-II, *İTÜ Baskı Atölyesi*, İstanbul.
- SELEZNEVA, A. 2007. Modelling and Sythesis of Tracking Control for the Belt Drive System. Master's Thesis, *Lappeenranta University of Technology*, Lappeenranta, Finland.
- TOSUNER, M. 2004. Otomasyon Atölyesi Ders Notları, Kocaeli Anadolu Teknik ve Endüstri Meslek Lisesi, <http://www.kumanda.org>

WOELFEL, M. 1999. Introduction to electronic cam, *Assembly Automation.*, Vol. 19, No. 1, 17-24.

YAOBIN, C. 2003. Dual-loop Feedback Control of Servo Motor Systems Using Singular Perturbation Method, IEEE, volume:6, issue 4-6.

YOUNKIN, G.W. 1996. Industrial Servo Controls Systems, Marcel Dekker, Newyork.

URL-1: <http://www.mersin.edu.tr/>

URL-2: <https://www.codesys.com/>

URL-3: <http://www.can-cia.org/>

URL-4: <http://www.plcopen.org/>

URL-5: <http://www.barissamanci.net/>

## 7. EKLER

### EK-1: Pres grubu için geliştirilen fonksiyon bloğu program kodu

IF bir\_sonraki\_adima\_gec=TRUE AND baslangic\_adimina\_don=FALSE THEN

I:=I+1;

qx\_servo\_git:=FALSE;

pres1\_geri\_don:=FALSE;

pres2\_geri\_don:=FALSE;

pres3\_geri\_don:=FALSE;

pres4\_geri\_don:=FALSE;

pres5\_geri\_don:=FALSE;

pres6\_geri\_don:=FALSE;

pres1\_geri\_dondu:=FALSE;

pres2\_geri\_dondu:=FALSE;

pres3\_geri\_dondu:=FALSE;

pres4\_geri\_dondu:=FALSE;

pres5\_geri\_dondu:=FALSE;

pres6\_geri\_dondu:=FALSE;

bir\_sonraki\_adima\_gec:=FALSE;

END\_IF

IF baslangic\_adimina\_don=TRUE OR reset\_blok THEN

I:=1;

qx\_servo\_git:=FALSE;

pres1\_geri\_don:=FALSE;

pres2\_geri\_don:=FALSE;

pres3\_geri\_don:=FALSE;

pres4\_geri\_don:=FALSE;

pres5\_geri\_don:=FALSE;

pres6\_geri\_don:=FALSE;

```
pres1_geri_dondu:=FALSE;
pres2_geri_dondu:=FALSE;
pres3_geri_dondu:=FALSE;
pres4_geri_dondu:=FALSE;
pres5_geri_dondu:=FALSE;
pres6_geri_dondu:=FALSE;
bir_sonraki_adima_gec:=FALSE;
baslangic_adimina_don:=FALSE;
yapilan_adim_adeti:=0;
END_IF
IF servo_pos[I]<>0 THEN
  IF qx_servo_git=0 AND calisma_guvenligi AND NOT servo_pos_ok AND NOT
  Acil_Stop_Basildi THEN
    qx_servo_pos:=servo_pos[I];
    qx_servo_git:=TRUE;
  ELSIF
    qx_servo_git=TRUE AND calisma_guvenligi AND NOT servo_pos_ok AND
    Acil_Stop_Basildi THEN
    qx_servo_git:=FALSE;
    qx_servo_pos:=servo_pos[I]-servo_actuel_pozisyon;
  END_IF
  STGIT_TON (IN:= qx_servo_git=FALSE AND calisma_guvenligi AND NOT
  servo_pos_ok AND Acil_Stop_Basildi,PT:= T#500MS);
  IF STGIT_TON.Q THEN
    qx_servo_git:=TRUE;
    Acil_Stop_Basildi:=FALSE;
  END_IF
  IF servo_gitti_bilgisi=TRUE OR servo_pos_ok THEN
```

```
qx_servo_git:=FALSE;
servo_pos_ok:=TRUE;

IF olcu_p1aktif[I]=1 AND sw_pres1_geri=1 AND NOT pres1_geri_don AND NOT
pres1_geri_dondu THEN

qx_pres1:=TRUE;

ELSIF olcu_p1aktif[I]=0 THEN
pres1_geri_dondu:=TRUE;
END_IF

IF qx_pres1=1 AND sw_pres1_ileri=1 AND P1ILTON.Q THEN

qx_pres1:=FALSE;
pres1_geri_don:=TRUE;
END_IF

IF pres1_geri_don=TRUE AND sw_pres1_geri=1 AND P1GETON.Q THEN
pres1_geri_dondu:=TRUE;
END_IF

IF olcu_p2aktif[I]=1 AND sw_pres2_geri=1 AND NOT pres2_geri_don AND NOT
pres2_geri_dondu THEN

qx_pres2:=TRUE;

ELSIF olcu_p2aktif[I]=0 THEN
pres2_geri_dondu:=TRUE;
END_IF

IF qx_pres2=1 AND sw_pres2_ileri=1 AND P2ILTON.Q THEN

qx_pres2:=FALSE;
pres2_geri_don:=TRUE;
END_IF

IF pres2_geri_don=TRUE AND sw_pres2_geri=1 AND P2GETON.Q THEN
pres2_geri_dondu:=TRUE;
END_IF
```

```
IF olcu_p3aktif[I]=1 AND sw_pres3_geri=1 AND NOT pres3_geri_don AND NOT  
pres3_geri_dondu THEN
```

```
qx_pres3:=TRUE;
```

```
ELSIF olcu_p3aktif[I]=0 THEN
```

```
pres3_geri_dondu:=TRUE;
```

```
END_IF
```

```
IF qx_pres3=1 AND sw_pres3_ileri=1 AND P3ILTON.Q THEN
```

```
qx_pres3:=FALSE;
```

```
pres3_geri_don:=TRUE;
```

```
END_IF
```

```
IF pres3_geri_don=TRUE AND sw_pres3_geri=1 AND P3GETON.Q THEN
```

```
pres3_geri_dondu:=TRUE;
```

```
END_IF
```

```
IF olcu_p4aktif[I]=1 AND sw_pres4_geri=1 AND NOT pres4_geri_don AND NOT  
pres4_geri_dondu THEN
```

```
qx_pres4:=TRUE;
```

```
ELSIF olcu_p4aktif[I]=0 THEN
```

```
pres4_geri_dondu:=TRUE;
```

```
END_IF
```

```
IF qx_pres4=1 AND sw_pres4_ileri=1 AND P4ILTON.Q THEN
```

```
qx_pres4:=FALSE;
```

```
pres4_geri_don:=TRUE;
```

```
END_IF
```

```
IF pres4_geri_don=TRUE AND sw_pres4_geri=1 AND P4GETON.Q THEN
```

```
pres4_geri_dondu:=TRUE;
```

```
END_IF
```

```
IF olcu_p5aktif[I]=1 AND sw_pres5_geri=1 AND NOT pres5_geri_don AND NOT  
pres5_geri_dondu THEN
```

```
qx_pres5:=TRUE;
ELSIF olcu_p5aktif[I]=0 THEN
pres5_geri_dondu:=TRUE;
END_IF

IF qx_pres5=1 AND sw_pres5_ileri=1 AND P5ILTON.Q THEN
qx_pres5:=FALSE;
pres5_geri_don:=TRUE;
END_IF

IF pres5_geri_don=TRUE AND sw_pres5_geri=1 AND P5GETON.Q THEN
pres5_geri_dondu:=TRUE;
END_IF

IF olcu_p6aktif[I]=1 AND sw_pres6_geri=1 AND NOT pres6_geri_don AND NOT
pres6_geri_dondu THEN
qx_pres6:=TRUE;
ELSIF olcu_p6aktif[I]=0 THEN
pres6_geri_dondu:=TRUE;
END_IF

IF qx_pres6=1 AND sw_pres6_ileri=1 AND P6ILTON.Q THEN
qx_pres6:=FALSE;
pres6_geri_don:=TRUE;
END_IF

IF pres6_geri_don=TRUE AND sw_pres6_geri=1 AND P6GETON.Q THEN
pres6_geri_dondu:=TRUE;
END_IF

END_IF

IF pres1_geri_dondu=1 AND pres2_geri_dondu=1 AND pres3_geri_dondu=1 AND
pres4_geri_dondu=1 AND pres5_geri_dondu=1 AND pres6_geri_dondu=1 THEN
```

```
yapilan_adim_adeti:=yapilan_adim_adeti+1;
servo_pos_ok:=FALSE;
IF vurum_adedi[I]=yapilan_adim_adeti THEN
yapilan_adim_adeti:=0;
bir_sonraki_adima_gec:=TRUE;
ELSE
qx_servo_git:=FALSE;
pres1_geri_don:=FALSE;
pres2_geri_don:=FALSE;
pres3_geri_don:=FALSE;
pres4_geri_don:=FALSE;
pres5_geri_don:=FALSE;
pres6_geri_don:=FALSE;
pres1_geri_dondu:=FALSE;
pres2_geri_dondu:=FALSE;
pres3_geri_dondu:=FALSE;
pres4_geri_dondu:=FALSE;
pres5_geri_dondu:=FALSE;
pres6_geri_dondu:=FALSE;
END_IF
END_IF
ELSE
baslangic_adimina_don:=TRUE;
END_IF
urun_bitti:=baslangic_adimina_don;
```

// Bekleme Zamanları

```
P1ILTON ( IN:= qx_pres1=1 AND sw_pres1_ileri=1,
          PT:= Ileride_Bekleme_Zamani);

P1GETON ( IN:= pres1_geri_don=TRUE AND sw_pres1_geri=1,
          PT:= Geride_Bekleme_Zamani);

P2ILTON ( IN:= qx_pres2=1 AND sw_pres2_ileri=1,
          PT:= Ileride_Bekleme_Zamani);

P2GETON ( IN:= pres2_geri_don=TRUE AND sw_pres2_geri=1,
          PT:= Geride_Bekleme_Zamani);

P3ILTON ( IN:= qx_pres3=1 AND sw_pres3_ileri=1,
          PT:= Ileride_Bekleme_Zamani);

P3GETON ( IN:= pres3_geri_don=TRUE AND sw_pres3_geri=1,
          PT:= Geride_Bekleme_Zamani);

P4ILTON ( IN:= qx_pres4=1 AND sw_pres4_ileri=1,
          PT:= Ileride_Bekleme_Zamani);

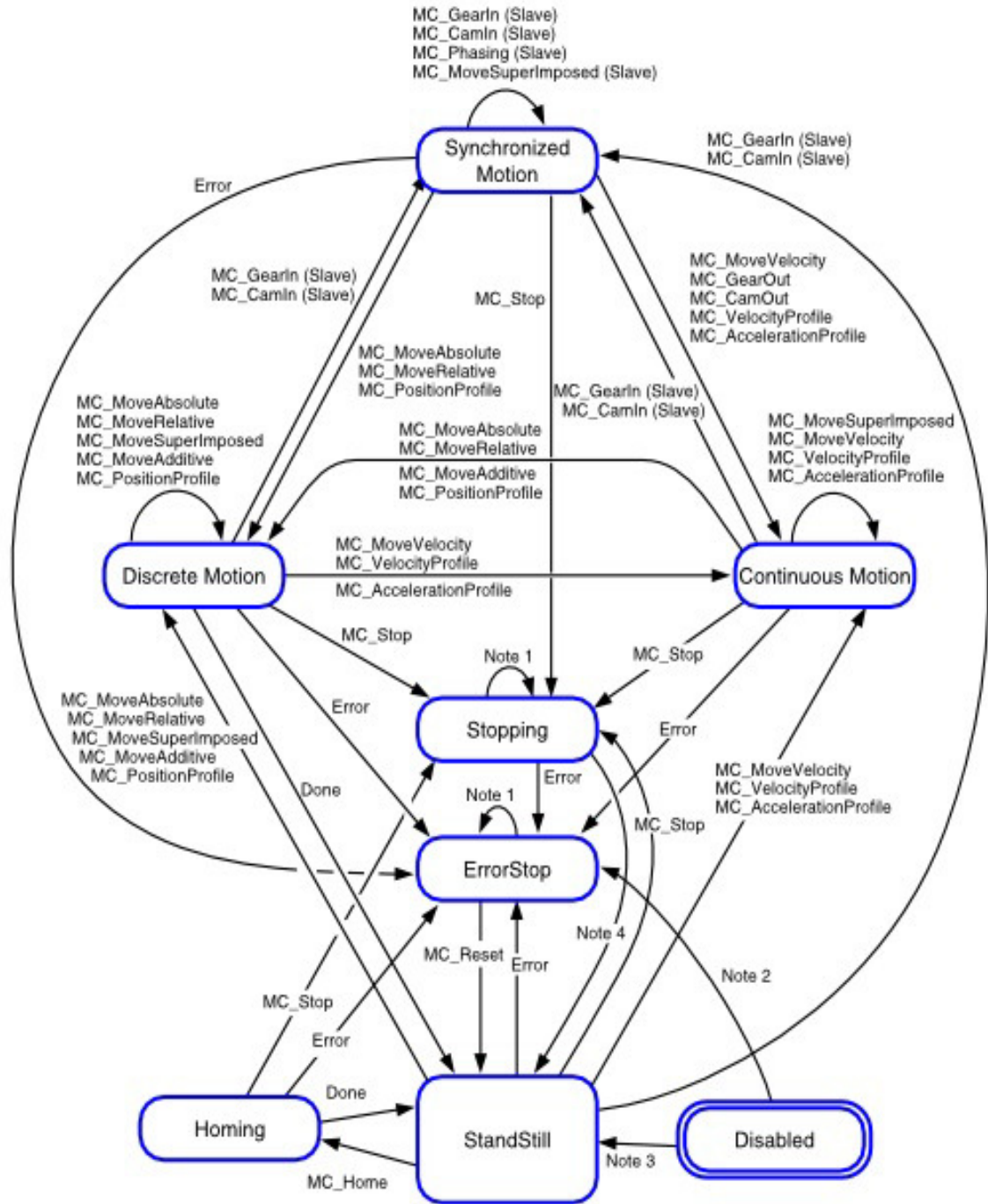
P4GETON ( IN:= pres4_geri_don=TRUE AND sw_pres4_geri=1,
          PT:= Geride_Bekleme_Zamani);

P5ILTON ( IN:= qx_pres5=1 AND sw_pres5_ileri=1,
          PT:= Ileride_Bekleme_Zamani);

P5GETON ( IN:= pres5_geri_don=TRUE AND sw_pres5_geri=1,
          PT:= Geride_Bekleme_Zamani);

P6ILTON ( IN:= qx_pres6=1 AND sw_pres6_ileri=1,
          PT:= Ileride_Bekleme_Zamani);

P6GETON ( IN:= pres6_geri_don=TRUE AND sw_pres6_geri=1,
          PT:= Geride_Bekleme_Zamani);
```

**EK-2: PLCopen motion blokları durum diyagramı**

**EK-3: Tüm Sistemin PLC ve HMI programının bulunduğu proje dosyası.  
(Schneider SoMachine) (CD kayıt.)**

**EK-4: Otomasyon panosunun elektrik projesi.  
(CD kayıt.)**

**EK-5: Hattın kurulduğu yerde tam kapasite çalışırken çekilmiş videosu.  
(CD kayıt.)  
( <https://youtu.be/CUk8VP8NeE0> )**



## ÖZGEÇMİŞ

### OZAN KOŞUN

Tel : 0 (555) 562 56 44  
E-posta : [kosunozan@gmail.com](mailto:kosunozan@gmail.com)  
Adres : Arapsuyu Mah. 620 Sok.  
Berkay Apt. No:5  
Konyaaltı/ANTALYA



1991’de Ankara’da doğdu. İlköğrenimini Ankara’da O.D.T.Ü. Vakfı İlköğretim Okulu’nda tamamladıktan sonra Antalya’ya taşındı ve lise öğrenimini Aldemir Atilla Konuk Anadolu Lisesi’nde tamamladı. 2008 yılında ÖSS ile Selçuk Üniversitesi Elektrik-Elektronik Müh. Bölümü’ne yerleşti. Burada 3 sene Onur Öğrencisi olduktan sonra Gazi Üniversitesi Elektrik-Elektronik Müh. Bölümü’ne Farabi Geçiş Programıyla geçiş yaptı. 2012 yılında Lisans eğitimini burada tamamladı. 2013 yılında Cantek Soğutma A.Ş. şirketinde Ar-Ge Mühendisi olarak çalışmaya başladı. 6 ay çalıştıktan sonra 2014 yılında kısa dönem er olarak 6 ay askerlik yaptı. Askerlik bittikten sonra, 2014 yılında Botek Elektrik ve Otomasyon San. Tic. Ltd. Şti. şirketinde Otomasyon Sorumlusu olarak çalışmaya başladı ve halen çalışmaya devam etmektedir. Aynı zamanda 2014 yılından beri Akdeniz Üniversitesi Fen Bilimleri Enstitüsü Elektrik-Elektronik Müh. Anabilim Dalı’nda Yüksek Lisans eğitimine devam etmektedir.