İSTANBUL TEKNİK ÜNİVERSİTESİ * FEN BİLİMLERİ ENSTİTÜSÜ

ENDÜSTRİYEL KONTROL DONANIMLARINDA HABERLEŞME SİSTEMLERİ

YÜKSEK LİSANS TEZİ

Müh. Hacer FEYİZ

Anabilim Dalı : Uzay Bilimleri ve Teknolojisi

Programı : Uzay Bilimleri ve Teknolojisi

HAZİRAN 1995

İSTANBUL TEKNİK ÜNİVERSİTESİ * FEN BİLİMLERİ ENSTİTÜSÜ

ENDÜSTRİYEL KONTROL DONANIMLARINDA HABERLEŞME SİSTEMLERİ

YÜKSEK LİSANS TEZİ

Müh. Hacer FEYİZ

Tezin Enstitüye Verildiği Tarih : 12 Haziran 1995

Tezin Savunulduğu Tarih : 30 Haziran 1995

Tez Danışmanı : Y.Doç.Dr. Turgut Berat KARYOT

Diğer Jüri Üyeleri : Y.Doç.Dr. Alper ORAL
Doç.Dr. Şakir KOCABAŞ

HAZİRAN 1995

ÖNSÖZ

Yüksek lisans öğrenimim boyunca değerli katkılarından dolayı öncelikle, danışmanım Y.Doç.Dr. Turgut Berat KARYOT'a ve tüm hocalarıma teşekkür ederim. Bu tezin hazırlanmasında her türlü imkanı sağlayan PiOMAK Otomasyon ve Makina Sanayi A.Ş. Genel Müdürü Oral AVCI'ya, Teknik Müdürü Orhan NALBANTLI'ya, Araştırma-Geliştirme bölümünde çalışma arkadaşlarım Şükrü ERGÜR ve Enis SARAÇ beyler ile tüm şirket çalışanlarına teşekkürlerimi sunarım.

Hacer FEYİZ, 1995

SEMBOL LİSTESİ

PLC	Programmable Logic Controller = Programlanabilir Mantık Denetleyici
I/O	Input/Output = Giriş/Çıkış
CP	Communication Processor= Haberleşme işlemcisi
CPU	Central Processing Unit = Merkezi İşlem Birimi
PS	Power Supply = Güç Kaynağı
VME	Versa Modular Eurocard standartı
RAM	Random Acces Memory = Rastgele Erişimli Bellek
ROM	Read Only Memory = Salt Okunur Bellek
EPROM	Erasable Programmable Read Only Memory = Silinebilir Programlanabilir Salt Okunur Bellek
DOS	Disk Operating System = Disk İşletim Sistemi
IEC	International Electrotechnic Commitee
DCS	Ditributed Control System = Dağıtılmış Kontrol Sistemi
BCC	Block Check Character = veri bloğu kontrol karakteri

ŞEKİL LİSTESİ

İÇİNDEKİLER

ÖZET

Bu tez çalışmasında, öncelikle Siemens S5-115 U kontrol sisteminin, diğer bilgisayar ve kontrolcülerle haberleşmesinde kullanılan 3964R Protokolü incelenmiştir. Kişisel bilgisayar veya VMEbus tabanlı mikrobilgisayar ile Siemens PLC arasında bu protokole uygun olarak, RS232C üzerinden haberleşmenin sağlanması amaçlanmıştır. Bu çerçevede, hedef konuyu açıklamak üzere, genel birtakım konuların açıklanması da gerekmektedir. Bu sebeple, giriş bölümünün ardından ikinci bölümde, bir kontrol sisteminin genel yapısını tanıtmaya ayrılmıştır. Üçüncü bölümde programlanabilir kontrol elemanlarında donanım ve yazılım konularına ayrılmıştır. Dördüncü bölümde, endüstriyel kontrol donanımlarındaki haberleşme sistemleri açıklanmıştır. Beşinci bölüm S5-115 U programlanabilir kontrol sistemine, altıncı bölüm CP524 haberleşme işlemcisine ve yedinci bölüm 3964R haberleşme protokolü ile uygulanmasına ayrılmıştır. Tez çalışmasının ikinci aşamasını, VMEbus tabanlı mikrobilgisayarla PLC arasında iletişim kurulması oluşturmaktadır. Bu amaçla yedinci bölümde VMEbus ve bu sistemde kullanılan OS-9 işletim sisteminin genel yapısı tanıtılmaktadır. Sonuçlar ve öneriler kısmında yapılan uygulama ve araştırmalar değerlendirilmiştir.

COMMUNICATION TOOLS IN INDUSTRIAL CONTROL SYSTEMS

SUMMARY

In this thesis, industrial control systems and their communication hardware and software are examined. Features of the programmable logic controllers, which are commonly used in automation sector, are explained. As a specific application, 3964R communication procedure, which is used by CP524 communication processor of Siemens S5-115 U programmable controller, was realised between PC and CP524 via RS232 serial connection. Besides, the same communication ability is used for VMEBus based microcomputer. In result, using the programmes which were developed in term of this work, serial communication between Siemens programmable controller and PC or VMEBus computer, is provided without any additional hardware and software.

In Section 1, there is an introduction to the industrial control systems and historical knowledge about improvements in hardware was given.

Basic elements of control systems are explained in Section 2. These are:

i. Network

ii. Sensors

iii. Actuators

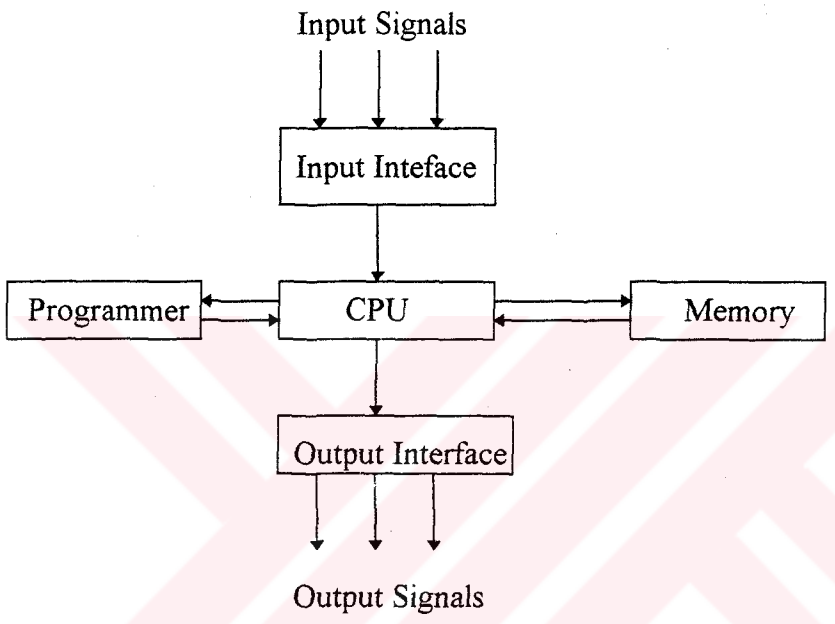
iv. Processors

v. Software

After that, there is a short explanation about fundamentals of digital control.

Section 3 was assigned to hardware and software developments in programmable logic controllers. National Electrical Manufacturers Association in

USA, described a programmable controller as a digital electronic tool which has a programmable memory that provides some logic, arithmetic, timing, sequencing, counting instructions to control machines and processes.



Programmable logic controller takes input signals by means of input interface and central processing unit of controller processes input signals in compliance with control program, then transmits output signals to the machines and processes via output intefaces.

Basic elements of PLC:

i. Processors

ii. Input/Output Modules

iii. PLC inteface units

After that, explanations about software and standard programming languages of PLC take place in second part of this section.

- Assembly languages
- High level languages
- Package program (macro, menu, graphics)

are commonly used in PLC programming.

In Section 4, communication tools in industrial control systems and especially fieldbus are examined. The most famous communication tools are:

- Monitoring tools
- Keyboards, hand-terminals
- Local area network
- printers

Fieldbus term which was mentioned in this section, expresses a low level communication architecture that provides data exchange beetween PLC and field elements like sensors and actuators, in two direction.

Siemens S5-115U industrial controller was introduced in Section 5. S5-115U has a modular structure that enables to expand the system to meet the system requirements. It can be used for various application from sample loop control to complex closed loop control.

Basic modules of S5-115 U controller are:

i. power supply unit

ii. central processing unit

iii. input/output units

iv. intelligent input/output units

v. communication processors

vi. expanding interfaces

Explanations about Siemens SIMATIC S5 CP524 communication processor take place in Section 6. Data is tranmitted between Siemens PLC and personal computer or VMEbus based industrial computer via RS232 connector on CP524 by using 3964R procedure.

In 3964R procedure, units, which communicate with each other, posts telegrams that include necessary information, reciprocal. Details about telegram structure are explained in second part of this section.

In Section 7, VMEbus and OS-9 operating system were mentioned. VMEbus, which was formed by experts from Motorola, Mostek, Signetic firms as a result of corporation, is a computer bus architecture like ISAbus, Multibus and Microbus.

VMEbus, which have 32 bits adress, 32 bits data bus and supports multiprocessor and 7 interrupt levels, uses master/slave architecture.

OS-9 is a comlete real-time operating system for the Motorola 680X0 family of microprocessors. The OS-9 operating system features a modular, scalable architecture for maximum flexibility and is the only operating system that can be used across the entire spectrume of 680X0 family systems.

OS-9 combines significant operating system concepts and real time capabilities with an overall architecture that is very compact, highly efficent and incorporates many key UNIX features. The OS-9 development environment provides for advanced programming languages, networking, man/machine interfaces.

OS-9 was designed for maximum efficency with minimum overhead in real-time applications. All speed-critical portions of OS-9 were coded in 680X0 assebly language for optimum performance. Various features were built into the OS-9 Kernel and unified I/O system to the facilitate real-time programming. These features include a priority-based, pre-emptive task scheduler, a full set of process execution control functions, fast interrupt service routine dispatching and a flexible set of interprocess communication facilities.

In results and suggetions, basic concepts of the research about communication tools in industrial control systems were evaluated. In conclusion, a communication interface program which applies 3964R procedure between Siemens CP524 communication processor and personal computer or VMEbus based industrial computer was developped.

BÖLÜM 1

GİRİŞ

Programlanabilir kontrol elemanları, Türkiye'de günden güne daha yaygın bir kullanım alanına yayılmaktadır. Sanayi sektöründe, binalarda, trafik, haberleşme ve diğer sinyalizasyon işlerinde otomasyona geçilmesi, bu konudaki çalışmaları hızlandırmış ve popülerlik kazandırmıştır. Donanım aşamasında, çeşitli firmalar, artık standartlaşan çok sayıda ürünleriyle dünya piyasasındaki yerlerini alırken, bu konuda ülkemizde, pek fazla geliştirme imkanı bulunamayacağı görülmektedir. Ancak, ürünlerin çok özel amaçlarla, çok farklı sistemler için uyarlanmasının gerekliliği; bu konudaki yazılım çalışmalarının önemini, donanımdan daha fazla artırmaktadır. Özellikle kontrol elemanlarının birbirleriyle ve diğer sistemlerle etkin ve güvenilir haberleşmesi, araştırmacılar için oldukça değişik konular içermektedir.

1.1. Programlanabilir Denetleyicilerin Tarihçesi

Endüstriyel tesislerde denetim sistemlerinin ilk kullanılışı XIX. yüzyılın sonlarında olmuştur. Bu yıllarda montaj hatlarının çeşitli kısımlarının otomasyonu için, özel olarak tasarlanmış, mekanik düzenler kullanılmıştır. 1920'li yıllardan itibaren ise röleler ve kontaktörler kullanılmaya başlamış ve böylece denetim sistemlerinde daha ileri bir düzeye ulaşılmıştır. Rölelerin ömürlerinin uzun olmaması ve çalışma frekanslarının sınırlı olması gibi nedenler, anahtarlama amaçları ile kullanılabilecek, yeni elemanlar aranmasına yol açmış ve 1950'li yıllarda Germanyum transistörler uygulamaya girmiştir. Daha sonraki aşama, 1970'li yıllarda, ayrık elemanların yerine tümleşik elemanların alması ile gerçekleşmiş ve böylece çok karmaşık denetim sistemlerinin yapılandırılması mümkün olmuştur.

Programlanabilir kontrol sistemleri ya da programlanabilir denetleyicilerin tarihçesi 1968 yılına kadar uzanır. General Motors firmasının Hydramatic bölümünde, esnek olmayan ve maliyeti yüksek röleli denetim sistemleri yerine kullanılabilecek, bilgisayar temelli, esnek ve endüstrideki mühendislerce kolayca programalanabilecek

ve bakımı yapılabilecek bir denetim sistemi tasarlanmıştır. Sonuç olarak ortaya çıkan cihazlar, sadece rölelerin yerini almakla kalmadılar, ancak, uygulama alanları tekrarlı işlemler yapan makina ve süreçlerle sınırlı kaldı. Çünkü bunlar sadece açık/kapalı denetimi yapabilecek yeteneğe sahiptiler.

1970-74 yılları arasında mikro işlemci teknolojisindeki ilk gelişmelerle birlikte programalanabilir denetleyicilerin esnekliği ve akıllılığı arttı. Operatörle etkileşim, aritmetik işlem, veri üzerinde işlem ve bilgisayarlarla iletişim gibi yetenekler programlanabilir kontrolörlerin uygulama alanlarına yeni boyutlar ekledi. Cihazların bir ekran aracılığı ile, alışılagelmiş röle sembolleri kullanarak kolaylıkla programlanabilmesi mümkün oldu. Aritmetik işlem yeteneği ve daha gelişmiş komut setleri programlanabilir kontrolörlerin sayısal data veren duyargalarla doğrudan kullanılabilmelerini ve bunlardan gelen verilere dayalı mantık ve sıralama işlemlerinin yapılabilmesini sağladı.

1975-79 yılları arasında ise gerek donanım gerekse yazılım açısından yeni ilerlemeler kaydedildi. Bunlar arasında daha yüksek bellek kapasitesi, analog denetim, konumlandırma denetimi (position control) gibi yetenekler sayılabilir. Daha yüksek bellek kapasitesi daha büyük uygulama programlarının, değişken koşullar altında farklı reçetelerin kullanılabilmesine olanak sağladı. Analog denetim kolaylığı ise açık/kapalı denetimin büyük bir eksikliğini ortadan kaldırdı ve analog verilerin doğrudan programlanabilir kontrolöre verilebilmesi ile analog denetimin, kontrolör içindeki açık/kapalı denetim ile etkileşerek yapılmasını sağladı.

Konumlandırma denetimi yeteneği adımlayıcı motor çıkışı ve kodlayıcı geribeslemesi ile gerçekleştirildi. Böylelikle süreç (process) kontrolünde gereken bir başka özellik de sağlanmış oldu.

Donanımda kaydedilen bu gelişmeler yazılımda da paralel ilerlemeleri gerektirdi. Analog denetim, konumlandırma denetimi gibi yeteneklerin kolaylıkla kullanılabilmesini sağlayacak, bilgisayar dillerine benzer diller geliştirildi. O zamana kadar kullanılan röle türü veya boolean cebri türündeki komutlarla bu yeteneklerden yararlanmanın çok zor ve hatta olanaksız olacağı görülmektedir.

1980'li yıllarla birlikte PLC endüstrisinde yeni teknolojik gelişmeler kaydedildi. "Bit Slice" teknolojisinin kullanımı ile daha hızlı bir tarama yapılabilmesi, çok az sayıda röle kullanan sistemler yerine kullanılabilecek düşük fiyatlı PLC'lerin piyasaya sürülmesi, mikroişlemci temelli zeki giriş/çıkışla dağıtılmış işleme ("distributed processing", örneğin, PID ve ASCII iletişim arabirimleri), ısı ve basınç ölçen duyargalar (thermocouple ve strain gauge) doğrudan PLC'ye bağlanmasına imkan veren arabirimler donanım açısından kaydedilen ilerlemelerden bazılarıdır. [1]

BÖLÜM 2

KONTROL SİSTEMLERİNİN TEMEL UNSURLARI

Makina ve süreç (process) kontrolünü sağlamak üzere bir sistem kurulurken çeşitli disiplinlerden faydalanır. Özellikle : makine, elektrik, elektronik, endüstri ve işletim mühendisliği konularının aynı çerçevede birleştirilmesi gerekliliği ortaya çıkar.

Kontrol sisteminin temel unsurları, kontrol mühendisliğinde beş köşe adı verilen:

i- Şebeke

ii- Duyargalar

iii- Hareketlendiriciler

iv- İşlemciler

v- Yazılım ' dır.

i- Şebeke: Bir kontrol uygulamasında enerji (elektrik, pnömatik,hidrolik) veya bilgi (sinyal) iletimini sağlayan bağlantı sistemidir. Şebeke, besleme gerilimi için elektrik kablosu; hava ve yağ beslemesi için hortum, ince cidarlı boru ve ek parçaları; hava ve yağ hazırlama üniteleri olabileceği gibi, sesör ve hareketlendiriciye giden elektrik kablosu ve cam elyaf kablo (fiber-obtic) olabilir. Aynı sistem içerisinde bu tür şebekelerin bir veya daha fazlası yer alabilir.

ii- Duyargalar (sensors): Sistemin duyu oırganaları olan bu elemanlar saha ile ilgili bilgileri toplamak için gereklidir. Kontrol sistemi ancak çevreden bilgi toplayabildikçe çalışabilir. Sistemi oluşturan diğer bileşenlere göre genellikle çok daha küçük ve ucuz olmalarına rağmen, birinin devre dışı kalması, bir tesisin çalışmasını durduracak kadar önem taşır. Modern endüstriyel otomasyonda çoğu zaman: optik sensörler, endüktif sensörler, basınç sensörleri, manyetik sensörler, mekanik sensörler ve kapasitif sensörler kullanılmaktadır. Çok değişik uygulamalara yönelik olarak 1000' den fazala tipte sensör mevcuttur.

iii- Hareketlendiriciler (actuators): Aktif eleman adı verilen hareketlendiriciler; hareket ettirme, aktarma, besleme, döndürme, taşıma, çevirme ve kilitleme işlemlerini kumanda eder. Endüstride yerleşmiş olan terminolojide hareketlendiriciler genellikle;

- valfli/solenoid valfli silindirler (dönel/doğrusal hareketli silindirler)

- elektrik motor kumandalı lineer eksen veya pnömatik/servopnömatik kumandalı

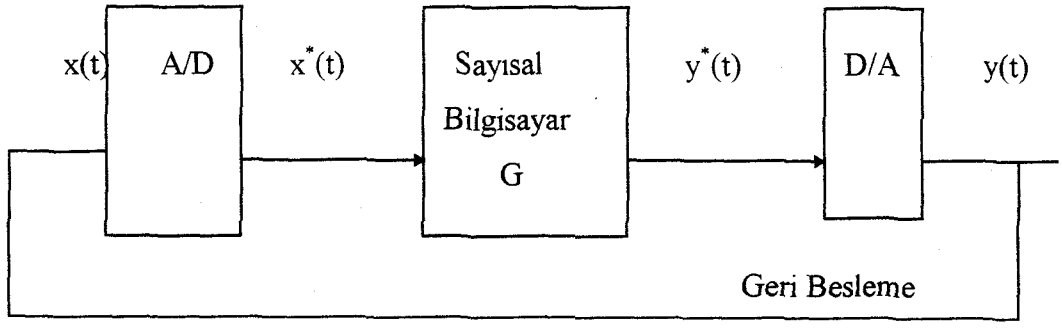
olmaktadır.

iv- İşlemciler (processors): (Duyargalardan) bilgi ve (bir yazılım programı ile) süreç (process) bilgisi alabilen ve valf/silindir kombinasyonundan oluşan bir hareketlendiriciye bilgi gönderilebilen sistemdir. Esas itibarıyla bir işlemci: bir bilgisayar sistemi ya da bir PLC (programlanabilir mantık kontrol) birimidir.

v- Yazılım: Bir programlama biçimidir. Yazılım programı kontrol sisteminde yer alan devrelerin gerçek bilgilerini içerir. Yazılım programı; gerçek uygulama programaları için kullanılabilmekle birlikte; programalanabilir mantık kontrol birimlerine eğitim ve program geliştirme amacına yönelik bir araç olarak da kullanılabilir. Gerek devre dizaynı ve simülasyonu, gerekse verimli eğitim amacıyla hazırlanmış yazılım araçları mevcuttur. [2]

2.1. Sayısal Kontrol Esasları

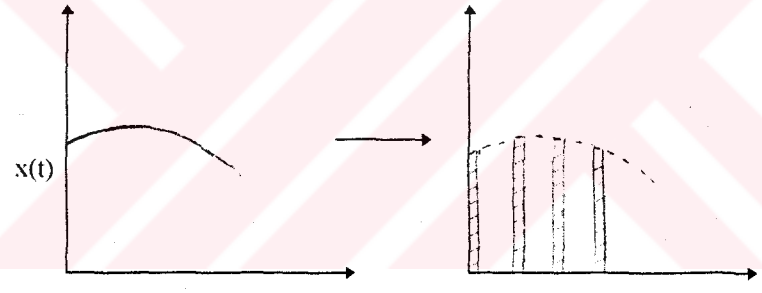
Bir sayısal kontrol sistemini temel olarak Şekil 2.1. deki blok diyagramı ile ifade edilebilir.



Şekil 2.1. Sayısal kontrol sistemi blok diyagramı

x(t) sistem giriş büyüklükleridir. Geri besleme yolundan gelen işaretlerde sistem giriş büyüklüklerine dahil edilebilir. Set points veya genel olarak giriş işaretleri çoğunlukla analog biçimdedir. Girişte bulunan A/D bloğu x(t) işaretini $x^*(t)$ sayısal işaretine dönüştürür. Sayısal bilgisayar çoğunlukla G(z) transfer fonksyonuna sahip bir sayısal kontrol kuralını uygular. Sayısal çıkış işaretleri $y^*(t)$ yine bir D/A çevirici ile y(t) analog işaretlerine dönüştürülebilir.

Sayısal kontrol sisteminde önemli bir yer tutan A/D çevirici bloğunu daha iyi anlamak için örneklenmiş işaret kavramına daha yakından bakmak gerekir. Şekil 2.2 de sayısal kontrol sisteminin analog giriş işaretlerini ve aynı işaretin örneklenmiş şekli görülmektedir. Örnekleme peryodu Tve örnek alma süresi τ saniyedir. Aslında τ süresi örnek alma peryodu T yanında ihmal edilecek kadar kısa olduğundan örneklenmiş işaret genliği giriş işareti ile orantılı impulse dizisidir.



Şekil 2.2. Sürekli işaret ve örneklenmiş işaret

İdeal örnekleme işareti $\delta_T(t)$

$$\delta_T(t) = \sum_{n=-\infty}^{+\infty} \delta(t - nT)$$

biçimiyle tanımlanan $\delta(t)$ birim impulse ve $\delta(t - nT)$ ise $t= nT$ ye ötelenmiş impulse'dır. Çıkış işareti

$$x^*(t) = x(t) \cdot \sum_{n=-\infty}^{+\infty} \delta(t - nT)$$

şeklinde düşünülebilir. x(t) fiziksel bir işaret olduğundan $t<0$ için $x(t)=0$ ve

$$x^*(t) = \sum_{n=0}^{+\infty} x(n).T \cdot \delta(t - nT)$$

yazılabilir. Fourier dizisi açıldığında

$$\delta_T(t) = \sum_{n=-\infty}^{+\infty} C_n e^{jnw_s t}$$

ve

$$C_n = \frac{1}{T}\int \delta_T(t)\, e^{-jnw_s t}\, dt$$

bulunur. w_s örnekleme frekansı olup $2\pi/T$ rad/s dir. $\delta_T(t)$

$$\int_0^T \delta_T(t)\, e^{-jnw_s t}\, dt = 1$$

olarak tanımlandığından $C_n = 1/T$ dir ve

$$\delta_T(t) = 1/T \sum_{n=-\infty}^{+\infty} e^{jnw_s t}$$ şeklindedir. Şekil 2.3.den

$x^*(t) = \delta_T(t) \cdot x(t)$ olduğundan

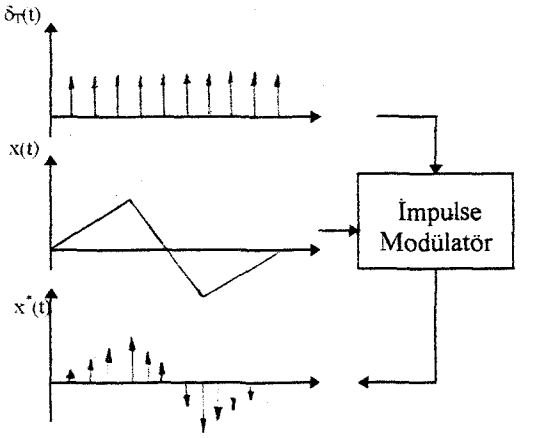
$$x^*(t) = 1/T \sum_{n=-\infty}^{+\infty} x(t)\, e^{jnw_s t}$$

bulunur. Bu ifadenin Laplace dönüşümü

$$x^*(s) = L[x^*(t)] = 1/T \sum_{n=-\infty}^{\infty} x(s - jnw_s)$$

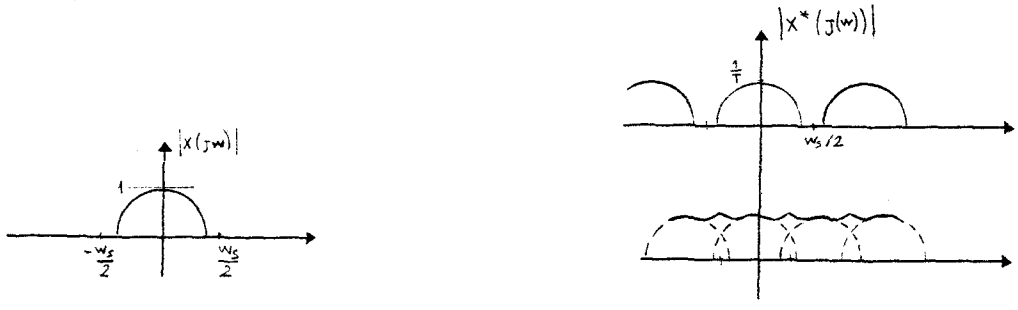
$$x^*(jw) = 1/T \sum_{n=-\infty}^{\infty} x[j(w - nw_s)]$$

olarak bulunur.



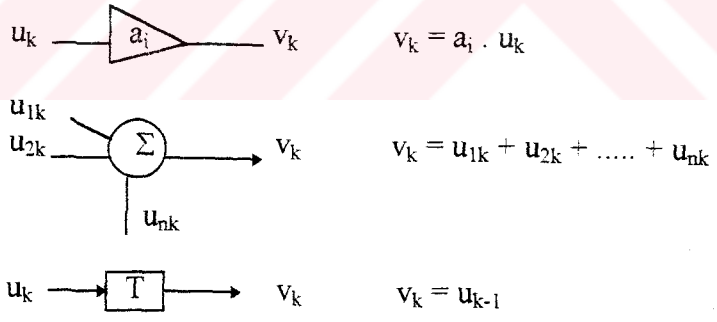
Şekil 2.3. Örneklenmiş işaretin elde edilmesi

Son ifadeden x(t) işaretinin örneklenmiş halinin frekans spektrumu jws aralıklarla tüm frekans alanına yayılmıştır. Şekil 2.4.'den örneklenmiş bir işaretin orjinal işaretin taşıdığı bilgiyi içermesi için minimum örnekleme frekansının $w_s > 2w_m$ olduğu görülür. Bu kriter Nyquist kriteri olarak bilinir.



Şekil 2.4.Minimum örnekleme frekansı

Sürekli zaman kontrol sistemlerindeki elemanlar yerlerini çarpma, toplama ve geciktirme elemanlarına bırakırlar. Ayrık zaman kontrol sisteminin bu elemanları:



diyagram ve ifadeler ile belirlenir. Sürekli zaman sistemlerinden yukarıda tanımlanan elemanları içeren ayrık zaman sistemine geçişe bir örnek olarak, kontrol teorisinde "durum denklemlerine geçiş" adı verilen yöntem gösterilebilir.

Durum denklelerinin basit biçimi:

$[\dot{x}] = [A]\,[x] + [B]\,[u]$ şeklindedir. Matris gösteriminde durum değişkeni sayısı n ise; [A] nxn biçiminde bir matristir. $[\dot{x}]$ ifadesi ayrık zamanda

$\dot{x}(t) = \frac{x(t) - x(t-1)}{\Delta t}$ biçiminde tanımlanabilir.

Bu ifade durum denklemlerinde yerine konulduğunda

$$x(t) = x(t-1) + \Delta t\,[A]\,x(t) + [B]\,u\,\Delta t$$

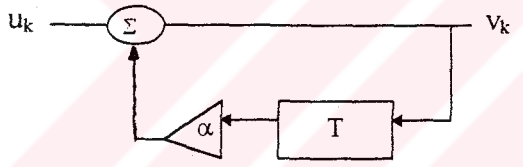
elde edilir. Bu ifadede kaynak fonksyonlarından gelen $[B]\,.u\,\Delta t$ yerine u_k konulunca

$x(t)(1-\Delta tA) = x(t-1) + u_k$ elde edilir ve bu ifadede $(1-\Delta tA)$ yerine $1/\alpha$ konulursa

$x(t) = \alpha\, x(t-1) + u_k$ ve t=k yazıldığında

$x_k = \alpha\, x_{k-1} + u_k$ ayrık ifadesi bulunur.

Ayrık zamanda elde edilen bu ifadenin blok diyagramı temel elemanlar kullanılarak Şekil 2.5. teki gibidir.



Şekil 2.5. Sistemin blok diyagramı

Sürekli zaman kontrol sistemleri ve transfer fonksyonlarıile ayrık zaman kontrol sitemleri arasındaki ilişki z dönüşümüyle açıklanabilir. Sürekli zamanda x(t) işaretinin z dönüşümü

$$x(z) = \sum x(kT)\, z^{-k}$$

olarak tanımlanır.

Çözüm yöntemlerinin ortak noktası sürekli zamandaki s domeninden ayrık zamanda z domenine olan geçiştir. Genelde bu geçiş

$$z = e^{st}$$

şeklinde bir bağıntı ile sağlanır.

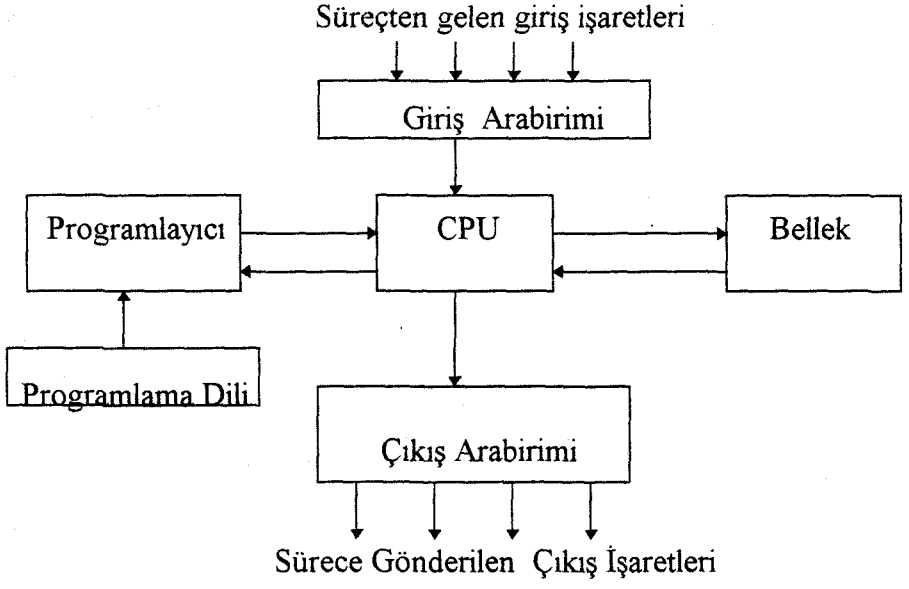
Ayrık zaman sistem elemanları kullanılarak ve transfer fonksiyonundan gidilerek, klasik P, PI, PD, PID kontrol prensiplerini gerçekleyen sistemler kurulabilir. Modern kontrol teorisinde ayrık zamanda korelasyon analizi, Fourier transformları, spektrum analizi günümüzde haberleşme ve kontrolda yoğun biçimde kullanılmaktadır.

BÖLÜM 3

PROGRAMLANABİLİR MANTIK KONTROL ELEMANLARI

Genel olarak bir programlanabilir denetleyici, belleğindeki programın akışı içinde, girişleri (ki bunlar kontrol edilecek sistemin denetim veya geribesleme işaretleridir.) okuyup programda istenen denetim işaretlerini üreten ve çıkışlara yazan özel amaçlı bir mikrobilgisayardır. Amerika'da National Electrical Manufacturers Association (NEMA) kuruluşu programlanabilir kontrolörü şu şekilde tanımlar: Bir programlanabilir denetleyici, makina ve süreçleri denetlemek için, mantık, sıralama (sequencing), zamanlama, sayma ve aritmetik gibi bazı belirli işlevelri gerçekleştirmeyi sağlayacak komutların depolandığı programlanabilir bir belleği olan bir sayısal elektronik araçtır. [6]

Bütün PLC'ler, büyüklük, karmaşıklık ve fiyat gibi faktörlerden bağımsız olarak Şekil.3.1.'de gösterilen bazı temel kısımlardan oluşurlar. Bunlardan bazıları donanım birimleridir, bazıları ise programlanabilir kontrolör yazılım veya programlarının işlevsel özelliklerini yansıtır.



Şekil 3.1. Programlanabilir kontrolcünün temel unsurları

3.1. Programlanabilir Mantık Kontrol Elemanlarında Donanım Mimarileri

3.1.1. Giriş/Çıkış Modülleri ve İşlemciler

PLC teknolojisindeki temel değişiklikler ve gelişmeler açısından en somut yenilikler, donanım mimarisinde "akıllılık" tanımının giriş/çıkış (I/O) modüllerine doğru yaygınlaşmasında görülmektedir. I/O sistemleri PLC'lerin şebekeleşmiş iletişim ve arıza teşhis imkanlarıyla sistemin temel unsurlarındandır. Günümüzde giriş/çıkış mimarisi alışılagelmiş yapının dışına çıkmaktadır. Bu yönde başlıca iki eğilim dikkati çekmektedir.

- daha fazla iletişim olanağı

- daha ileri düzeyde akıllılık

Tek bir nokta başına iletişim imkanının maliyeti düştükçe, uzak giriş/çıkış yapıları bundan mutlaka etkilenecektir. Bugün geçerli olduğu şekliyle I/O modülleri grup grup ele alınmayacaktır. Giderek azalan sayıda sinyal işleyen I/O birimleri sahaya yayıla bilecektir.

Bu gelişmeye paralel olarak saha enstrümanlarının durum, ölçüm ve arıza teşhis raporlaması yapabilen "akıllı" modellerinin yaygın kullanımı da neredeyse tek tek noktasal bazda uzak I/O sistem mimarisine doğru gidişi hızlandırmaktadır.

Bir yandan da bu uç noktalara yüklenen "akıllılık" düzeyi giderek artmaktadır. Başlangıçta sadece sinyal şartlandırma, mühendislik birim çevrimi, teşhis ve benzeri özelliklerle ifade edilen artık çok daha yüksek seviyeli fonksiyonel bir takım halini almakta; kontrol algoritmalarındaki temel yapılar (PID kontrolü gibi) dahi bu tanımın içinde kalmaktadır.

Doğal olarak bu tür gelişme PLC'lerin işini yapabilecek diğer alternatif donanımları da gündeme getirmektedir. Örneğin; çeşitli yazılım imkanlarıyla donatılmış bir kişisel bilgisayar (PC), programalnabilir mantık ünitesinin fonksiyonlarını yerine getirebilir. Son yıllarda "Soft-PLC" denilen türde PC'ler kullanıma sunulmuştur. Ülkemizdeki kontrol uygulamalarında, çoğu kez bu durum PLC ve PC'lerin doğrudan karşılaştırılmasına yolaçmaktadır. Bu durum güvenilirlik ve

gerçek-zamanlı (real-time) olmak bakımından değerlendirilirse: PLC'lerde oluşan hataların, anlaşılabilir veya tahmin edilebilir şartları dahilinde olduğu görülür. PC'lerde ise hata oluşumunu algılamak daha zor ve belirsizdir. (Ancak bu proje dahilinde üzerinde durulan VMEbus tabanlı mikrobilgisayarda gerçek-zamanlı ve çok-görevli (multi-tasking) OS/9 işletim sisteminin kullanılması bu tür olumsuzlukları bertaraf etmektedir). PC programaları bir yerde kilitlenebilir veya bloke olabilir. PLC'lerde ise; her taramada kendi durumlarını yeniden teşhis etme imkanları olduğundan, böyle bir durum meydana gelmez.

3.1.2. PLC Arabirim Üniteleri

PLC'lerin alışılagelmiş şekilde yapılan kablo bağlantıları, büyük ölçüde montaj ve devreye alma masrafına yol açmaktadır. Bu yüzden PLC arabirimleri ile kullanıcıya giriş/çıkış seviyesinde konforlu ve hızlı bir şekilde bağlantı imkanı sağlanır. Bu arabirim elemanları piyasadaki PLC markalarına uygun olarak bulunmaktadır.[5]

PLC sinyallerinin iletişimi, çift sözcük (doubleword) modlu 40 pinli yassı bant konnektör ile veya bir byte modlu 4 adet 10'lu yassı bant konnektörle yapılır.

PLC ön adaptörü, PLC ile mekanik olarak bağlanır ve 32 giriş/çıkış sağlar.

Pasif arabirimler, durum göstergeli veya göstergesiz sinyal arabirim modüllerinden, 3 iletkenli (initiator) terminallerine kadar uzanan ürünleri kapsar.

Aktif sinyal bağlantıları, çoklu optokapl veya röle kuplajları ile yapılır. Pahalı PLC güç kaynağı modülleri yerine, opyo veya röle kuplaj modülleri kullanılabilir.

PLC arabirim üniteleri klemens (TS32) ve otomat (TS35) raykarına monte edilebilir.

PLC arabirimlerinin kullanımı ile:

- Bağlantı hataları azaltılır.
- Fişli-konnektör kullanımı ile kuplaj işlemi en aza indirilir.
- Devreye alma süresi azalır.

- Montaj ve servis işlemleri açık ve kısa tanımlamalardan dolayı kolaylaşır.

3.2. Programlanabilir Denetleyicilerde Yazılım

Tasarlanan ya da hazır kullanılan kontrol donanımları üzerinde kişisel bilgisayar programları yürütülür. Kontrol programları aşağıda sıralanan yöntemler kullanılarak gerçeklenebilir.

i- Simgesel dil

ii- Yüksek seviyeli dil

iii- Uygulamaya yönelik diller

iv- Paket programlar

- Makrolar
- Menüler
- Grafikler

Özellikle işlem hızının yüksek olduğu uygulamalarda simgesel dil kullanılır. Simgesel dilde hazırlanan program derlenerek makina kodu elde edilir. Simgesel dilde program yazma belirli bir deneyim gerektirir. Her simgesel dilde yazılan program en kısa (veya en hızlı) kod değildir. Yüksek seviyeli bir dilin ürettiği kod daha kısa olabilir. Uygulamalarda, donanımı doğrudan yönlendiren alt düzey yazılımı simgesel dilde hazırlanır ve genellikle tüm programın küçük bir bölümü olur. Programın kullanıcıya yönelik kısmı ise yüksek seviyeli dilde hazırlanır. Yüksek seviyeli dillerde donanımı yönlendiren komutlar yer alır. Böylelikle yüksek seviyeli dilin yeteneklerini programı geliştirmede kullanabiliriz. Ancak programın yapısı karmaşık ise programı geliştirme işlemi güçleşir ve uygun olmayan çözümler sonucu, işlem süresi uzar. Bu durumda uygulamaya yönelik özel bir dil kullanma, hem program geliştirme süresinin kısalmasında hem de hızlı kod üretilmesinde yaralı olmaktadır.

Kontrol programlarını geliştirmede önemli bir sorun mevcuttur. Bu sorun, kişisel bilgisayar ve ek donanım yapılarının bilinmesinin zorluğudur. Kullanıcı, ek donanıma ilişkin her türlü alt düzey yazılımı hazırlamalıdır. Bu ise türlü bilgi

gerektirir; donanım bilgisi ve bunlara ilişkin etkin program kodu üretme tekniği. Bu güçlüğü gideren paket programlar ise, genellikle aynı firmanın geliştirdiği donanımlar üzerinde işlemlerini yaparlar. Esnek yapılı donanımlar kişisel bilgisayara isteğe bağlı modüller olarak yerleştirilmektedir. Bu donanımlara alt düzeyden erişen program parçaları (makrolar, alt programlar, kütüphane programları) hazırlanmıştır. Bunların kullanımı ile program geliştirme süresi kısalır. Paket programların gelişmiş türlerinde ise menüler kullanıcının uygulama yazılımının tümünü hazırlamasına olanak tanır. Böylelikle daha az bilgi ile sonuca gidilebilir. Özellikle lojik denetlemeye yönelik olarak grafik kullanan paketler mevcuttur. Bu paketlerde lojik işlemler ve değişkenler grafik simgeleri olarak ekranda yer alır. Kullanıcının belirlediği simgeler arasında bağlantıları kurmasıyla program geliştirilmiş olur.

3.2.1. Programlama Dili Standartlaşma Çalışmaları

1979 yılında başlayan çalışmalarla PCL'ler için çeşitli standartların oluşturulmasına başlanmıştır. Bu amaçla Uluslararası Elektronik Komitesi'nin (IEC) görevlendirdiği bir uzmanlar grubu ilk raporlarını 1982 yılında hazırlamışlardır. Ancak konu çok boyutlu olduğundan daha sonra standartların ayrı ayrı ele alınmasına karar verilmiştir. IEC-1131 koduyla bilinen bu çalışma

i- Genel Bilgi

ii- Ekipman ve Test Gereksinimleri

iii- Programlama Dilleri

iv- Kullanıcı Klavuzları

v- İletişim

konularını kapsamaktadır. Programlama dilleri için on yıldır sürdürülen çalışmalar 1992 yılında IEC'nin yayınladığı 1131-3 standartlarıyla son halini almış durumdadır.

Bu standart LD (Ladder Diagram= Merdiven Diagramı) ve FBD (Function Block Diagram = Fonksiyon Blok Diagramı) grafik dillerini ve ayrıca IL (Instruction List = Talimat Listesi) ve ST (Structured Text= Yapısallaştırılmış Metin) metin dillerini tanımlamaktadır. 1131-3 aynı zamanda yapısallaştırılmış programlamayı,

kuramsal veri tiplerini ve data ve prosedür kalıplarını da içermektedir. Ayrıca bu standart, uygulama programının değişik bölümlerinin, değişik dillerde yazılmasını ve ortak bir yürütme programına irtibatlandırılmasına da imkan tanımaktadır.

Bu standartlaştırma çalışmaları hem kullanıcılara hem de PLC üreticilerine yarar sağlayacaktır. Kullanıcılar standartların tarif ettiği programlama dilleri sayesinde tek bir eğitim programı çerçevesinde değişik üreticilerin PLC'lerinde progrtam yapabilme imkanına kavuşacaktır.

Mevcut programlama dilleriyle kazanılmış tecrübeler yeni standartların daha iyi anlaşılmasına ve yazılımın özelliklerinin daha iyi kavranmasına destek olacaktır. Sistem kurucuları, bakımcılar v.b. kadrolar bu standartlar sayesinde bilgili ve muktedir bir şekilde çeşitli donanımların tasarımlarını, uygulamalarını yapabileceklerdir.

IEC-1131 global düzeyde yaygınlaştığında beklenen gelişmeler şöyle sıralanabilir:

- Almanya ve Fransa gibi Avrupa ülkelerinde kullanılan ulusal standartlar yerini IEC-1131'e bırakacaktır. Dolayısıyla bu ülkelerdeki PLC üreticileri de üretimlerini uyumlu hale getireceklerdir.
- Kullanıcılar eğitim ihtiyaçlarını sadece üreticilerden değil, üçüncü kişilerden örneğin sistem evlerinden de karşılayabileceklerdir.
- Standartlaştırılmış programlama dilleri kendine özgü dillerin eğitim bağımlılığını kaldıracağından; tesislerde yeni uygulamalarda kullanılacak donanımlarda gerçek rekabet; fiyat, performans ve diğer kabiliyetler açısından geçerli olacaktır.
- Gelecekteki kullanıcıların değişik sistemler arasında programları değiş tokuş ettiği ortak kütük formatında bir yapıya kavuşmak mümkün gözükmektedir. Böyle bir formatla, özel uygulamalar için fonksiyon bloğu paketleri geliştirilebilir ve mevcut sisteme kolaylıkla entegre edilebilir. Dolayısıyla büyük ve geniş sistemler paketlenmiş alt sistem çözümlerinin kombinasyonu şeklinde hayata geçirilebilirler. Bu noktadan itibaren yazılım, üzerinde çalıştığı donanımdan bağımsız hale gelecektir.
- Fonksiyon blokları uygulamasının dağıtılmış kontrol sistemi alanına uzanması halinde IEC'nin 1131-3 ile oluşturduğu kavrama uygun saha-iletişim hatlarının

(fieldbus) özelliklerinin bilinmesi gereklidir. IEC'nin field-bus standartı SC65C/WG6 olup kullanıcıların, tanımlanmış iletişim imkanlarının uygulama yazılımıyla nasıl bütünleştirileceği hususunu anlamaları gerekir. Böylelikle çoklu kontrol cihazlarının, merkezi işlemcilerin ve akıllı sensörlerin bulunduğu bir dağıtılmış kontrol sisteminde satndartlaştırılmış bir dille yazılan uygulama programlarının aslında sistemi "tek-vücut" haline getirdiği düşünülebilir.

3.2.2. Yeni Kontrol Yazılım Paketleri

Kontrol yazılım paketleri genel olarak, kontroldonanımıyla (DCS, PLC'ler, akıllı tip G/Ç üniteleri) operatör istasyonları ve göztimsel (supervisory) fonksiyonları bağdaştıran unsurlardır.

Gerek kontrol mimarisindeki gerekse de iletişim ve şebekeleşme teknolojilerindeki son gelişmeler, kontrol yazılım paketlerini de etkilemiştir. Herşeyden önce, bu gelişmeler ışığında sahanın her yerine yayılan hesaplama ve kontrol fonksiyonlarını gerçekleştiren mikro-işlemci veya merkezi işlemcileri (CPU) birarada ahenkli bir şekilde çalıştırabilmek zorunluluğu doğmaktadır. Bütün bir sistemdeki amaç ve görevleri organize edecek ve daha sonra parça parça bu görevlerin sahadaki hangi CPU tarafından yerine getirileceğine karar vererek; emirlerini iletişim kanalı aracılığıyla transfer edecek kontrol yazılımı paketlerine ihtiyaç vardır. Günümüzde işin en zor tarafı kontrol donanımlarının topladığı değişik standartlardaki verileri, bir başka deyişle; prosesin parçası olan cihazların davranışlarını algılayıp tanımlayabilen yazılımları tasarımlamaktadır. Çünkü, sistemler çok çeşitli donanımlardan oluşabilmektedir. En azından yazılımcılar bu imkanı yartacak şekilde düşünmek zorundadırlar. Bu imkan ise nesne yönelimli (object oriented) programlama sistemlerinin yaygınlaşması ve bu nesnelerin de standartlaşması ile sağlanacaktır. Böylece bir sistemin tüm veri tabanı, sistemi oluşturan ayrı ayrı işlemcilerin ortak malı olabilecektir.

Burada en önemli husus; sadece otamatik kontrol işlevlerinin yerine getirilmesi değil bütün bu tür ekipman/makina bazında sağlanan fonksiyonel işlemlerle birlikte

tüm bir tesis denetiminin hammadde stok sahasından satınalmasına kadar her türlü girdi/çıktısıyla yapılabilmesidir.

Bu amaçla, en üst seviyedeki MIS (Management Information System) yazılımlarıyla olsun ya da MRP (Manufacturing Resource Planning) gibi üretim kaynakları planlaması yapabilen yazılımlarla olsun, üretim alanında aktarılarak oluşturulan istatistiki verilerin, bütünleşmiş bir bilgisayar ağı içerisinde "on-line" kullanılabilmesi için kontrol yazılımlarında kullanılan "nesne" tanımlarının standartlaştırılmasına; dolayısıyla çok çeşitli platformlarda paylaşılabilir hale getirilmesine çalışılmaktadır.

Bu çalışmalar IBM, DEC ve HP gibi büyük şirketlerin oluşturduğu OMG (Object Management Group - Nesne Yönetimi Grubu) tarfından sürdürülmektedir. Diğer birçok kontrol yazılımı üreten firmalar bu grubun oluşturmaya çalıştığı CORBA (Common Object Requester Broker Architecture - Ortak Nesne Aracı Mimarisi) standartına uyugn paketler hazırlamaktadırlar. Böylece üst seviyede kurulacak ya da mevcut olan bilgisayar destekli ileri kontrol sistemlerine kolaylıkla taşınabilir/aktarılabilir veri paketleri hazır olacaktır.

Günümüzde bu imkanı yaratacak bazı yazılım teknikleri ve yazılım mimarileri de artık yaygın biçimde kullanılmaktadır. Bunlar arasında en çok bilinen standartlar windows altında çalışan DDE (Dynamic Data Exchange - Dinamik Veri Alışverişi), DDL (Dynamic Link Library - Dinamik Kütüphane Hattı) ve OLE (Object Linking and Embedding - Nesne Bağlantısı ve Yerleştirimi) metodlarıdır. Bu metodların yapısal içerikleri ayrı bir konu olmakla birlikte ne şekilde kullanıldıklarını ve yukarıda bahsedilen amaca nasıl hizmet ettiklerini somutlaştırmak için şöyle bir örnek verebiliriz. Bir kontrol sisteminde kullanılan çeşitli (birden fazla) uygulama programları olabilir. Bu uygulama programlarından herhangi birinde kullanılan bir "nesne" tanımlı fonksiyon ki; herhangi bir metin ya da veri içeriyor olabilir, bir başka uygulama programı tarafından da kullanılması gerekiyorsa aynı fonksiyonun yeniden yaratılmış olmasına gerek kalmaksızın OLE sayesinde ilgili programlara ulaşılır (linking) ve diğer programdaki yerine de aktarılır (embedding). Şebekeleşmiş bir sistemde bu tür imkanların yaratacağı esneklik ise hayal dünyamızın sınırlarını zorlamaktadır. PLC'ler tüm bu gelişmelerin ortasında önemli yapı taşlarıdır. Üretim

alanından bilgiyi toplayan, işleyen ve aktaran özellikleriyle bu gelişmelerin kaçınılmaz birer parçasıdırlar. Özellikle bilginin aktarılmasında kullanılan iletişim yazılımları kontrol yazılımlarındaki bu yeniliklere ayak uydurmak durumundadır. (Aslında DCS'lerde de durum pek farklı değildir). Esasen belli başlı tüm PLC üreticileri geleceği yakalamak amacıyla ISO'nun (Open System Architecture - Açık Sistem Mimarisi) referans modeline uygun ürünlerinin sayısını giderek arttırmaktadır. Teknolojinin bugünkü imkanları bu amaca ulaşmayı kolaylaştırmaktadır.

Gelecekte kontrol sistemlerinin donanım tarafında ne olursa olsun en büyük uğraş, en büyük yatırım ve maliyet yazılım tarafında olacaktır. Hatta öyle öngörüler vardır ki; PLC'lerin operatör istasyonları ve programlama cihazları ile iç içe girerek tek bir ünite halini alması eklenmektedir. [7]

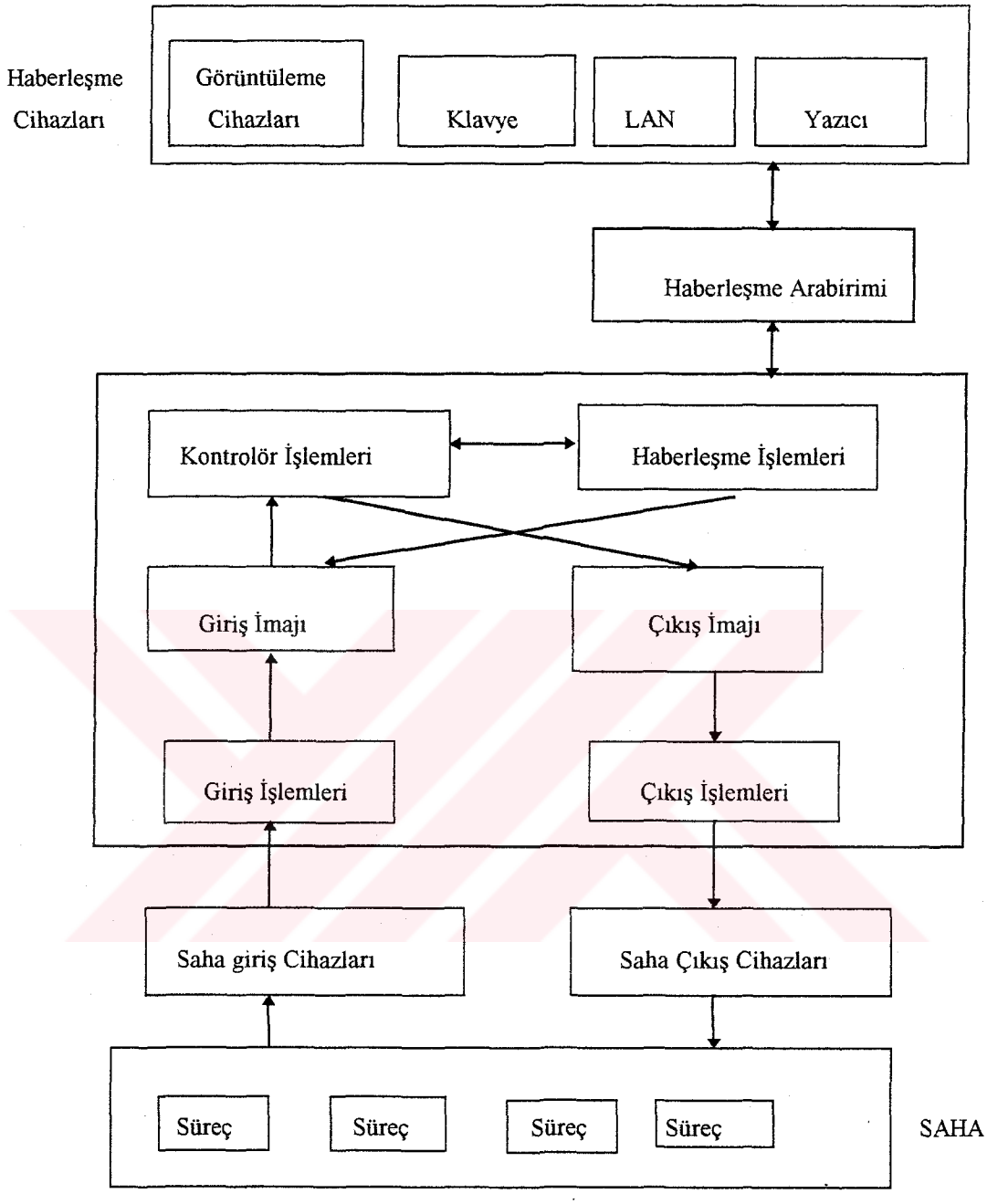
BÖLÜM 4

ENDÜSTRİYEL KONTROL DONANIMLARINDA HABERLEŞME SİSTEMLERİ

Gelişen kontrol mimarisi beraberinde giriş/çıkış ünitelerinin birbirlerine, merkezi işllemcilere ve bilgisayarlara ne ile ve nasıl bağlanabileceği sorusunu da getirmiştir. Bu etkileşim tek yönlü değildir yani iletişim sahasındaki yenilikler de donanımların farklı tasarlanmasına yol açabilmektedir. Endüstriyel bir ortamda her türlü elektriksel gürültüye karşı güvenceli bir işletim temin etmek oldukça güçtür. Mevcut seçenekler içinde en genel tercih: çok noktalı (multidrop veya multipoint) standard iletişim metodlarını kullanan açık ya da kapalı tip protokollerdir. Bunun dışında üreticilerin kendilerine özel iletişim metodları da mevcuttur.

Üreticiler PLC sistemlerindeki iletişim yapılarını daha geniş bir müşteri kitlesine ulaştırmak amacıyla açık protokollere dayalı duruma getirmektedirler. Kurulacak sistemlerde özellikle ülkemizde de olduğu gibi tesis standartlarının bulunmadığı ortamlarda bu çok önem kazanmaktadır. Başlangıçta yapılan seçimler, ileride yeni sistemlerin integrasyonuna engel olabilmektedir. En yaygın çok noktalı iletişim spesifikasyonu RS-485 hattıdır. Hem I/O modülleri hem de işlemciler arasındaki iletişimde kullanılmaktadır. İletişim spesifikasyonu protokoller tarafından tanımlanmadığından dolayı, birbirleriyle uyumlu olmayan çok sayıda RS485 tabanlı sistem mevcuttur.

Diğer taraftan, bir kontrol sisteminde yer alan birden fazla ana işlemcinin bir üst düzeydeki kontrol sistemine bağlanmasını sağlayan şebekelerde durum biraz farklıdır. En önemli fark, şebekelerde iletilen veri bloğunun yapısıdır. Bu alanda en çok LAN (yerel bilgisayar ağları) bazında çeşitli şebeke protokolleri mevcuttur. (Örnek; ETHERNET)



Şekil 4.1. Bilgisayar tabanlı kontrol sisteminde haberleşme işlemleri[4]

Genel olarak PLC'lere bakıldığında iki tür şebeke vardır. İlki daha önce bahsedilen I/O ünitelerini bağlayan şebekelerdir. Bunlar tipik efendi/köle (master/slave) uygulamalarıdır. Küçük veri paketlerinin yüksek hızda ve sık aralıklarla iletilmesini gerektiren saha iletişim hatları (fieldbus) olarak bir genelleme yapabiliriz.

Şebeke tanımı daha çok sistemde birden fazla sayıda bulunan ve değişik görevelri olan PLC ve/veya kontrol ünitelerinin birbirine bağlanmasında kullanmak daha yerinde olacaktır. Bunlar da uygulama açısından daha çok bire-bir (peer-to-peer) kategorisinde şebekelerdir. Bu şebekelerde de son yıllarda hızlı bir değişim ve yenilenme izlenmektedir. Mesela; büyük ve geniş alan şebekelerinde kullanılan MAP (Manufacturing Area Protocol) değiştirilerek hücresel (ekipman) bazında bağlantı sağlayan Mini-MAP protokolü oluşturulmuştur. Zaman-kritik uygulamalarda ve hızlı veri iletişimi gereken durumlar için ise MAP-EPA (Enhanced Performance Architecture) tasarlanmıştır. Diğer tarafta, ETHERNET şebekesinde iletişim uygulama programlarının çözümlerine açık olan mesaj transferi, kütük ulaşımı ve transferi, terminal iletişimi, şebeke yönetimi gibi servislerin bulunduğu 7. iletişim tabakasına MMS (Manufacturing Message Spesification) protokolü yerleştirilerek "danışıksız" (non-contention) iletişim yönetimine (token ring/bus) bir alternatif getirilmiştir.

4.1. Fieldbus Yapısı

Fieldbus'ın, saha bus'ı şeklinde kullanılmasının çok doğru bir karar olmadığını düşünerek, bu tabiri şimdilik orjinal haliyle kullanmak durumunda kalıyoruz. Fieldbus, CIM modelinin en alt seviyesine karşılık gelen, duyarga ve hareketlendiriciler gibi saha elemanlarıyla kontrol birimleri (PLC) arasında sayısal ve iki yönlü haberleşme olanağı sağlayabilen bir haberleşme alt yapısıdır. Aynı zamanda üretim otomasyonu, denetleme ve veri tabanı oluşturma amacıyla kurulan TOP ve MAP gibi üst seviye ağlarına da bağlantı olanağı sağlar.

Fieldbus'ın hizmet verdiği seviye düşünülürse kendi bünyesinde gevşek gerçek zamanlı gereksinimleri (Soft Real -Time Requirements) garantileyecek birtakım özel mekanizmalara da sahip olması gerektiği açıkça görülmektedir. Kısacası bu tür

sistemlerde denetim esnasında yapılan hesaplamaların doğrul olması yanında, sonucun elde edildiği zaman da önemlidir. Sıkı gerçek zamanlı sistemlerde sonuç belli bir süreden önce ve/veya istenilen belli bir anda elde edilebilmelidir. Fieldbus'ın haberleşme alt yapısının da mevcut yapıyı destekleyecek gerçek zamanlı haberleşme yapısına sahip olması gerekir.

Fieldbus taşıt (bus) yapısına sahiptir. Bu yapı ile yukarıdaki tanımda adı geçen tüm birimler bir tek veri hattı ile birbirine bağlanır. bu yüzden kablolama masrafi az; bakımı, arıza tespiti ve kurulumu kolaydır. Teorik olarak hatta yeni bir birim eklemek mevcut birimleri etkilemez ve sonsuz sayıda birim bağlanabilir. Pratikte ise bu sayı tarama (scan) zamanına ve haberleşme yoğunluğuna bağlıdır.

Fieldbus'ın bu yapısının diğer bir avantajı ise yapı olarak hem merkezi, hem de dağıtılmış denetime uygun oluşudur. Yani birimlerin taşıta erişimi tek bir birimin denetiminde (centralised) gerçekleştirilebileceği gibi denetim hakkı sırasıyla diğer birimlere de verilebilir (decentralised-flying master). Herhangi bir arızanın ya da olumsuzluğun yaratacağı sonucun ciddiyeti kullanılan denetim yapısına bağlı olarak değişir. Merkezi denetimde merkezin çökmesi ya da hattın kopması genelde tüm sistemin çökmesi anlamına gelir. Dağıtılmış denetimde ise sadece hattın kopması sorun yaratır. Doğal olarak arızalanan birimlerin hattı etkilememesi için bu birimler "By-pass" edilebilmelidir. Bu aşamada şunu da söylemek faydalı olacaktır. Sistemin güvenilirliği ve performansı üst seviyede (uygulama seviyesinde) ve ağ düzenleme kısmında (network management) sunulan hizmetlere de bağlıdır.

4.1.1. Teknolojik Durum:

Özellikle sayısal denetim teknolojisinin gelişmesi ve fiyatların hızla düşmesi sonucu Fieldbus yapısı içerisinde kullanılan en küçük birim bile akıllı birim haline gelmiştir. Örneğin ölçüm elemanları sadece ölçüm değil artık aynı zamanda Fieldbus'a doğrudan bağlanıp diğer birimlerle haberleşme yapabilmekte, bir kısım denetim ve gözlemleme işlemlerini yürütebilmektedir. Bu akıllı birimler sayesinde artık dağıtılmış denetim daha iyi yapılabilmekte, yük dağıtımı daha iyi sağlanabilmekte ve sonuçta gerçek zaman gereksinimleri daha iyi gerçekleştirilebilmektedir. Günümüzde

Fieldbus'da olduğu gibi dağıtılmış denetim yapısının merkezi denetim yapısına göre getirdiği avantajlar nedeniyle gittikçe daha fazla benimsendiği ve kullanıldığı düşünüldüğünde bunun çok önemli bir özellik olduğu görülmektedir.

Şu ana kadar oluşturulmaya çalışan Fieldbus'larda özellikle açıklık özelliği gözönüne alınarak tasarımlar yapılmış ve ISO/OSI modeline uyumluluğu tercih edilmiştir. Bir başka değişle oluşturulan sistem, herkesin elde edebileceği ve uyabileceği temel özelliklere uygun olarak geliştirilir. Bu sayede diğer firmalar aynı network üzerinde birçok firmanın ürününün kullanılabilmesini sağlamak için herkesin elde edip uyabileceği özelliklere sahip olacak şekilde tasarım yaparlar. Ancak, Fieldbus yapılarında ISO/OSI modelinin sadece birinci (fiziksel seviye), ikinci (veri bağlantı seviyesi) ve yedinci seviyeleri (uygulama seviyesi) gerçekleştirilmiştir. Fieldbus'ın yapısı ve amacından ötürü diğer katmanların kullanılması gereksizdir. Gerçek zamanlı sistem gereksinimleri bu üç seviyenin uygun şekilde tanımlanması ve uygulamaya geçirilmesi ile başarılmaya çalışılmaktadır.

4.1.2. Uluslararası Standartlaşma Çalışmaları ve Standart Belirleme Grupları

Şu ana kadar FIP, Profibus, Interbus-S, Bitbus, CAN, Modbus gibi birçok Fieldbus yapısı oluşturuldu. Bu yapılar değişik ulusal gruplar ve ülkeler tarafından desteklenmekte ve doğal olarak bu gruplar oluşturdukları ya da oluşturmakta oldukları kendi standartlarını uluslararası bir standart olarak benimsenmesini sağlamak için tüm baskılarını sürdürmekteler.

Önceleri ulsal çapta süren standartlaşma çalışmalrı daha sonra uluslararası bir boyut kazandı. Sonuçta "IEC/ISA Joint Committee adında bir komite kurularak bir uluslararası standartın oluşturulması çalışmasına başlandı. IEC/ISA tarafından oluşturulan standarta uyan ürünleri bir arada kullanarak interkama ve ISA fuarlarında gösteriler yapmak, elde edilen tecrübeleri diğer Fieldbus standartlaşma çalışmaları yapan gruplar ile paylaşmak ve sonuçları halka duyurmak amacıyla 1990 yılında uluslararası bir Fieldbus konsorsiyumu (International Fieldbus Consortium-IFC) da oluşturuldu.

Bu satandartlaşma çalışmaları sırasında en büyük çekişme FIP ve Profibus grupları arasında meydana geldi. Her iki grup kendi yapılarının kabul edilmesini sağlamak için büyük baskı yaptı. Uluslararası tanıtımda Fıp grubu açık bir tanıtımla başarılı oldu ve çok farklı ülke ve şirketle ilişki kurdu. Bunun üzerine Profibus grubu birçok kuruluşu yanına alarak ISP adında yeni bir yapı oluşturdu. ISP Fieldbus yapısının en basit bir tanımla IEC/ISA, Profibus, Fip ve HART gibi yapıların bazı kısımların alınıp birleştirilmesi sonucu oluşturulduğu söylenebilir. WORLFIP temel olarak FIP'i değiştirmeyi ve IEC/ISA'nın tanımladığı standartı kullanmayı kabul etmekte. Şimdi gözler bu iki grup üzerinde odaklanmış olup, gelişmeler merakla izlenmektedir. Hangi yapının bu yarıştan galip çıkacağı ise şimdilik belirsizliğini korumakta. Bu gelişmeler 4-20 mA standardının oluşumunu anımsatmakla beraber 4-20 mA "de facto" standart olarak daha kısa sürede oturmuş ve birçok firma tarafından kabul edilmiş ve kullanılmıştı. Şu ana kadar çekişmenin yoğunlaştığı alan daha çok teknik özellikler oldu. Bir başka deyişle, herkes teknik açıdan kendi yapılarının en iyisi olduğunu kabul ettirmeye çalıştı. Ancak bir Fieldbus'ın uluslararası alanda başarılı olabilmesi için birçok üreticinin ürünleri ile o yapıyı desteklemekleri gerekmektedir. Daha da önemlisi değişik alanlarda uygulanması ve yaygınlaşması gereklidir.

4.1.2.1. FIP ve Profibus'ın Temel Özellikleri

Aşağıda sadece FIP ve Profibus ile ilgili bilgiler verilmiş ve karşılaştırmalar yapılmıştır. Yalnız tüm özelliklerini açıklayıp karşılaştırmaktan daha ziyade Fieldbus yapıları için en önemli gerçek zamanlı gereksinimlerin karşılanması açısından ele alınmış ve karşılaştırılmışlardır.

FIP

Destekleyen Firmalar:

Özellikle Fransız ve İtalyanlar'ın kurduğu, FIP klüp tarafından desteklenen FIP, diğer birçok Avrupalı ve Amerikalı firma tarafından da destek görmektedir. Sayısı 125'I aşan firmalardan bazıları şöyle sıralanabilir.

ABB KENT TAYLOR, ALCATEL CETT, CEGELEC I.C. BAILEY ESACONTROL, ASAP Composants, DIGIMETRE, EFISYSTEME, ENDRESS & HAUSER, ENTRELEC, HARMANN & BRAUN, INFA, MERLIN GERIN.

FIP'ın Uyduğu Standartlar:

Genel yapısı UTE C 46 - 601 standardına uygundur. Katmanların uyduğu standartlar ise:

- Fiziksel seviye; C 46 - 604 ve C 46 - 601
- Veri bağlantı seviyesi; C 46 - 603
- Uygulama seviyesi; C 46 - 602 ve C 46 - 606

Network yönetimi ise C 46 - 605 satandartına uymaktadır. Yalnız bu standartların bir kısmı halen "draft standart" halindedir ve çalışmalar sürdürülmektedir. Oluşturulan standartların çoğu test edilmiştir.

FIP'ın Temel Özellikleri:

FIP'in en önemli özelliği "broadcasting"dir. İstasyon isimleri yerine değişken isimleri kullanılır. Her bir müşteri istasyon sürekli taşıtı dinler ve kendisini ilgilendiren bir bilgi görürse onu alır ve değerlendirir. Üretici istasyon ise bilgiyi kimin kullanacağını bilmez ve bilmesi de gerekmemektedir. Görevi sadece gerekli bilgiyi üretmek ve zamanı geldiğinde taşıta vermektir. kimin ne zaman taşıtı kullanıp bilgi dağıtacağını ise Bus Arbitratör, (B.A.) adında bir denetleyici organize eder. Fakat bu bir master-slave ilişkisinden çok farklıdır. B.A.'nın yaptığı sadece bir tarama (scan) tablosu kullanarak doğru zaman ve belli bir süreden önce önemli bilgilerin üretilmesini sağlamaktır. Çoğu Fieldbus yapısında önem verilen konu bilginin belli bir süre içinde üretilmesidir. Fakat dikkat edilirse FIP'te bilginin üretildiği an da önemlidir. Bu da sıkı gerçek zamanlı gereksinimleri karşılamak için çok önemli bir özelliktir. Tarama işleminden arka kalan zamanlarda ise periyodik olmayan mesaj ve proses verilerinin iletimi yapılır. Bir başka deyişle normal tarama zamanları dışında kalan zamanlarda istasyonlar birbiri ile konuşabilir. Aynı zamanda bir istasyon başka bir istasyondan bilgi alışverişi yapabilir. Bu sayede hem merkezi hem de dağıtılmış denetim yapısına sahiptir. Bu sistemin kötü yanı Bus Arbitratörün çökmesi durumunda sistemin de

çökmesi anlamına gelebilmesidir. Aynı zamanda tarama tablosu dinamik olarak değiştirilmediğinden sistemde değişiklik yapmak zorlaşmaktadır.

Profibus

Destekleyen Firmalar:

Siemens, Bosch, Klöckner-Möller, ABB, AEG, Bauer, Danfoss

Profibus'ın Uyduğu Standartlar:

Genel yapısı DIN 19245 standardına uygundur. Katmanların uyduğu standartlar ise:

- Fiziksel seviye; DIN 19245 Part 1
- Veri bağlantı seviyesi; DIN 19245 Part 1
- Uygulama seviyesi; DIN 19245 Part 2

Bu standarların bir kısmı halen "draft standart" halindedir ve çalışmalar sürdürülmektedir.

Profibus'un Temel Özellikleri:

"Master-Slave" yapısı ile merkezi, "Token Passing" yöntemi ile de dağıtılmış denetim yapısına yani hibrit bir yapıya olanak sağlar. FIP temel olarak Broadcasting yapısı üzerine kurulmuş olmasına rağmen Profibus daha çok "Polling" yapısı üzerine kurulmuştur. Aktif ve pasif olmak üzere iki grup istasyon vardır. Aktif istasyonlar daha ziyade denetim elemanlarıdır ve aralarında bir "Token" dolaşır. Sorgulama sadece bu "Master" adı verilen aktif istasyonlar tarafından ve sadece "Token" kendilerine aktarıldığında yapılabilir. Aktif istasyonlar "Token"a bir turda bir kere sahip olurlar ve kendilerine tanınan süre içinde hem kritik hem de normal mesajlama işlemlerini yerine getirirler. "Slave" olarak isimlendirilen pasif istasyonlar ise herhangi ibr aktif istasyon tarafından kendilerinden bilgi sorulduğunda hatta çıkar ve istenilen bilgiyi aktarırlar. Sistemin performansı ve R.T. gereksinimlerinin sağlanması büyük ölçüde "Token"in dönme hızına ve mesajlama protokolün yapısına bağlıdır. Bu zaman istasyonların mesaj transferlerini rahatça yapabilecekleri kadar uzun, gerçek zaman gereksinimlerini karşılayabilecek kadar kısa olmalıdır. Seçilecek bu süre çok önemlidir ve azami değerin seçilmesi gerekir. Eğer "Token" ele geçirildiğinde gereksiz mesajlaşmalar ile

"token" geç teslim edilirse veya bir istasyon gereğinden fazla "Token" beklerse kritik mesajlaşmalar yapılamaz ve sistem zor bir durumla karşı karşıya kalır. Öte yandan dönme hızı sistem kurulumu sırasında hesaplanabilir ve daha sonra sistem çalışırken değiştirilebilir. Bu özellik ise sisteme sonradan "master" veya "slave" eklenebilmesini ve esnek bir yapı oluşmasını sağlar. [8]

BÖLÜM 5

S5-115 U KONTROL SİSTEMİNİN YAPISI

Siemens, SIMATIC S5-115 U programlanabilen kontrolörü dünya çapında çok geniş bir uygulama alanında kullanılmaktadır. bu sistemi meydana getiren ayrılabilir modüllerden her biri özel bir görevi yerine getirir. Bu yüzden sistem, ihtiyaçları karşılayacak şekilde modüller eklenerek genişletilebilir. 3 tipte olan iletişim sistemleri, bilgiyi çoklu kontrol elemanları boyunca iletirler. S5-115 U sistemi operatör panellerini, görüntüleme cihazlarını ve ihtiyaca göre çeşitli programlayıcıları destekler. STEP5 programlama dili ve yazılımla ilgili kapsamlı kataloglar sayesinde programlanması oldukça kolaylaşır. [10]

5.1. Uygulama Alanı

Çok çeşitli endüstriler S5-115 U sistemini kullanır. her otomasyon işi farklı olmasına rağmen, S5-115 U asit açık döngü (open-loop) kontrolünden, çok karmaşık kapalı döngü (closed-loop) kontrolüne kadar çok çeşitli uygulamalar için kullanılabilmektedir.

Bu kontrol sisteminin kullanıldığı alanlardan bazıları şunlardır:

- Otomobil endüstrisi (otomatik delme, montaj, test cihazları, boyama işleri)
- Plastik endüstrisi (Isıl kalıp makinaları, sentetik üretim sistemleri)
- Ağır sanayi (Kalıp ekipmanları, sanayi fırınları, hareketli miller vs.)
- Kimya, ilaç, gıda ve meşrubat sektörü (Karıştırıcılar, fırınlar,...)
- Ulaşım sektöründe (Ürünlerin sınıflandırılmasında, ambarlarda, vinç ve taşıma bantlarında)
- Enerji, gaz ve su dağıtım istasyonlarında

- Bina otomasyonunda (asansör, havalandırma, ısıtma ve ışıklandırma)
- ve paketleme, ağaç işleri, çeşitli makinaların kontrolü, kaynak işleri, acil durum ikaz sistemleri v.s.

5.2. Sistem Elemanları

S5-115 U sistemi ayrı modüller halinde elemanlardan oluşur. Bunlar:

- Güç Ünitesi (PS)
- Merkezi İşlem Birimi (CPU)
- Giriş ve Çıkış Modülleri (I/O)
- Zeki Giriş/Çıkış Modülleri (IP, WF)
- Haberleşme İşlemcileri (CP)' dir.

5.2.1. Güç Ünitesi

Güç ünitesi modülü dışardan sistemi besleyen elektrik gerilimini iç işletim gerilimine çevirir. S5-115 U 24 V doğru akım, 115 V alternatif akım ve 230 V alternatif akım gerilimlerini destekler. Vidalı terminaller vasıtasıyla, gerilim hatları, güç ünitesinin alt kısmına bağlanır. En fazla üç çıkış akımı mümkündür. Kullanılan modüllerin sayısına veya bunların güç tüketimine bağlı olarak, 3A, 7A ve 15A' lık akımlardan biri seçilir.

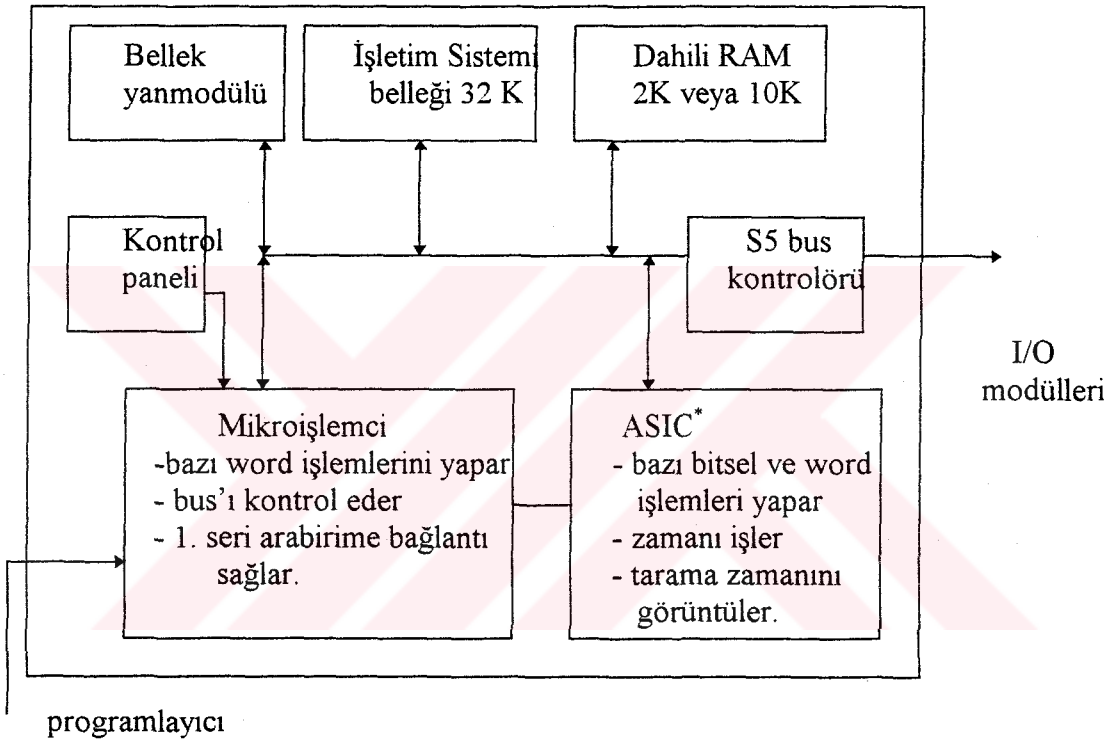
Güç kesintisi ve arızalarında, program belleği, sayaç, zamanlayıcı ve flaglar gibi bazı gerekli bilgiler güç ünitesindeki bir lityum pili sayesinde korunur. Pilde bir arıza olması durumunda ilgili ışıklı diyot (LED) sinyal verir.

5.2.2. Merkezi İşlem Birimi (CPU)

Merkezi işlem birimi, sistemin kontrol programlarını çalıştıran beynidir. Kullanılacağı işlemin gerektirdiği performansa uygun olarak CPU941, CPU942, CPU943, CPU944 modüllerinden biri seçilebilir. Daha güçlü bir CPU seçildiğinde programın çalıştırma süresi kısalacak ve kullanıcı bellek alanı genişleyecektir. CPU941'den 944'e kadar olan modüller, bu CPU'ların işletim sistemleri tümleşik PID

kontrol algoritmalarını kullandığından, PID kontrolü için de kullanılabilir. PID kontrol döngüleri için 100ms'den başlayan örnekleme zamanları mümkündür. Bu işlemcilerle en fazla sekiz tane PID kontrol döngüsü uygulanabilir.

CPU943 ve 944 üniteleri, (her ikisinin de iki adet seri arabirimi vardır) tümleşik donanım saati (integral hardware clock) sayesinded, süreç (process) kontrolüne ek bir takım olanaklar sağlarlar.



Şekil 5.1. CPU 941'in şematik gösterimi (ASIC=application spesific integ.circuit)

5.2.3. Giriş/Çıkış Modülleri (I/O)

Giriş/çıkış modülleri, makinalardaki veya kontrol edilen sistemdeki duyargalar (sensors) ve hareketlendiriciler (actuators) ile kontrol sisteminin bağlantısını sağlayan arabirimlerdir. S5-115 U sisteminin modülleri

- Hızlı yükleme (installation)

- Mekanik kodlama ve
- Geniş etiketleme alanı,

özellikleri sayesinde kullanımı kolaylaştırırlar.

Dijital Modüller:

makinalarda ölçülen gerilim ve akım değerlerini, bir adaptasyon gerektirmeksizin programalanabilir kontrol biriminin mevcut düzeyine uyarlarlar.

Analog Modüller:

Programalnabilir kontrolörün performans derecesi, analog değerleri işlemesi imkanına göre artar. Analog giriş/çıkış modülleri özellikle otomatik seviye ölçümü, sıcaklık ve hız kontrolü gibi kapalı döngü kontrolü uygulamalarında kullanılır.

5.2.4. Zeki Giriş/Çıkış Modülleri

Çok hızlı darbe dizilerinin sayılması; konum değişikliklerinin tesbiti ve işlenmesi; hız ve zaman ölçümü; kapalı döngü kontrolü ve konumlandırma; zamanlamnın son derece önem taşıdığı uygulamalardan bir kaçıdır. PLC'nin merkezi işlemcisi asıl kontrol işine ek olarak, bu tür işlemleri yeterince hızla yerine getiremez. Bu amaçla, S5-115 U sisteminde zeki giriş/çıkış modülleri tasarlanmıştır. Bu modüller kullanılarak, kapalı ve açık döngü kontrolleri ve ölçme işlemleri işletilen programa paralel olarak yürütülebilir. Çoğu modüllerin görevlerini bağımsız olarak yerine getirmeleri için yüksek hızda çalışan kendi işlemcileri vardır.

5.2.5. Haberleşme İşlemcileri

S5-115 U sistemi, insan-makina veya makina-makina arasındaki iletişmi kolaylaştıran çok sayıda özel haberleşme işlemcisi imkanını sağlar. Haberleşme işlemcilerinin iki ana grubu: Yerel bilgisayar ağları (LAN) için olanlar ile bağlantı (linking) işaretleme (signalling) ve kütükleme (logging) için olanlardır.

5.3. Genişletme Olanağı

Merkezi kontrol elemanının bağlantı imkanı, sistem gereklerini karşılamadığı takdirde; kapasite, genişletme birimleri sayesinde artırılabilir. Arabirim modülleri,merkezi kontrol birimini, genişletme birimine ve genişletme birimlerini de birbirine bağlar.

5.3.1. Merkezi Konfigürasyon

Bu tür düzenlemede bir merkezi kontrolcüye üç taneye kadar genişletme elemanı bağlanması olanağı vardır. Arabirim modülleri, bus hatlarını ve kaynak gerilimini genişletme birimlerine bağlarlar. Bu yüzden bu tür düzenlemede her genişletme biriminin ayrı bir güç kaynağına ihtiyacı yoktur. Ayrı ayrı kontrolcüler arasındaki kabloların uzunluğu 2.5 m.'yi aşmamalıdır.

5.3.2. Dağıtılmış (Distributed) Konfigürasyon

Bu tür düzenlemede, genişletme birimleri, sistemdeki duyargalara ve hareketlendiricilere yakın yerleştirilmesi imkanı vardır. Böylece kablolama harcamaları önemli ölçüde azaltılır.

5.4. Haberleşme Sistemleri

Kontrol ünitesinin esnekliği, üretimin verimliliği açısından çok önem taşır. Karmaşık kontrol işleri, geniş esnekliğin sağlanması için çeşitli kontrol elemanlarına bölünür ve dağıtılır. Dağıtımın şu gibi avantajları vardır.

- Küçük birimlerin idaresi kolay olacağından, sistemin planlama, yenileme ve geliştirme, çalıştırıp durdurma gibi işleri ile tüm sürecin (process) izlenmesi kolaylaşacaktır.

- Sistemin imkanları artacaktır çünkü, bir birim arızalansa dahi, diğerleri işlerine devam edebileceklerdir.

Dağıtılmış kontrol elemanları arasında,

- programlanabilir kontrolcüler arasında veri alışverişinin
- üretim sisteminin merkezi işlem kontrolü ve görüntülenmesinin
- üretim ve depolama gibi çeşitli konularda yönetim için gerekli bilgilerin sağlanması için bilgi akışı olmalıdır.

Bu sebeplerle S5-115 U programlanabilir kontrolcüleri için şu haberleşme imkanları önerilmektedir.

- CP524 ve CP525 haberleşme işlemcileri ile noktadan noktaya bağlantı
- SINEC L1 network yoluyla yerel bilgisayar ağı haberleşmesi imkanı
- SINEC H1
- SINEC L2
- 943 ve 944 merkezi işlem birimlerinin (CPU) noktadan noktaya bağlantı imkanı
- Yazıcı ve klavye bağlantısı için CPU 943 ve 944'de ASCII arabirimi
- 3964 veya 396R protokolü ile bilgisayar bağlantısı imkanı

5.5. İşlemci-Süreç Haberleşmesi, Görüntüleme ve Programlama

Bugün artık kullanıcılar, kontrol sistemlerine gerektiğinde müdahale imkanı tanıyan ve görsel olarak sürecin (process) rahatlıkla izlenebildiği tasarımları beklemektedirler. Çok basit sistemler için dahi, indikatör lambalar, çeşitli düğme, anahtar, buton ve potansiyometre gibi donanımları kullanmayı öncelikle tercih ederken, sistemin gereklerine ve karmaşıklığına göre video görüntüleme terminallerini de kullanabilmektedirler.

S5-115 U sistemi çok yaygın ve çeşitli (küçük el terminallerinden, renkli video terminallerine kadar) operatör panellerini ve görüntüleme cihazlarını

desteklemektedir. PG605U PG615 el terminal programlayıcı PG635 2CD göstergeli portatif terminal, PG685, PG710.

5.6. Yazılım

Bugüne kadar donanım gereçelerinin fiyatları sürekli olarak düşerken yazılım gereçleri fiyatları sürekli bir artış eğilimi göstermektedir. Bu:

- otomasyonu yapılacak sistemlerin gittikçe daha karmaşık hale gelmesi
- emniyet şartlarının iyileştirilmesi
- personel giderlerinin artışı
- ergonomik kuralların yertleşmesi

gibi çeşitli sebeplerin sonucunda ortaya çıkmıştır.

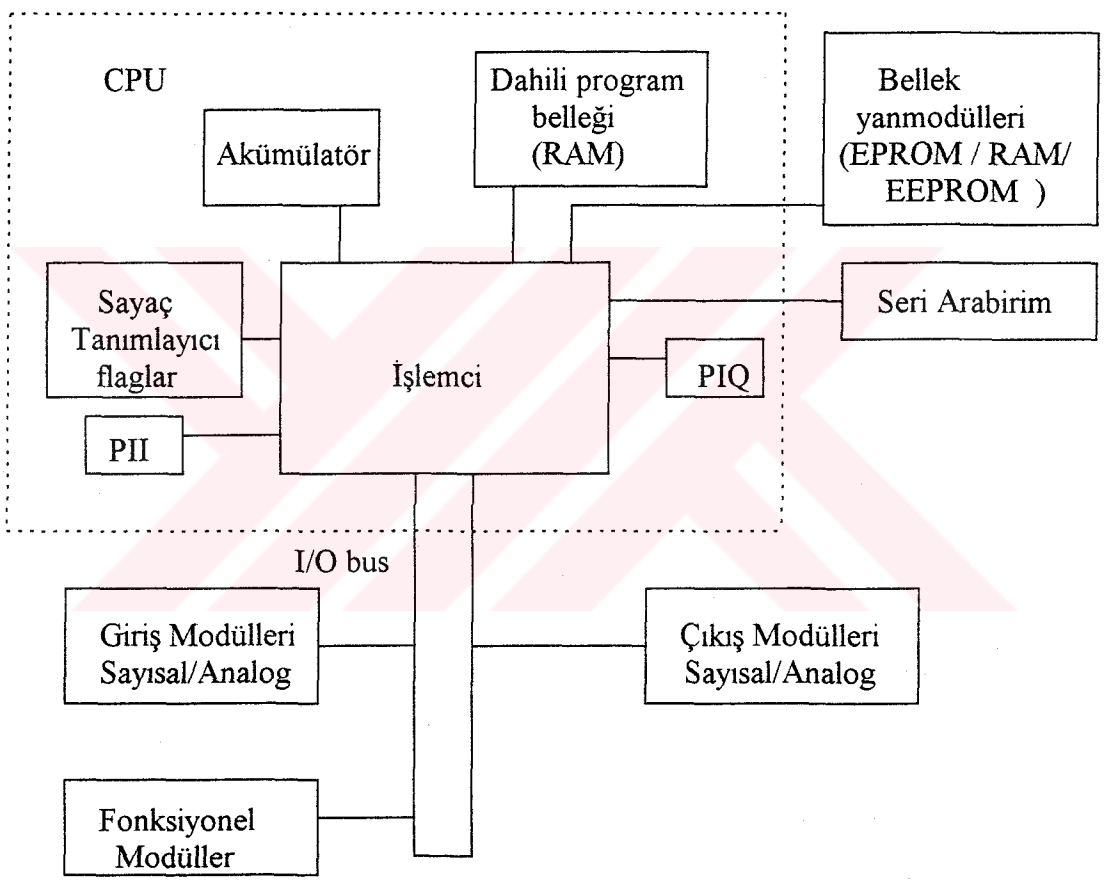
Siemens firması, SIMATIC ile yazılım giderlerini düşürmek üzere şu üç çözümü sağlamıştır.

- 4 gösterim metodu ve uygun yapılandırma imkanlarına sahip olan kullanıcı-dostu (user-friendly) STEP-5 programlama dili
- kapsamlı bir yazılım kataloğu
- kullanıcı-dostu programlayıcılar

5.7. Teknik Tanımlama

5.7.1. Modüler Tasarım

S5-115 U, özel bir probleme yönelik olarak, uygun bir çözüm oluşturan çeşitli fonksiyonel birimler içerecek şekilde dizayn edilmiştir.



Şekil 5.2. S5-115U yapısının şematik gösterimi

i- Güç kaynağı modülü (PS951) : 115V AC/230V AC veya 24V DC güç sistemi gerilimin PLC'nin işletim gerilimine dönüştürür. Bu modül RAM (Rastgele erişimli bellek= Random Acces Memory) 'i yedeklemek üzere bir pil veya bir dış güç kaynağını kullanır.

ii- Merkezi İşlem Birimi (CPU): giriş sinyal durumlarını işler, control programını yürütür ve çıkış sinyallerini işler. Program tarama fonksiyonlarına ek olarak, CPU iç bayrakları (flags), zamanlayıcı ve sayaçları da düzenler. Sistemi çalışmaya başlatan programın çalışıp çalışmadığını ve varsa hataları CPU üzerindeki LED (ışıklı diyot) lerden görmek mümkündür.

iii- Haberleşme İşlemcileri (CP): makineler arasındaki ve insan-makine arasındaki haberleşmeyi sağlamak için kullanılır. Aşağıdaki fonksiyonları yerine getirirler.

- makine fonksiyonlarının veya süreci meydana getiren bölümlerin kontrolü ve görüntülenmesi
- makina ve süreç durumlarının dökümü ve raporlanması

iv- Giriş/Çıkış Modülleri (I/O s):

- Dijital giriş modülleri, sayısal sinyaller girişlerini , S5-115 U iç sinyal seviyesine uyarlar
- Sayısal çıkış modülleri S5-115 U'nun iç sinyal seviyesindeki işaretleri sayısal işlem sinyallerine dönüştürür. (Örneğin; role ve solenoid valflerin sünülmesi için)
- Analog giriş modülleri, analog işlem sinyallerini (örneğin; transducer veya dirençli termometreden alınan sinyalleri) S5-115 U'nun işleyebileceği sayısal değerlere uyarlar.
- Analog çıkış modülleri iç sayısal değerleri analog işlem sinyallerine dönüştürür. (Örneğin; hız kontrolcüleri için)

v- Arabirim Modülleri (IM):

S5-115 U, belli sayıda bağlantı lokasyonları (slots) olan bağlantı rafları üzerine yerleştirilir. Güç kaynağı, CPU, giriş/çıkış modülleri içeren konfigürasyona merkezi kontrolcü adı verilir. Eğer merkezi kontrolcünün bağlama rafi üzerindeki slotlar yeterli gelmezse, ek bağlama rafları üzerine genişletme birimleri yüklenebilir. Arabirim modülleri genişletme birimlerini merkezi kontrolcüye bağlar.

vi- Bağlantı Rafları: bütün modüllerin mekanik olarak üzerine tutturulduğu bir alüminyum ray sisteminden oluşur. Ve modülleri birbirlerine elektriksel olarak bağlayan bir veya iki arkaplanı (backplane) bulunur.

vii- Seri Arabirim:

- Programcı
- Operatör paneli
- SINEC L1 bus terminali

bu ara birime bağlanır.

viii- Bellek Yan Modülleri

ix- Pil Yerleştirme Bölümü

5.7.2. Fonksiyonel Birimler

Program Belleği (Dahili program belleği, bellek yan modülleri) :

Kontrol programı, dahili program belleğine (RAM) veya bellek yan modüllerine depolanır. CPU943 ve CPU944 tüm programı dahili RAM'de tutabilirler. Programın kaybolmasını önlemek için, dış EPROM veya EEPROM bellek yan modülüne atılır. Bu bellek yan modüllerinin aksine dahili RAM veya RAM bellek yan modülleri şu özelliklere sahiptir.

- Bellek içeriği hızlı bir şekilde değiştirilebilir.
- Kullanıcı verileri depolanabilir ve değiştirilebilir.
- Güç kaynağı arızalanırsa ve yedekleme pili de bulunmuyorsa bellek içeriği silinir.

Süreç İmajları (PII, PIQ):

Giriş/çıkış modüllerindeki sinyallerin durumları CPU'da süreç imajlarında toplanır. Süreç imajları CPU RAM içinde saklı alanlardır.

- PII: Giriş sinyalleri süreç imajı
- PIQ: Çıkış sinyalleri süreç imajı

Seri Arabirim:

Programlayıcının, operatör panelinin ve monitörlerin bağlandığı arabirimdir. Ayrıca bütün CPU'lar SINEC L1 yerel bilgisayar ağına seri arabirim vasıtasıyla bağlanır. Ek olarak:

- Diğer programalnabilir kontrolcülerle noktadan noktaya bağlantı kurulması
- Bağlanacak yazıcı ve klavyeler için ASCII sürücüsü imkanı
- Tümleşik gerçek zaman saati (Integral real-time clock)
- PC ile haberleşme bağlantısı (3964/3964R protokolünü kullanan) imkanları da mevcuttur.

Zamanlayıcılar, Sayaçlar, Flaglar:

Her CPU dahili sayaç, zamanlayıcı ve flagları ile donatılmıştır. Flaglar özel işaretlerin durmlarının saklandığı bellek bölümleridir.Flaglar, zamanlayıcı ve sayaçların içeriği güç kesintisinde silinmezler.

Akümülatör (ACCUM):

Akümülatör, örneğin dahili sayısal değerlerin ve zaman değerlerinin yüklenmesi için kullanılan aritmetik yazmaçtır. Karşılaştırma, dönüştürme ve diğer aritmetik işlemler de akümülatörde yürütülür.

İşlemci:

İşlemci, program belleğindeki durumları sırasıyla çağırır ve kontrol programına uygun olarak yerine getirir. Giriş sinyalleri süreç imajından edinilen bilgiyi, dahili zamanlayıcı, sayaç ve flagların durumlarını da gözönünde bulundurarak işler.

I/O Bus:

I/O bus, CPU ile merkezi kontrolcüye ve genişletme birimine bağlanan diğer modüller arasındaki bütün sinyal alışverişini sağlayan bağlantıyı kurar.

Bellek Yan Modülleri:

S5-115 U' da, kontrol programını depolamak veya programı PLC'ye iletmek için kullanılan 3 tür bellek yanmodülü imkanı vardır.

- EPROM yanmodülleri
- EEPROM yanmodülleri
- RAM yanmodülleri

BÖLÜM 6

SIMATIC S5 CP 524 HABERLEŞME İŞLEMCİSİ

CP 524 ve 525 haberleşme işlemcileri SIMATIC S5 U serisi için (S5-115 U, S5-135 U, S5-150 U ve S5-155 U) için kullanılabilmektedir. CP 524 haberleşme işlemcisi COM 525 programlama paketi eşliğinde kullanılarak şu işlemleri yerine getirir.

- Süreç durumlarının ve süreçten alınan mesajların kütüklenmesi
- Programlanabilir kontrolcünün diğer kontrolcüler ve bilgisayarlarla haberleşmesi

Eğer PLC otomasyon ağının bir parçasını oluşturuyorsa;

- bir veya daha fazla kontrolcü ile veya bir süpervizör bilgisayar ile veri alışverişi yapması sağlanır.

CP524 bu görevleri bağımsız olarak yerine getirir.

CP524 ile programlanabilir kontrolcünün merkezi işlem birimi arasındaki veri alışverişi bir minimum değerle kısıtlıdır. Bunun için, yalnızca değişken veri alışverişi gerçekleştirilir.

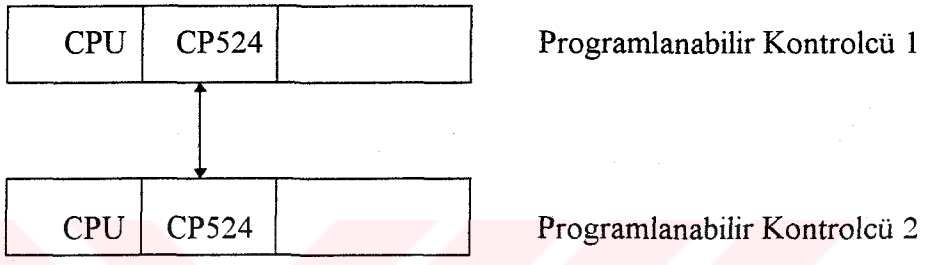
CP524, değişmez verilerin saklandığı bir belleğe sahiptir.

6.1. CPU arabirimi olarak : çift kapılı (Dual-port) RAM

CPU ile CP524 arasındaki data alışverişi yaygın bir bellek alanı boyunca gerçekleştirilir. (Dualport RAM) CP524 üzerindeki çift kapılı RAM, bir posta kutusu gibidir. CP524 ve CPU birbirlerine gönderecekleri mesajları bu alana bırakırlar. Data alışverişinde programlanabilir kontrolcünün CPU'su daima yönetici (master) durumundadır. CP524, CPU'nun veri iletimi yapıp yapamayacağını sormak ve beklemek zorundadır. Bu karar verme işlemi CPU'daki standart fonksiyon blokları tarafından yerine getirilir.

6.2. 3964R Haberleşme Protokolünü Kullanarak Haberleşme Bağlantısının Kurulması

İki programlanabilir kontrolcü (2 CPU) veya bir programlanabilir kontrolcü ile bir başka uç nokta arasında veri alışverişi, bir haberleşme bağlantısıyla mümkündür. Veri alışverişi CP524'ün RK512 interpreter ve 3964R prosedürleri kullanılarak gerçekleştirilebilir. Noktadan noktaya bağlantısı boyunca data alışverişi şöyle gösterilebilir.



Veri alışverişi iki yolla yerine getirilebilir.

- PC1 kendi başlatıcısındaki (initiator) veriyi PC2'ye iletir. Bu durumda CPU1, kendi CP525'indeki Gönder (SEND) komutunu çağırmalıdır.
- PC1, PC2'den kendi başlatıcısına gelen datayı kabuledebilir. Bu durumda CPU1, kendi CP525'indeki Kabul et (FETCH) komutunu çağırmalıdır.

İki programlanabilir kontrolörün haberleşmesi için herbirinde iki programın bulunması gerekir.

- CPU'daki STEP5 kullanıcı programı
- CP524'deki CP524 kullanıcı programı

COM 525 programlama paketi bu tür işlemlerin yapılamsını sağlar.

Haberleşme işlemi için bazı parametrelerin belirlenmesi gerekir.

Öncelik (Priority):

Bağlanan her iki eleman da iletim hattına mesaj göndermek istediklerinde aralarında bir önceliğin tespit edilmesi gerekir. Yüksek önceliğe sahip olan alet mesaj gönderdiğinde, diğerinin beklemesi gerekir.

Veri Hızı:

110 bps (bits/s)
150 bps
300 bps
600 bps
1200 bps
2400 bps
4800 bps
9600 bps
19200 bps'den biri seçilebilir.

(19200 bps ancak RS 422-A veya 485 arabirimleri ile mümkündür. Bunların dışında 3964R için diğer parametreler: 8 data biti, 1 durdurma (stop) biti ve çift (even) parity olacak şekilde sabittir.

3964R protokolü, 3964 protokolü ile hemen hemen aynı yapıdadır. Ancak 3964R farklı olarak gönderdiği veri paketlerinin bir blok kontrol değeri (BCC=Block Check Character) ekler. BCC gönderilen datanın "even longitudinal parity" değeridir. Bu değer hesaplanırken: her bir byte'ta önce 0. bit değerleri toplanır sonuç tek sayı ise BCC byte'ının 0. bitine 1 aksi halde 0 yazılır. 7. bite kadar bu işlem tekrarlanır. Sonuçta elde edilen byte BCC değeri olarak veri paketinin sonuna eklenir.

3964R yapısında komutlar, belirli bir telgraf mesajı yapısında iletilir. İstenen bir adresteki veriyi okuma işlemi FETCH, belirlenen bir adrese bir değerin yazılması işlemi için de SEND komutlarıyla yerine getirilir.

SEND komutu telgrafı hem telgraf başlığını hem de veriyi içerirken, FETCH komutu telgrafı yalnızca telgraf başlığını içerir.

Telgraf Başlığı Yapısı:

1	2	3 4	5 6	7 8	9 10
00hex. (FF)hex.	00hex.	komut	hedef/kaynak adresi	veri dizisi uzunluğu	CPU no/ CF

telgraf başlığında yer alan byte'ların gösterdikleri değerler:

Byte 1: telgraf tanımlayıcısı (00 ve ek telgraf ise FF değerini taşır.)

Byte 2: telgraf tanımlayıcısı (00 hexadesimal)

Byte 3: komut, SEND işlemi için ('A' veya 'O'), FETCH işlemi için ('E') karakterleri yer alır.

Byte 4: Veri türü . Siemens PLC kontrol programı yazılırken çeşitli veri tipleri tanımlanır. Okunacak ve yazılacak veri hangi tür ise o türü belirtirken karakter 4.byte'ta yazılır.

'D' = Data bloğu

'E' = Giriş byteları

'M'= Flag byteları

'Z' = Sayaç lokasyonları

'S' = Mutlak değer adresler

'Q' = Genişletilmiş giriş/çıkışlar

'X' = Genişletilmiş data bloğu

'A' = Çıkış byteları

'P' = Giriş/çıkış byteları

'T' = Zamanlayıcı lokasyonları

'B' = Sistem adresleri

Byte 5 ve 6: SEND işlemi için hedef adresi, FETCH işlemi için kaynak adresi yazılır.

Byte 7 ve 8: Data türüne bağlı olarak söz konusu verinin uzunluğunun kaç byte veya word olduğu yazılır.

Byte 9 : Koordinasyon flag (CF) numarasıdır. Herhangi biri belirlenmemişse bu byte'a hexadesimal FF değeri yazılır.

Byte 10 : En anlamlı 4 bitine (4-7 bitleri) CPU numarası yazılır. Bu 1'den 4'e kadar bir sayı olabilir. En az anlamlı 4 biti ise; koordinasyon flagdaki (CF) bit sayısı belirlenir. Eğer CPU numarası ve koordinasyon flag özellikle belirlenmemişse bu byte hexadesimal FF değerini taşır.

Cevap Telgrafı Yapısı:

1	2	3	4
00 hex./ FF hex.	00 hex.	00 hex.	hata numarası

Byte 1: telgraf tanımlayıcısı (Normalde 00hex. Ek cevap mesajında FF hex. olur.)

Byte 2: telgraf tanımlayıcısı hex. 00 değerini taşır.

Byte 3: hexadesimal 00 değerini taşır.

Byte 4: oluşan bir hata varsa ilgili değeri taşır.

6.2.1. SEND (Veri Gönderilmesi) Fonksiyonunun İşleyişi

PC veya PLC1			PLC2
	------	hex. 02 = Başlatma Karakteri	------>
	<----	hex. 10 = Pozisyon Bilgisi	-------
	------	hex. 00	------>
	------	hex. 00	------->
telgraf	------	hex. 41 = Send İşlemi Kodu	------->
başlığı	------	hex. XX= Veri Tipi	------->
	------	hex. XX= Hedef Adres Byte 1	------->
	------	hex. XX= Hedef Adres Byte 2	------->
	------	hex. XX= Veri uzunluğu Byte 1	------->
	------	hex. XX= Veri uzunluğu byte 2	------->
	------	hex. FF = CF belirlenmemesi halinde	------>
	------	hex. FF = CPU belirlememsi halinde	------>
	------	hex. XX= Veri byte 1	------->
	------	hex. XX= Veri byte 2	------->
Veri dizisi		.	
		.	
	------	hex. XX= Veri byte n.	------->
	------	hex. 10 = Sonlandırma karakteri	------->
	------	hex. 03 = Sonlandırma karakteri	------->
	------	BCC karakteri	------->
	<------	hex. 10 = Pozisyon bilgisi	-------
	<------	hex. 02 = Başlatma karakteri	-------
	------	hex. 10 = Pozisyon bilgisi	------->
	<------	hex. 00	-------
cevap	<------	hex. 00	-------
telgrafi	<------	hex. 00	-------
	<------	hex. 00 veya hata numarası	-------
	<------	hex. 10 = Sonlandırma karakteri	-------
	<------	hex. 03 = Sonlandırma karakteri	-------
	<------	BCC karakteri	-------
	------	hex. 10 = Pozisyon bilgisi	------->

gönderilmek istenen verinin uzunluğu 128 byte'ı aşarsa bundan sonra gönderilecek her 128 byte için bir ek mesaj (işleyişi aşağıda açıklanıyor) gönderilir. Veri byte değeri kontrol karakterlerinden hex. 10'a eşit ise bu byte iki kez tekrarlanır.

EK SEND İşlemi:

PC veya PLC1			PLC2
	------	hex. 02 = Başlatma Karakteri	------->
_____	<-----	hex. 10 = Pozisyon Bilgisi	-------
	------	hex. FF = Ek mesaj tanımlayıcısı	------->
ek telgraf	------	hex. 00	------->
başlığı	------	hex. 41 = Send İşlemi Kodu	------->
_____	------	hex. XX= Veri Tipi	------->
	------	hex. XX= Veri Byte 129	------->
	------	hex. XX= Veri Byte 130	------->
Veri Dizisi		. .	
		. .	
		. .	
_____	-------	hex. XX= Veri Byte n.	------->
	-------	hex. 10 = Sonlandırma karakteri	------->
	-------	hex. 03 = Sonlandırma karakteri	------->
	-------	hex. XX= BCC karakteri	------->
	<-------	hex. 10 = Pozisyon bilgisi	-------
	<-------	hex. 02 = Başlatma karakteri	-------
	-------	hex. 10 = Pozisyon bilgisi	------->
_____	<-------	hex. FF = ek mesaj tanımlayıcısı	-------
ek cevap	<-------	hex. 00	-------
telgrafi	<-------	hex. 00	-------
_____	<-------	hex. 00 veya hata kodu	-------
	<-------	hex. 10 = Sonlandırma karakteri	-------
	<-------	hex. 03 = Sonlandırma karakteri	-------
	-------	hex. 10 = Pozisyon bilgisi	------->

6.2.2. FETCH (Veri Okuma) Fonksiyonunun İşleyişi

PLC veya PLC1		PLC2
	------- hex. 02 = Başlatma karakteri	------>
_____	<------ hex. 10 = pozisyon bilgisi	------
	------- hex. 00	------>
	------- hex. 00	------>
	------- hex. 45 = FETCH işlemi kodu	------>
telgraf	------- hex. XX= Veri tipi	------>
	------- hex. XX= kaynak adres byte 1	------>
başlığı	------- hex. XX= kaynak adres byte 2	------>
	------- hex. XX= veri uzunluğu byte 1	------>
	------- hex. XX= veri uzunluğu byte 2	------>
	------- hex. FF = CF belirlenmemiş halinde	------>
_____	------- hex. FF = CPU no belirlenmemesi halinde	------>
	------- hex. 10 = Sonlandırma karakteri	------->
	------- hex. 03 = Sonlandırma karakteri	------->
	------- BCC karakteri	------->
	<------ hex. 10 = pozisyon bilgisi	-------
	<------ hex. 02 = başlatma karakteri	-------
_____	------- hex. 10 = pozisyon bilgisi	------->
cevap	<------ hex. 00	-------
telgrafı	<------ hex. 00	-------
	<------ hex. 00	-------
_____	<------ hex. 00 veya hata kodu	-------
	<------ hex. XX= veri byte 1	-------
veri dizisi	<------ hex. XX= veri byte 2	-------
	. .	
	. .	
_____	<------ hex. XX= veri byte n.	--------
	<------ hex. 10 = Sonlandırma karakteri	--------
	<------ hex. 03 = Sonlandırma karakteri	--------
	<------ BCC karakteri	--------
	------- hex. 10 = pozisyon bilgisi	-------->

Veri dizisi içinde hex.10 karakteri geçiyorsa 2 kez tekrarlandığından ardarda gelen iki hex. 10 değerinden biri veri olarak kaydedilir. Veri dizisinin 128 byte'tan daha uzun olması halinde ek FETCH işlemi yürütülür.

Ek FETCH İşlemi:

PLC veya PLC1			PLC2
	------	hex. 02 = Başlatma karakteri	------>
	<-----	hex. 10 = pozisyon bilgisi	------
	------	hex. FF = Ek mesaj tanımlayıcısı	------>
ek telgraf	------	hex. 00	------>
başlığı	------	hex. 45 = FETCH işlemi kodu	------>
	------	hex. XX= Veri tipi	------>
	------	hex. 10 = Sonlandırma karakteri	------>
	------	hex. 03 = Sonlandırma karakteri	------>
	------	BCC karakteri	------>
	<------	hex. 10 = Pozisyon bilgisi	------
	<------	hex. 02 = Başlatma karakteri	------
	-------	hex. 10 = Pozisyon bilgisi	------>
	<------	hex. FF = Ek mesaj tanımlayıcı	------
ek cevap	<------	hex. 00	------
telgrafi	<------	hex. 00	------
	<------	hex. 00 veya hata kodu	------
	<------	hex. XX= veri byte 129	------
	<------	hex. XX= veri byte 130	------
		. .	
Veri dizisi		. .	
	<-----	hex. XX= veri byte n.	-------
	<------	hex. 10 = Sonlandırma karakteri	-------
	<------	hex. 03 = Sonlandırma karakteri	-------
	<------	BCC karakteri	-------
	------	hex. 10 = Pozisyon bilgisi	------->

6.3. 3964R Haberleşme Protokolünü Uygulayan Programın Tanıtımı

PC veya VME ile SIEMENS PLC arasında 3964R protokolüne uygun olarak seri iletişimi sağlamak üzere C dilinde program, aşağıda açıklanan düzene göre çalışan fonksyonlar ile PLC'den istenen datanın okunması veya PLC'de belirlenen yerlere datanın yazılması sağlanmaktadır.

6.3.1. Haberleşmeyi Sağlayan Fonksiyonların Çalışma Düzenleri

PREAD ve PWRITE fonksyonları işleme başladıklarında PLC'ye yapılacak işlemi tanıtmak üzere 10 byte'tan oluşan Header (başlık) mesajını gönderirler.Bu mesajın içeriği:

1.Byte: hex.00
2.Byte: hex.00
3.Byte: işlem kodu.
4.Byte: DataTürü
5.Byte: DataAdresi
6.Byte: DataAdresi
7.Byte: Data sayısı (8-15 bitleri)
8.Byte: Data sayısı (0-7 bitleri)
9.Byte: Coordination Flag (CF) byte numarası.
10.Byte: 0-3 bitleri coordination flag bit numarasını gösterir. 4-7 bitleri CPU numarasını gösterir.

Header mesajının içeriği ileride verilen tablolarda da açıklanacaktır.

6.3.2. PREAD Fonksiyonu

PREAD(DataTürü,DataBlokNo,DataWordNo,Datasayısı)

DataTürü :

PLC'den okunmak istenen datanın tipine bağlı olarak bu değişken yerine ( ) içindeki karakterler yazılmalıdır. Bu Siemens PLC'leri programlamada kullanılan STEP5 dilinde tanımlı değişken türleridir.

Data Block(DB),
System Data (RS),
Absolute Adress (AS)
Extended Data Block (DX),
Flag byte (FY),
Input Byte (IB),
Output Byte (QB),
I/O Byte (PB),
Timer Location (TB),
Counter Location (CB),
extended I/O (OB)

DataBlokNo :

Okunacak DataTürü DB veya DX olduğunda bu değişkene verilen 0-255 arasındaki integer değer PLC'den okunacak DataBloğunun numarasını ifade eder. Diğer DataTürlerinde bu değişkene atanan değer göz önüne alınmaz.

DataWordNo:

Okunacak DataTürü DB veya DX olduğunda bu değişkene verilen 0-255 arasındaki integer değer PLC'deki DataBloğun kaçıncı Word'ünden itibaren okuma yapılacağını gösterir. Diğer DataTürlerinde bu değişkene atanan integer değer datanın adresini gösterir.

DataSayısı:

Bu değişkene verilen integer değer DataTürüne bağlı olarak kaç byte veya word okunacağını ifade eder.

PREAD İşleminin Kodu:

Header mesajının 3.byte'ına, PREAD işlemi için 3964r'de fetch işlemine karşılık gelen 'E'=hex.45 değeri atanır.

TABLO 6.1. Okunacak datanın türüne göre Header mesajının içeriği

DataTürü	CF seçimi yapılabilir?	Header 5.ve 6.Byte	Header 7.ve 8.Byte
DB	evet	DBno(0-255)DWno(0-255)	words (1-2048)
DX	evet	DBno(0-255)DWno(0-255)	words (1-2048)
FY	hayır	Byte adres (0-254)	bytes (1-128)
IB	hayır	Byte adres (0-126)	bytes (1-64)
QB	hayır	Byte adres (0-126)	bytes (1-64)
CB	hayır	Sayaç No.(0-255)	words(1-256)
TB	hayır	Timer No.(0-255)	words(1-256)
PB	hayır	I/O adres (0-254)	bytes(1-128)
RS	hayır	System adres(0-511)	words(1-512)
AS	hayır	Abs. adres(0-65535)	words(1-32767)
OB	hayır	I/O adres(0-254)	byte (1-128)

PREAD fonksyonunun çalışma düzeni:

DURUM1:

Başlangıç karakterini (STX = hex.02) gönder.

DURUM2:

Girişe bir karakter gelip gelmediğini kontrol et.

- DLE karakteri (hex.10) geldi ise DURUM3'e geç
- QVZ (2 s) süresi içinde bir bilgi gelmediyse veya DLE'den farklı bir karakter geldi ise DURUM2 sayacını artır.
- Durum2 sayacı 6'dan büyükse, ilgili hata mesajını ver, NAK (hex.15) karakteri gönder ve işlemi bitir, değilse DURUM1'e geç.

DURUM3:

Daha önce belirlenen HEADER mesajına DLE ,ETX (hex.03) karakterleri eklenerek elde edilen dizinin BCC değerini hesapla. BCC (even longitudinal parity) diziyi oluşturan byte'ların aynı numaralı bitlerinden 1 olanları sayar tek sayıda 1 olduğunda BCC byte'ının aynı nolu biti 1 değerini alır. HEADER mesajını, DLE, ETX ve BCC karakterlerini gönder ve Durum4'e geç.

DURUM4:

Girişe bir karakter gelip gelmediğini kontrol et.

- DLE geldi ise DURUM5'e geç
- QVZ (2 s) süresi içinde bir bilgi gelmediyse veya DLE'den farklı bir karakter geldi ise DURUM2 sayacını artır.
- Durum4 sayacı 6'dan büyükse, ilgili hata mesajını ver, NAK (hex.15) karakteri gönder ve işlemi bitir, değilse DURUM1'e geç.

DURUM5:

Girişe bir karakter gelip gelmediğini kontrol et.

- STX geldi ise DURUM6'ya geç
- ZVZ (220 ms) süresi içinde okuma yapılamadıysa veya STX'den farklı bir karakter geldi ise DURUM5 sayacını artır.
- Durum5 sayacı 6'dan büyükse, ilgili hata mesajını ver, NAK (hex.15) karakteri gönder ve işlemi bitir, değilse DURUM1'e geç.

DURUM6:

DLE (0x10) karakteri gönder.

DURUM7:

-Ardarda DLE ETX karakterlerini tesbit edene kadar giriş buffer'ını oku ve bir pointer'a kaydet.

- İki okuma işlemi arasında ZVZ süresi aşılırsa ,ilgili hata kodunu yaz, NAK karakterini gönder ve işlemi bitir.
- pointer'a kaydedilen 4. karakter 0'dan farklı ise ilgili hata kodunu yaz , NAK karakteri gönder ve işlemi bitir.
- DLE ETX karakterlerinden sonra okunan byte BCC değeridir. Kaydedilen karakterlerin BCC değerini hesapla ve gelen BCC ile karşılaştır.Aynı değere

sahiplerse DLE karakteri gönder ve DURUM8'e geç. Farklılarsa, Durum7 sayacını artır.

- Durum7 sayacı 6'dan büyükse ilgili hata kodunu ver, NAK karakteri gönder ve işlemi bitir. Değilse DURUM1'e geç

DURUM8:

Pointer'ın ilk 4 byte ile son DLE ETX karakterileri arasındaki byte'ları Data Bilgisini oluşturur. Data bilgisi içinde iki kez tekrarlanan hex.10 değeri 1 kez kaydedilir.

Okunmak istenen data dizisinin uzunluğu 128 byte'ı aşmıyorsa DURUM8'den sonra okuma işlemi sona erer. 128 byte'tan uzun dizilerde ilk 128 byte yukarıda açıklandığı gibi okunduktan sonra DURUM1'e dönülür. Her defasında en fazla 128 byte okunacak şekilde geri kalan datanın okunması için aynı işlemler tekrarlanır, yalnız DURUM3'deki HEADER mesajının yerini EK HEADER mesajı alır.Ek header mesajı 4 byte içerir.

1.Byte: hex.ff

2.Byte: hex.00

3.Byte: işlem kodu.

4.Byte: DataTürü

6.3.3. PWRITE Fonksiyonu

PWRITE(DataTürü,DataBlokNo,DataWordNo,Datasayısı,DBTipi)

DataTürü :

PLC'ye yazılmak istenen datanın tipine bağlı olarak bu değişken yerine ( ) içindeki karakterler yazılmalıdır.

Data Block(DB),

System Data (RS),

Absolute Adress (AS)

Extended Data Block (DX),

Flag byte (FY),

Input Byte (IB),

Output Byte (QB),

I/O Byte (PB),

Timer Location (TB),

Counter Location (CB),

extended I/O (OB)

DataBlokNo :

Yazılacak DataTürü RS veya AS olduğunda bu değişkene verilen değer göz önüne alınmaz. Diğer DataTürlerinde bu değişkene atanan 3-255 arasındaki integer değer, datanın PLC'de kaçıncı DataBloğa yazılacağını gösterir.

DataWordNo:

Yazılacak DataTürü RS veya AS olduğunda bu değişkene verilen değer yazılmaya başlama adresini gösterir. Diğer DataTürlerinde bu değişkene atanan integer değer, DataBloğun kaçıncı DataWord'unden itibaren yazılacağını gösterir.

DataSayısı:

Bu değişkene verilen integer değer DataTürüne bağlı olarak kaç byte veya word yazılacağını ifade eder.

DBtipi:

Data PLC'de bir extended DataBloğa yazılmak istendiğinde 1,diğer durumlarda 0 seçilir.DBtipi değerine göre işlem kodu değişecektir.

PWRITE İşleminin Kodu:

Datanın PLC'de bir DataBloğuna,Sistem adress değerine veya Absolute adres değerine yazılması durumunda işlem kodu 'A'=hex.41 dir.(DBtipi=0)

Eğer PLC'de bir Ext.DataBloğa yazılacaksa işlem kodu 'O'=hex.4F olur.(DBtipi=1)

TABLO 6.2. Yazılacak datanın türüne göre Header mesajının içeriği
(işlem kodu:hex.41 için)

DataTürü	CF seçimi yapılabilir?	Header 5.ve 6.Byte	Header 7.ve 8.Byte
DB	evet	DBno(3-255)DWno(0-255)	words (1-2048)
DX	evet	DBno(3-255)DWno(0-255)	words (1-2048)
FY	evet	DBno(3-255)DWno(0-255)	bytes (1-256)
IB	evet	DBno(3-255)DWno(0-255)	bytes (1-128)
QB	evet	DBno(3-255)DWno(0-255)	bytes (1-128)
CB	evet	DBno(3-255)DWno(0-255)	words(1-256)
TB	evet	DBno(3-255)DWno(0-255)	words(1-256)
PB	evet	DBno(3-255)DWno(0-255)	bytes(1-256)
RS	hayır	adres (0-511)	words(1-512)
AS	hayır	adres(0-65535)	words(1-32767)
OB	evet	DBno(3-255)DWno(0-255)	byte (1-256)

TABLO 6.3.Yazılacak datanın türüne göre Header'ın içeriği
(işlem kodu:hex.4F için)

DataTürü	CF seçimi yapılabilir?	Header 5.ve 6.Byte	Header 7.ve 8.Byte
DB	evet	DBno(3-255)DWno(0-255)	words (1-2048)
DX	evet	DBno(3-255)DWno(0-255)	words (1-2048)
FY	evet	DBno(3-255)DWno(0-255)	bytes (1-256)
IB	evet	DBno(3-255)DWno(0-255)	bytes (1-128)
QB	evet	DBno(3-255)DWno(0-255)	bytes (1-128)
CB	evet	DBno(3-255)DWno(0-255)	words(1-256)
TB	evet	DBno(3-255)DWno(0-255)	words(1-256)
PB	evet	DBno(3-255)DWno(0-255)	bytes(1-256)
OB	evet	DBno(3-255)DWno(0-255)	byte (1-256)

PWRITE fonksyonunun çalışma düzeni:

DURUM1:

Başlangıç karakterini (STX = hex.02) gönder.

DURUM2:

Girişe bir karakter gelip gelmediğini kontrol et.

- DLE karakteri (hex.10) geldi ise DURUM3'e geç
- QVZ (2 s) süresi içinde okuma yapılamadıysa veya DLE'den farklı bir karakter geldi ise DURUM2 sayacını artır.
- Durum2 sayacı 6'dan büyükse, ilgili hata mesajını ver, NAK (hex.15) karakteri gönder ve işlemi bitir, değilse DURUM1'e geç.

DURUM3:

Burada PREAD işleminden farklı şekilde, HEADER kısmına ek olarak gönderilecek data katarı (uzunluğu 128 byte'ı aşmayan) yer alır. Data dizisi içinde

hex.10 karakteri yer alıyorsa iki kez tekrarlanır. Dizinin sonuna DLE ETX karakterleri eklendikten sonra BCC değeri hesapla ve BCC karakterini de diziye eklenerek gönder. Durum4'e geç.

DURUM4:

Girişe bir karakter gelip gelmediğini kontrol et.

- DLE geldi ise DURUM5'e geç
- QVZ (2 s) süresi içinde okuma yapılamadıysa veya DLE'den farklı bir karakter geldi ise DURUM4 sayacını artır.
- Durum4 sayacı 6'dan büyükse, ilgili hata mesajını ver, NAK (hex.15) karakteri gönder ve işlemi bitir, değilse DURUM1'e geç.

DURUM5:

Girişe bir karakter gelip gelmediğini kontrol et.

- STX geldi ise DURUM6'ya geç
- ZVZ (220 ms) süresi içinde okuma yapılamadıysa veya STX'den farklı bir karakter geldi ise DURUM5 sayacını artır.
- Durum5 sayacı 6'dan büyükse, ilgili hata mesajını ver, NAK (hex.15) karakteri gönder ve işlemi bitir, değilse DURUM1'e geç.

DURUM6:

DLE karakteri gönder.

DURUM7:

Buffera gelen karakterleri oku.

- ilk 4 karakter 0'dan farklı ise hata kodunu yaz , NAK karakteri gönder ve işlemi bitir.
- değilse, sonraki DLE ETX BCC karakterlerini dikkate alarak, gelen karakterlerin hesaplanan BCC değeri ile okunan BCC değerini karşılaştır.

Eşit olmaları durumunda DLE karakteri göndererek işlemi bitir.Aksi halde Durum7 sayacını artır.

- Durum7 sayacı 6'dan büyükse ilgili hata kodunu ver, NAK karakteri gönder ve işlemi bitir. Değilse DURUM1'e geç.

PLC'ye gönderilecek data dizisinin uzunluğu 128 byte'ı aşmıyorsa DURUM7'den sonra yazma işlemi sona erer. 128 byte'tan uzun dizilerde ilk 128 byte

yukarıda açıklandığı gibi gönderildikten sonra DURUM1'e dönülür. Her defasında en fazla 128 byte yazılacak şekilde geri kalan datanın gönderilmesi için aynı işlemler tekrarlanır, yalnız DURUM3'deki HEADER mesajının yerini aşağıdaki yapıda olan EK HEADER mesajı alır.

1.Byte: hex.ff

2.Byte: hex.00

3.Byte: işlem kodu.

4.Byte: DataTürü

6.3.4. Program Sonuç Mesajları

Veri alışverişi tamamlandığında, ekrana yapılan işlemin ne şekilde sonuçlandığını gösteren mesaj yazılır.

Seri iletişim kapılarının düzenlenmesi ile ilgili mesajlar:

"OpenCommError"

"BuildCommError"

"SetCommStateError"

Seri kapının açılması veya parametrelerinin seçilmesi ile ilgili hata oluştuğunu gösterirler.

"DLE gelmedi" : Fonksyonların çalışma düzeninde açıklanan durumlarda, PLC'den DLE karakterinin gelmemesi halinde işlemin kesildiğini gösterir.

"STX gelmedi" : Fonksyonların çalışma düzeninde açıklanan durumlarda, PLC'den gelecek mesajın başlayacağını gösteren STX karakterinin gelmemesi halinde işlemin kesildiğini ifade eder.

"QVZ timeout" : Fonksyonların çalışma düzeninde açıklanan durumlarda, PLC'den QVZ ile belirlenen süre içinde bir cevap gelmemesi halinde işlemin kesildiğini gösterir.

"ZVZ timeout" : Fonksyonların çalışma düzeninde açıklanan durumlarda, PLC'den ZVZ ile belirlenen süre içinde bir cevap gelmemesi halinde işlemin kesildiğini gösterir.

"BCC error" : datanın doğru olarak okunamadığını gösterir.

"ReplyMessError XX" : PLC nin gönderdiği cevabın hata içerdiğini gösterir.XX degeri yerinde hatanın hexadesimal kodu yer alır.

hex OA: Veri türü veya kaynak/hedef adres değerleri geçerli değildir. Okunacak veya yazılacak veri türünün tanımlı ve geçerli bir adreste olması gerekir.

hex. 14 veya OC: Okunacak veya yazılacak data bloğunun bulunamadığı veya yeterince uzun olmadığını gösterir. PLC programının kullandığı Data blok ve word numaraları kullanılmalıdır.

hex. 10 : Telgraf başlığı yapısı hatalı

hex. 12: Sistem komutunun geçersiz olduğunu gösterir.

hex. 16: Telgraf başlığını ilk karakteri hatalı

hex. 34: Mesajın geçerli uzunlukta olmadığını gösterir. (128 byte'tan daha uzun bir mesaj alınması durumunda)

hex. 36: Bağlantı kurulan haberleşme işlemcisinin senkronize olmadığını gösterir. Telgraf sıralamasında hata vardır.

hex 2A: PLC programının çalışır durumda olmadığını gösterir.

hex. 32: Data bloğunun koordinasyon flağı ile etkisiz hale getirildiğini gösterir. Koordinasyon flagları reset edilmelidir.

"Işlem Dogru" : okuma veya yazma işleminin hatasız olarak tamamlandığını gösterir.

BÖLÜM 7

VMEBus MİKROBİLGİSAYAR BUS'I ve OS-9 İŞLETİM SİSTEMİ

7.1. VMEBus

VMEbus; Multi bus, ISA bus, Micro bus gibi bir bilgisayar mimarisidir. VME teknolojisi "Versa Modular Eurocard" için kullanılan bir kısaltmadır ve ilk kez 1981 yılında, onu tanımlayan üretici grup, standardı tanımlamak için birlikte çalışan Motorola, Mostek ve Signetic firmalarından gelen uzmanlar tarafından oluşmuştu. "Bus" terminolojisi, dolayısıyla VMEbus terminolojisi, bilgisayarın veri yolunu tanımlayan genel bir terimdir.

VMEbus, Motorola firmasının kendi mikro işlemcisi olan 68000 için 1979 yılında tanımladığı VERSA bus adındaki eski bir standardın geliştirilmesiyle elde edilmiştir. VERSA bus günümüzde hala kullanılıyor olsa da, VMEbus'un sahip olduğu geniş kullanımalanına ulaşamamıştır. VERSA bus, başlangıçta Multibus, IBM-PC, STD bus, S-100, Q-bus ve diğer birçokları ile rekabet etmiştir.

Mikrobilgisayar bus (yol) endüstrisi mikroişlemcilerin gelişmesi ile başlamış ve 1981'de birçok bus kendini göstermiştir. Çoğu yalnızca bir ya da iki mikro işlemci tipi ile çalışabiliyordu ve dar bir adresleme aralığına sahip olmakla beraber oldukça yavaştılar. VMEbus mimarları, mikroişlemcilerden bağımsız, 16'dan 32 bit mikroişlemciye kolaylıkla yükseltilebilecek, güvenilir bir mekanik standarta sahip olan ve rekabet edebilecek ürünler üretebilen bağımsız üreticilerin doğmasına imkan verecek yeni bir bus'ın tanımlanması gereği ile yüklendiler. Üçüncü şahısların yeni ürünler geliştirmesine imkan verebilmek için, bu yeni bus'la ilgili olarak, herhangi bir kişi ya da firmaya herhangi bir hak tahsis edilmemiştir. Herkes, VMEbus tabanlı ürünleri, patent ya da lisans hakkı için herhangi ibr ücret ödemeden geliştirip, üretebilir.

İlgili çalışmaların birçoğu VERSA bus üzerinde yapıldığından bu, yeni bus için bir çerçeve teşkil etmiştir. Buna ek olarak, mekanik standart olarak "Eurocard" standardı seçilmiştir. Eurocard, IEC 603-2 konnektör, DIN 41494 ve IEC 297-2 rack standartları temel alınarak DIN 41612 standartına göre üretilen ürünleri tanımlayan bir terimdir. VMEbus ilk geliştirildiği zaman, Eurocard formatı Avrupa'da yıllardır kullanılan bir satandarttı. Mekanik donanımın büyük kısmını teşkil eden kart yuvaları, konnektörler ve alt rack grupları hazır durumdaydılar. Pin ve soket konnektörlerinin yapıları, baskılı devre kenar konnektörlerine (edge connectors) göre mekanik yapıya daha uyumludurlar.

Versa bus'ın elektriksel özellikleri ile Eurocard formatının birleşmesi VMEbus'ı ortaya çıkarmıştır. VME bus, telekomünikasyon, entrümantasyon kontrolü, robot uygulamaları, askeri sistemler, hava kontrol sistemleri, uzay teknolojisi ve benzer birçok alanda uygulama olanağı bulmasının yanısıra endüstriyel ve proses kontrol alanlarında PLC ve DCS sistemlerine de alternatif olmaya başlamıştır.

Bir sistem geliştirildiği zaman, planlayıcının üç seçeneği vardır: Kendi kartlarını üretmek, standart bir bilgisayarı uygun şekilde değiştirmek ya da standart mikrobilgisayar kartlarını kullanmak. Pazara girme zamanı, maliyet, miktar, teknoloji, uyugnluk ve bazen "biz donanım yapmaktayız" düşüncesi, seçimi etkileyen unsurlardır.

Belli bir müşteriye göre üretilen "custom system"ler, çok zaman alan ve pahalı olan sistemlerdir. Bunların temel avantajı, belli bir uygulamaya uygun olarak adapte edilebilmesi ve yüksek miktarlarda olduğunda ucuza mal edilmesidir.

Standart bir mini ya da mikrobilgisayarın adapte edilmesi, düşük miktarlı uygulamalarda fiyat açısından en etkili olanıdır. Standart bir sistemin adapte edilebileceği uygulamalarda, uygulamaya yönelik ürün geliştirmeye kalkışmak kaynakların heba olmasına sebep olmak demektir. Bu sistemin en büyük dezavantajı, birim fiyatlarının yüksek olması ve esnek olmamasıdır. Eğer özel fonksiyonlar, I/O ve mekanik paketleme gerekirse, standart sistemler esnekliklerini tamamen yitirirler.

Mikrobilgisayar kartlarının entegrasyonu, "custom" ve standart sistemler arasında yer alan bir çözümdür. Entegrasyon, pazarda bulunan parçalar bir araya getirilerek minimum maliyetle oluşturulur.

Mikrobilgisayar bordlarının diğer bir avantajı çok esnek olmalarıdır. Bunun sebebi standart kartların sisteme rahatlıkla eklenebilmesidir.

Mikrobilgisayar bus'ın kullanılmasına kadar verildiğinde, kullanılacak olanı belirlemek için seçim yapılmalıdır. Yeni tanıtılanlarla birlikte birçok bus bulunmaktadır. Seçim; uygulamaya bağlıdır ve pazara girme zamanı, maliyet, adet, teknoloji, uygunluk gibi sebeplerden etkilenir.

7.1.1. Özellikleri

VMEbus 32-bit adres, ki 4 gigabyte bellek demektir ve 32-bit veri yoluna sahiptir. Ayrıca çok işlemcili çalışmaya ve 7 kesme seviyesine imkan vermektedir. Adres ve veri bus'ları dinamik olarak şekillendirilebilir, yani boyutları otomatik olarak değiştirilebilir. Bu mikrobilgisayar teknolojisi ilerledikçe sistemin genişletilmesine imkan verir.

VMEbus master-slave mimarisini kullanır. Birçok master bus üstünde yer alabildiğinden buna multiprocessing bus denir. Bir master bus'a çıkmadan önce sistem kontrolörünün arbiter kısmından istekte bulunmak zorundadır. Arbiterin görevi bus'a hangi masterin çıkacağını belirlemektir.

Bütün bus aktivitesi 4 adet alt-bus tarafından idare edilir. Veri transferi Veri bus'ı ve Arbitrasyon da Veri Arbitrasyon bus'ı üzerinde gerçekleşir. Eğer interruplar kullanılıyorsa, bu Priority interrup bus tarafından kontrol edilir. Dördüncüsü, Utility bus denilen, 16 Mhz saat ve power-up reset gibi jenerik sinyalleri taşır.

7.1.2. Backplane

VMEbus modülleri, üzerinde 2'den 21 slot'a kadar backplaneler üzerinde bir araya getirilirler. Sistemin yapısına bağlı olarak bir ya da iki backplane birarada kullanılabilir.

P1/J1 backplane, 24 adres bit ve 16 veri biti ile birlikte power-ground ve bütün kontrol sinyallerini taşır. İkinci bir backplane olan P2/J2, VME'nin adres ve verisini 32 bite tamamlar. İki backplane, seçilen stile göre, ayrı ayrı ya da tek bir ünite olarak beraber monte edilebilir. [9]

7.2. OS-9 İşletim Sistemi

OS-9, Motorola 680X0 ailesinin mikroişlemcileri için geliştirilmiş, gerçek zamanlı (real-time) ve çok görevli (multi-tasking) bir işletim sistemidir. OS-9 maksimum esneklik sağlayacak şekilde modüler (modular) ve ölçeklenebilir (scalable) bir yapıya sahiptir, ayrıca 680X0 ailesinden olan tüm sistemlerde kullanılabilen tek işletim sistemidir. OS-9, ortak programlama dilleri ve yazılım araçları ile 680X0 tabanlı sistemlerin performansını arttırmada önemli rol oynar.

OS-9, gerçek zaman uygulamaları için yüksek performans, düşük maliyet ve modüler çözümler gerektiren bilimsel ve endüstriyel çevrelerde geniş bir kabul görmüştür.

OS-9 dünyadaki bütün bilgisayar ortamlarına erişebilmiştir. OS-9 bugün, proses kontrolü, görüntüleme, iletişim, havacılık, robotik, kişisel bilgisayarlar gibi çok çeşitli gerçek zaman uygulamaları için kullanılmaktadır. OS-9'ın blok yapı mimarisi, onu, küçük ROM tabanlı (ROM-based) bilgisayarlardan, geniş bilgisayar ağı tabanlı (network-based), çok kullanıcılı geliştirme sistemlerine kadar, 680X0 ailesi spektrumunda yeralan tüm sistemler için tek uygun işletim sistemi yapmaktadır.

7.2.1. Özellikleri

OS-9 maksimum etkinlik ve gerçek zaman uygulamalarından minimum sapmalar sağlayacak şekilde, dikkatle tasarlanmıştır. OS-9'da hızın önemli olduğu bölümler, optimum performans temini için 680X0 assembler dilinde kodlanmıştır. OS-9 çekirdeğinde (kernel) yapılandırılmış çeşitli özellikler, gerçek zamanlı programlamayı kolaylaştırmak için giriş/çıkış (I/O) sistemine birleştirilmiştir. Bu özellikler priority-based (öncelik tabanlı), pre-emptive task scdeduler (sıralı komut işlemi), hızlı interrupt service (kesme servisi) işlemleri, kontrol fonksiyonları ve esnek bir dizi interprocess communication (işlemler arası) iletişim imkanlarını içermektedir. En belirgin özellikleri kısaca açıklanacak olursa:

Multi-tasking: Multi-tasking (çok görevli) işletim sistemi, çok sayıda programın birarada çalışmasını sağlar. Bilgisayarın işlemcisi bir anda yalnız bir programı yürütebilir. Burada multi-tasking'in yaptığı; bir programı kısa bir süre çalıştırıp durdurmak ve diğer bir prıgramı çalıştırmak suretiyle, çok sayıda programı birlikte işletmektir.

Real-time: Real-time (gerçek zamanlı) terimi genllikle yanlış kullanılır ve yanlış anlaşılır. burada kastedilen, gerçek dünya olaylarının meydana gelmeleri esnasında işlem görmeleridir.Gerçek zamanlı sistem, bir dış olaya belirli bir süre içinde cevap vermelidir. Bu süre, uygulamadan uygulamaya ve uygulama içindeki durumların çeşidine göre çok değişiklikler gösterebilir.

ROMable: OS-9'ın tek bellek modülü yapısı, doğal olarak işletme sistemi ve uygulama programlarının ROMable olmasını gerektirir. Ayrıca modüllerin ROM içinde nerede bulunduklarına dair bir bilgiye ihtiyaç duyulmaz. OS-9 kernel'i (çekirdek) çalıştığında tüm ROM alanını tartar ve modüllerin nerede bulunduklarını gösteren bir directory kurar.

Modüler yapı: OS-9 işletim sistemi kendiliğinden bellek modüllerine ayrılabilir. Bu sayede, belli bir fonksiyon gerekmediğinde ilgili modül iptal edilebilir veya ek bir fonksiyon gerektiğinde başka modüller eklenebilir. [9]

TABLO 7.1. İşletim sistemlerinin karşılaştırılması

	OS-9	UNIX	DOS
Çok görevli (Multi-tasking)	evet	evet	hayır
Çok Kullanıcılı (Multi-user)	evet	evet	hayır
Gerçek Zamanlı (Real time)	evet	hayır	hayır
Moduler	evet	hayır	hayır
Geniş sistemlerde kullanılabilir	evet	evet	hayır
Kişiselbilgisayarlarda kullanılır	evet	hayır	evet
Bellek sınır	yok	yok	640K
Bellek yönetim birimi gerektirir	hayır	evet	hayır
ROMable	evet	hayır	hayır
Kullanıcının seçebileceği çekirdek (kernel)	evet	hayır	hayır
Kullanıcının seçebileceği I/O dosya yönetimi	evet	hayır	hayır
Donanım üreticileri ile uygunluk	hayır	evet	evet
Kullanılan işlemciler	Motorola Intel	birçok	Intel

SONUÇLAR VE ÖNERİLER

Endüstriyel kontrol donanımları ve kullanılan haberleşme imkanları ile ilgili yapılan bu çalışmada, bu sahada kullanılan çeşitli sistemler incelendi. Yaygın olarak kullanılan programlanabilir mantık denetleyicileri ile ilgili konular değerlendirildi.

Uygulama olarak, Siemens S5 115U programlanabilir kontrolörünün CP524 haberleşme işlemcisi ile kullandığı, 3964R protokolüne uygun olarak, RS232 seri bağlantısı üzerinden iletişim kuruldu.

Endüstriyel sahada mikrobilgisayar busı baz alan sistemlerin daha avantajları göz önünde bulundurularak; VMEbus tabanlı bilgisayar üzerinde çalışmalar yapıldı. Bu sistemin gerçek-zamanlı, çok-görevli ve modüler işletim sistemi yapısı, diğer işletim sistemleri ile karşılaştırılarak değerlendirildi. Kişisel bilgisayar ile PLC arasında kurulan seri iletişim VMEbus tabanlı mikrobilgisayar ile PLC arasında da tesis edildi. Çalışma sonucunda, programlanabilir denetleyicide belirlenen adreslerdeki değerlerin PC'den veya VMEbus tabanlı mikrobilgisayardan okunabilmesi veya değiştirilmesi imkanı sağlandı.

Bu çalışmalar, kontrol sisteminin diğer sistemler ve bilgisayar ağları ile bağlantı kurabileceği; daha açık bir sisteme geçiş için hazırlık teşkil etmektedir. Yazılımlar geliştirilerek, kullanımı daha elverişli (user-friendly) programlar oluşturulabilir.

KAYNAKLAR

[1] KAYNAK, OKYAY,M., Programlanabilir Denetleyiciler, Boğaziçi Üniversitesi Mühendislik Fakültesi Yayınları, 1988

[2] BAUER ,O., Kontrol Mühendisliğinin 5 köşesi, Türkiye'de ve Dünyada Otomasyon Dergisi, s.86-87,Kasım 1994

[3] BENNETT, S., Real-time Computer Control , Prentice Hall, 1988

[4] GUPTON, J. A., Computer Controlled Industrial Machines, Processes and Robots, Prentice Hall, 1986

[5] İNAL, İ., PLC Arabirim Ürünleri , Kaynak ELEKTRİK Uluslararası Enerji, Elektrik, Elektronik ve Otomasyon Mühendisliği Dergisi, s.110-111, Mayıs-Haziran 1994

[6] JOHANNESSON,G.,Translation: NICHOLL,Andrew G.McK., Programmable Control Systems, Chartwell- Bratt LTD., 1982

[7] ARAPKİRLİOĞLU, A., PLC Dünyasında Yeni Teknolojik Eğilimler, Türkiye'de ve Dünyada Otomasyon Dergisi,s.58-61, Ağustos 1994

[8] NARLI ,E., Fieldbus , Türkiye'de ve Dünyada Otomasyon Dergisi, s.90-94 Ekim 1994

[9] ÖZBOZKURT, U., FEYİZ ,H., VMEbus İle Tanışmak, Türkiye'de ve Dünyada Otomasyon Dergisi, s.94-96, Şubat 1995

[10] SIMATIC S5 - S5 115U Programmable Controller Manual

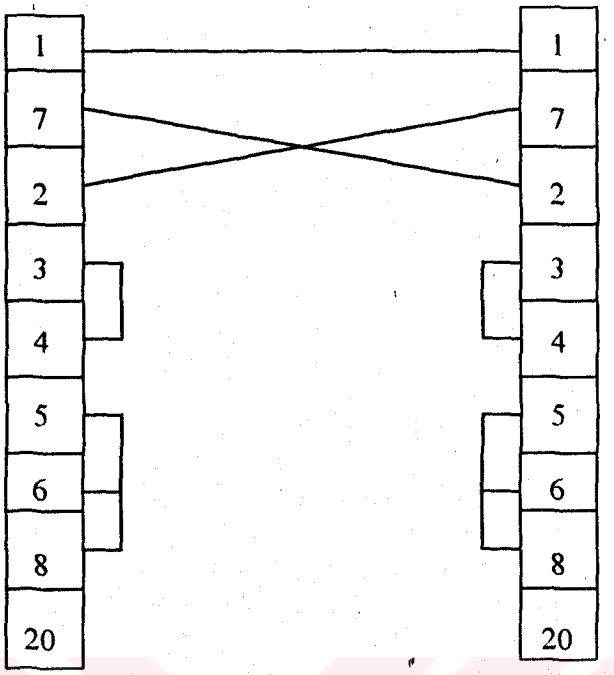
EK.A.

RS-232 Standardı

RS 232 konnektörün 25 ucundan yalnız 20 tanesi EIA (Elektronic Industry Association) tarafından tanımlanmış olup, bunlardan en çok kullanılanları Şekil A.1. de gösterilen 1-8 ve 20 nolu uçlardır. 4,5,6,8 ve 20 nolu uçlar "handshaking" işaretleridir. Bu tür bağlantıların minimum konfigürasyonu 3 telle Şekil A.2.de gösterildiği gibi yapılır.

EIA etiketi	Pin Nosu	İşlevi
AA	1	Koruyucu Toprak
AB	7	İşaret Toprağı
BA	2	Gönderilen Veri
BB	3	Alınan Veri
CA	4	Request To Send
CB	5	Clear To Send
CC	6	Data Set Ready
CF	8	Carier Detect
CD	20	Data Terminal Ready

Şekil A.1. RS 232 'de yaygın olarak kullanılan 9 uç



Şekil A.2. Üç telli RS 232 bağlantısı

EK-B.

PC ile Siemens PLC CP524 Haberleşme İşlemcisi arasında 3964R protokolüne uygun seri iletişimi sağlayan program listesi

```
/*****************************************************************************
 AÇIKLAMALAR: userprc.c, p3964r.def,menucmd.rc dosyalarını içeren p3964r.prj
proje dosyası WINDOWS ortamında derlenerek çalıştırılabilir. İletişim, 1 nolu seri kapıdan
COM524 üzerindeki seri kapıya  RS232 bağlantısı ile sağlanır. Program çalıştırıldığında ekrana
gelen seçeneklerden; PLC'de bir adresteki deger okunmak     istendiğinde READ seçeneği seçilir ve
okunacak verinin belirtilen adres değerleri girilir. PLC de bir adresteki değer değiştirilmek
istendiğinde WRITE seçeneği seçilir ve diyalog kutusuna içeriği değiştirilecek  adres ve yazılacak
olan değerler girilir.
*****************************************************************************/

/*****************************************************************/
/*  DOSYA ADI : userpr.c                                         */
/*              3964r protokolünü uygulayan program.             */
/*****************************************************************/
#include "p3964r.h"

#if _MSC_VER >= 700
#pragma warning (disable:4028)
#endif

HWND hWnd1;
HDC hDC;
PAINTSTRUCT ps;
DCB dcb;
COMSTAT cs;
char sonmesaj[100],text[150],szDataTypc[20]="DB";
int row=0,col,son,APP_ACTIVE=0,TIMER_ACT=1,chkport=0,ISLEM=0;
int chart,overf,cctime,cctime1,cctime2,cbas,komut,cBCC; //SAYACLAR
int yazdeger=0;
int DataTyp,DBno,DWno,Number;
int mess=1,outp,rtnval;
int  islem,say,vrb,wbc,TotalBytes,TotalChar,szHeader,idComDev,err,dc;
time_t tstart,tend;
char  BCC_cal,                  // hesaplanan BCC degeri
      BCC_gel,                  // okunan BCC degeri
      point[300],
      Buffer[300],
      datapnt[5000],            // gönderilecek verilerin yazıldığı poiter
      DATA[5000];               // gelen verilen yazıldığı pointer
char islem1,dchar;

int PASCAL WinMain(HANDLE hInstance,HANDLE hPrevInstance,LPSTR
                                        lpszCmdLine,int nCmdShow )
{
MSG msg;

if (!hPrevInstance)
{
   if ( !CommInit(hInstance) )
            return FALSE;
}
hWnd1 =CreateWindow("P3964R","3964R",WS_OVERLAPPEDWINDOW
         |WS_SYSMENU,0,0,800,600,NULL,NULL,hInstance,NULL);
while(!SetTimer(hWnd1,1,1,NULL))
          return FALSE;

ShowWindow(hWnd1, nCmdShow);
UpdateWindow(hWnd1);
```

```
while (GetMessage(&msg, NULL,0,0)){
    TranslateMessage(&msg);
    DispatchMessage(&msg);
}
return (int)msg.wParam;
}

/*************************************************************************/
/*  Function Name:  CommInit()                                           */
/*************************************************************************/
BOOL CommInit(HANDLE hInstance)
{
WNDCLASS wccomclass;

wccomclass.hCursor      = LoadCursor(NULL, IDC_ARROW);
wccomclass.hIcon        = LoadIcon(NULL, IDI_APPLICATION);
wccomclass.lpszMenuName  = "CGAPMENU";
wccomclass.lpszClassName = "p3964r";
wccomclass.hbrBackground = GetStockObject(WHITE_BRUSH);
wccomclass.hInstance     = hInstance;
wccomclass.style         = CS_VREDRAW | CS_HREDRAW;
wccomclass.lpfnWndProc   = ComWndProc;
wccomclass.cbClsExtra    = 0;
wccomclass.cbWndExtra    = 0;
if (!RegisterClass(&wccomclass))
   return FALSE;
return TRUE;
}

/*************************************************************************/
/*  Function Name : ComWndProc()                                         */
/*************************************************************************/
long FAR PASCAL ComWndProc(HWND hWnd,unsigned message,unsigned int
                                                wParam,LONG lParam)
{
static int ij;
static FARPROC lpfnMsgProc,lpfnMsgProc2;
static HWND hInst,hInst2;

switch(message){
        case WM_CREATE:
            hDC = GetDC(hWnd);
            hInst = ((LPCREATESTRUCT) lParam)->hInstance;
            lpfnMsgProc=(DLGPROC)
            MakeProcInstance((FARPROC)WS_AdressMsg,hInst);
            hInst2 = ((LPCREATESTRUCT) lParam)->hInstance;
            lpfnMsgProc2=(DLGPROC)
MakeProcInstance((FARPROC)WS_AdressMsg2,hInst2);
            Open_ComPorts();
            if( outp == 1){    // seri port acilamadi
                  sprintf(text,"COMPORT1 hatali secim");
                  MessageBox(hWnd,text,"ERROR",MB_OK);
                  TIMER_ACT = 1;
            }
            DBno=100;
            DWno=1;
            Number=20;
            DataTyp=DB;
            return TRUE;
```

```
case WM_TIMER:
            if( TIMER_ACT == 1){
                  break;}
            if ( APP_ACTIVE == 1){
                  TIMER_ACT = 1;
                  if (mess == 0){
                        tend=time(NULL);
                        sprintf(text,"gecen sure: %f s",difftime(tend,tstart));
                        TextOut(hDC,250,row,text,strlen(text));
                        DEGERLENDIR(outp);
                        sprintf(text,"sonuc:%s",sonmesaj);
                        TextOut(hDC,50,row,text,strlen(text));
                        row+=20;
                        if ((outp==10)&&(ISLEM==0x45)){
                           TextOut(hDC,0,row,"okunan degerler:",16);
                           col=10;
                           row+=20;
                           for(ij=0;ij<TotalBytes;ij++){
                                 sprintf(text,"%02X",DATA[ij]);
                                 TextOut(hDC,col,row,text,strlen(text));
                                 col+=30;
                                 if(col > 600){
                                   col=10;
                                   row+=20;
                            }
                             }
                        }
                        row+=40;
                        ISLEM=0;
                        APP_ACTIVE = 0;
                        break;
                  }
                     else{
                        Appl_period();
                        TIMER_ACT = 0;
                        break;
                     }
               }
            break;

case WM_COMMAND:
            if (row > 500)
                  row=0;
            switch(wParam){
                  case CM_READF:
                        ISLEM=0x45;
                           if ( APP_ACTIVE == 0){    //fetch işlemi
                              DialogBox(hInst2,(LPSTR)"DLG_ADRESS2",hWnd,
                                 lpfnMsgProc2);
                              DataTyp=FindDataTyp(szDataType);
                              if(DataTyp == 0){
                                 TextOut(hDC,400,row,"Taninmayan Data Tipi",20);
                                 row+=20;
                                 break;
                        }
                           PREAD();
                           APP_ACTIVE = 1;
                           mess=1;
```

```
                    TIMER_ACT = 0;
                    tstart=time(NULL);
                    break;
            }
                    break;
                case CM_WRITEF:
                    if ( APP_ACTIVE == 0){    // send islemi
                        DialogBox(hInst,(LPSTR)"DLG_ADRESS",hWnd,
                            lpfnMsgProc);
                        DataTyp=FindDataTyp(szDataType);
                        if(DataTyp == 0){
                            TextOut(hDC,400,row,"Taninmayan Data Tipi",20);
                            row+=20;
                            break;
                    }
                        PWRITE(0);
                        APP_ACTIVE = 1;
                        mess=1;
                        TIMER_ACT = 0;
                        tstart=time(NULL);
                    break;
                }
                        break;

                    case CM_TRANS:
                        if ( APP_ACTIVE == 0){    // send islemi
                            DialogBox(hInst,(LPSTR)"DLG_ADRESS",hWnd,
                                lpfnMsgProc);
                            DataTyp=FindDataTyp(szDataType);
                            if(DataTyp == 0){
                                TextOut(hDC,400,row,"Taninmayan Data Tipi",20);
                                row+=20;
                                break;
                            }
                            PWRITE(1);
                            APP_ACTIVE = 1;
                            mess=1;
                            TIMER_ACT = 0;
                            tstart=time(NULL);
                            break;
                        }
                        break;

                    case CM_HELPABOUT:
                        MessageBox(hWnd,szCMDWINAbout,"About Program"
                                ,MB_OK);
                        break;

                    case CM_U_EXIT:
                        Close_ComPorts();
                        DestroyWindow(hWnd);
                        break;

                    default :
                        break;
            }
        break;

case WM_PAINT:
```

```
            row=0;
            BeginPaint(hWnd, &ps);
            EndPaint(hWnd, &ps);
            break;

case WM_DESTROY:
          if (chkport == 0)
                    Close_ComPorts();
             PostQuitMessage(0);
             break;
          default:
             return( DefWindowProc(hWnd,message,wParam,lParam));
   }
return (0l);
}

/*************************************************************************/
/* Fonksyon Adı:    Open_ComPorts()                                      */
/* Amacı        :   Inolu seri kapıyı açar ve paramaetrelerini baud rate=9600,even   */
/*                  parity, 8 data, 1 stop bit olacak şekilde düzenler.  */
/*************************************************************************/
int Open_ComPorts()
{
int i;
idComDev = OpenComm("COM1",1000,1000);
if (idComDev < 0)
    outp=1;
err = BuildCommDCB("COM1:9600,e,8,1", &dcb);
if (err < 0)
    outp=1;
err = SetCommState(&dcb);
if (err < 0)
    outp=1;
return 0;
}
/*************************************************************************/
/* Fonksyon Adı:    Close_ComPorts()                                     */
/* Amacı        :   Seri kapıyı kapatır.                                 */
/*************************************************************************/
void Close_ComPorts(){
          CloseComm(idComDev);
}

/*************************************************************************/
/* Fonksiyon Adı :  PREAD()                                              */
/*  Amacı           : Okuma işlemi için mesajın başlığını hazırlar.      */
/*************************************************************************/
void PREAD()
{
islem=1;
dc=0;
// datanın uzunluğu byte cinsinden belirleniyor
TotalBytes=Number;
if ((DataTyp == DB)||(DataTyp == DX)||(DataTyp == CB)||(DataTyp == TB)||
   (DataTyp == RS)||(DataTyp == AS))
    TotalBytes=Number*2;
overf=0;
// HEADER mesajı oluşturuluyor
Buffer[0]=0;
```

```
Buffer[1]=0;
Buffer[2]=0x45;
Buffer[3]=DataTyp;
if((DataTyp == DB)||(DataTyp == DX)){
        Buffer[4]=DBno;
        Buffer[5]=DWno;
}
else {
        Buffer[4]=(DWno & 0xff00) >> 8;
        Buffer[5]= DWno & 0xff;
}
Buffer[6]=(Number & 0xff00) >> 8;
Buffer[7]= Number & 0xff;
Buffer[8] = 0xff;
Buffer[9] = 0x1f;
Buffer[10] = 0x10;
Buffer[11] = 0x03;
Buffer[12] = CAL_BCC(Buffer,12);   //HEADER'in BCC degeri hesaplandi
szHeader=13;
// farkli durumlara ait sayaclar set edildi
say=0;chart=1;cctime=1;cctime1=1;cctime2=1;cbas=1;komut=1;cBCC=1;
}
/*************************************************************************/
/*  Fonksiyon Adi :  PWRITE()                                            */
/*  Amacı         :  Okuma işlemi için mesajın başlığını hazırlar.       */
/*************************************************************************/
void PWRITE(int secim)
{
int i,k;

TotalChar=Number;
if ((DataTyp == DB)||(DataTyp == DX)||(DataTyp == CB)||(DataTyp == TB)||
   (DataTyp == RS)||(DataTyp == AS))
            TotalChar=Number*2;
YAZILACAKDEGER();
overf=0;
Buffer[0]=0;
Buffer[1]=0;
if ( secim == 0){
        Buffer[2]=0x41;
        islem=2;
}
if ( secim == 1){
        Buffer[2]=0x4F;
        islem=3;
}
Buffer[3]=DataTyp;

if((DataTyp == RS)||(DataTyp == AS)){
        Buffer[4]=(DWno & 0xff00) >> 8;
        Buffer[5]= DWno & 0xff;
}
else {
        Buffer[4]=DBno;
        Buffer[5]=DWno;
}
Buffer[6]=(Number & 0xff00) >> 8;
Buffer[7]= Number & 0xff;
Buffer[8] = 0xff;
```

```
Buffer[9] = 0x1f;
i=9;
wbc=0;
for( k=0;k < TotalChar ;k++){
        if ( k > 127)
       break;
        Buffer[++i]=datapnt[k];
       wbc++;
        if (datapnt[k] == 0x10 )
                Buffer[++i]=0x10;
}
Buffer[++i] = 0x10;
Buffer[++i] = 0x03;
Buffer[++i] = CAL_BCC(Buffer,i);
szHeader=i+1;
vrb=wbc;
// sayaclarin baslangic degerleri
say=0;chart=1;cctime=1;cctime1=1;cctime2=1;cbas=1;komut=1;
}

/*************************************************************/
/*  Fonsiyon adi:    YAZILACAKDEGER(void)                    */
/*  Amacı            :  verilen adreslere yazılacak degeri hesaplar.  */
/*************************************************************/
YAZILACAKDEGER(void)
{
int j;

for (j=0;j<TotalChar;j++){
        datapnt[j]=(yazdeger & 0xff00) >> 8;
    j++;
        datapnt[j]=yazdeger & 0x00ff;
}
}
/*************************************************************/
/* Fonksiyon Adı:          Appl_period()                     */
/* Amacı          : Data alış verişini düzenler.             */
/*************************************************************/
char Appl_period()
{
int i,j,k;

   switch (mess){
           case 1 :                // STX gönder
                   TransmitCommChar(idComDev,STX);
                   mess=2;
                if (komut != 8)
                     komut=1;
                   break;

           case 2 :
                   ReadDLE();
                   break;

           case 3 :                        // Header gönder
                   WriteComm(idComDev,Buffer,szHeader);
                   komut=3;
                   mess=2;    // DLE oku
                   break;
```

```
case 4 :        // STX oku
        GetCommError(idComDev,&cs);
        if (cs.cbInQue <= 0){
                ++cctime1;
                if (cctime1 > ZVZ){
                    outp=HataMesaj(ERROR3);
                 mess=0;
                    break;
                }
                break;
        }
        ReadComm(idComDev,point,cs.cbInQue);
        for (i=0;i<cs.cbInQue;i++){
                 if( point[i] == STX  ){
                    mess=5;       // STX geldi ,DLE gönder
                    break;
                 }
                 if( point[i] == NAK  ){
                        TransmitCommChar(idComDev,DLE);
                        TransmitCommChar(idComDev,NAK);
                 }
                 if( cctime2 > RPT  ){
                        outp=HataMesaj(ERROR3);
            mess=0;
                        break;
                  }
                    else{
                        ++cctime2;
                        mess=1;
                        break;
                    }
        }

case 5 :        // DLE gönder
        TransmitCommChar(idComDev,DLE);
        mess=6;
        break;

case 6 :        //  gelen bilgiler okunuyor
        GetCommError(idComDev,&cs);
        if (cs.cbInQue > 0){
                chart=1;
                ReadComm(idComDev,point,cs.cbInQue);
                for (j=0;j<cs.cbInQue;j++){
                        Buffer[say++]=point[j];
                        overf++;
          }
             for (j=say-cs.cbInQue;j<say;j++){
                   if (Buffer[j]== DLE){
                          if (Buffer[j+1]== ETX){  //ardarda DLE ETX geldi
                                mess=7;
                           overf=0;
                                break;
                           }
                     }
                }
                if (overf >256){
                        outp=HataMesaj(ERROR5);
```

```
                    mess=0;
                }
                    break;
                }
                else{  // giris boş
                    chart++;
                     if (chart > ZVZ){
                        outp=HataMesaj(ERROR5);
                    mess=0;
                        break;
                     }
                     break;
            }
case 7 :
            if (Buffer[3] != 0 ){       // PLC nin cevap mesajı kontrol ediliyor
                outp=0xff00 | Buffer[3];
                    TransmitCommChar(idComDev,DLE);
                    TransmitCommChar(idComDev,NAK);
                    outp=0xff00 | Buffer[3];
                    mess=0;
                    break;
            }
            else{
                    BCC_gel=Buffer[say-1];   // okunan BCC değeri
          BCC_cal=CAL_BCC(Buffer,say-1);
                                    // gelen datanın BCC değeri hesaplandı
                 if (BCC_cal == BCC_gel){       // data doğru olarak okundu
                    TransmitCommChar(idComDev,DLE);  // DLE gonder
                    if (islem==1){
                          err=Select_Data(Buffer,say-3);
                          if (err  == 0){ // datanın okunması tamamlandı
                        outp=10;
                              mess=0;
                              break;
                               }
                               else{          // Datanın uzunlugu 128 byte'ı aştı.
                                                Ek mesaj gönder
                                    mess=8;
                                    break;
                               }
                    }
                    else if((islem==2)| (islem==3)){
                              if ((TotalChar - wbc) == 0){
                            outp=10;
                                    mess=0;
                              break;
                               }
                               else{
                                  mess=8;"
                                  break;
                               }
                    }
            }
          else{  //  data doğru olarak okunamadı
                    ++cBCC;
                    outp=HataMesaj(ERROR6);
                    if ( cBCC > RPT ){
                    mess=0;
                       break;
```

```
                    }
                    mess=1;say=0;cctime=1;cctime1=1;cctime2=1;  cbas=1;
                komut=1;
                    break;
                }
            }
        case 8:  /* datanın uzunluğu 128 byte dan uzun : ek mesaja başla*/
                mess=1;say=0;cctime=1;cctime1=1;cctime2=1;cbas=1;komut=8;
         cBCC=1;
                break;

        case 9:  /* ek header mesajı  */
                Buffer[0]=0xff;
                Buffer[1]=0;
                if (islem == 1){
                        Buffer[2]=0x45;
                        Buffer[3]= DataTyp;
                        Buffer[4]=0x10;
                        Buffer[5]=0x03;
                        Buffer[6] = CAL_BCC(Buffer,6);
                        WriteComm(idComDev,Buffer,7);
                        komut=3;
                        mess=2;
                        break;
                }
               else{
                    if (islem == 2)
                        Buffer[2]=0x41;
                    if (islem == 3)
                        Buffer[2]=0x4F;
                    Buffer[3]= DataTyp;
                i=3;
                    for( k=vrb;k<TotalChar;k++){
                        if ( k > (vrb+127) )
                        break;
                      Buffer[++i]=datapnt[k];
                      wbc++;
                        if (datapnt[k] == 0x10 )
                                Buffer[++i]=0x10;
                    }
                    Buffer[++i] = 0x10;
                    Buffer[++i] = 0x03;
                    Buffer[++i] = CAL_BCC(Buffer,i);
                    WriteComm(idComDev,Buffer,i+1);
                vrb=wbc;
                    komut=3;
                    mess=2;   // DLE oku ya git
                    break;
                }
        default:
                break;
    } //  end of mess switch

return 0;
}
```

```
/*************************************************************************/
/* Fonksiyon Adi  : HataMesaj();                                         */
/* Parametreler   : int hatano (olusan hatanin numarasi)                 */
/* Sonuc Degeri   : int      ( islem sonucunu  belirten sayi)            */
/* Amaci          : PREAD ve PWRITE islemlerinin nasil sonuclanacagini belirler */
/*************************************************************************/
int HataMesaj(int hatano)
{
int sonuc;

TransmitCommChar(idComDev,DLE);
TransmitCommChar(idComDev,NAK);
switch(hatano){
        case ERROR1:    sonuc=4;        break;
        case ERROR2:    sonuc=5;        break;
        case ERROR3:    sonuc=3;        break;
        case ERROR5:        sonuc=6;    break;
        case ERROR6:    sonuc=9;        break;
}
return sonuc;
}
/*************************************************************************/
/* Fonksiyon Adi  : DEGERLENDIR();                                       */
/* Parametreler   : int sonuc ( PREAD veya PWRITE islemlerinin sonuc degeri) */
/* Amaci          : PREAD ve PWRITE islemlerinin nasil sonuclandigini aciklar */
/*************************************************************************/
void DEGERLENDIR(int sonuc)
{
if (sonuc == 1)
    strcpy(sonmesaj,"OpenCommError ");
else if (sonuc == 3)
    strcpy(sonmesaj,"ZVZtimeout.STX gelmedi");
else if (sonuc == 4)
    strcpy(sonmesaj,"DLE gelmedi.NAK");
else if (sonuc == 5)
    strcpy(sonmesaj,"QVZtimeout.DLE gelmedi");
else if (sonuc == 6)
    strcpy(sonmesaj,"ZVZtimeout.DATA");
else if (sonuc == 9)
    strcpy(sonmesaj,"BCC Error ");
else if (sonuc == 10)
    strcpy(sonmesaj,"Islem Dogru ");
else
    sprintf(sonmesaj,"ReplyError%02x",sonuc & 0xff);
}
/*************************************************************************/
/* Fonksiyon Adi  : CAL_BCC();                                           */
/* Parametreler   : char datapnt[500] ( BCC degeri hesaplanacak dizi )   */
/*                  int  size        ( dizinin uzunlugu (byte) )         */
/* Sonuc Degeri   : char     ( data dizisinin BCC degeri)                */
/* Amaci          : datapnt ile verilen dizinin BCC  degerini hesaplar   */
/*************************************************************************/
char CAL_BCC(char datap1[200],int size )
{
char BCC_r=0;
int i;
char c[8]={0,0,0,0,0,0,0,0};
```

```
for (i=0;i<size;i++){
        if ( (datap1[i] & 0x01) == 1 ) c[0] +=1;
        if ( (datap1[i] & 0x02) == 2 ) c[1] +=1;
        if ( (datap1[i] & 0x04) == 4 ) c[2] +=1;
        if ( (datap1[i] & 0x08) == 8 ) c[3] +=1;
        if ( (datap1[i] & 0x10) == 16 ) c[4] +=1;
        if ( (datap1[i] & 0x20) == 32 ) c[5] +=1;
        if ( (datap1[i] & 0x40) == 64 ) c[6] +=1;
        if ( (datap1[i] & 0x80) == 128) c[7] +=1;
}
for (i=0;i<8;i++){
   if((c[i] % 2)==1)
        c[i]=1;
   else
        c[i]=0;
}
for (i=0;i<8;i++)
BCC_r += c[i]*( 0x01 << i) ;
return BCC_r;
}

/*************************************************************************/
/* Fonksiyon Adi  : Select_Data();                                       */
/* Parametreler   : char datap[300] ( gelen DATA dizisi )                */
/*                  int son       (gelen data dizisinin uzunlugu)        */
/* Sonuc Degeri   : int   ( -Datanin okunmasi bitmediyse kalan datanin uzunlugu */
/*                          -Datanin okunmasi tamamlandiysa 0 degeri ile doner ) */
/* Amaci          : PREAD  islemi ile okunan datayi kaydeder             */
/*************************************************************************/
int Select_Data(char datap[300],int son)
{
int i,fark;

for (i=4;i<son;i++){
     DATA[dc]=datap[i];
     ++dc;
     if ( datap[i] == 0x10 )
          ++i;
}
fark=TotalBytes-dc;
return fark;
}

/*************************************************************************/
/* Fonksyon Adı:   ReadDLE()                                             */
/* Amacı       :   Seri kapıya gelen değerler arasında DLE=hex.10 karakteri gelip */
/*                 gelmediğini kontrol eder.                             */
/*************************************************************************/
char ReadDLE()
{
int i;

GetCommError(idComDev,&cs);
if (cs.cbInQue <= 0){
    ++cctime;
    if (cctime > QVZ){
            ++cbas;
            if (cbas > RPT ){
                    outp=HataMesaj(ERROR2);
```

```
                mess=0;
                    return 0;
                }
        }
        mess=2;
        return 0;
}
ReadComm(idComDev,point,cs.cbInQue);
cctime=1;       // reset QVZ counter: timeout yok
for (i=0;i<cs.cbInQue;i++){
   if( point[i]==DLE){
      if( komut == 1){   // Header a git
                mess=3;
                return 0;
      }
      if( komut == 3){
                mess=4;   // STX oku ya git
                return 0;
      }
      if ( komut == 8){  // ek header a git
                mess=9;
                return 0;
      }
   }
   if( point[i]== NAK){
            TransmitCommChar(idComDev,DLE);
            TransmitCommChar(idComDev,NAK);
   }
}
if( cbas > RPT ){
     outp=HataMesaj(ERROR1);
     mess=0;
     return 0;
}
++cbas;
mess=1;
return 0;
}

/************************************************ /
/*                  FindDataTyp()                 */
/************************************************/
int FindDataTyp(char *strn)
{
int sonuc;

if (strcmp(strn,"DB")==0 ){
   sonuc=DB;
   return sonuc;
}
else if (strcmp(strn,"FY")==0 ){
   sonuc=FY;
   return sonuc;
}
else if (strcmp(strn,"IB")==0 ){
   sonuc=IB;
   return sonuc;
}
else if (strcmp(strn,"QB")==0 ){
```

```
    sonuc=QB;
    return sonuc;
}
else if (strcmp(strn,"PB")==0 ){
    sonuc=PB;
    return sonuc;
}
else if (strcmp(strn,"TB")==0 ){
    sonuc=TB;
    return sonuc;
}
else if (strcmp(strn,"CB")==0 ){
    sonuc=CB;
    return sonuc;
}
else if (strcmp(strn,"RS")==0 ){
    sonuc=RS;
    return sonuc;
}
else if (strcmp(strn,"AS")==0 ){
    sonuc=AS;
    return sonuc;
}
else if (strcmp(strn,"DX")==0 ){
    sonuc=DX;
    return sonuc;
}
else if (strcmp(strn,"OB")==0 ){
    sonuc=OB;
    return sonuc;
}
return 0;
}
/*********************************************************************/
/*  Fonksyon Adı: WS_AdressMsg()                                     */
/*   Amacı        : WRITE veya WRITEEX menüsü seçildiğinde kullanılacak olan  */
/*                  diyalog kutusunu  hazırlar.                     */
/*********************************************************************/
BOOL FAR PASCAL WS_AdressMsg(HWND hWndDlg, WORD Message,WORD
                                    wParam, LONG lParam)
{
 BOOL lpfTranslated;

 switch(Message)
 {
   case WM_INITDIALOG:
      SetDlgItemText(hWndDlg,DLG_DTYPE,szDataType);
      SetDlgItemInt(hWndDlg,DLG_DBNO,DBno,FALSE);
      SetDlgItemInt(hWndDlg,DLG_DWNO,DWno,FALSE);
      SetDlgItemInt(hWndDlg,DLG_LENGTH,Number,FALSE);
      SetDlgItemInt(hWndDlg,DLG_VALUE,yazdeger,FALSE);
      break;
   case WM_CLOSE:
      PostMessage(hWndDlg, WM_COMMAND, IDCANCEL, 0L);
      break;
   case WM_COMMAND:
      switch(wParam)
      {
            case IDOK:
```

```
            GetDlgItemText(hWndDlg,DLG_DTYPE,szDataType,10);
            DBno =GetDlgItemInt(hWndDlg,DLG_DBNO,&lpfTranslated,FALSE);
            DWno =GetDlgItemInt(hWndDlg,DLG_DWNO,&lpfTranslated,FALSE);
            Number=GetDlgItemInt(hWndDlg,DLG_LENGTH,&lpfTranslated,FALSE);
            yazdeger=GetDlgItemInt(hWndDlg,DLG_VALUE,&lpfTranslated,FALSE);
            if(Number >2048)
                        Number=2048;
            EndDialog(hWndDlg, TRUE);
                break;
        case IDCANCEL:
                EndDialog(hWndDlg, FALSE);
                break;
        }
        break;
    default:
        return FALSE;
 }
 return TRUE;
}
/*********************************************************************/
/* Fonksyon Adı: WS_AdressMsg2()                                     */
/* Amacı       : READ menüsü seçildiğinde kullanılacak olan diyalog  */
/*               kutusunu hazırlar.                                  */
/*********************************************************************/
BOOL FAR PASCAL WS_AdressMsg2(HWND hWndDlg, WORD Message,WORD
                                        wParam, LONG lParam)
{ BOOL lpfTranslated;
switch(Message) {
   case WM_INITDIALOG:
      SetDlgItemText(hWndDlg,DLG_DTYPE,szDataType);
      SetDlgItemInt(hWndDlg,DLG_DBNO,DBno,FALSE);
      SetDlgItemInt(hWndDlg,DLG_DWNO,DWno,FALSE);
          SetDlgItemInt(hWndDlg,DLG_LENGTH,Number,FALSE);
          break;
   case WM_CLOSE:
      PostMessage(hWndDlg, WM_COMMAND, IDCANCEL, 0L);
      break;
   case WM_COMMAND:
      switch(wParam)
      {
           case IDOK:
              GetDlgItemText(hWndDlg,DLG_DTYPE,szDataType,10);
              DBno =GetDlgItemInt(hWndDlg,DLG_DBNO,&lpfTranslated,FALSE);
              DWno =GetDlgItemInt(hWndDlg,DLG_DWNO,&lpfTranslated,FALSE);
              Number=GetDlgItemInt(hWndDlg,DLG_LENGTH,&lpfTranslated,FALSE);
              if(Number >2048)
                        Number=2048;
              EndDialog(hWndDlg, TRUE);
               break;
          case IDCANCEL:
               EndDialog(hWndDlg, FALSE);
               break;
          }
          break;
          default:
             return FALSE;
 }
 return TRUE;
}
```

```
/*****************************************************************/
/*  DOSYA ADI : p3964r.def                                       */
/*              program tanım dosyası.                           */
/*****************************************************************/
NAME         PiCOMDrv

DESCRIPTION    '3964r Driver Program '

EXETYPE      WINDOWS

CODE         PRELOAD MOVEABLE DISCARDABLE
DATA         PRELOAD MOVEABLE MULTIPLE

HEAPSIZE     2048
STACKSIZE    8192

EXPORTS      ComWndProc     @1
             WS_AdressMsg   @2
             WS_AdressMsg2  @3

/*****************************************************************/
/*  DOSYA ADI : menucmd.rc                                       */
/*              menülerin tanımlandığı dosya.                    */
/*****************************************************************/
#include "p3964r.h"

CGAPMENU MENU
BEGIN
        MENUITEM "&Read",       CM_READF
        MENUITEM "&Write",      CM_WRITEF
        MENUITEM "Write&EX",    CM_TRANS
        MENUITEM "E&xit",       CM_U_EXIT

        POPUP "\a&Help"
        BEGIN
                MENUITEM "&About", CM_HELPABOUT
        END

END

#include "p3964r.dlg"

/*****************************************************************/
/*  DOSYA ADI : p3964.dlg                                        */
/*              diyalog kutuları dosyası.                        */
/*****************************************************************/
1 DLGINCLUDE "P3964R.H"

DLG_ADRESS DIALOG 6, 18, 120, 105
STYLE DS_MODALFRAME | WS_POPUP | WS_VISIBLE | WS_CAPTION | WS_SYSMENU
CAPTION "Hedef Adres Bilgisi"
FONT 8, "MS Sans Serif"
BEGIN
   LTEXT        "Data Turu:",     -1, 6, 5,  80, 8
   LTEXT        "Data Blok No:",  -1, 6, 20, 80, 8, NOT WS_GROUP
   LTEXT        "Ilk Word No:",   -1, 6, 35, 80, 8, NOT WS_GROUP
   LTEXT        "Data adedi:",    -1, 6, 50, 80, 8, NOT WS_GROUP
   LTEXT       "Yazilacak deger:",-1, 6, 65, 80, 8, NOT WS_GROUP
```

```
  EDITTEXT      DLG_DTYPE, 70, 3, 32, 12, ES_AUTOHSCROLL  | WS_GROUP
  EDITTEXT      DLG_DBNO,  70, 18, 32, 12, ES_AUTOHSCROLL  | WS_GROUP
  EDITTEXT      DLG_DWNO,  70, 33, 32, 12, ES_AUTOHSCROLL  | WS_GROUP
  EDITTEXT      DLG_LENGTH,70, 48, 32, 12, ES_AUTOHSCROLL | WS_GROUP
  EDITTEXT      DLG_VALUE, 70, 63, 32, 12, ES_AUTOHSCROLL | WS_GROUP
  PUSHBUTTON    "Ok", IDOK, 70, 85, 40, 14, WS_GROUP
END

DLG_ADRESS2 DIALOG 6, 18, 120, 105
STYLE DS_MODALFRAME | WS_POPUP | WS_VISIBLE | WS_CAPTION | WS_SYSMENU
CAPTION "Kaynak Adres Bilgisi"
FONT 8, "MS Sans Serif"
BEGIN
  LTEXT         "Data Turu:",     -1, 6, 5,  80, 8
  LTEXT         "Data Blok No:",  -1, 6, 20, 80, 8, NOT WS_GROUP
  LTEXT         "Ilk Word No:",   -1, 6, 35, 80, 8, NOT WS_GROUP
  LTEXT         "Data adedi:",    -1, 6, 50, 80, 8, NOT WS_GROUP
  EDITTEXT      DLG_DTYPE, 70, 3, 32, 12, ES_AUTOHSCROLL  | WS_GROUP
  EDITTEXT      DLG_DBNO,  70, 18, 32, 12, ES_AUTOHSCROLL  | WS_GROUP
  EDITTEXT      DLG_DWNO,  70, 33, 32, 12, ES_AUTOHSCROLL  | WS_GROUP
  EDITTEXT      DLG_LENGTH,70, 48, 32, 12, ES_AUTOHSCROLL | WS_GROUP
  PUSHBUTTON    "Ok", IDOK, 70, 85, 40, 14, WS_GROUP
END

/*********************************************************************/
/*  DOSYA ADI : p3964.h                                              */
/*              header dosyası.                                      */
/*********************************************************************/
#include <string.h>
#include <stdio.h>
#include <time.h>
#include <windows.h>

#define QVZ 2000
#define ZVZ 200
#define RPT 6

#define STX 0x02
#define DLE 0x10
#define ETX 0x03
#define NAK 0x15

#define ERROR1 1
#define ERROR2 2
#define ERROR3 3
#define ERROR4 4
#define ERROR5 5
#define ERROR6 6
#define ERROR7 7

#define DB 0x44    /* data block          */
#define FY 0x4D    /* flag bytes          */
#define IB 0x45    /* input bytes         */
#define QB 0x41    /* output bytes        */
#define PB 0x50    /* I/O bytes           */
#define TB 0x54    /* time counter        */
#define CB 0x5A    /* counter location    */
#define RS 0x42    /* system data         */
#define AS 0x53    /* absolute addresses  */
```

```
#define DX 0x58     /* extended data blocks  */
#define OB 0x51     /* extended I/O's        */

#define CM_READF   100
#define CM_WRITEF  101
#define CM_TRANS   102
#define CM_U_EXIT  103
#define CM_HELPABOUT 104

// **** connect dialog box controls
#define DLG_DTYPE          401
#define DLG_DBNO           402
#define DLG_DWNO           403
#define DLG_LENGTH         404
#define DLG_VALUE          400

#define  szCMDWINAbout "PC-PLC 3964r Driver\n 1995 PiOMAK\n
                                                    written by Hacer Feyiz"
// Fonksyonların bildirimi
int Open_ComPorts();
void Close_ComPorts();
void PREAD();
void PWRITE(int secim);
char CAL_BCC(char datapnt[500],int size );
int  Select_Data(char datapnt[500],int son);
int  HataMesaj(int hatano);
void DEGERLENDIR(int sonuc);
char Appl_period();
char ReadDLE();
int FindDataTyp(char *strn);
YAZILACAKDEGER(void);
BOOL CommInit(HANDLE hInstance);
long FAR PASCAL ComWndProc(HWND hWnd,unsigned message,unsigned int
                                    wParam,LONG lParam);
BOOL FAR PASCAL WS_AdressMsg(HWND hWndDlg, WORD Message,WORD wParam,
                                    LONG lParam);
BOOL FAR PASCAL WS_AdressMsg2(HWND hWndDlg, WORD Message,WORD
                                    wParam, LONG lParam);
```

ÖZGEÇMİŞ

Hacer FEYİZ. 1969 yılında İstanbul'da doğdu. İlk ve orta öğrenimini İstanbul'da yaptı. 1987 yılında Kadıköy İmam Hatip Lisesi'nden mezun oldu. Aynı yıl İ.T.Ü. Uçak ve Uzay Bilimleri Fakültesi, Uzay Bilimleri ve Teknolojisi Bölümüne girdi. 1991'de Yüksek Lisans İngilizce Hazırlık programına, 1992 yılında Yüksek Lisans Programına başladı.