

T.C.

GEBZE YÜKSEK TEKNOLOJİ ENSTİTÜSÜ

MÜHENDİSLİK VE FEN BİLİMLERİ ENSTİTÜSÜ

VANA OTOMASYONU

ORÇUN DURSUN

YÜKSEK LİSANS TEZİ

TASARIM VE İMALAT MÜHENDİSLİĞİ ANABİLİMDALİ

DANIŞMANI

YRD. DOÇ. DR. İLYAS KANDEMİR

GEBZE

2008

ÖZET

TEZİN BAŞLIĞI: VANA OTOMASYONU

YAZAR ADI: ORÇUN DURSUN

Bu çalışmada vanaların kontrol mekanizmaları ele alınmıştır. Çalışmada bir vana otomasyonu tasarımı için gerekli olduğu düşünülen konular ele alınmıştır. Bununla birlikte bir vana sistemi alınarak, servo, step ve DC motorlar kullanılmak üzere vana kontrol sistemleri tasarlanmıştır. Motor kontrolü için gereken H-Köprülerinden üç çeşit kullanıldığı ve servo motor kullanılan tasarım için H-Köprüsü kullanılmadığı için aslen 4 tasarım yapılmıştır.

Vana kontrol sistemi tasarımı için gerekli olduğu düşünülen konulardan, önce vanaların sınıflandırılması incelenmiştir. Daha sonra vanada akış konusu ele alınmıştır. Bunlar ikinci ve üçüncü bölümlerdir.

Çalışma, tasarım için gerekli konuların tartışılmasından sonra tasarlanan devrelerin dördüncü bölümde incelenmesi bu devrelere ait elemanların beşinci bölümde ele alınması ile devam etmektedir. Ayrıca bu çalışmada, kontrol devresi için yapılan yazılımların kodları ekler bölümünde verilmiştir fakat bu kodların anlamları beşinci bölümün sonunda ele alınmıştır.

Bu çalışmada bir vana sistemi alınmış, değişik kontrol mekanizmaları tasarlanmış, yazılımı gerçekleştirilmiş, mikrodenetleyecilere yüklenmiş, devresi yapılmış ve denenmiştir.

SUMMARY

NAME OF THE THESIS: VALVE AUTOMATION

NAME OF THE WRITER: ORÇUN DURSUN

This study focuses on the valve control mechanisms. In this piece of study the issues which are considered important while designing valve control systems are discussed. Also a valve system is taken and control systems are designed, using servo, step and normal DC motors. Three types of H-Bridges are used for the designs (one for each) except one which uses servo motor. So, four designs are produced for this study.

The first issue that is discussed in this work that is thought to be important for valve control system design is the classification of valves. Then the issue of flow through valves is handled. These are in parts two and three in the study.

The study continues by analyzing the circuits that are designed for this study in part four and discussing the hardware used in these circuits in part five. Also the software designed for this study is give in the appendix part but the meaning of the algorithms is given at the end of Part Five.

In this study a valve system is taken. A control mechanism is designed. The software for this design is created and loaded on the microcontoler. The curcuit for this design is made and finally the design is tested.

TEŞEKKÜR

Bu çalışmayı yapmamı sağlayan Sayın Yrd. Doç. Dr. İlyas Kandemir'e teşekkür ederim.

İÇİNDEKİLER DİZİNİ

ÖZET	iv
SUMMARY	v
TEŞEKKÜR	vi
İÇİNDEKİLER	vii
SİMGELER VE KISALTMALAR DİZİNİ	xi
ŞEKİLLER DİZİNİ	xiv
ÇİZELGELER DİZİNİ	xvii

1. GİRİŞ	1
2. VANALARIN SINIFLANDIRILMASI	3
2.1 Sızdırmazlık Kriterine Göre Sınıflandırma	3
2.2 Vana Yataklamasına Göre Sınıflandırma	3
2.2.1 Metal Yataklama	3
2.2.2 Yumuşak Yataklama	4
2.3 Vana Stem Salmastırasına Göre Sınıflandırma	4
2.4 Vana Genel Yapısına Göre Sınıflandırma	5
2.4.1 Manuel Vanalar	5
2.4.1.1 <i>Gate</i> Vanalar	7
2.4.1.2 <i>Glob</i> Vanalar	10
2.4.1.3 <i>Piston</i> Vanalar	12
2.4.1.4 <i>Plug</i> Vanalar	13
2.4.1.5 <i>Küresel</i> Vanalar	15
2.4.1.6 <i>Kelebek</i> Vanalar	16
2.4.1.7 <i>Pinch</i> Vanalar	17
2.4.1.8 <i>Diyafram</i> Vanalar	18
2.4.2 Çek Vanalar	19
2.4.2.1 <i>Lift</i> Çek Vanalar	20
2.4.2.2 <i>Swing</i> Çek Vanalar	21
2.4.2.3 <i>Tilting Disk</i> Çek Vanalar	21
2.4.2.4 <i>Diyafram</i> Çek Vanalar	22
2.4.3 <i>Pressure Relief</i> Vanalar	22

2.4.4 Patlama Kapakları	25
3. VANADA AKIŞ	27
3.1 Sürtünme Katsayısı	27
3.2 Akış Katsayısı K_v	34
3.3 Akış Katsayısı A_v	34
3.4 Vananın Açıklığıyla Sürtünme Katsayısı Arasındaki İlişki	35
3.5 Vanalardaki Kaviteasyon	37
3.6 Vanalardaki Su Çekici Olayı	40
3.7 Vana Sesinin Azaltılması	43
4. VANA TAHRİK DEVRESİ	45
4.1 Servo Motor Devresi	45
4.2 Step Motor Devresi	46
4.3 L293D – DC Motor Devresi	47
4.4 LMD18200 – DC Motor Devresi	48
5. VANA KONTROL SİSTEMİ DONANIM SEÇİMİ	49
5.1 PIC16F877 Donanımı	49
5.1.1 Çevresel Özellikleri	49
5.1.2 CMOS Geliştirilmiş FLASH/EEPROM Teknolojisi	54
5.1.3 PIC16F877 Hafıza Haritası	55
5.2 PIC16F84A Donanımı	56
5.2.1 Çevresel Özellikleri	56
5.2.2 CMOS Geliştirilmiş FLASH/EEPROM Teknolojisi	58
5.2.3 PIC16F84 Hafıza Haritası	59
5.3 PIC 16F84’de ve PIC 16F877’de Konfigürasyon Kelimesinin Bit Açılımları	60
5.4 Motor Stem İlişkisi	63
5.5 DC Motorlar	64
5.6 Step Motor	68
5.7 Servo Motor	74
5.8 Kristal	75
5.9 Rezonatör	76

5.10	Direnç	77
5.11	Led	77
5.12	H-Köprüsü	78
5.13	Diyot	80
5.14	Transistör	81
5.15	Kapasitör	82
5.16	7805 Regülatör Entegre Devre	83
5.17	Devre Eleman Listeleri	84
5.19.1	Adım Motor Devresi	84
5.19.2	Servo Motor Devresi	84
5.19.3	DC Motor (L293D) Devresi	85
5.19.4	DC Motor (LMD18200T) Devresi	85
5.18	Yazılım İçin Seçilen Programlama Dili	85
5.19	Yazılım İçin Seçilen Derleyiciler	86
5.20	Yazılımların Algoritmaları	87
5.22.1	Servo2.asm	87
5.22.2	Step2.c	88
5.22.3	DCL293D.c	89
5.22.4	LMD18.c	89
6.	SONUÇ	91
7.	KAYNAKLAR	93
8.	EKLER	95
8.1	PIC 16F84 ve 16F877’de Kullanılan Komut Seti	95
8.2	Yazmaçlar	105
8.3	Yazılım ve Açıklaması	115
8.3.1	Delay.c	115
8.3.2	Delay.h	116
8.3.3	Pic.h	117
8.3.4	Pic1684.h	119
8.3.5	Pic1687x.h	124
8.3.6	P16877A.inc	150

8.3.7	P16F84A.inc	149
8.3.8	Servo2.asm	154
8.3.9	Step2.c	161
8.3.10	DCL293D.c	164
8.3.11	DCL293D.asm	165
8.3.12	DCLMD18.c	167

SİMGELER VE KISALTMALAR DİZİNİ

σ_a :	Eksenel Gerilim
σ_l :	Yanal Gerilim
ρ_0 :	Operasyonel Akışkanın Yoğunluğu
Δp_0 :	Referans Basınç Farkı
Δh :	Potansiyel Enerji Farkı
Δp :	Basınç Kaybı
ΔP :	Normalin Üzerindeki Basınç Yükselişi
Δp :	Operasyonel Basınç Farkı
ν :	Poisson Oranı
μ :	Sürtünme Katsayısı
a :	Basınç Dalgasının Yayılma Hızı
A :	Kesit Alanı
c :	Boru Kısıtlama Faktörü
C :	Kavitasyon İndeksi
K_v, A_v :	Akış Katsayısı
D :	Borunun İç Çapı
e :	Boru Duvarının Kalınlığı
E :	Boru Malzemesinin Elastisitesinin Young Modülü
F_F :	Sürtünme Kuvveti
F_L :	Yük Kuvveti

F_{PF} :	Önceden Yüklenen Kuvvet
G :	Yerçekimi
g :	Yerçekimine Bağlı Yerel İvme
J_{LS} :	Vida Atalet Momenti
J_M :	Motor Atalet Momenti
J_T :	Toplam Atalet Kuvveti
K :	Akışkanın Bulk Modülü
K_g :	Gazın Bulk Modülü
K_l :	Sıvının Bulk modülü
p :	Vida Adım Aralığı.
P_d :	Vana Yatağından Boru Çapının 12 Katı Kadar Sonrasındaki Basınç
P_u :	Vana Yatağından Boru Çapının 3 Katı Kadar Öncesindeki Basınç
P_v :	Atmosfer Basıncına Göre Buhar Basıncı (negatif)
Q :	Debi
T_F :	Sürtünme Torku
T_L :	Motor Taftındaki Yük Torku
v :	Akışkanın Hızı
V_D :	Geniş Boğazda Akış Hızı
V_d :	Dar Boğazda Akış Hızı
V_g :	Gazın hacmi
V_L :	Lineer yük hızı
V_l :	Sıvının hacmi
V_t :	Toplam hacim

W_M : Motor hızı

z : Vida verimi

ξ : Sürtünme katsayısı

ρ_g : Gazın yoğunluğu

ρ_l : Sıvının yoğunluğu

ρ : Referans akışkanın yoğunluğu

ŞEKİLLER DİZİNİ

<u>Şekil</u>	<u>Sayfa</u>
2.1 Paralel Gate Vana Kesiti	7
2.2 Conduit Gate Vana Kesiti	8
2.3 Bıçak Gate Vana Genel Görünüşü	8
2.4 Alt Kaviteli Açılı Gate Vana	9
2.5 Alt Kavitesi Olmayan Açılı Gate Vana	9
2.6 Standart Patern Glob Vana Kesiti	10
2.7 Açılı Patern Glob Vana Kesiti	11
2.8 Oblique Paternli Glob Vana Kesiti	11
2.9 Üç Yollu Glob Vana	12
2.10 Piston Vananın İki Değişik Kesiti	13
2.11 Silindirik Plug Vana Kesiti	14
2.12 Plug Vanalarda Çoklu Çıkış Kombinasyonları	14
2.13 Tam Geçişli Küresel Vana Kesiti	15
2.14 Redüksiyonlu Küresel Vana	16
2.15 Kelebek Vana Yataklama Modelleri	16
2.16 Kelebek Vana Kesiti	17
2.17 Açık Gövdeli Pinch Vana	17
2.18 Kapalı Gövdeli Pinch Vana	18
2.19 Bentli ve Bentsiz Diyafram Vanalar	19
2.20 Açılı Patern Lift Çek Vana	20
2.21 Oblique Patern Lift Çek Vana	20
2.22 Swing Çek Vana Kesiti	22
2.23 Swing Çek Vana Çalışma Prensibi	23
2.24 Tilting Disk Çek Vana Kesiti	22
2.25 Diyafram Çek Vana Genel Yapısı	22
2.26 Emniyet Vana Kesiti	23
2.27 Emniyet Relief Vanası Kesiti	23
2.28 Sıvı Relief Vana Kesiti	24
2.29 Vakum Relief Vana Kesiti	24

2.30 Doğrudan Yüklenen Pressure Relief Vana	25
2.31 İleri Dom Tipli Patlama Kapakları	26
2.32 Ters Burkulma Tipli Patlama Kapakları	26
3.1 Glob Vanaların Açılmasının Sürtünme Katsayısına Yaklaşık Etkileri	30
3.2 Gate Vanaların Açılmasının Sürtünme Katsayısına Yaklaşık Etkileri	30
3.3 Diyafram Vanaların Açılmasının Sürtünme Katsayısına Yaklaşık Etkileri	31
3.4 Kelebek Vanaların Açılmasının Sürtünme Katsayısına Yaklaşık Etkileri	31
3.5 Tam Açık Daralan ve Genişleyen Akışlı Gate Vanalarda Sürtünme Katsayısı	33
3.6 Vanaların Yapısal Akış Karakteri	36
3.7 Akış oranı, Vana Açıklık Pozisyonu ve Basınç Kaybı Arasındaki İlişki	37
3.8 Bazı Vanaların Kavitasyon Karakteristiği	39
3.9 Teorik ve Deneysel Veriler ile Bir Basınç Dalgasının Değişik Hava İçeriğine Göre Yayılma Hızı	43
4.1 Servo Motor Devresi	45
4.2 Step Motor Devresi	46
4.3 L293D ile DC Motor Devresi	47
4.4 LMD18200T ile DC Motor Devresi	48
5.1 PIC 16F877 PDIP Pin Yayılımı	50
5.2 PIC 16F877'nin Hafıza Haritası	55
5.3 PIC 16F84 PDIP, SOIC Pin Yayılımı	57
5.4 PIC 16F84A'nın Hafıza Haritası	59
5.5 Mıknatıslı DC Motor	65
5.6 DC Motor geri besleme kontrol blok şeması	67
5.7 Adım Motor Bağlantıları	71
5.8 Bi-polar sürücü ile unipolar motoru sürmek için yapılan bağlantılar	72
5.9 Kristal Osilatör Devresi	76
5.10 Rezonatör Devresi	77
5.11 L293D Bağlantı Şeması	78
5.12 L293D Pinleri	78
5.13 LMD18200T Pinleri	79
5.14 LMD18200 Bağlantı Şeması	79
5.15 ULN2003 Serisi Şeması	80
5.16 Diyotun Şematik Gösterimi ve Resmi	80
5.17 Diyotun Anahtar Şeklinde Gösterimi	81

5.18 L7805CV Çeşitleri	83
5.19 L7805 Bağlantı Şeması	83

ÇİZELGELER DİZİNİ

<u>Çizelge</u>	<u>Sayfa</u>
2.1 Akış Regülasyonuna göre Vanalar	5
2.2 Vana Akım Çizelgesi	6
3.1 Tam türbülanslı akışta, tam açık vanalardaki yaklaşık sürtünme katsayıları	29
5.1 PIC 16F877A'nın Pin Tablosu	51
5.2 PIC 16F84A'nın Pin Tablosu	57
5.3 PIC16F877 ile PIC 16F84 Arasındaki Farklar	60
5.4 Bir fazlı tam adım	70
5.5 İki fazlı tam adım	70
5.6 İki fazlı yarım adım	70
5.7 Bi-polar adım Motoru	71
5.8 Kristal Kullanımı İçin Kapasitör Seçimi	76
6.6 PIC16F84 Yazmaç Haritası	105
6.8 PIC16F877 Yazmaç Haritası	111

1. GİRİŞ

Vana en önemli akış kontrol elemanıdır. Bu çalışmada da proses kontrolünün en önemli ayaklarından biri olan vanaların elektriksel olarak tahrik edilerek çalıştırılması üzerinde durulmuştur. Bu çalışmanın yapılmasının tetikleyici etkeni ulaşılması zor veya tehlikeli bölgelerdeki vanaların kullanılamaması yüzünden endüstrideki üretim hataları ve buna bağlı zararın görülmesidir. Personelin, sağlık açısından tehlikeli gördüğü için veya tamamen ulaşılması zor olduğu için bir vanayı kontrol etmemesi sonucu, endüstride oluşacak zarar tahmin edilenin de ötesindedir. Bu yüzden, bu çalışmada bir vana sistemi alınmış, kontrol mekanizması tasarlanmış, yazılımı gerçekleştirilmiş, mikrodnetleyecilere yüklenmiş, devresi yapılmış ve denenmiştir. Değişik tasarımlar üzerinden tartışma yürütüldüğü için bu çalışma, tam olarak saptanan proses ihtiyacı için önemli bir yardımcı etmen olabilir. Benzer sorunlara odaklanmış değişik tasarımların ele alındığı bu çalışmanın, karşılaştırma ihtiyacının giderilmesine katkı sağlayabileceği düşünülebilir.

Bu çalışmanın 2. bölümünde vana sınıflandırılması ve 3. bölümünde vana içinden akış incelenmiştir. Çalışmanın 4. bölümünde tahrik kontrol devrelerinin şemaları verilmiş ve bu devreler tartışılmıştır. Bu çalışmanın 5. bölümü ise 4. bölümde tartışılmış olan devrelerin elemanları ile ilgilidir. Bu sözü edilen bölümde vanaları tahrik eden motor tiplerinden DC motorlar, step ve servo motorlar ve tahrik mekanizmasını kontrol eden devrelerin her elemanı kısaca ele alınmış, kontrol mekanizmasının beyni olan mikrodnetleyiciler de diğer devre elemanları gibi fakat daha detaylı bir şekilde ele alınmıştır. Ayrıca 5. bölümde C ve Assembler programlama dilleri ile programlanan bu yazılımların algoritmaları açıklanmış, bu yazılımları mikrodnetleyiciye aktarmak için kullanılan yöntemler incelenmiştir.

Kontrol mekanizması için PIC mikrodnetleyiciler seçilmiştir. Bu mikrodnetleyicilere ait assembler komutları ve C fonksiyonları kullanılarak yazılan programların kodları ve kodların içindeki fonksiyonların anlamları bu çalışmanın ekler kısmında verilmiştir. Ayrıca mikrodnetleyiciler için geçerli olan komut setleri

de yine ekler kısmında verilmiştir. Ayrıca mikrodnetleyicilerdeki yazmaçlar ekler kısmında incelenmiştir.

2. VANALARIN SINIFLANDIRILMASI

Vanalar genel yapılarına, sızdırmazlıklarına, salmastralarına, yataklarına, stem (kapama elemanını yönlendiren mil) salmastralarına ve büyüklüklerine göre sınıflandırılabilirler [1].

2.1 Sızdırmazlık Kriterine Göre Sınıflandırılması

Sızdırmazlık açısından vananın sınıflandırılması, vananın kapanma elemanından akışkanın kılcal damarlar yoluyla hat dışına çıkma hızına göre yapılabilir. Bu şekilde dört grup sızdırmazlık tanımlayabiliriz: 1- Düşük Sızdırmazlık Sınıfı, 2- Nominal Sızdırmazlık Sınıfı, 2- Buhar Sızdırmazlık Sınıfı, 3- Atom Sızdırmazlık Sınıfı [2; 3].

Nominal ve düşük sızdırmazlık gurubundaki vanalar, aslında tam olarak kapanıp kapanmamaları çok önemli olmayan hatlarda kullanılır. Fakat buhar sızdırmazlık sınıfındaki vanalar, adlarından anlaşılacağı üzere buhar kontrolü için endüstride çok sıkı sızdırmazlık gerektiren yerlerde kullanılmaktadır. Buhar sınıfı salmastra metresi başına 10'dan $100\mu\text{g/s}$ gaz sızdırmazlığına sahip olmalıdır. Aynı zamanda buhar sınıfı salmastra metresi başına 0,1'dan $1\mu\text{g/s}$ sıvı sızdırmazlığına sahip olmalıdır. Atom sınıfı vanalar ise uzay teknolojisinde ve güç santrallerinde kullanılır. Atom sınıfı salmastra metresi başına 10^{-3} 'dan $10^{-5}\mu\text{g/s}$ gaz sızdırmazlığına sahip olmalıdır [2; 3].

2.2 Vana Yataklamasına Göre Sınıflandırma

2.2.1 Metal Yataklama

Yataklama sadece yumuşak yataklama ile sınırlı değildir. Eğer vananın koyulacağı hat korozyon ya da içinde parçacıklar taşıyan bir akışkan için tasarlanmış ise metal yataklama daha uygun bir çözümdür. Yataklama için kullanılan malzemeler ayrıca vananın içinden geçen akışkan içinde çözünebilir ise metal yataklama

kullanmak tek çare gibi gözükebilir. Örnek olarak karbon bazlı akışkanlar (petrol bazlı yakıtlar, her türlü yağ vs.) vanaların yataklamaları için kullanılan PTFE'yi (politetrafloroetilen) zaman içinde çözerek vanayı işlevsiz hale getirebilirler. Özellikle bu sözü geçen akışkanlar yüksek sıcaklıkla hareket ediyorsa bu erozyon daha hızlı olur [1; 4].

2.2.2 Yumuşak Yataklama

Yumuşak yataklamalar çok etkilidir fakat yüksek sıcaklık ve basınçlarda kullanımlarında sınırlamalar vardır. Bazı yumuşak yataklamalar bazı akışkanlar içinde pek kullanışlı sayılmazlar. Bu konu Bölüm 2.2.1 Metal Yataklamada incelenmiştir [1; 4; 5].

Yumuşak yataklamadan, vana yataklarında daha yumuşak malzemenin kullanılmaktadır. Plastik ve benzeri malzemeler daha yumuşak ve esnek olmaları yüzünde vananın kapanma elemanı yatağa tam oturduktan sonra diğer yataklamalardan çok daha verimli bir şekilde sızdırmazlık sağlar. Bu özelliği de bu tür yataklamaları daha tercih edilir kılmaktadır [1; 4; 5].

2.3 Vana Stem Salmastrasına Göre Sınıflandırma

Compression tipi, *Lip* tipi, *Squeeze* tipi, *Thrust* Tipi, Diyafram Vana Stem tipi olmak üzere beş tipi vardır. *Compression* tipi stem salmastraları direkt olarak stem'e uyguladıkları yanıl gerilime bağılı olduğı için yanıl gerilimin Denklem 2.1'deki gibi hesaplanması önem kazanır [1].

$$\sigma_l = \sigma_a \left(\frac{1-\nu}{\nu} \right) \quad (2.1)$$

Burada:

- σ_l : yanıl gerilim
- σ_a : eksenel gerilim
- ν : Poisson oranı

Burada poisson oranına göre malzemelerin seçimi yapılır.

Değişik akışkanlara göre vananın tasarım özelliklerine göre değişik stem salmastra tipleri seçilir. [1]

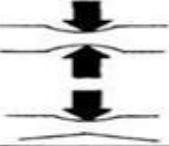
2.4 Vana Genel Yapısına Göre Sınıflandırma

2.4.1 Manuel Vanalar

Manuel vanalar kapatma elemanlarına göre dörde ayrılır. Her vana grubunda çeşitli vana türleri ve bunların da çeşitli varyasyonları mevcuttur. Bu dört grup şöyle listelenebilir [1; 5]:

- Stopper tipi kapatma elemanı – glob, iğne
- Dikey kayan – *gate*
- Dönen tip – küresel, plug, kelebek
- Esnek tip – diyafram

Çizelge 2.1 Akış Regülasyonuna Göre Vanalar [1]

Akış Kontrolüne Göre Gruplama	Vana Tipi
Kapama 	Glob Vana Piston Vana
Kayma 	Paralel Gate Vana Wage Gate Vana
Dönme 	Plug Vana Küresel Vana Kelebek Vana
Esnek Vana Gövdesi 	Pinch Vana Diyafram Vana

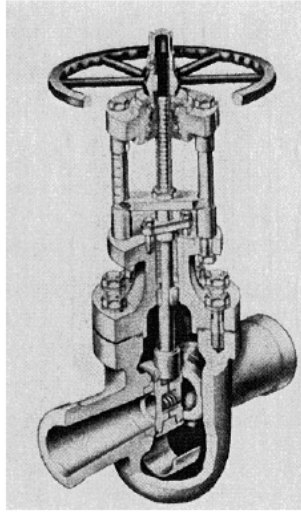
Bu çalışmanın devamında örnekleri verilecek olan vanaları kullanılabilecekleri alanlara göre bir çizelgede gösterebiliriz. Bu oluşturulan çizelge (Çizelge 2.2) aynı zamanda vana seçim çizelgesidir [1; 5].

2.2 Vana Akım Çizelgesi [1]

	Akış Regüle Eden Modlar					Sıvı İçinde Bulunan Katılar			
Grup	Tip	Aç-Kapa	Kısılabilen	Saptırıcı Olan	Katı Madde Bulundurmayan	Aşındırıcı Olmayan	Aşındırıcı	Yapışkan	Sağlıklı
Aşağı Doğru Kapanan Tip	Glob:								
	Standart patern	Var	Var		Var				
	Açılı patern	Var	Var		Var	Özel	Özel		
	<i>Oblique</i> patern	Var	Var		Var	Özel			
	Çoklu patern			Var	Var				
	Piston:	Var	Var		Var	Var	Özel		
Kayan Tip	Paralel Gate :								
	Konvensiyonel	Var			Var				
	<i>Conduit</i>	Var			Var	Var	Var		
	Bıçak	Var	Özel		Var	Var	Var		
	Açılı Gate:								
	Alt Kaviteli	Var			Var				
	Alt Kavitesiz (plastik yataklı)	Var	Orta seviyede		Var	Var			
Dönen Tip	Plug:								
	Yağlaması Olmayan	Var	Orta seviyede	Var	Var	Var			Var
	Yağlamalı	Var		Var	Var	Var	Var		
	Eksantrik Plug	Var	Orta seviyede		Var	Var		Var	
	Lift Plug	Var		Var	Var	Var		Var	
	Küresel:	Var	Orta seviyede	Var	Var	Var			
	Kelebek:	Var	Var	Özel	Var	Var			Var
Esnek Tip	Pinch:	Var	Var	Özel	Var	Var	Var	Var	Var
	Diyafram:								
	Bentli	Var	Var		Var	Var		Var	Var
	Düz	Var	Orta seviyede		Var	Var		Var	Var

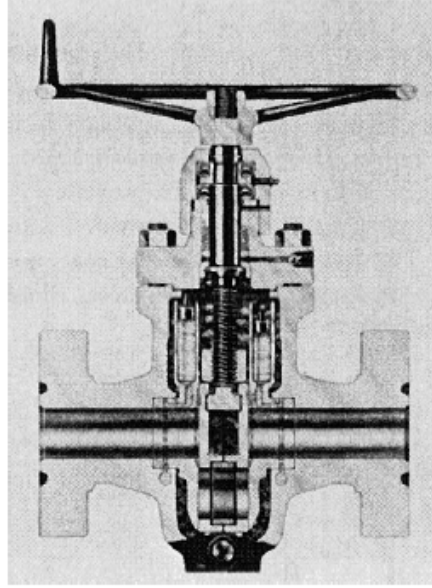
2.4.1.1 *Gate* Vanalar

Disk yatağı dik olarak iner. Hattaki basınç diskin hareketini sürtünme kuvvetini arttıracak için direkt etkiler. Hattaki basınç arttıkça diski açıp kapamak kayda değer biçimde zorlaşır. Basıncın etkisini azaltmak için disk akışa 90 derece dik olarak değil açılı biçimde yatağı oturacak şekilde tasarlanır. Bu sebeplerle *gate* vanalar iki ana gruba ayrılabilir: 1- Paralel, 2- Açılı [1; 5].

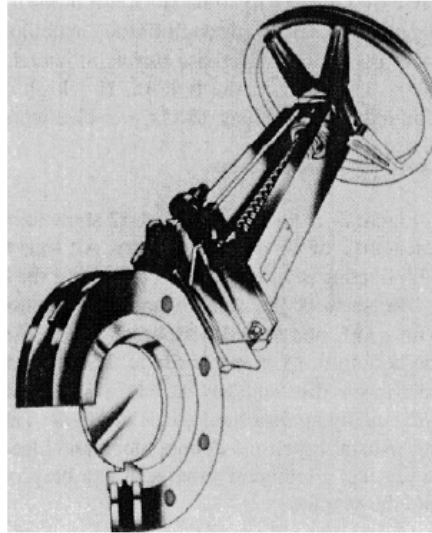


Şekil 2.1 Paralel *Gate* Vana Kesit [1]

Şekil 2.1’de örneği verilen *gate* vana, standart, tipik (konvansiyonel) kullanılan bir türdür. Aşağıda paralel tip *gate* vanaların diğer iki çeşidi bulunmaktadır. İlk örnek *conduit* tipidir ikinci örnek ise bıçak tipidir. Glob vanalara piyasada su vanası da denmektedir [1;5;6].



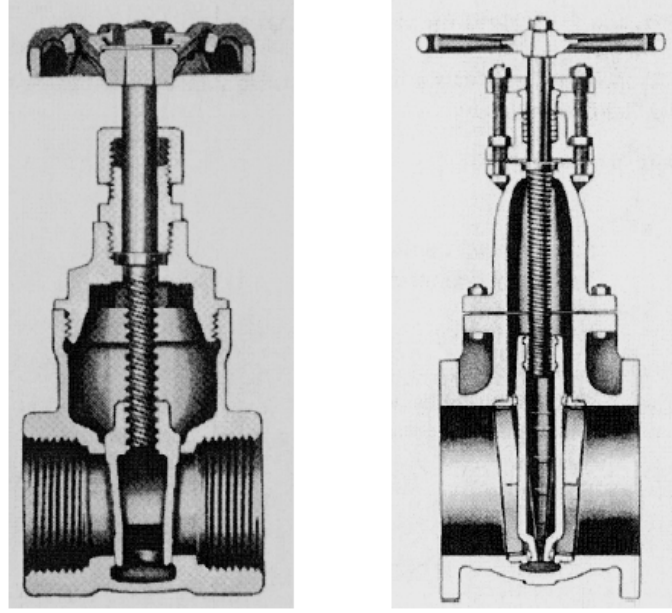
Şekil 2.2 *Conduit Gate* Vana Kesiti [1]



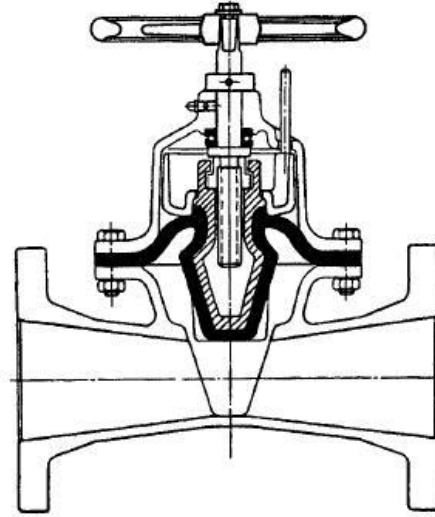
Şekil 2.3 Bıçak *Gate* Vana Genel Görünüşü [1]

Paralel *gate* vanalar akış durdurma ve başlatma ve sürekli kullanım gerektirmeyen yerlerde kullanılabilirler. Paralel *gate* vanalar, gazlar ve sıvılar için olduğu kadar vakum ortamlar için ve kriyojenik (düşük ısıya sebep olan) sıvılar için de seçilebilecek bir vana türüdür. Bu tür vanalar ayrıca içinde katı olan akışkanlar için ve özellikle bıçak *gate* vanalar granül, toz, lif, çamur tipi akışkanlar için kullanılabilir [5; 6].

Şekil 2.4’de açılı *gate* vanalara örnekler mevcuttur. Şekil 2.2.4’deki vanalar kavitesi olan açılı *gate* vanalara örnektir. Şekil 2.2.5’deki örnek ise alt kavitesi olmayan açılı *gate* vanalara örnek teşkil eder [1].



Şekil 2.4 Alt Kaviteli Açılı *Gate* Vana [1]



Şekil 2.5 Alt Kavitesi Olmayan Açılı *Gate* Vana [1]

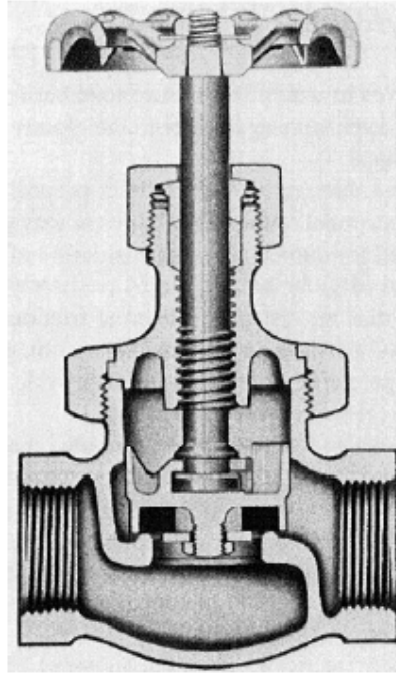
Açılı *gate* vanalar, hatta akışkanın yüksek basıncından dolayı yatakta çok fazla sürtünme oluşmasından dolayı diskin yukarı çıkamamasından doğan vana sıkışmalarını önlemek amacı ile tasarlanmıştır [1; 5].

Açılı *gate* vanalar akış durdurma ve başlatma ve sürekli kullanım gerektirmeyen yerlerde kullanılabilirler. Açılı *gate* vanalar, gazlar ve sıvılar için olduğu kadar vakum ortamlar için ve kriyojenik (düşük ısıya sebep olan) sıvılar için de seçilebilecek bir vana türüdür. Bu tür vanalar içinde alt kavitesi olan ve plastik yataklı açılı *gate* vanalar, katı olan akışkanlar için kullanılabilir [1; 5; 6].

2.4.1.2 Glob Vanalar

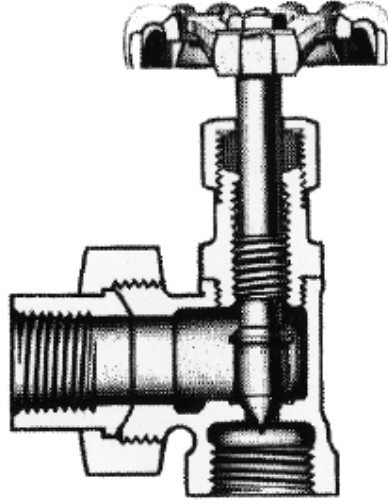
Glob vanalarda disk yatağa paralel olarak iner. Kapama elemanına disk denmektedir. Bu tip vanaları fiziksel özelliklerine göre üçe ayırabiliriz: 1- Standart, 2- *Oblique*, 3- Açılı. Glob vanalar endüstride en çok görülen vana çeşitlerindendir [5].

Şekil 2.6, 2.7 ve 2.8’de değişik paternde olan glob vanalara örnekler mevcuttur.



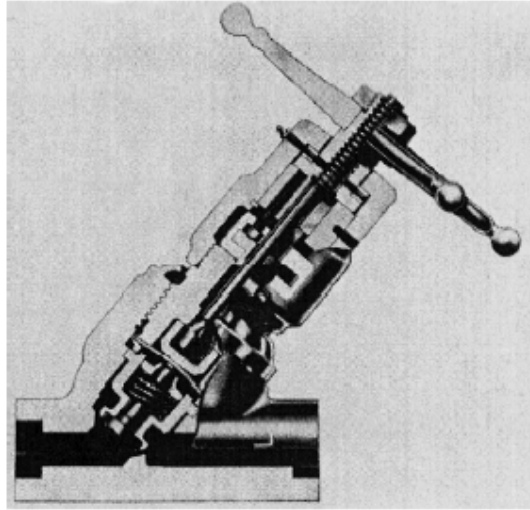
Şekil 2.6 Standart Patern Glob Vana Kesiti [1]

En sık görülen glob vana türü standart paterndir. Çok sık kullanılmasına rağmen akışa en fazla direnç gösteren yapıya bu glob vana sahiptir [5].



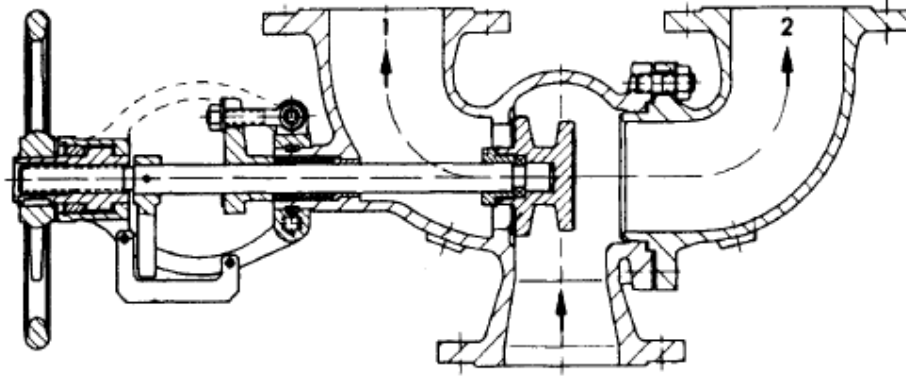
Şekil 2.7 Açılı Patern Glob Vana Kesiti [1]

Gerekli görüldüğü taktirde yer kazanma amaçlı olarak kullanılır. Açılı glob vana hem standart türden çok daha az akışa direnir hem de boru hattı tasarımında önemli yer kazançları sağlar [1; 5].



Şekil 2.8 *Oblique* Patern Glob Vana Kesiti [1]

Oblique patern glob vanaların amacı akış direncini minimuma indirmektir. Dirençte önemli azalmalar sağlar. Şekil 2.9'da üç yollu glob vanalara bir örnek verilmektedir [1].

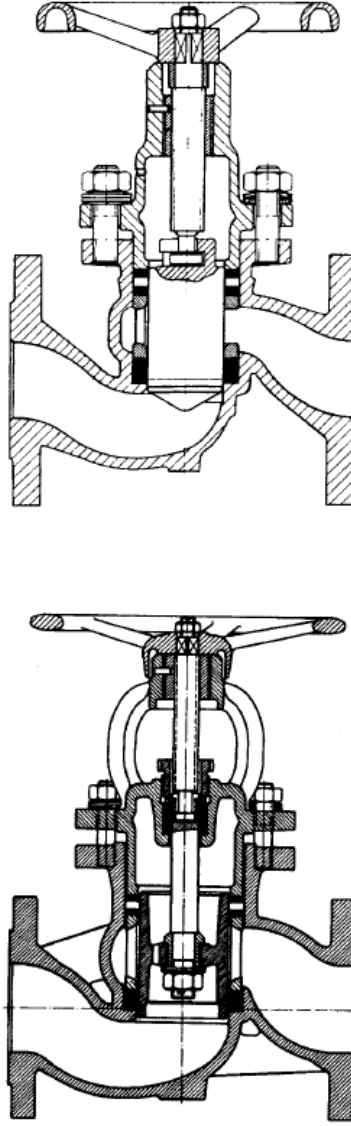


Şekil 2.9 Üç Yollu Glob Vana [1]

Glob vanalar akış kontrolü, akış durdurma ve başlatma ve sürekli kullanım gerektiren yerlerde kullanılabilirler. Glob vanalar, içinde katı ihtiva etmeyen gazlar ve sıvılar için olduğu kadar vakum ortamlar için ve kriyojenik (düşük ısıya sebep olan) sıvılar için de seçilebilecek bir vana türüdür [6].

2.4.1.3 Piston Vanalar

Disk yatağa paralel olarak iner fakat akış Diskin içinde hareket ettiği silindirden kesiti belli bir orifizden çıktığı için bu tip vanalar akış debisinin en kolay kontrol edildiği vanalardır. Pratikte de debinin en verimli kontrol edildiği vana tipidir [1].

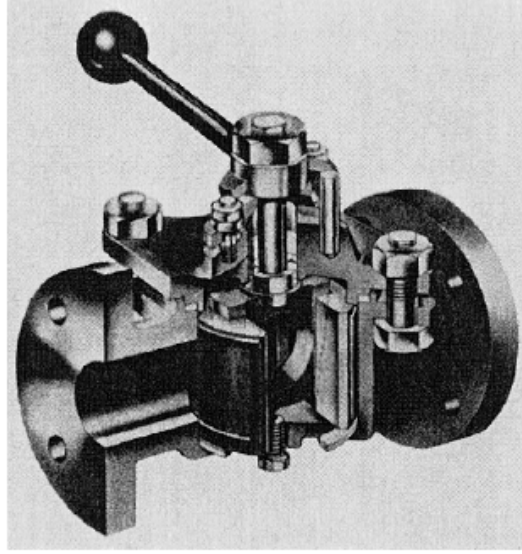


Şekil 2.10 Piston Vananın İki Değişik Kesiti [1]

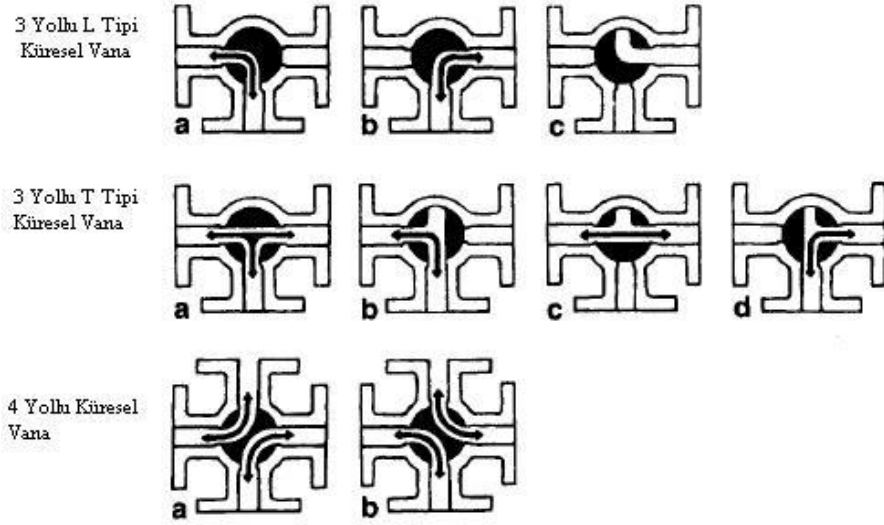
Piston vanalar akış kontrolü, akış durdurma ve başlatma için kullanılabilirler. Piston vanalar, gazlar ve sıvılar için olduğu kadar vakum ortamlar için ve içinde katı akışkanlar için de seçilebilecek bir vana türüdür [1; 6].

2.4.1.4 Plug Vanalar

Disk yatağa dik olarak fakat kendi eksenini etrafında dönecek biçimde hareket eder. Disk üzerindeki orifiz vana çeperinden hatla paralel hale gelerek akışa izin vermektedir. Kapama elemanı 90° derece dönerek hattı akışa kapatır veya açar [1; 4].



Şekil 2.11 Silindirik Plug Vana Kesiti [1]



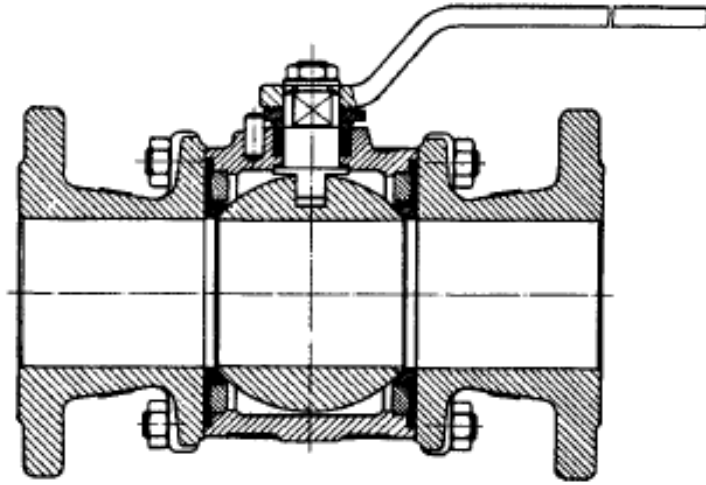
Şekil 2.12 Plug Vanalarda Çoklu Çıkış Kombinasyonları [1]

Şekil 2.12'deki örnekte plug vanaların kapama elemanlarının dönme özelliği kullanılarak kolayca çok yönlü vana olarak kullanılabilecek tasarımlar ortaya çıktığı görülebilir [1].

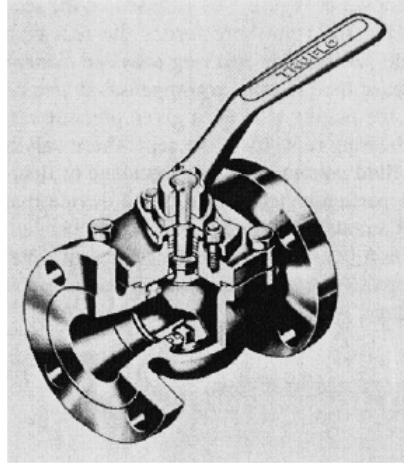
Plug vanalar akışı durdurma ve başlatma, kapanmada uygunluk ve akışı saptırma istendiğinde kullanılabilir. Plug vanalar gazlar, sıvılar ve korozif olmayan çamurlar için kullanılabilir. Yağlı plug vanalarda korozif çamurlar kullanılabilir. Yapışkan akışkanlar eksantrik ve lift plug vanalarda kullanılabilir. İlaç ve gıda sanayide, sıhhi ortamlarda plug vanalar kullanılabilir. Ayrıca vakum ortamlarda da kullanılabilirler [1; 4; 5].

2.4.1.5 Küresel Vanalar

Plug vana cinsidir. Kapanma elemanları (disk) daima küre biçimindedir. En fazla kullanılan vana türlerindendir. Her çeşit boru hattında görmek mümkündür. Bir plug vana çeşidi olduğu için plug vanaların prensiplerine uyar. Çok çeşitli amaçlarla çok değişik hatlarda kullanıldıkları için içlerinden geçen akışkana göre ve hattın özelliklerine göre çok değişik yataklama özelliklerine sahiptirler. En sık kullanılan yataklama malzemesi ise PTFE'dir [4; 5].



Şekil 2.13 Tam Geçişli Küresel Vana Kesiti [1]

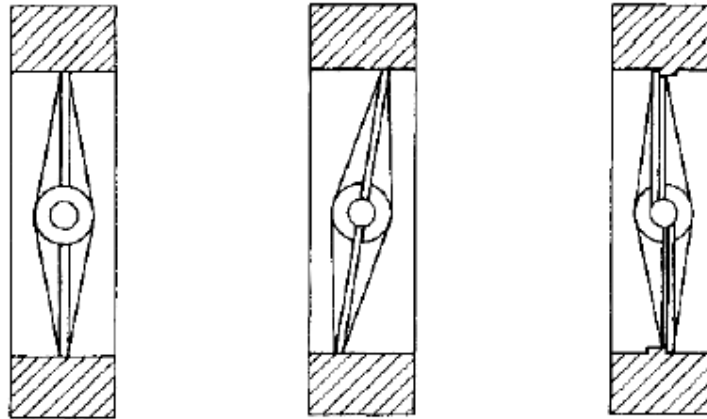


Şekil 2.14 Redüksiyonlu Küresel Vana [1]

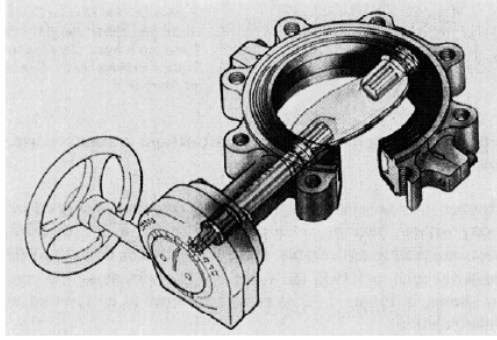
Küresel vanalar akışı durdurma ve başlatma, kapanmada uygunluk ve akışı saptırma istendiğinde kullanılabilir. Küresel vanalar gazlar, sıvılar ve korozif olmayan çamurlar için kullanılabilir. Ayrıca vakum ortamlar ve kriyojenik (düşük ısıya sebep olan) sıvılar için de seçilebilecek bir vana türüdür [1; 4; 5].

2.4.1.6 Kelebek Vanalar

Şekil 2.15’de kelebek vanalarda kullanılan yataklama modelleri görülmektedir. Kelebek vanalar tam açıkken akışa çok az direnç gösterirler. 15° ile 70° arasında açıkken ise çok ince akış kontrolü sağlayabilirler [1; 2; 4; 5; 6].



Şekil 2.15 Kelebek Vana Yataklama Modelleri [1]

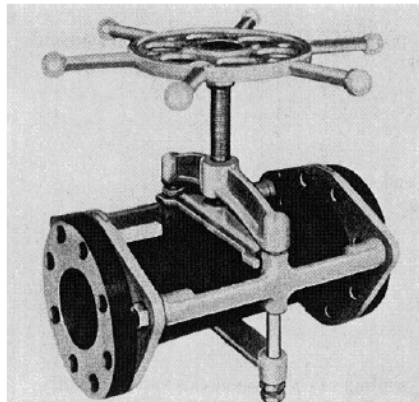


Şekil 2.16 Kelebek Vana [1]

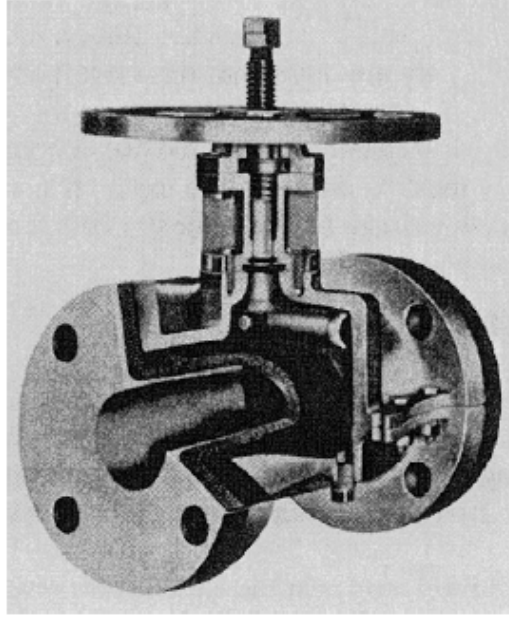
Kelebek vanalar açma, kapama ve akış kontrolü için kullanılabilirler. Ayrıca gazlar, sıvılar, çamur, toz ve granüller için kullanılabilirler. Ayrıca kelebek vanalar vakum ortamlarda ve sıhhi tesisatlarda, ilaç sanayide ve yiyecek sektöründe kullanılabilir [1; 4; 5].

2.4.1.7 *Pinch* Vanalar

Pinch vanalar esnek yapıya sahip bir borunun mekanik olarak veya hat basıncı ile sıkılıp bırakılması ile çalışır. *Pinch* vanalar açık gövdeli ve kapalı gövdeli olmak üzere ikiye ayrılırlar. Aşağıdaki örneklerden ilki açık gövdeli *pinch* vanalara örnektir. İkincisi ise kapalı gövdeli vanalara örnektir. Açık gövdeli *pinch* vanalar tabî ki vanayı kontrol etmek. Tıkanıklıkları saptamak veya vananın çalışmasını görmek açısından daha tercih edilebilir olanıdır fakat eğer vana gövdesinde herhangi bir sebepten oluşabilecek boşluk veya boşluklardan sızıntı her koşulda tahammül sınırları dışında ise kapalı gövdeli *pinch* vana kullanılmalıdır [1; 2; 4; 6].



Şekil 2.17 Açık Gövdeli *Pinch* Vana [1]



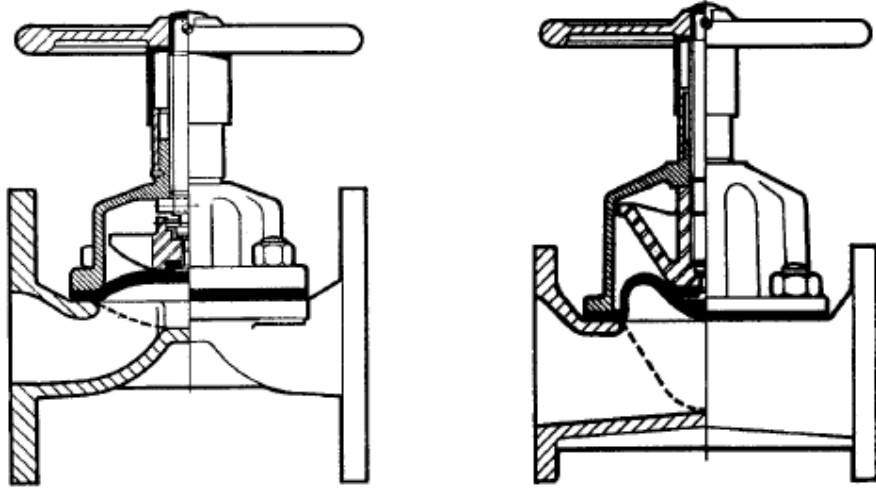
Şekil 2.18 Kapalı Gövdeli *Pinch* Vana [1]

Pinch vanalar açma, kapama ve akış kontrolü için kullanılabilirler. Ayrıca sıvılar, korozif çamur, toz ve granüller için kullanılabilirler. Ayrıca *pinch* vanalar sıhhi tesisatlarda, ilaç sanayide ve yiyecek sektöründe kullanılabilir [1; 5].

2.4.1.8 Diyafram Vanalar

Diyafram vanalar ikiye ayrılmaktadır. Şekil 2.19’da verilen iki vana arasındaki fark açıkça görülmektedir. Bentli diyafram vanalar diyaframın esnek kısmında kullanılacak malzemenin seçim yelpazesini büyütürken daha az esnek fakat dayanıklı (PTFE gibi) malzemelerin kullanılmasına olanak sağlamak amacı ile tasarlanmışlardır. Bentsiz tam geçişli diyafram vanalarda ise akışkan geçişi daha kolay olmakla beraber diyafram için kullanılan malzeme daha esnek olmalıdır [1; 4; 6].

Diyafram vanaların *pinch* vanalara benzer operasyonel limitleri mevcuttur. Aynı *pinch* vanalar gibi kapanırken akışkanın bir kısmını geri iterler [1;6].



Şekil 2.19 Bentli ve Bentsiz Diyafram Vanalar [1]

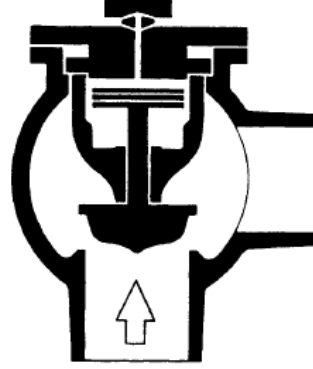
Diyafram vanalar açma, kapama ve akış kontrolü için kullanılabilirler. Bentli Diyafram vanalar, içinde katı madde barındıran gazlar ve sıvılar, viskoz akışkanlar, sızıntısı tahammül dışı tehlikeli sıvılar için kullanılabilirler. Aynı zamanda bentli diyafram vanalar sıhhi tesisatlarda, ilaç sanayide ve yiyecek sektöründe ve vakum ortamlar için kullanılabilir. Bentsiz tam geçişli diyafram vanalar da aynı bentliler gibi içinde katı madde barındıran gazlar ve sıvılar, viskoz akışkanlar ve vakum ortamlar için kullanılabilirler. Ayrıca tortu, korozyif çamur ve kuru malzeme akışı içinde kullanılabilir [1].

2.4.2 Çek Vanalar

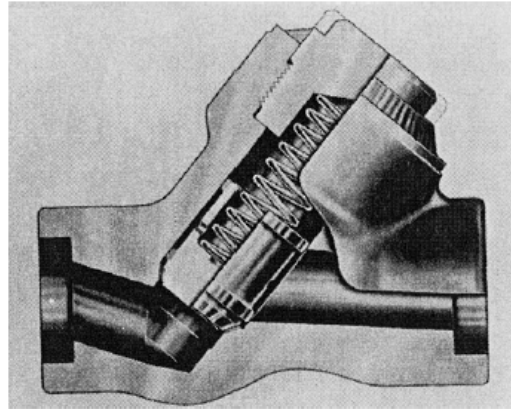
Çek vanalar kapatma elemanlarına göre dörde ayrılır. Bu tür vanaların asli işlemi geri yönde akışı önlemektir. Fakat aynı zamanda ani kapanmanın etkisiyle oluşacak zararlı etkileri olan ters akımı mümkün oldu kadar önleyecek biçimde tasarlanmalıdırlar [1; 4].

1.4.2.1 *Lift* Çek Vanalar

Kapama elemanı yatağa dik olarak hareket eder [1].



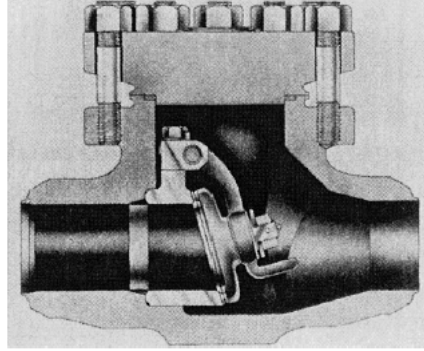
Şekil 2.20 Açılı Patern Lift Çek Vana [1]



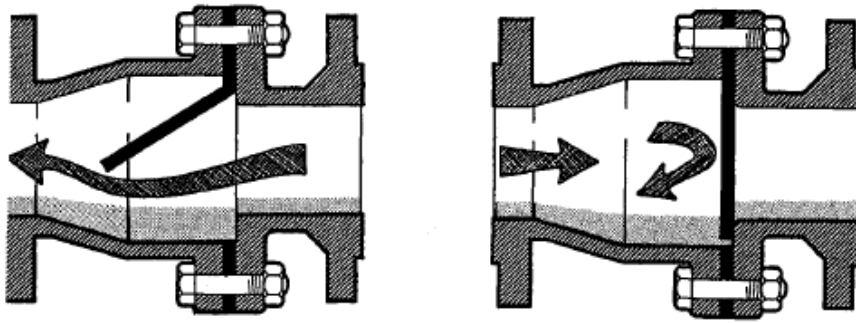
Şekil 2.21 *Oblique* Patern Lift Çek Vana [1]

1.4.2.2 *Swing* Çek Vanalar

Kapama elemanı yatağın dışından menteşelenmiş biçimde akış yönüne doğru sallanarak hareket eder [1].



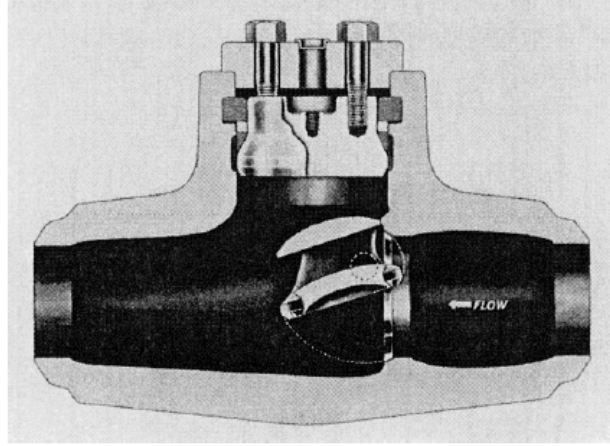
Şekil 2.22 *Swing* Çek Vana Kesiti [1]



Şekil 2.23 *Swing* Çek Vana Çalışma Prensibi [1]

1.4.2.3 *Tilting Disk* Çek Vanalar

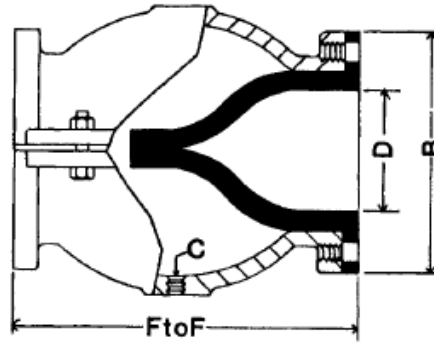
Kapama elemanı yatağa yakın fakat üstünde bir yerden menteşeli bir şekilde yatarak ve eğilerek çalışır. Bu tür çek vanalar çok sıklıkla kullanılmaz. *Swing* çek vanaların bir sebeple kullanılamayacağı yerlerde kullanılırlar [1; 4; 5].



Şekil 2.24 *Tilting Disk* Çek Vana [1]

1.4.2.4 Diyafram Çek Vanalar

Kapama elemanı diyaframdır ve yatağa doğru veya yataktan sonra kıvrımlı olarak yerleştirilir.



Şekil 2.25 Diyafram Çek Vana [1]

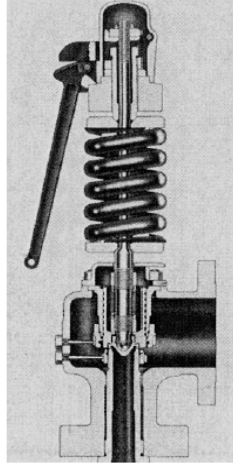
2.2.3 *Pressure Relief* Vanalar

Pressure Relief vanaları iki ana guruba ayrılmaktadır.

- Direkt yüklenmiş *pressure relief* vanalar
- *Piloted pressue relief* vanalar

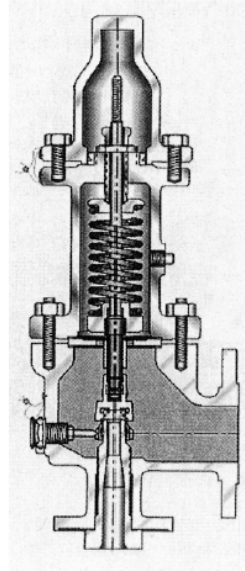
Doğrudan yüklenen sistemlerde vana direk olarak sistemin basıncı tarafından aktive edilir. Pilotlu sistemlerde ise vana bir “pilot” (kontrol elemanı) tarafından kontrol edilir. Bir başka çeşitte dıştan güç alarak opere edilen pilotlu *pressure relief* vanalardır [1;2;4;6].

Emniyet vanaları da bir çeşit *pressure relief* vana çeşitidir.



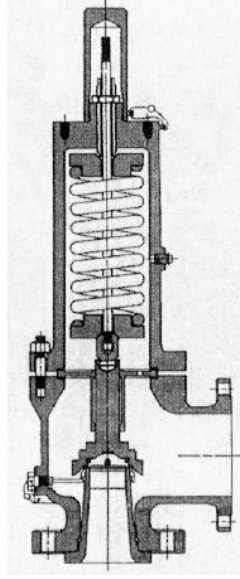
Şekil 2.26 Emniyet Vanası Kesiti[1]

Emniyet vanaları dışında bir de emniyet relief vanaları diye bir kategori açarsak genel amaçlı hem gaz hem de sıvı basınç ayarlı vanaları sınıflandırabiliriz. Aşağıda konvansiyonel bir emniyet relief vanası görebiliriz [1; 2; 4; 5; 6].



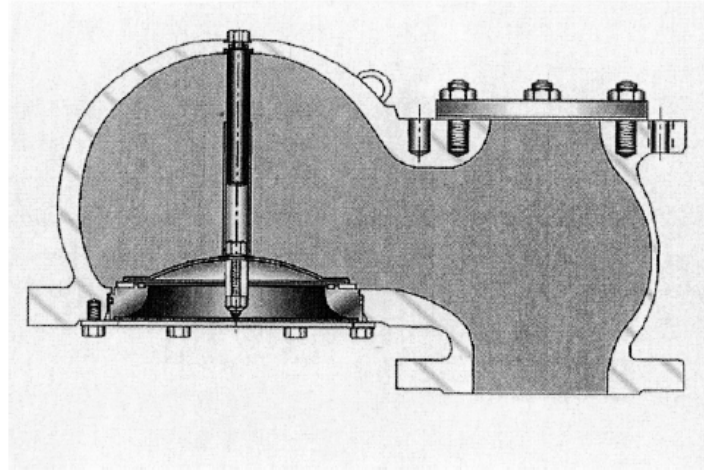
Şekil 2.27 Emniyet Relief Vana Kesiti[1]

Sıvı relief vanaları, konvasiyonel sıvı relief vanalarından sadece çok küçük farklarla ayrılırlar. Bu türler, sıvılara özgü performans değerleri için geometride küçük değişiklikler yapılmış emniyet relief vanaları olarak da düşünülebilir [1; 5].



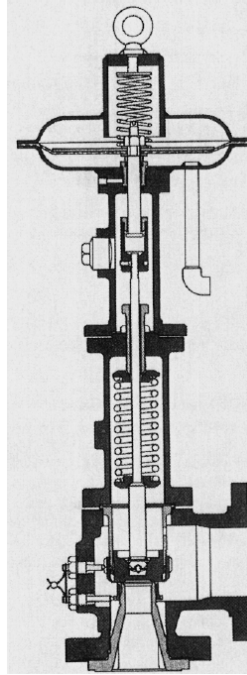
Şekil 2.28 Sıvı *Relief* Vana Kesiti [1]

Vakum relief vanaları ise daha değişik bir yapıya sahiptir.



Şekil 2.29 Vakum *Relief* Vana Kesiti [1]

Ayrıca endüstride, yardımcı olarak aktüatör kullanan doğrudan yüklenen *pressure relief* vanalarından çok yaygın olarak kullanılmaktadır. Şekil 2.30’da bir örneği görülebilir [1; 2; 4; 5; 6].

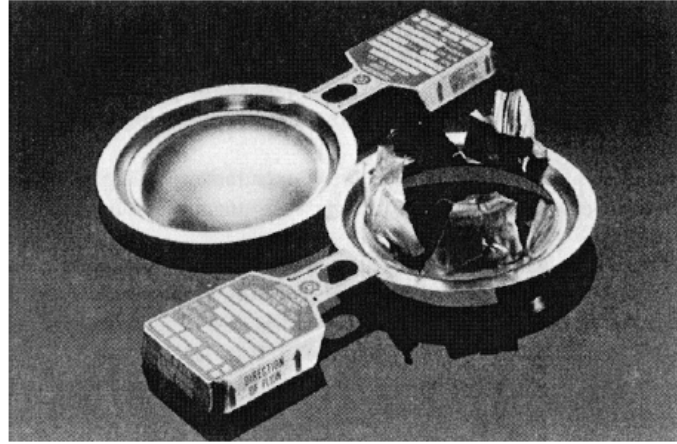


Şekil 2.30 Doğrudan Yüklenen *Pressure Relief* Vana [1]

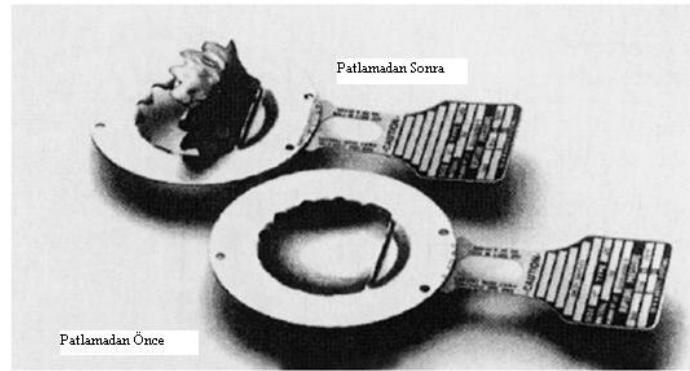
Pilot ile opere edilen *pressure relief* vanaları direkt çalışan pilotlar ve indirekt çalışan pilot tipleri olarak ikiye ayrılabilir [1].

2.2.4 Patlama Kapakları

Patlama kapaklarında geri kapanmayan *pressure relief* vanaları diyebiliriz. Patlama kapakları *pressure relief* vanaları ile birlikte veya tek başlarına kullanılabilirler. İleri dom’lu (*forward Domed*) ve ters burkulma (*Reverse – Buckling*) tipler patlama kapakları çeşitleri olarak ortaya çıkmaktadır. Bu çeşitler kontrolü sağlanmak istenen basınca göre seçilirler. Vakum veya sistem basıncından yüksek basınçlar için değişik patlama kapakları kullanılır. Hatta seçilen patlama kapakları için yardımcı patlatma elemanları dahi kullanılabilir [1].



Şekil 2.31 İleri Dom Tipli Patlama Kapakları Patlamadan Önce ve Sonra. [1]



Şekil 2.32 Ters Burkulma Tipli Patlama Kapakları Patlamadan Önce ve Sonra. [1]

3. VANADA AKIŞ

Vanaları orifis kontrol cihazları olarak tanımlarsak, doğal olarak orifis alanının azalıp, artması direkt olarak akışkan sürtünmesini etkiler. Denklem 3.1 ve Denklem 3.2'deki bağıntılarla bu durum anlatılabilir [1; 2; 7; 8].

$$v \approx (\Delta h)^{1/2} \quad (3.1)$$

$$v \approx (\Delta p)^{1/2} \quad (3.2)$$

Burada:

- v: Akışkanın Hızı
- Δh : Hmax'da oluşacak farkı
- Δp : basınç kaybı

Vananın kapalılık oranına göre birçok bağıntı, deneysel verilere göre yazılmıştır. Ortak kabul edilen ve nominal vana çapını belirlemek için kullanılan parametreler şöyledir:

- Sürtünme katsayısı: ξ
- Akış Katsayıları: C_v, K_v, A_v

[1; 2; 7; 8]

3.1 Sürtünme Katsayısı

Bir boru hattındaki vana içerisindeki sürtünmeden dolayı oluşan kayıplar Denklem 3.3 ve 3.4 ile ifade edilebilir [1; 2; 7].

$$\Delta h = \xi \frac{v^2}{2g} \quad (3.3)$$

$$\Delta p = \xi \frac{v^2 \rho}{2} \quad (3.4)$$

Burada:

g : Yerçekimine bağlı yerel ivmesi

ρ : Akışkanın yoğunluğu

Bu denklemler tek fazlı newton sıvılarda hem türbülanslı hem de lineer akış için geçerlidir. Ayrıca düşük Mach sayıları için bu denklemler gaz akışı içinde kullanılabilir [1; 2; 7].

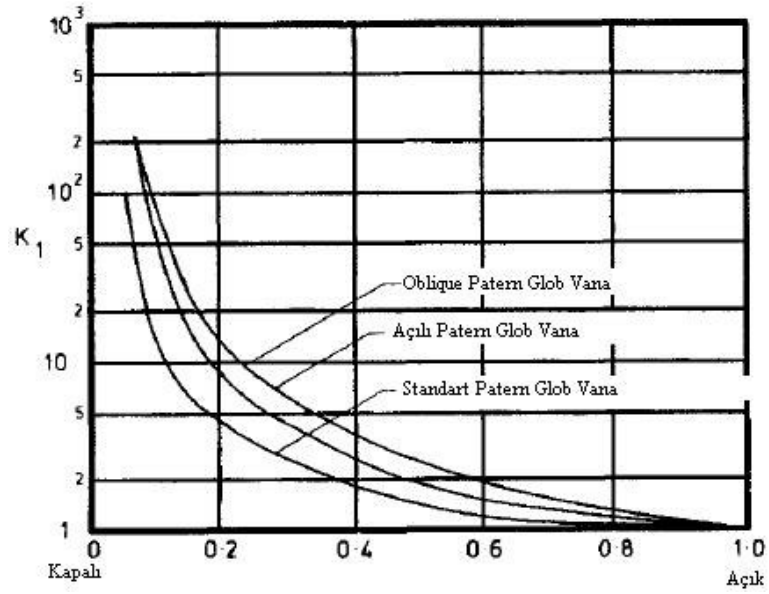
Vana girişindeki Mach değeri 0.2'ye ulaştığında sıkıştırılabilirlik algılanabilir bir düzeye ulaşabilir fakat bu değer 0.5'e ulaşana kadar ciddi bir değişikliğe yol açması beklenmez [1; 2; 7].

Aynı tip olan fakat değişik imalatçılar tarafından üretilen veya aynı imalatçı tarafından üretilen değişik boyutlardaki aynı tip vanalar geometrik olarak benzer değildir. Bu yüzden belli bir boyuttaki, belli bir tipteki ve belli bir imalatçı tarafından üretilen vanaların sadece kendilerine ait sürtünme katsayıları vardır. Aynı tip ve aynı boyuttaki fakat değişik imalatçılar tarafından üretilen vanaların sürtünme katsayıları ihmal edilemez şekilde değişik olabilir. Bu sebeple Çizelge 3.1 sadece tipik sürtünme katsayıları vermektedir. Bu katsayılar tam açık ve $Re \geq 10^4$ için geçerlidir [1; 2; 7].

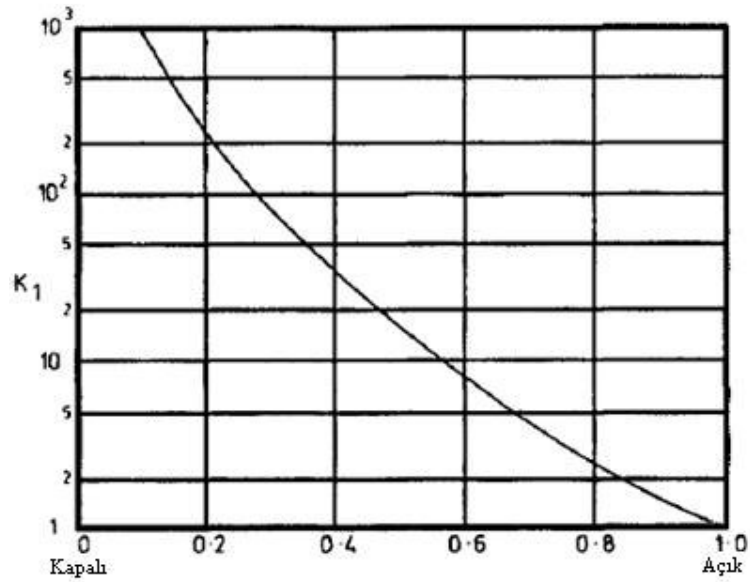
Çizelge 3.1 Tam türbülanslı akışta, tam açık vanalardaki yaklaşık Sürtünme Katsayıları [1]

Tip ve özellik	Katsayı (ξ)
Glob Vana, Standart Tip	
Tam Geçişli, Çelik	4.0-10.0
Tam Geçişli, Döküm (Sadece küçük boyutlardakiler)	5.0-13.0
Glob Vana, Eğik Tip	
Tam geçişli, Çelik	1.0-3.0
Glob Vana, Açılı Tip	
Tam Geçişli, Çelik	2.0-5.0
Tam Geçişli, Döküm (Sadece Küçük Boydakiler)	1.5-3.0
<i>Gate</i> Vana	
Tam Geçişli <i>Gate</i> Vana	0.1-0.3
Küresel Vana	
Tam Geçişli Küresel Vana	0.1
Plug Vana, Dikdörtgensel Geçiş	
Tam Akış Alanı	0.3-0.5
%80 Akış Alanı	0.7-1.2
%60 Akış Alanı	0.7-2.0
Plug Vana Dairesel Geçiş	
Plug Vana Dairesel Tam Geçiş	0.2-0.3
Kelebek Vana	
Kelebek Vana (Paravanın Kalınlığına Göre Değişken Olarak)	0.2-1.5
Diyafram Vana	
Setli (Weir) Tip	2.0-3.5
Tam Geçişli (Vana Yatağından)	0.6-0.9
Çek Vana	
Sallanan Tip	1.0
Kayan Disk Tipi	1.0

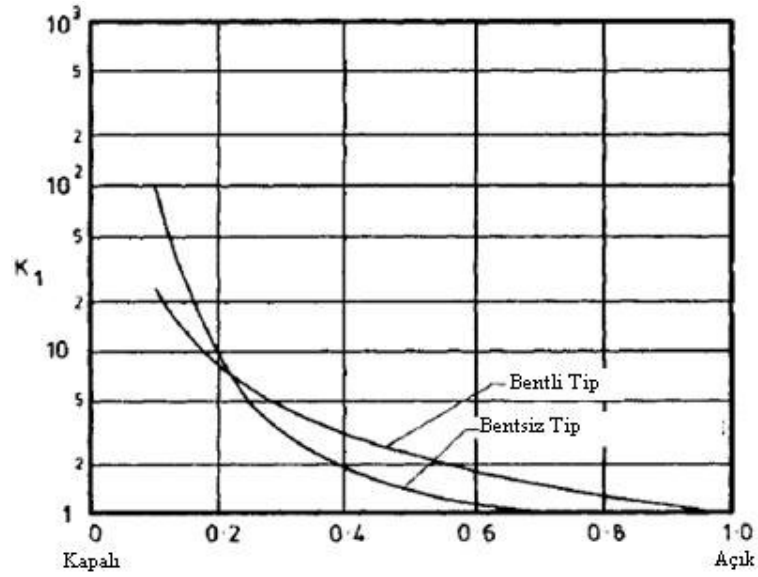
Tam açık olmayan vanalar için gerekli olan düzeltme katsayıları 3-1, 3-2, 3-3, 3-4 numaralı grafiklerde görülebilir [1].



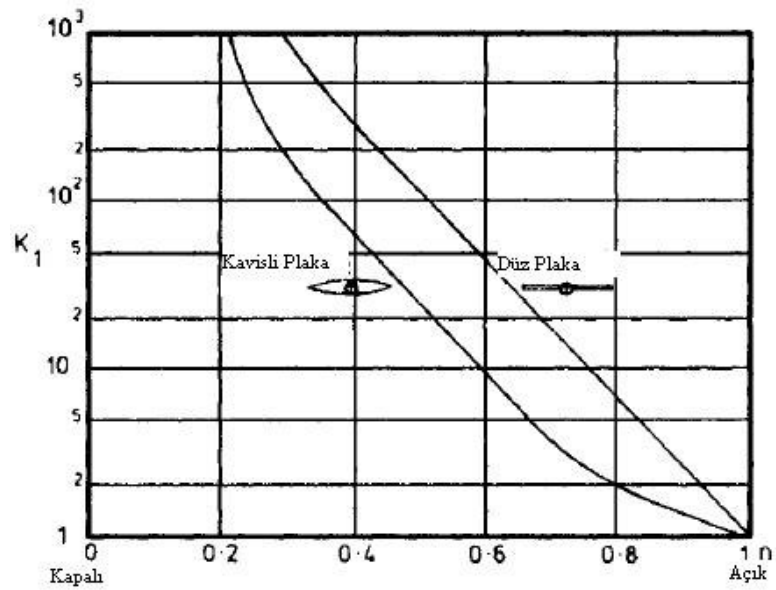
Şekil 3.1 Glob Vanaların Açılmasının Sürtünme Katsayısına Yaklaşık Etkileri [1]



Şekil 3.2 Gate Vanaların Açılmasının Sürtünme Katsayısına Yaklaşık Etkileri [1]



Şekil 3.3 Diyafram Vanaların Açılmasının Sürtünme Katsayısına Yaklaşık Etkileri [1]



Şekil 3.4 Kelebek Vanaların Açılmasının Sürtünme Katsayısına Yaklaşık Etkileri [1]

Şekil 3.5'te venturi tip bir *gate* vana çıkışı basınç kaybını göstermektedir. Burada da görülmektedir ki statik enerjiden geri kazanım çaplar (d^2/D^2) arasındaki orana, redüksiyon açısına ($\alpha/2$) ve vana yatakları çıkışı boru uzunluğuna (L) bağlıdır. Burada d vana boğaz çapı, D boru çapıdır [1; 7].

Eğer vana boğazından sonra $L \geq 12D$ o halde basınç kaybı Denklem 3.5 ile belirtilen Borda-Carnot kaybını geçemez [1; 7].

$$\Delta h = \frac{(V_d - V_D)^2}{2g} \quad (3.5)$$

Burada:

V_d = Dar boğazda akış hızı

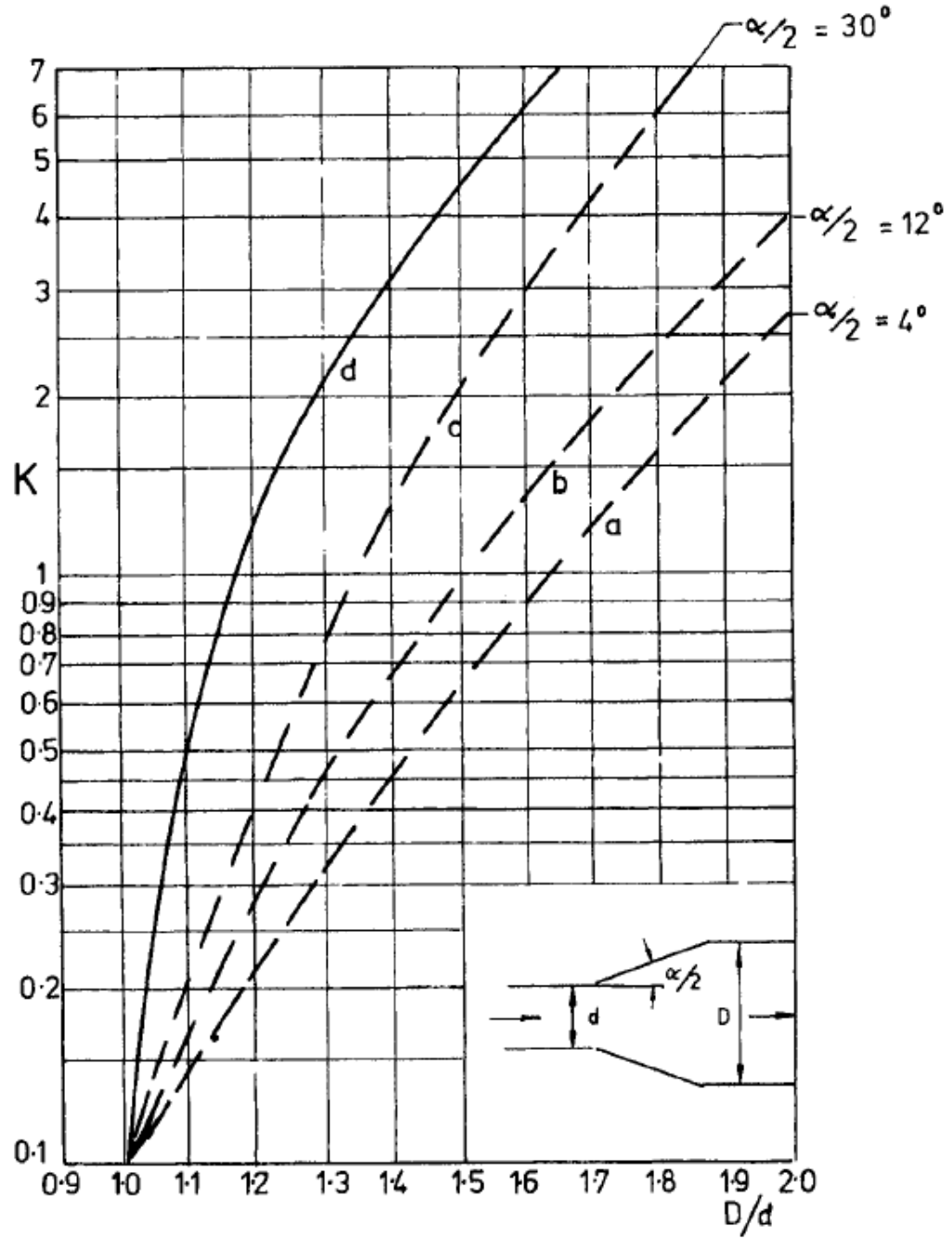
V_D = Geniş boğazda akış hızı

Maksimum basınç kaybı, $\alpha/2 > 30^\circ$ olduğunda oluşur. Açının azalması ile basınç kaybı da düşerek 4° derecede en düşük seviyesine ulaşır [1; 7].

Eğer vana boğazından sonra $L < 12D$ o halde basınç kaybı maksimum değere ulaşır. Burada kinetik enerjiye dönüşen statik enerji tamamen kaybolur. Bu durumda:

$$\Delta h = \frac{(V_d^2 - V_D^2)}{2g} \quad (3.6)$$

[1; 7]



Şekil 3.5 Tam Açık Daralan ve Genişleyen Akışlı *Gate* Vanalarda Sürtünme Katsayısı (a'dan c'ye kadar olan bütün eğriler $L \geq 12D$, d ise $L = 0$ 'ı gösterir. Yani redüksiyonun boyu yoktur. Redüksiyon mevcut değildir ve akış *gate* vananın venturi boğazından geçerek direkt olarak boruya devam eder.) [1; 2; 7]

3.2 Akış Katsayısı K_V

Akış katsayısı K_V , bir vanada 5°C ile 40°C arasındaki sıcaklıktaki akış (m^3 /saat) ve özgün bir açılış noktasında bir bar basınç kaybı olacak şekilde denklem haline getirilmiş hali Denklem 3.9'daki gibidir [1; 7].

$$K_V = Q \left(\frac{\Delta p_0}{\Delta p} \frac{\rho}{\rho_0} \right)^{1/2} \quad (3.9)$$

Burada:

Q : m^3 /saat

Δp_0 : referans basınç farkı (1 bar)

Δp : operasyonel basınç farkı (bar olarak)

ρ : referans akışkanın yoğunluğu (suyun yoğunluğu 1000 kg/m^3)

ρ_0 : operasyonel akışkanın yoğunluğu (kg/m^3)

G : Yerçekimi

ρ/ρ_0 spesifik yerçekimi ve $\Delta p_0 = 1$ olduğu için Denklem 3.9, Denklem 3.10'daki gibi kullanılabilir [1; 7].

$$K_V = Q \left(\frac{G}{\Delta p} \right)^{1/2} \quad (3.10)$$

[1]

3.3 Akış Katsayısı A_V

Akış katsayısı A_V , bir vanada 5°C ile 40°C arasındaki sıcaklıktaki akış (m^3 /saniye) ve özgün bir açılış noktasında bir paskal basınç kaybı olacak şekilde denklem haline getirilmiş hali Denklem 3.11'deki gibidir [1; 7].

$$A_V = Q \left(\frac{\rho}{\Delta p} \right)^{1/2} \quad (3.11)$$

Burada:

Q : m^3/saat

Δp : operasyonel basınç farkı (Pa olarak)

ρ : referans akışkanın yoğunluğu (suyun yoğunluğu kg/m^3)

Denklem 3.11, Denklem 3.4 'e uygulanırsa eğer:

$$Q = A \left(\frac{2}{\xi} \right)^{1/2} \left(\frac{\Delta p}{\rho} \right)^{1/2} \quad (3.12)$$

$$A \left(\frac{2}{\xi} \right)^{1/2} = Q \left(\frac{\rho}{\Delta p} \right)^{1/2} \quad (3.13)$$

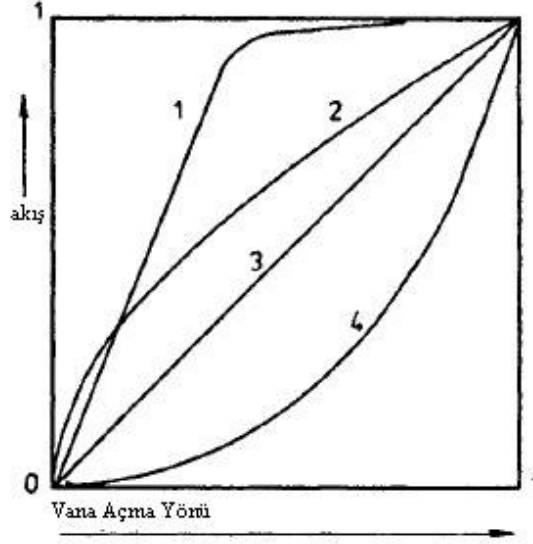
Burada:

A : kesit alanı

Denklem 3.13 gördüğümüz $A \left(\frac{2}{\xi} \right)^{1/2}$ ifadesi Denklem 3.11'de A_v sembolü ile ifade edilmektedir [1; 7].

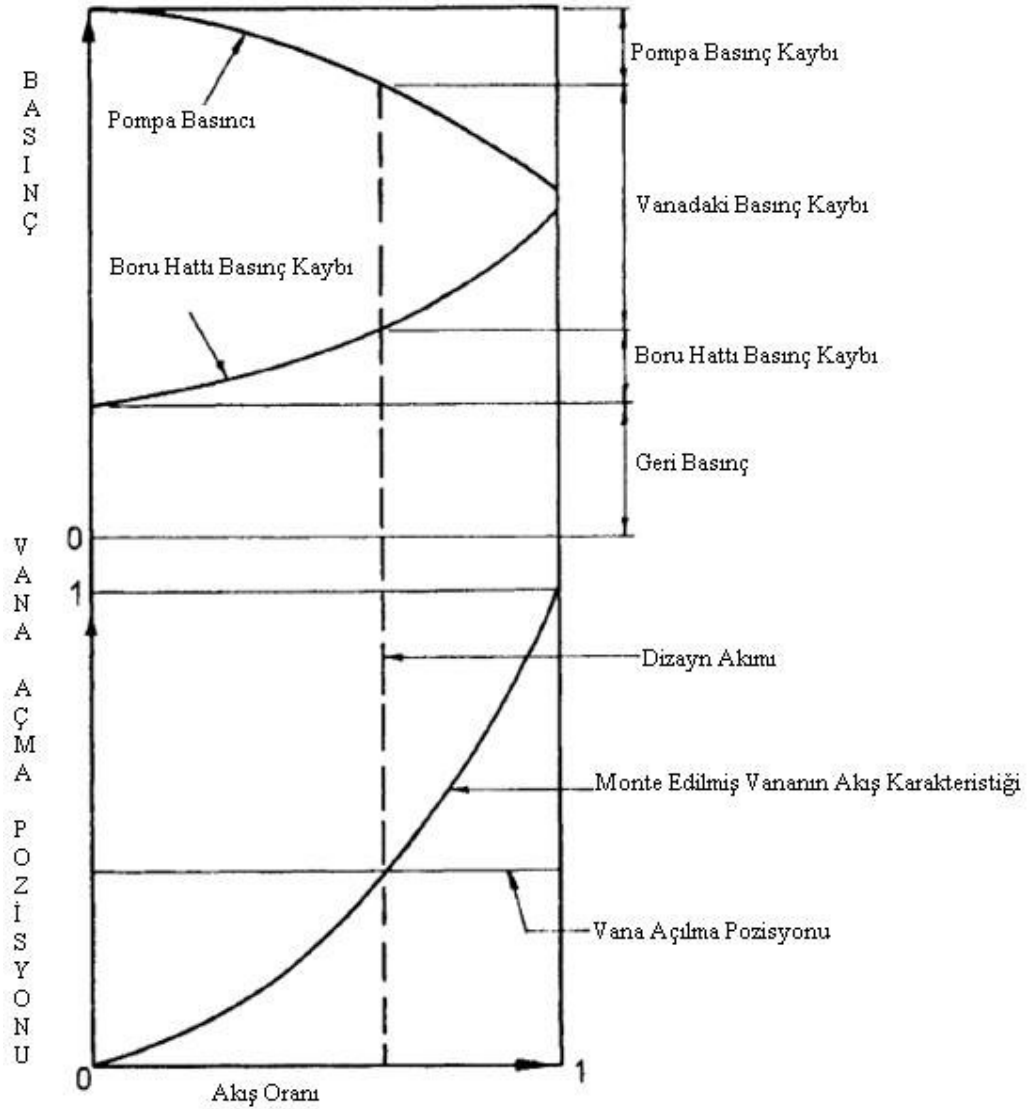
3.4 Vananın Açıklığıyla Sürtünme Katsayısı Arasındaki İlişki

Şekil 3.6, akış kontrol vanalarının vananın açılma pozisyonu ile aldıkları akış karakteristiğini göstermektedir [1; 5; 7].



Şekil 3.6 Vanaların Yapısal Akış Karakteri (1. Hızlı Açılma, 2. Karekök, 3. Lineer, 4. Eşit yüzdeliler) [1]

Birçok pratik uygulamada basınç kaybı vananın açılma pozisyonuna göre değişiklik göstermektedir. Bu grafik 3.7 'de olaya bir pompa da dahil edilerek gösterilmiştir. Grafiğin üst kısmı pompanın karakteristiğini göstermektedir. Bu üst kısımda pompanın basıncının akışla olan ilişkisi ve sistemin karakteristiği olan, boru hattı basınç kaybının akışla olan ilişkisi gösterilmiştir. Alt kısımda ise akış oranının vananın açıklık derecesine oranı gösterilmiştir. Bu karakteristik monte edilmiş vana karakteristiği olarak anılır ve her vana montajına göre ayrıdır. Vana daha fazla akış sağlamak için açıldıkça vana girişindeki basınç azalmaktadır. Bu durumda vana, yapısal karakterlerinde belirtilen vana açış hızından daha hızlı bir şekilde açılmalıdır ki aradaki basınç kayıpları sebebi ile yataklarda zarar oluşması önlenmelidir. Eğer pompa ve sistem karakteristiği vananın yüksek basınç düşüşüne maruz kaldığını gösteriyorsa bu durumlarda vananın bu basınç düşümünü kapalı duruma yakın durumlarda daha az hissetmesi için boru hattına konacak vana boyutlarının tekrar gözden geçirilmesi gerekir. Bu da çoğu zaman boru hattından daha küçük vana kullanılması sonucu verir [1; 5; 7].



Şekil 3.7 Akış oranı, Vana Açıklık Pozisyonu ve Basın Kaybı Arasındaki İlişki [1]

3.5 Vanalardaki Kaviteasyon

Kısmi olarak kapalı haldeki bir vanadan akışkan geçerken artan hızdan dolayı artan basınçtan dolayı, birde vana elemanı hareket etmeye başlayınca, vananın düşük basınç bölgesinde buharlaşma noktasına gelebilir. Düşük basınç bölgesindeki sıvı buharlaşma gösterince, buhar partikülleri sıvının içinde kaviteler oluştururlar. Bu kaviteler sıvı ile birlikte yol alan diğer gazlar ve katı yabancı maddeler etrafında

büyür. Daha sonra bu buhar partikülleri hattın statik basınç gösteren bir noktasına geldiklerinde çökerler ya da içe doğru patlarlar. Bu olaya kavitasyon denir [6; 8].

Kavitasyon, hatta yüksek fakat ani yüksek basınçlar sebep olur. Eğer bu ani basınç değişimleri boru atının çeperlerine yakın olursa oluşacak kısa ömürlü, ani şok darbeleri zamanla boru çeperlerinde pürüzlenmelere hatta büyük kavitelerin oluşmasına sebep olurlar [1; 6].

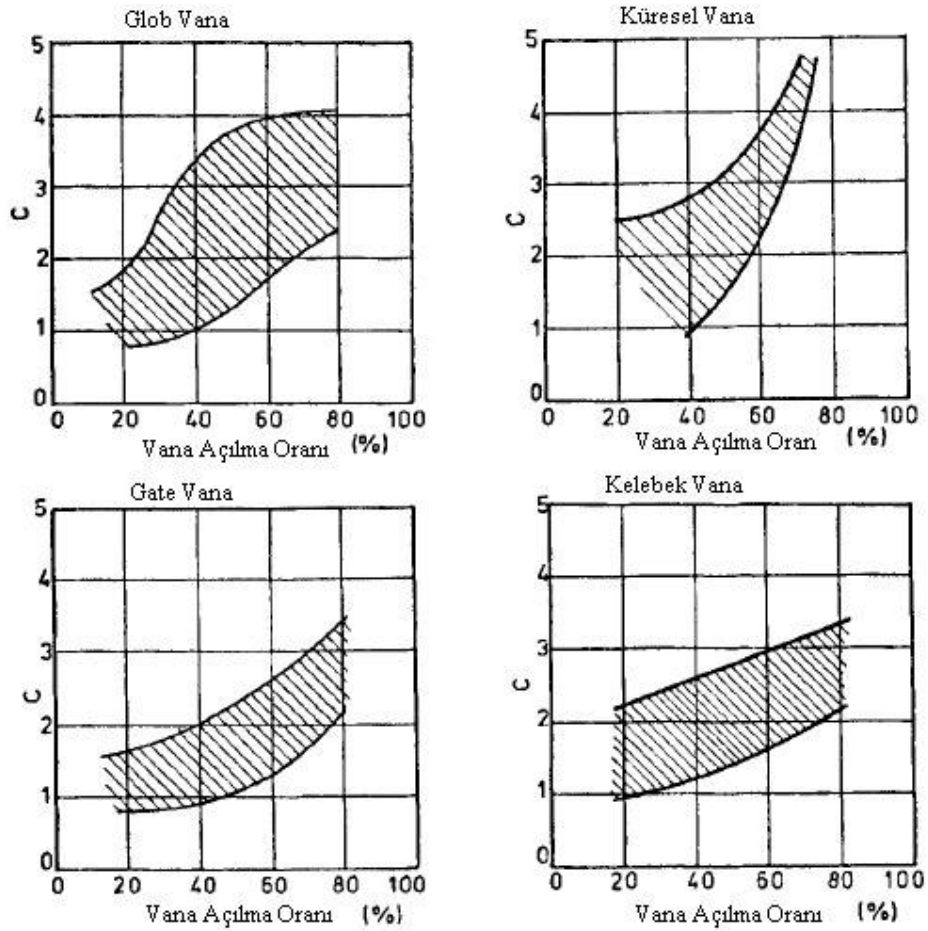
Her tip vananın kavitasyon kapasitesi o vanaya özgüdür. Kavitasyon olasılığını ya da derecesini göstermek için kullanılan indekse, kavitasyon indeksi denir. Denklem 3.20’de Amerikan Standartlarına göre bu indeksin denklemi mevcuttur [1; 6].

$$C = \frac{P_d - P_v}{P_u - P_d} \quad (3.20)$$

Burada:

- C: kavitasyon indeksi
- P_v : atmosfer basıncına göre buhar basıncı (negatif)
- P_d : vana yatağından boru çapının 12 katı kadar sonrasındaki basınç
- P_u : vana yatağından boru çapının 3 katı kadar öncesindeki basınç

Şekil 3.8 kelebek, *gate*, glob ve küresel vananın suya göre kavitasyon karakteristiğini göstermektedir. Fakat bu grafik kesin çevresel faktörlere dayanmadığı için daha çok bir yol gösterici olarak algılanabilir [1; 6].



Şekil 3.8 Bazı Vanaların Kaviteasyon Karakteristiği [1]

Kaviteasyon gelişmesini, basınç düşüşünü basamaklayarak önleyebiliriz. Vanadan sonra hatta basınçlı hava vererek kaviteasyona sebep olan buhar kabarcıkların sıvının içerisinde hapsolmesine sebep olabilir. Bu önlemin yan etkisi ise akış yönündeki ölçüm ekipmanının verdiği değerleri etkilemesidir [1; 6].

Vanadan sonra hattaki ani genişlemede vanayı ve boru hattını kaviteasyon hasarından koruyabilir. Bu ani genişleme bölümünün çapı, boru hattının çapının 1.5 katına ve boyu da boru hattının çapının 8 katına eşittir. Boy ölçüsünün içine çıkış redüksiyonu da dahildir. Bu uygulama özellikle iğne uçlu vanalarda çok iyi sonuç vermiştir [1; 6].

3.6 Vana Kullanımında Su Çekici Olayı

Vanalar kapanırken ve açılırken debideki değişim ve kinetik enerjideki değişim boru hattında geçici bir değişime sebep olur. Statik basınçtaki bu değişim bazen gürültü ve boru hattının sallanmasına yol açar bu sebepten bu olaya su çekici olayı denir (su darbesi). Su buharının basıncının değişiminde yaşanan olay ise buhar darbesidir [6; 8].

Bu darbe hattın tümünde anında hissedilmez fakat basınç değişiminin yaşandığı bölgede net biçimde görülür. Bu kinetik enerji sıvı moleküllerini hattın duvarlarına doğru sıkıştırır. Akış yönüne doğru hareket eden sıvı kütlesi ise sabit duran kütleye kadar orijinal hızı ile devam eder. Bu basınç alanın hızı sıvı içindeki ses hızına eşittir. Bu basınç alanı hattın sonuna ulaştığında sıvı durgundur fakat normal statik basınçtan daha yüksek bir basınç gösterir ve boru hattının çeperlerinde genişleme görülür. Bu aşamada dengede olmayan basınç dalgası ters yönde bir akışa sebep olur ve bu normal statik basıncı normal hale getirir ve boru hattının çeper genişlemesini durdurarak normal hale döndürür. Bu basınç düşüşü vanaya ulaştığında tüm sıvı kolonu tekrar normal statik basınç altındadır fakat hala hat girişine doğru devam etmek istemektedir. Bu yüzden bir düşük basınç dalgası daha oluşur. Bu dalgada bir tam tur yaptıktan sonra artık akış normal yönünde normal statik basınç altındadır. Bu döngü sıvının kinetik enerjisi tükenene kadar devam eder [1; 6; 8].

Jouskowsky'e göre ani vana kapamaları hattaki statik basıncı Denklem 3.21'de ifade edildiği şekilde yükseltir.

$$\Delta P = av\rho/B \quad (3.21)$$

Burada:

ΔP : normalin üzerindeki basınç yükselişi

a: basınç dalgasının yayılma hızı

[1; 6; 8]

$$a = \left[\frac{K}{\frac{\rho}{B} \left(1 + \frac{K D c}{E e} \right)} \right]^{1/2} \quad (3.22)$$

Burada:

- K: Akışkanın bulk modülü
- E: Boru malzemesinin elastisitesinin Young modülü
- D: borunun iç çapı
- e: Boru duvarının kalınlığı
- c: boru kısıtlama faktörü (c = 1.0 kısıtlamasız borular için)
- B: 1 (metrik sistemde) (İngiliz sisteminde, fps, g = 32.174 ft/s²)

D/e oranı 35 olan çelik bir boruda akışkan olarak su kullanılırsa eğer, basınç dalgası yaklaşık 1200 m/s (4000 ft/s) ile hareket eder. Statik basınç her 1 m/s 'lik hız değişimi için 13.5 bar yükselir (ya da her 1ft/s için 60 lb/in²). Eğer vana anında kapanmayıp da basınç dalgasının bir tam tur atması süresi olan 2L/a'nın vereceği değerinde kapanırsa (L burada borunun uzunluğudur.) bu da karşılıklı hareket eden basınç dalgalarının birbirlerini sönmömlmelerini engelleyeceği için anlık kapama etkisi yapar. Sanki anlık kapandığı zamandaki gibi yüksek basınç gözlemlenir. Bu kapama hızlı olarak nitelenebilir. Eğer 2L/a'dan daha uzun bir zamanda kapanırsa o zaman karşılıklı basınç dalgalarının birbirini sönmömlmesine fırsat verilmiş olunur ve kapama sırasında ulaşılacak maksimum basınç azaltılmış olur. Bu kapama yavaş olarak nitelenebilir. Manüel olarak kullanılan vanalar, akış hızında uniform değişim olması için yavaş kapanmalıdır. Çek vanaların kapanma hızı, öteki taraftan, vana tasarımının bir fonksiyonu gibi akışkanın özelliklerine göre olduğundan daha kontrol dışı sayılabilir [1; 6; 8].

Eğer ani basınç değişimi pompanın durmasından dolayı ise enerji kesildikten sonraki pompa hızının değişim oranı dikkate alınmalıdır [1; 6; 8].

Akışkanın basınç ve hız formülasyonu, zaman ve boru hattındaki pozisyona göre birkaç şekilde yapılabilir. Basit olaylar için grafikler veya cebrik hesaplar kullanılabilir fakat daha karışık problemler veya daha kesin değerler için nümerik metotlar ile bilgisayarlar devreye girmelidir [1; 6; 8].

Bazı durumlarda su darbesini, vana karakteristikleri ile oynayarak önlemek mümkün değildir veya pratikte değildir. Bu durumda boru hattının karakteristiğini değiştirme yoluna gidilmelidir. Bunun için, boru hattının belirli yerlerine sıvı ile direkt temasta gaz bulunduran elemanlar, sıvı ile gaz arasında esnek ince bir duvar içeren elemanlar veya *pressure relief* vanalar kullanılabilir [6; 8].

Su darbesi ayrıca birde akışkanın akustik özelliklerini değiştirerek de önlenabilir. Akışkana kasıtlı bir biçimde akışkan sıvıda çözünmeyen gaz kabarcıkları gönderilerek bu metot uygulanabilir. Bu işlem sıvının yoğunluğunu ve Bulk modülünü düşürür. Aynı etki boru hattı boyunca döşenmiş bir hortum içinde veya çeperleri esnek başka tür bir hat içinde gaz ile de sağlanabilir [6; 8].

Sadece küçük bir miktar gaz mevcutsa bile 3 – 20 numaralı denklem artık geçersizdir. Değişmiş olan akustik hız Denklem 3.23'deki gibi ifade edilir [8].

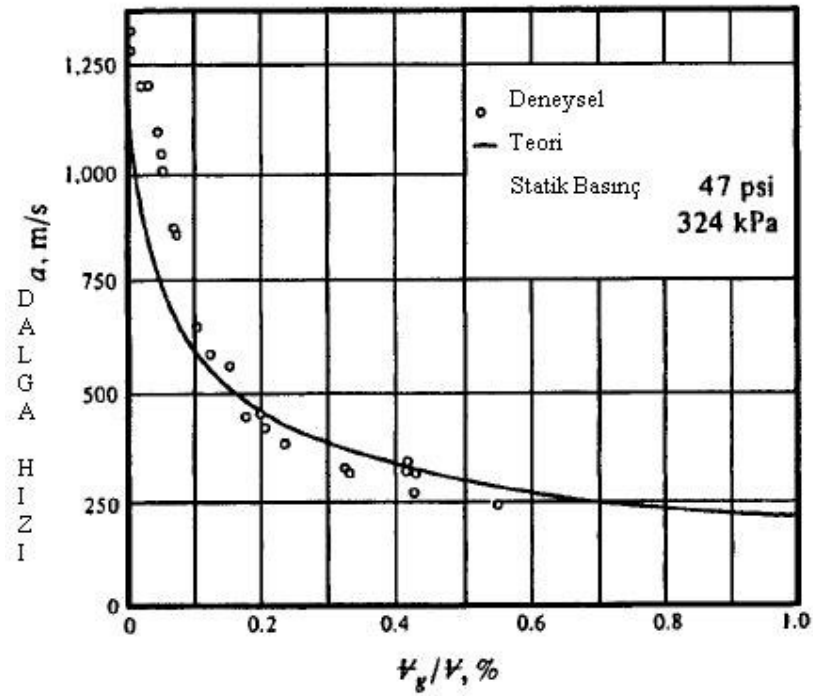
$$a = \sqrt{\frac{BK}{\rho}} \quad (3.23)$$

$$K = \frac{K_1}{1 + \left(\frac{V_g}{V_t}\right)\left(\frac{K_1}{K_g} - 1\right)} \quad (3.24)$$

$$\rho = \rho_g \frac{V_g}{V_t} + \rho_l \frac{V_l}{V_t} \quad (3.25)$$

Burada:

- K_g : Gazın Bulk modülü
- K_l : Sıvının Bulk modülü
- V_g : Gazın hacmi
- V_l : Sıvının hacmi
- V_t : Toplam hacim
- ρ_g : Gazın yoğunluğu
- ρ_l : Sıvının yoğunluğu



Şekil 3.9 Teorik ve Deneysel Veriler ile Bir Basınç Dalgasının Değişik Hava İçeriğine Göre Yayılma Hızı (a) [8]

Şekil 3.9 su için bünyesindeki havanın basınç dalgası hızına etkisini göstermektedir. Akışkan içindeki küçük miktarlardaki hava bile akışkandaki dalga hızının havadaki ses hızının (20°C (68°F)'deki kuru havadaki hızı: 318 m/s (1042ft/s)) dahi altına düşmesine sebep olur [8].

3.7 Vana Sesinin Azaltılması

Yüksek basınç alanlarından düşük basınç alanlarına gaz akışının sağlanması rahatsız edici bazen de tahammül edilemez ses oluşumuna sebep olur. Ses yüksek hıza sahip gaz jetinin vananın çıkış tarafındaki akışı kırmasından meydana gelmektedir. Delikli difüzer, bir bölücü vazifesi görerek sesin önlenmesinde başarılı olduğu görülmüştür. Bu şekilde gaz birçok küçük orifizden geçmek zorunda bırakılmıştır [1; 6 ;9].

Difüzer düşük ve orta frekanstaki sesleri sönümler fakat gaz orifizlerden geçerken yüksek frekansta sesler oluşmasına sebep olur. Fakat bu yüksek frekanstaki

ses akış içindeki havada ve boru hattı boyunca, hattın özelliğinden dolayı sönümlenir. Difüzerin başka bir faydası da akışı daha eşit bir biçimde dağıtıyor olmasıdır [1; 6; 9].

Ingard'a göre boru hattında dikey perfore (delikli sac) plakanın normalize edilmiş akustik direnci, hem orifislerden geçen akışın Mach sayısı ile hemde Denklem 3.26'da ifade edilen faktörle direkt orantılıdır [1; 6; 9].

$$\left(\frac{1-\sigma}{\sigma}\right)^2 \quad (3.26)$$

Burada σ , difüzör üzerindeki deliklerin oranıdır. Ses azalması için Mach sayısı mümkün olduğunca büyük, σ mümkün olduğunca küçük olmalıdır. Pratik sebeplerle Mach sayısı maksimum Mach sayısı olarak 0,9 tavsiye edilmektedir. Eğer difüzör etrafındaki basınç düşüşü limite tabi olmak zorunda ise o zaman daha düşük bir Mach sayısı seçilebilir. Difüzerdeki açık alan oranı pratik olarak 0.1 ile 0.3 arasında seçilir. Eğer 0.1'den küçük seçilirse difüzer gereğinden fazla büyük olur. 0.3'ten büyük olursa seste gerekli düşme sağlanamayabilir [1; 6; 9].

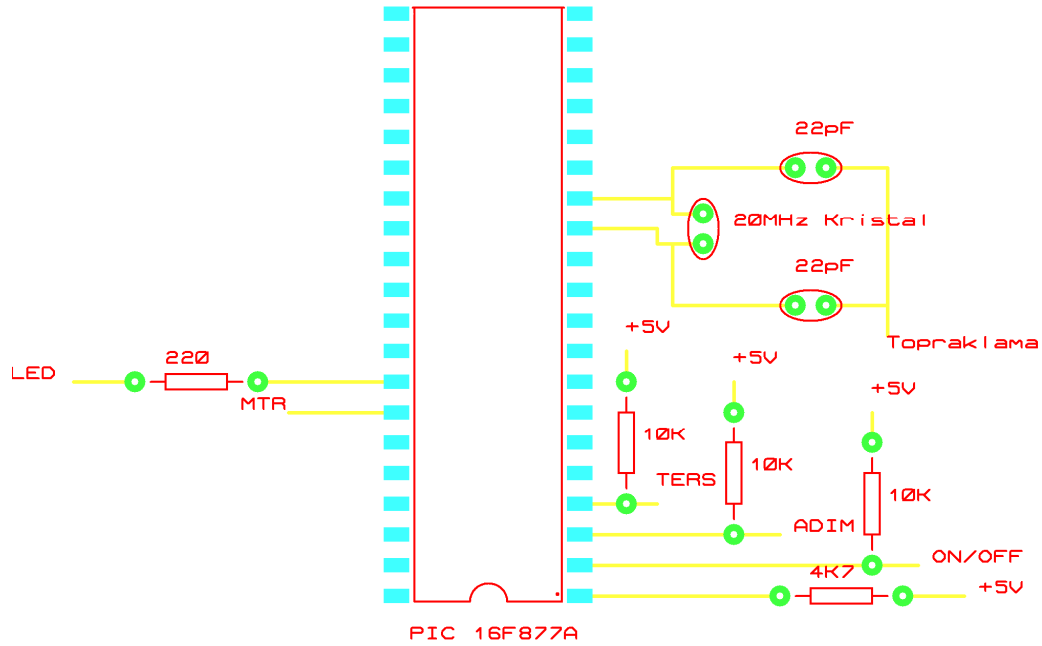
Ayrıca jetin yarattığı pik frekans orifisin çapıyla ters orantılıdır. Sesin düşürülmesi açısından bakıldığında perfore (delikli) sacın deliklerinin çaplarının mümkün olduğunca küçük olması gerekir. Deliklerin tıkanmasını önlemek için de minimum olarak 5mm çap genellikle kullanılmaktadır. Eğer difüzer sonrası akış hızı yüksekse, bu da hatta vananın yarattığı sese yakın bir ses yaratabilir. Bu sorun ise boru içi akış için Mach sayısını 0.3'e yaklaşık altında alarak çözülebilir. [1; 6; 9]

4. VANA TAHRİK DEVRESİ

Bu bölümde tasarlanan 4 tip kontrol sisteminin motor kontrol devreleri verilmiştir. Bu devreler Proteus Ares Programı tarafından çizilmiştir. Bu bölüm devrelerin birbirine benzerlikleri ve birbirlerinden farklarını da tartışma konusundan yararlıdır.

4.1 Servo Motor Devresi

Bölüm 4.1’de tartışılan kontrol biçimi daha hassas stem kontrolünü sağlayan servo motor kontrolüdür. Şekil 4.1’de verilen devre PIC 16F877A mikrodnetleyicisi kullanılarak yapılmış bir servo motor kontrol ünitesidir. Bu devrede 5 volt ile beslenen bir DC servo motor kontrol etmek için ara kademe olarak bir H-Köprüsü devre bölümüne veya entegresine ihtiyaç duyulmamıştır. Bu devre ve PIC mikrodnetleyicisi içerisindeki kod, motorun anahtar kontrolü ile ileri veya geri adım atmasını sağlar [10; 11; 12].

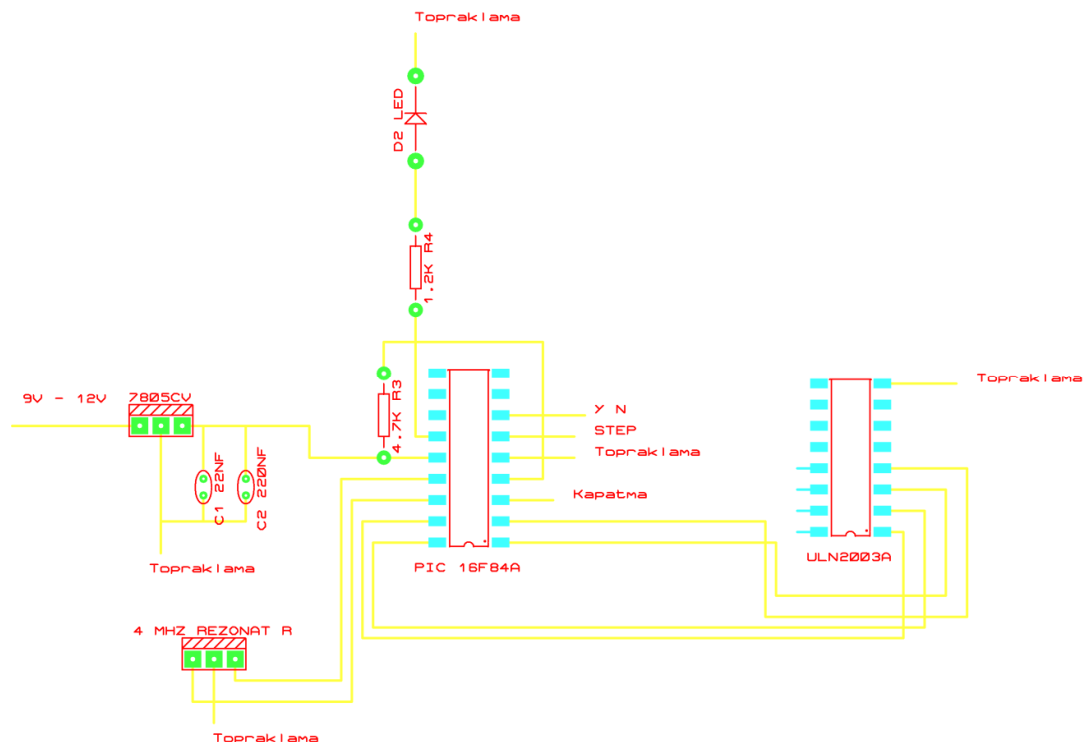


Şekil 4.1 Servo Motor Kontrol Devresi [11; 12]

Bu devre proses kontrol için geliştirilmeye karar verilse bile PIC 16F84 ve benzeri mikrodentleyiciler ile yapılan ve aynı işi yapan diğer devreler kadar karmaşık bir hale gelmez. Nedeni ise PIC 16F877A'nın çok daha gelişmiş bir mikrodentleyici olması ve diğer devre elemanlarına ihtiyaç duyulacak tasarımlarda dahi bünyesinde diğer mikrodentleyicilerden farklı özellikler taşıdığından dolayı devrenin daha sade olmasını sağlamasıdır. Örnek olarak içerisinde analog-dijital çeviricisi mevcuttur [10; 11; 12; 13].

4.2 Step Motor Devresi

Step Motor da aynı servo motor gibi dışı sistemi olmadan vana kontrolü için seçilebilecek sistemlerdendir. Step (Adım) motor devresinde, servo motor devresinde olmayan H-Köprüsü kullanılmıştır. Şekil 4.2’de PIC 16F84A kullanılarak kurulmuş bir bi-polar step motor devresi görülmektedir. Bu devrede 12 Volt ile beslenen bi-polar step motor mikrodnetleyici tarafından ULN2003A entegre devresi ile yönetilmektedir. Bu devre sayesinde anahtarlar ve mikrodnetleyiciye yüklenen program yordamı ile step motora iki yönde de adım attırılabilmektedir. [14; 15; 16; 17]

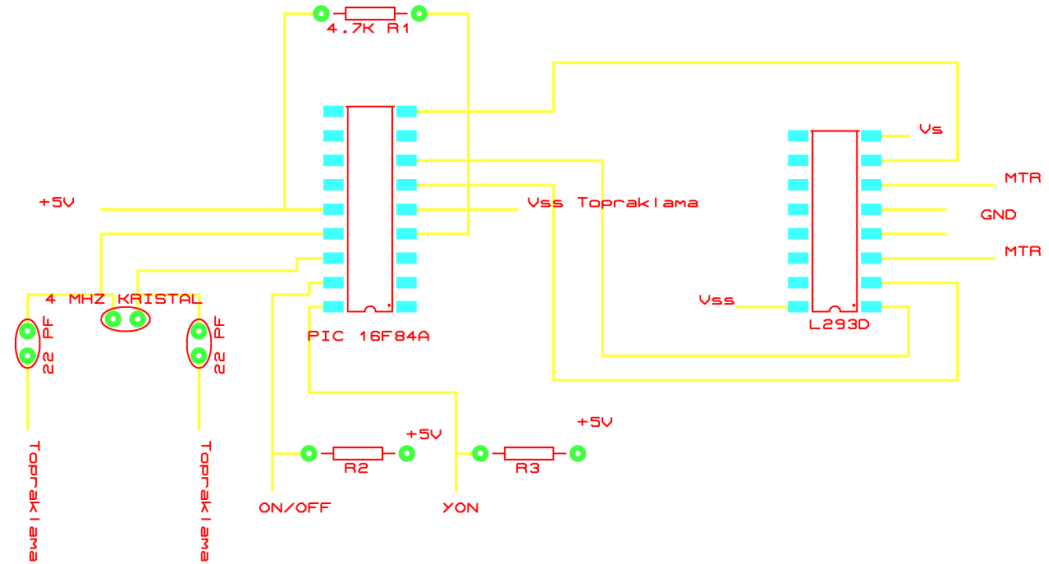


Şekil 4.2 STEP Motor Kontrol Devresi [15; 17]

ULN2003A H-Köprüsü aynı anda iki motoru kontrol edebilmektedir. PIC 16F84A'nın port sayısı 16F877A'dan daha az olduğu için proses kontrol geliştikçe devreye daha fazla 16F84 takviyesi gerekebilir [14; 15; 16; 17].

4.3 L293D – DC Motor Devresi

Bu çalışmada tartışılan iki H-Köprülü DC motor kontrolünden ilki olan kontrol biçimi Şekil 4.3'te verilmiştir. Burada L293D H-Köprüsü entegre devresi kullanılarak PIC 16F84A'nın bir DC motor ile iletişim kurması sağlanmıştır. Mikrodenetleyiciye yüklenen program ve L293D sayesinde anahtarlar ile opere edilerek bir DC motor hem saat yönü hemde saat yönünün tersine döndürülebilir [12; 13; 14; 15; 16]



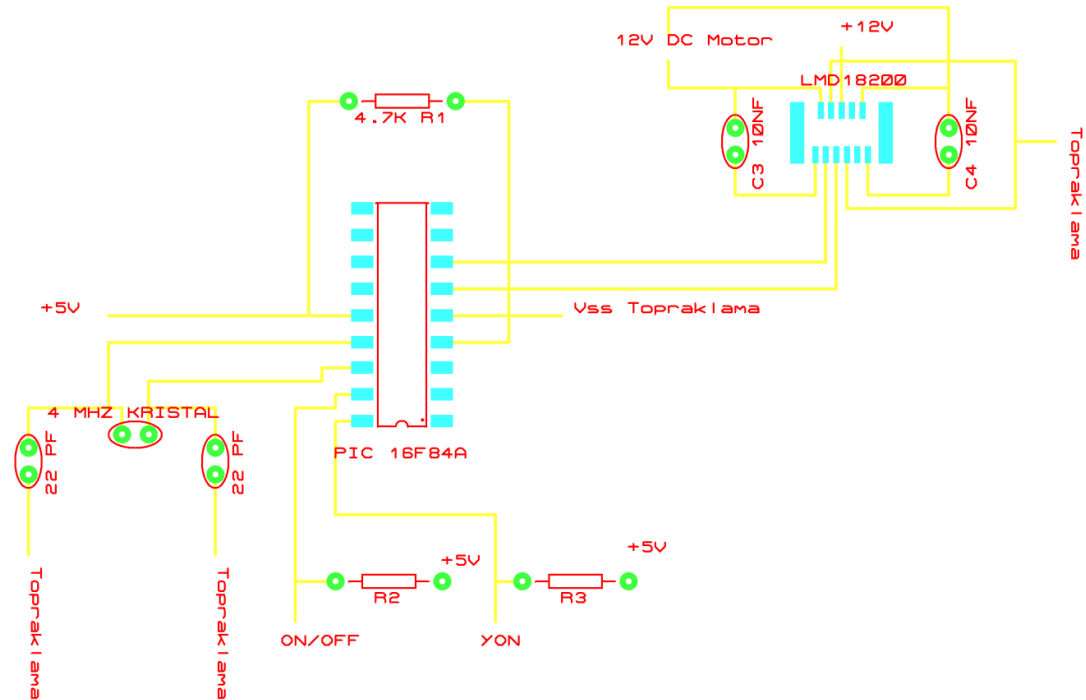
Şekil 4.3 L293D ile DC Motor Kontrol Devresi [15]

Bu çalışmada incelenen DC motor kontrol eden diğer H-Köprüsü entegre devresi olan LMD18200T'ye göre L293D ile kontrol edilebilen DC motor çeşit yelpazesi daha küçüktür [8]. Vana seçimi açısından düşünürsek, daha küçük çapta vanaları kontrol etmek için bu entegre düşünülebilir. DC motor bir dişli sistemi vasıtası ile vananın stemine bağlanmaktadır. Bu sebeple elektrik kesintilerinde step motor ve servo motor (stem'e direk bağlı olduğu düşünülürse) yapıları gereği küçük

farklarda olsa, tahrik kontrol mekanizması da devre dışı olduğu için stemin yeri değişebilmesine rağmen, bu sistmede stemin yeri değişmez [12; 13; 14; 15; 16].

4.4 LMD18200 – DC Motor Devresi

Bu çalışmada tartışılan ikinci DC motor kontrol tasarımı ise LMD18200 H-Köprüsü kullanılarak yapılmıştır. Şekil 4.4’de PIC 16F84A ve LMD18200 entegre H-Köprüsü sayesinde DC motor kontrolü sağlanmasına yarayan devrenin şeması görülmektedir. Mikrodenetleyiciye yüklenen program sayesinde DC motor hem saat yönüne hemde saat yönünün tersine döndürülebilmektedir [14; 15; 16].



Şekil 4.4 LMD18200T ile DC Motor Kontrol Devresi [15]

Şekil 4.4’de gösterilen devre şemasında LMD18200’ün T tipi paketlemesi kullanılmıştır. LMD18200T ile L293D’ye oranla daha çok DC motora hitap edilebilmektedir [9]. Bu sayde çok büyük olmasada daha geniş çapta vanalar tasarım yapılmak istenirse düşünülebilir. Bunun dışında programlama prensibi aynı L293D gibidir. Algoritmalarındaki değişiklik aslında port çıkış numaralarından ibarettir. L293D’de ayrıca *Enable* pinine de sinyal gönderilmelidir [14; 15; 16].

5. VANA KONTROL SİSTEMİ DONANIM SEÇİMİ

Bu bölümde 4. Bölümde tartışılan kontrol devre tasarımlarını oluşturan elemanlar daha detaylı bir şekilde ele alınmışlardır. Özellikle mikrodenetleyiciler üzerinde durulmuş ve aralarındaki farklar verilmiştir. Bu kıyaslama da tasarımları tartışırken verimli bir sonuca ulaşmada fayda sağlayabilir.

5.1 PIC 16F877 Donanımı

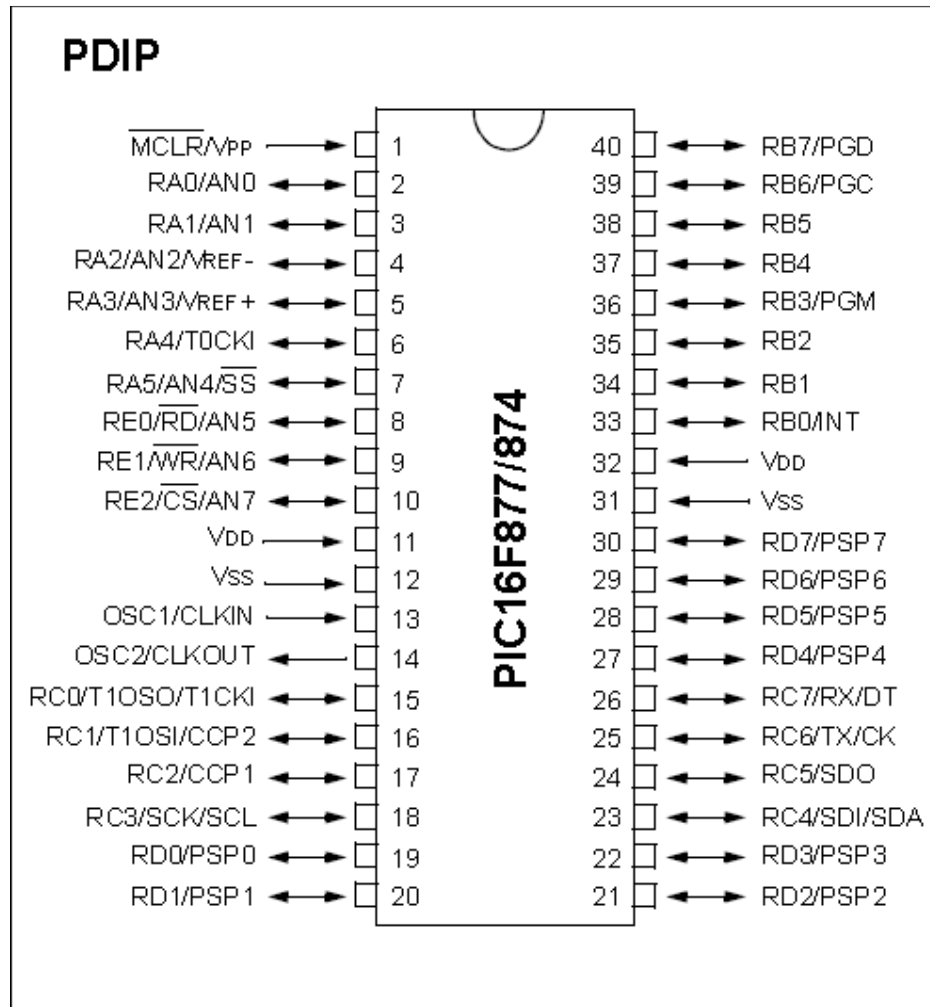
PIC 16F877'ler RISC CPU mimarisine göre tasarlanmışlardır. Toplam 35 assembler komutu ile programlanabilirler. Programda dallanma yapanlar dışında tüm komutlar tek çevrimdir. Programda dallanma yapanlar çift çevrimdir. Çalışma hızları; DC – 20 MHz saat girdisidir. (DC – 200ns komut çevrimi). 8K x 14 Word FLASH program hafızası, 368 x 8 byte data hafızası (RAM), 256 x 8 byte EEPROM data hafızası mevcuttur. 14 kaynağa kadar kesme özelliği mevcuttur. Sekiz seviye derinliğinde yığın özelliği mevcuttur. Direkt, indirekt ve relativ adresleme modlarına sahiptirler. Power-on Reset (POR), Power-up Timer (PWRT), Oscillator Start-up Timer (OST) özellikleri mevcuttur. Watchdog timer kendi osilatörü mikrokontrolere monte biçimde 16F877 yapısında bulunur. Kod koruması mevcuttur. Gücün daha verimli kullanılmasını sağlayan SLEEP (uyku) modu mevcuttur. Osilatör opsiyonları seçilebilen cinstendir [10; 12].

İki pin ile In-Circuit Seri Programlama (ICSP) özelliği mevcuttur. 5V ile in-Circuit Seri Programlama (ICSP) kapasitesi mevcuttur. İki pin ile In-Circuit debugging mümkündür. [10; 12]

5.1.1 Çevresel Özellikleri

PIC 16F877ler 8-bit önbölücülü (prescaler) 8-bit timer/counter (zamanlayıcı/sayıcı) 'a sahiptir (TMR0). 16-bit önbölücülü (prescaler) -bit timer/counter (zamanlayıcı/sayıcı) 'a sahiptir (dış saat veya kristal ile SLEEP durumunda dahi arttırılabilir.) (TMR1). 8-bit önbölücülü ve son bölücü (prescaler ve

postscaler) 8-bit timer/counter (zamanlayıcı/sayıcı) 'a sahiptir (TMR2). İki adet yakala (capture), karşılaştır (compare) ve PWM modülü vardır: capture 16-bitliktir ve maksimum çözünürlük 12,5 ns'dir, compare 16-bitliktir ve maksimum çözünürlük 200 ns'dir, PWM'nin maksimum çözünürlüğü 10-bit'tir. 10-bitlik çoklu kanal analog-dijital çeviricisi vardır. SPI (Master mode)'sı ve I²C (Master/Slave)'si olan Senkron Seri portu (Synchronous Serial Port :SSP) mevcuttur. PIC 16F877ler 9-bit adres saptayıcı üniversal senkron asenkron alıcı vericiye (Universal Synchronous Asynchronous Receiver Transmitter: USART/SCI) sahiptir. 8-bit genişlikte, dış $\overline{RD}$, $\overline{WR}$ ve $\overline{CS}$ kontrollü (sadece 40/44 pin) Parallel Slave Port (PSP) özelliği mevcuttur. Brown-out Reset (BOR) için Brown-out detection devresi mevcuttur. [10; 12]



Şekil 5.1 PIC 16F877 PDIP Pin Yayılımı [10]

Çizelge 5.1’de PIC 16F877A’nın pinleri tanımlanmıştır. Fakat bu çalışmada PDIP kullanıldığı için sadece bu modelin pin tablosu verilmiştir [10; 12].

Çizelge 5.1 PIC 16F877A’nın Pin Tablosu [10]

Pin İsmi	DIP No.	I/O/P Tipi	Buffer Tipi	Tanım
OSC1/CLKIN	13	I	ST/CMOS ⁴	Osilatör kristali girdisi / Dışsal saat kaynağı girdisi
OSC2/CLKOUT	14	O	-----	Osilatör kristali çıkışı. Kristal osilatör modunda kristale veya resonatöre bağlar. RC modunda OSC2 pini CLKOUT çıkışı olur. Bu OSC1’in frekansının 1/4’ü kadardır. Ayrıca komut çevrim oranını gösterir.
$\overline{\text{MCLR}}/\text{V}_{\text{PP}}$	1	I/P	ST	Master Clear (Reset) girdi/programlama voltaj girişi. Bu pin, ayarda, bir aktif düşük RESET’tir.
PORTA iki yöne de ayarlanabilen bir I/O port’udur.				
RA0/AN0	2	I/O	TTL	RA0 ayrıca analog girdi0 de olabilir.
RA1/AN1	3	I/O	TTL	RA1 ayrıca analog girdi1 de olabilir.
RA2/AN2/V _{REF-}	4	I/O	TTL	RA2 ayrıca analog girdi2 de olabilir veya negatif analog referans voltajı olabilir.
RA3/AN3/V _{REF+}	5	I/O	TTL	RA3 ayrıca analog girdi3 de olabilir veya negatif analog referans voltajı olabilir.
RA4/T0CKI	6	I/O	ST	RA4 ayrıca Timer0 timer/counter için saat girdisi de olabilir. Çıktı açık <i>drain</i> tipidir.

RA5/ $\overline{SS}$ /AN4	7	I/O	TTL	RA5
PORTB iki yöne de ayarlanabilen bir I/O port'udur. PORTB tüm girdilerde zayıf pull-up'lar için programlanabilir.				
RB0/INT	33	I/O	TTL/ST ¹	RB0 dış interrupt pini de olabilir.
RB1	34	I/O	TTL	
RB2	35	I/O	TTL	
RB3/PGM	36	I/O	TTL	RB3 ayrıca düşük voltaj programlama girdisi olabilir.
RB4	37	I/O	TTL	Kesme olduğunda değişen pin.
RB5	38	I/O	TTL	Kesme olduğunda değişen pin.
RB6/PGC	39	I/O	TTL/ST ²	Kesme olduğunda değişen pin ya da In-Circuit debugger pini. Seri programlama saati.
RB7/PGD	40	I/O	TTL/ST ²	Kesme olduğunda değişen pin ya da In-Circuit debugger pini. Seri programlama saati.
PORTC iki yöne de ayarlanabilen bir I/O port'udur.				
RC0/T1OSO/T1CKI	15	I/O	ST	RC0 ayrıca Timer1 osilatör çıktısı ya da Timer1 saat girdisi olabilir.
RC1/T1OSI/CCP2	16	I/O	ST	RC1 ayrıca Timer1 osilatör girdisi ya da Capture2 girdi/Compare2 çıktı/PWM2 çıktısı olabilir.
RC2/CCP1	17	I/O	ST	RC2 ayrıca Capture2 girdi/Compare2 çıktı/PWM2 çıktısı olabilir.
RC3/SCK/SCL	18	I/O	ST	RC3 ayrıca hem SPI hem de I ² C modları için senkron seri saat girdi/çıkı olabilir.
RC4/SDI/SDA	23	I/O	ST	RC4 ayrıca SPI data al(in) (SPI modunda) ya da data I/O (I ² C modu) olabilir.

RC5/SDO	24	I/O	ST	RC5 ayrıca SPI data ver(out) (SPI modunda)
RC6/TX/CK	25	I/O	ST	RC6 ayrıca USART asenkron verici ya da senkron saati olabilir.
RC7/RX/DT	26	I/O	ST	RC6 ayrıca USART asenkron alıcı ya da senkron data olabilir.
PORTC iki yöne de ayarlanabilen bir I/O port'udur ya da mikroprosesör bus'ı karşısında paralel slave portudur.				
RD0/PSP0	19	I/O	ST/TTL ³	
RD1/PSP1	20	I/O	ST/TTL ³	
RD2/PSP3	21	I/O	ST/TTL ³	
RD4/PSP4	22	I/O	ST/TTL ³	
RD5/PSP5	27	I/O	ST/TTL ³	
RD6/PSP6	28	I/O	ST/TTL ³	
RD7/PSP7	29	I/O	ST/TTL ³	
PORTE iki yöne de ayarlanabilen bir I/O port'udur.				
RE0/ $\overline{\text{RD}}$ /AN5	30	I/O	ST/TTL ³	RE0 ayrıca paralel slave port için okuma kontrol olabilir. Ya da analog girdi5 olabilir.
RE1/ $\overline{\text{WR}}$ /AN6	8	I/O	ST/TTL ³	RE1 ayrıca paralel slave port için yazma kontrol olabilir. Ya da analog girdi6 olabilir.
RE2/ $\overline{\text{CS}}$ /AN7	9	I/O	ST/TTL ³	RE2 ayrıca paralel slave port için kontrol seçme (select control) kontrol olabilir. Ya da analog girdi7 olabilir.
V _{SS}	12, 31	P	-----	Lojik ve I/O pinleri için topraklama
V _{DD}	11, 32	P	-----	Lojik ve I/O pinleri için pozitif kaynak
NC	-----		-----	Bu pinler içsel olarak bağlantılı değildir. Bu pinler birbirine bağlanmalıdır.

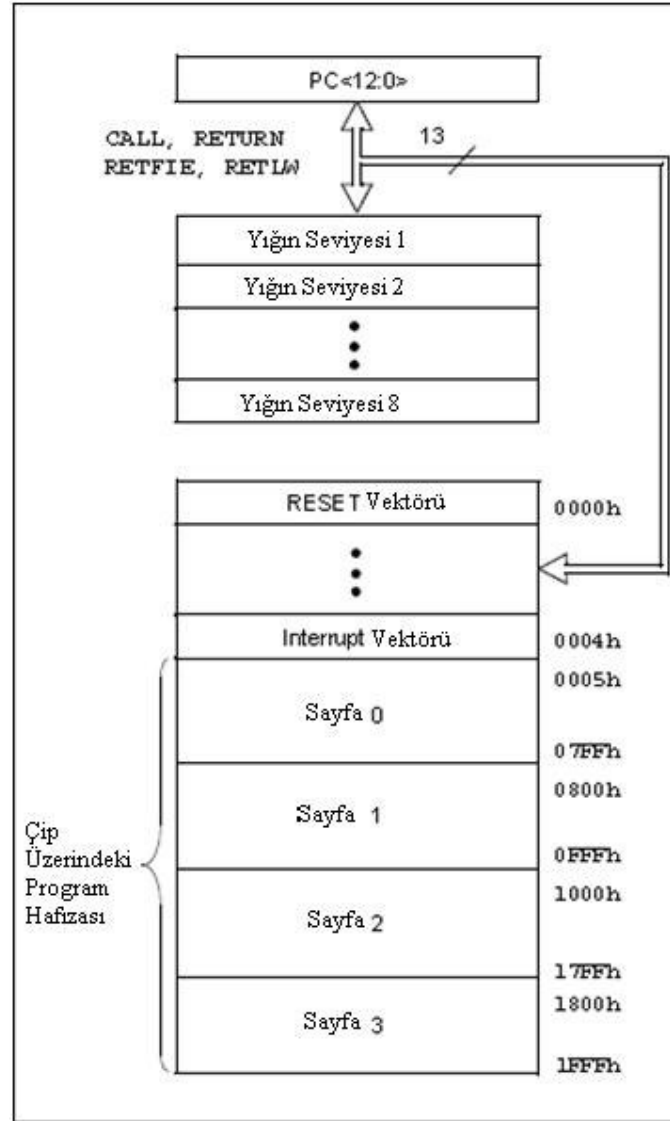
Legend: I = girdi, O = çıktı, I/O = girdi/çıktı, P = güç ----- = kullanılmayan TTL = TTL girdisi ST = Schmitt Trigger girdisi

- 1- Bu buffer, dış kesme olarak tanımlandığı zaman, Schmitt Trigger girdisidir.
- 2- Bu buffer, seri programlama modunda kullanıldığında, Schmitt Trigger girdisidir.
- 3- Bu buffer, genel amaçlı I/O olarak yaralandığında Schmitt Trigger girdisidir. Veya paralel slave modunda (mikroprosesör bus'ına karşı) kullanıldığında TTL girdisidir.
- 4- Bu buffer, RC osilatör modunda veya CMOS girdi modunda kullanıldığında Schmitt Trigger girdisidir. [10]

5.1.2 CMOS Geliştirilmiş FLASH/EEPROM Teknolojisi

Bu teknoloji düşük güç ve yüksek hız için geliştirilmiştir. Tamamen statik bir teknoloji olarak tasarlanmıştır. Bu teknolojiye geniş bir çalışma voltaj yelpazesi kullanılır (2,0V'dan 5,5V'ta). Ayrıca daha az güç harcanır (3V, 4 MHz için 0,6 mA'den daha az, 3V, 32 kHz için 20 μ A ve son olarak standby durumunda 0,5 μ A) [10; 12].

5.1.3 PIC 16F877'nin Hafıza Haritası



Şekil 5.2 PIC 16F877'nin Hafıza Haritası [10; 12]

Her PIC 16F87X MCU'da üç hafıza bloğu bulunmaktadır. Program hafızası ve data hafızasının ayrı "bus" 'ı vardır. Bu sebeple aynı anda adreslere ulaşım sağlanabilir. PIC 16F8X türü aygıtlarda 13-Bit program sayacı vardır. Bu sayaç 8K x14 program hafızası adresleme yeteneğine sahiptir. PIC 16F877 mikrokontrolerlerinde 8K x 14 word FLASH program hafızası üzerinde fiziksel adresleme tekrar başa dönmeye yol açar. Data hafızası 4 BANK'a ayrılmıştır. Her BANK 7Fh (128byte) kapasitesindedir. Her BANK'ın düşük numaralı bölgeleri özel

amaçlı yazmaçlar için ayrılmıştır. Bunların hemen üstünde genel amaçlı yazmaçlar bulunmaktadır. Bu bölge statik RAM'dır. [10; 12]

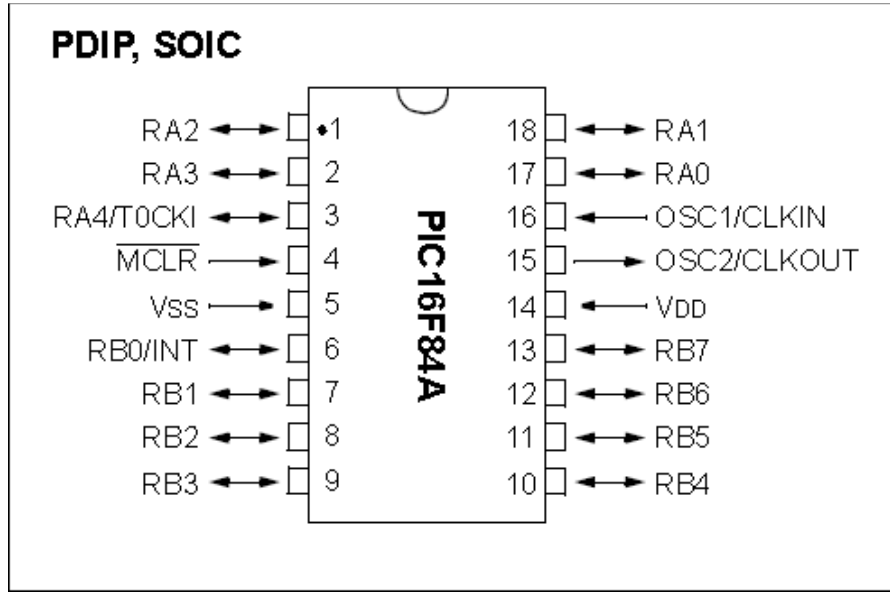
5.2 PIC 16F84 Donanımı

PIC 16F84'ler RISC CPU mimarisine göre tasarlanmışlardır. Toplam 35 assembler komutu ile programlanabilirler. Programda dallanma yapanlar dışında tüm komutlar tek çevrimdir. Programda dallanma yapanlar çift çevrimdir. Çalışma hızları; DC – 20 MHz saat girdisidir (DC – 200ns komut çevrimi). 1024 Word program hafızaları mevcuttur. 68 byte DATA RAM'ları bulunmaktadır. 64 byte DATA EEPROM'ları bulunmaktadır. Komutlar, 14 – bit Word genişliktedir . 8 – bit genişlikte data byte'ları mevcuttur. 15 özel fonksiyonlu donanım yazarı bulunmaktadır. 8 seviye derinliğinde donanımsal yığına sahiptirler. Direkt, indirekt ve relativ adresleme modlarına sahiptirler. Dört Adet kesme (interrupt) kaynağına sahiptirler: 1- Dışsal RB0/INT pini, 2- TMR0 timer taşması, 3- PORTB <7:4> değiştiğinde kesme özelliği (interrupt –on-change), 4- Data EEPROM yazımı bittiğinde. [13; 14]

5.2.1 Çevresel Özellikleri

PIC 16F84'nin 13 I/O pinlerinin yönleri tek tek kontrol edilebilir. Yüksek sağlayabilen veya alabilen (sink/source) LED sürücüsüne sahiptir (pin başına 25mA sink(alma) maksimum, 25mA verme (source) maksimum). 8-bit önbölücülü (prescaler) 8-bit timer/counter (zamanlayıcı/sayıcı) 'a (TMR0) sahiptir [13; 14].

PIC 16F84'nin, 10.000 sil/yaz çevrimli, Enhanced FLASH Program hafızası mevcuttur. Aynı zamanda 10.000.000 sil/yaz çevrimli, EEFLASH Data hafızası mevcuttur. EEPROM'da 40 yıl bilgi tutma kapasitesine sahiptir. İki pin ile in-Circuit Seri Programlama (ICSP) özelliği mevcuttur. Power-on Reset (POR), Power-up Timer (PWRT), Oscillator Start-up Timer (OST) özellikleri mevcuttur. Watchdog timer kendi osilatörü mikrokontrolere monte biçimde 16F84 yapısında bulunur. Kod koruması mevcuttur. Gücün daha verimli kullanılmasını sağlayan SLEEP (uyku) modu mevcuttur. Osilatör opsiyonları seçilebilen cinstendir. [13; 14]



Şekil 5.3 PIC 16F84 PDIP, SOIC Pin Yayılımı [14]

Çizelge 5.2’de PIC 16F84 PDIP paketinin pinleri gösterilmiştir.

Çizelge 5.2 PIC 16F84A’nın Pin Tablosu [10]

Pin İsmi	PDIP No.	I/O/P Tipi	Buffer Tipi	Tanım
OSC1/CLKIN	16	I	ST/CMOS ³	Osilatör kristali girdisi / Dışsal saat kaynağı girdisi
OSC2/CLKOUT	15	O	-----	Osilatör kristali çıkışı. Kristal osilatör modunda kristale veya resonatöre bağlar. RC modunda OSC2 pini CLKOUT çıkışı verir. Bu OSC1’in frekansının 1/4’ü kadardır. Ayrıca komut çevrim oranını gösterir.
$\overline{\text{MCLR}}$	4	I/P	ST	Master Clear (Reset) girdi/programlama voltaj girişi. Bu pin, aygıta, bir aktif düşük RESET’tir.
PORTA iki yöne de ayarlanabilen bir I/O port’udur.				
0RA0	17	I/O	TTL	
RA1	18	I/O	TTL	
RA2	1	I/O	TTL	
RA3	2	I/O	TTL	
RA4/T0CKI	3	I/O	ST	TMR0 timer/counter için saat girdisi olarak da ayarlanabilir. Çıkışı açık çıkış (drain) tipidir.

PORTB iki yöne de ayarlanabilen bir I/O port'udur. PORTB tüm girdilerde zayıf pull-up'lar için programlanabilir.				
RB0/INT	6	I/O	TTL/ST ¹	RB0/INT dış kesme pini olarak seçilebilir.
RB1	7	I/O	TTL	-----
RB2	8	I/O	TTL	
RB3	9	I/O	TTL	
RB4	10	I/O	TTL	Kesme olduğunda değişen pin.
RB5	11	I/O	TTL	Kesme olduğunda değişen pin.
RB6	12	I/O	TTL/ST ²	Kesme olduğunda değişen pin. Seri programlama saati
RB7	13	I/O	TTL/ST ²	Kesme olduğunda değişen pin. Seri programlama saati
V _{SS}	5	P	-----	Lojik ve I/O pinleri için topraklama
V _{DD}	14	P	-----	Lojik ve I/O pinleri için pozitif kaynak

Çizelge 5.2'nin *legend*'i:

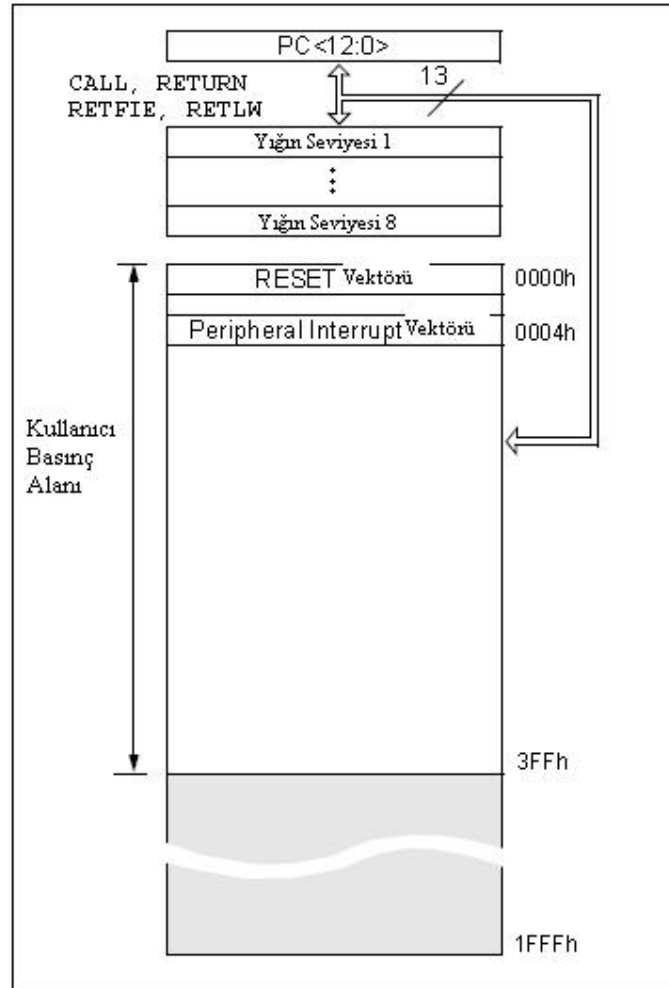
I = girdi, O = çıktı, I/O = girdi/çıkıtı, P = güç ----- = kullanılmayan TTL = TTL girdisi ST = Schmitt Trigger girdisi

- 1- Bu buffer, dış kesme olarak tanımlandığı zaman, Schmitt Trigger girdisidir.
- 2- Bu buffer, seri programlama modunda kullanıldığında, Schmitt Trigger girdisidir.
- 3- Bu buffer, RC osilatör modunda veya CMOS girdi modunda kullanıldığında Schmitt Trigger girdisidir [14].

5.2.2 CMOS Geliştirilmiş FLASH/EEPROM Teknolojisi

Bu teknoloji düşük güç ve yüksek hız için geliştirilmiştir. Tamamen statik bir teknoloji olarak tasarlanmıştır. Bu teknolojiye geniş bir çalışma voltaj yelpazesi kullanılır: 2,0V'dan 5,5V'ta. Ayrıca daha az güç harcanır (5V, 4 MHz için 2 mA'den daha az, 2V, 32 kHz için 15 μ A ve son olarak standby durumu için (2V) 0.5 μ A) [13; 14].

PIC 16F84'nın Hafıza Haritası



Şekil 5.4 PIC 16F84A'nın Hafıza Haritası [10]

PIC 16F84'ün iki adet hafıza bloğu bulunmaktadır. Bunlardan biri data biriside program bloğudur. Her bloğun ayrı "bus" 'ı vardır. Bu yüzden aynı osilatör çevrimi içinde ikisine de ulaşılabilir. Hafıza aynı zamandan özel amaçlı yazmaçlar (SFR: Special Function Registers) ve genel amaçlı RAM olarak da ikiye ayrılabilir. Data hafıza alanı yanı zamanda EEPROM'u da içerir. Bu alan hafıza alanında direkt olarak değil de indirekt olarak haritalanmıştır. 64 byte olan EEPROM'un adresi 0h-3Fh arasındadır. [13; 14]

Bu çalışmada kullanılan PIC Mikrokontrolerler arasındaki farklar Çizelge 5.3’de gösterildiği gibidir.

Çizelge 5.3 PIC16F877 ile PIC 16F84 Arasındaki Farklar [16]

Özellikle	PIC 16F877	PIC 16F84
Çalışma Hızı	DC – 20 MHz	DC – 20 MHz
Program Belleği	8 KWord	8 KWord
EEPROM Veri Belleği	256 Byte	256 Byte
Kullanıcı RAM	368 Byte	368 Byte
Giriş/Çıkış Port Sayısı	33	33
Zamanlayıcı	Timer0, Timer1, Timer2	Timer0
A/D Çevirici	8 Kanal, 10 Bit	Yok
Capture/Compare/PWM	16 Bit Capture, 16 bit Compare, 10 Bit PWM Çözünürlük	Yok
Seri Çevresel Arayüz	SPI (Master) ve I ² C (Master-Slave) Modunda SPI Portu ve Senkron Seri Port	Yok
Paralel Uydu Port	8 Bit, Harici, RD, WR ve CS Kontrolü	Yok
USART/SCI	9 Bit Adresli	Yok

5.3 PIC 16F84’de ve PIC 16F877’de Konfigürasyon Kelimesinin Bit Açılımları

PIC mikrodnetleyicilerini programlamaya başlamadan önce “__CONFIG” ifadesi ile (MPASM derleyicisinde) yapılan ayarlarla ilgili bitlerin açıklamaları bu bölümde incelenmiştir.

PIC 16F84’de konfigürasyon sözcüğünün bit açılımı:

13	12	11	10	9	8	7	6	5	4	3	2	1	0
CP	CP	CP	CP	CP	CP	CP	CP	CP	CP	PWRTE	WDTE	FOSCI	FOSC0

Bit 13 – 4: CP: Kod koruma biti

1 = Kod koruması var.

0 = Kod koruması yok.

Bit 3: $\overline{\text{PWRTE}}$: Power-up zamanlayıcı biti

1 = Power-up zamanlayıcı aktif

0 = Power-up zamanlayıcı pasif

Bit 2: $\overline{\text{WDTE}}$: Watch Dog Timer biti

1 = Watch Dog Timer aktif

0 = Watch Dog Timer pasif

Bit 1 – 0: FOSC1, FOSC0: Osilatör seçme biti

11 = RC osilatör seçildi.

10 = HS osilatör seçildi.

01 = XT osilatör seçildi.

00 = LP osilatör seçildi.

[13; 16]

PIC 16F877’de konfigürasyon sözcüğünün bit açılımı:

13	12	11	10	9	8	7	6	5	4	3	2	1	0
CP1	CP0	DEBUG		WRT	CPD	LVP	BODEN	CP1	CP0	$\overline{\text{PWRTE}}$	$\overline{\text{WDTE}}$	FOSC1	FOSC0

Bit 13-12 ve 5-4: CP1, CP0: FLASH Program belleği kod koruma biti

11 = Kod koruması yok

10 = 1F00h – 1FFFh arası kod korumalı bölge

01 = 1D00h – 1FFFh arası kod korumalı bölge

00 = 000h – 1FFFh arası kod korumalı bölge

Bit 11: DEBUG: Devre içi hata arama modu

1 = Devre içi hata arama pasif

0 = Devre içi hata arama aktif

Bit 10: Alan boştur. '1' olarak okunur.

Bit 9: WRT: FLASH program belleğine yazma biti

1 = Kod korumasız program belleğine EECON denetimi ile yazılabilir.

0 = Kod korumasız program belleğine EECON denetimi ile yazılamaz.

Bit 8: CPD: Veri EE Belleği kod koruma biti

1 = Kod koruması yok.

0 = Kod koruması var.

Bit 7: LVP: Düşük gerilim devre içi seri programlama biti

1 = RB3/PGM pini PGM işlevlidir. Düşük gerilim ile programlanabilir.

0 = RB3 sayısal pini I/O tanımlı, $\overline{\text{MCRL}}$ 'ye programlam için yüksek gerilim uygulanmalıdır.

Bit 6: BODEN: Gerilim alt ve üst limitleri aşarsa, programı yeniden başlatabilen (Brown out Reset Enable) biti

1 = BOR yeniden başlatma aktif (ON)

0 = BOR yeniden başlatma pasif (OFF)

Bit 3: $\overline{\text{PWRTE}}$: Power-up zamanlayıcı biti

1 = Power-up zamanlayıcı aktif

0 = Power-up zamanlayıcı pasif

Bit 2: $\overline{\text{WDTE}}$: Watch Dog Timer biti

1 = Watch Dog Timer aktif

0 = Watch Dog Timer pasif

Bit 1 – 0: FOSC1, FOSC0: Osilatör seçme biti

11 = RC osilatör seçildi.

10 = HS osilatör seçildi.

01 = XT osilatör seçildi.

00 = LP osilatör seçildi.

[13; 16]

5.4 Motor Stem İlişkisi

Vana otomasyonlarında DC vana ve kompleks bir dişli sistemi veya vana stemine direkt bağlı bir step veya servo motor sıkça görülebilir. Dişli sistemine bağlı DC motorlar genellikle steme direct lineer hareket verecek şekilde tasarlanmışlardır. Steme direct olarak bağlı sistemlerde ise motor stemi rotasyonel hareket ile aşağıya iletirmektedir (Aynı manuel sistemlerde bir *gate* veya glob vanayı simidini çevirerek kapatmak gibi). [15; 17]

Motorun hareket verdiği vida sistemi, yani vana stemi hesaplamaları için Denklem 5.1, 5.2, 5.3 ve 5.4'den oluşan sistem kullanılabilir.

Hız:

$$W_M = V_L P \quad (5.1)$$

Moment:

$$J_T = Mr^2 \left(\frac{1}{2\pi p} \right)^2 \frac{1}{z} + J_{LS} + J_M \quad (5.2)$$

Yük Torku:

$$T_L = \frac{1}{2\pi} \frac{F_L}{pz} + \frac{0,2}{2\pi} \frac{F_{PF}}{p} \quad (5.3)$$

Sürtünme Torku:

$$T_F = \frac{1}{2\pi} \frac{\mu M \cos A + M \sin \theta}{pz} \quad (5.4)$$

Burada:

W_M :	Motor hızı
V_L :	Lineer yük hızı
p :	Vida adım aralığı.
z :	Vida verimi
T_L :	Motor şaftındaki yük torku
T_F :	Sürtünme torku
F_L :	Yük kuvveti
F_{PF} :	Önceden yüklenen kuvvet
J_T :	Toplam atalet kuvveti
J_M :	Motor atalet momenti
J_{LS} :	Vida atalet momenti
F_F :	Sürtünme kuvveti
μ :	sürtünme katsayısı

[15; 17]

Bu denklemler bütünü glob, piston, *gate* ve *pinch* vanalar için kullanılabilir fakat küresel, kelebek ve plug vanalar için uygun değildir. [15; 17]

5.5 DC Motorlar

DC motorlar 4 guruba ayrılır [15]:

- 1- Mıknatıs kullanan motorlar
- 2- Seri sarımlı motorlar
- 3- Şönt sarımlı motorlar
- 4- Karışık sarımlı motorlar
- 5- Step motorları

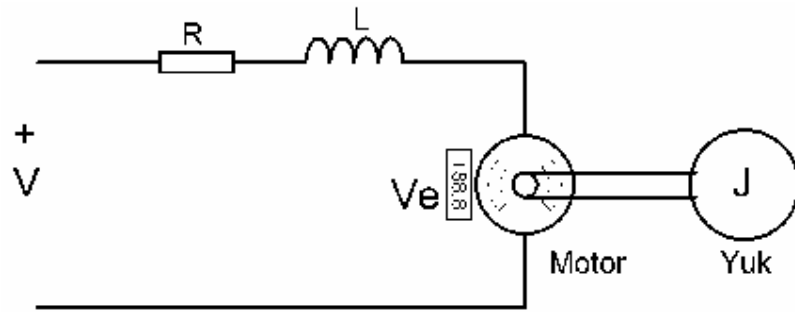
Seri sarımlı DC motorların hızları motora takılan yüke bağlı olarak değişmekte ve yük azaldıkça hız artmaktadır. Bu tür motorlarda manyetik alan motora seri olarak bağlanmış olan bir bobine akım vererek mümkün olur. Seri sarımlı motorlar daha çok başlangıçta büyük güçleri belirli bir hıza getirmek amacıyla

kullanılmaktadır. Bu tür motorlar mıknatıslı motorlardan daha güçlüdürler ve daha fazla akım çekerler. [15]

Şönt sarımlı DC motorlarda gerekli olan manyetik alan motor ile paralel bağlanmış olan bir bobine akım verilerek elde edilir. Bu tip motorlar daha çok hassas hız kontrolü uygulamalarında ve aynı zamanda yüksek tork gerektiren uygulamalarda kullanılmaktadır [15].

Karışık sarımlı DC motorlar seri sarımlı ve şönt sarımlı motorların karışımı olarak düşünülebilir. Bu motorlardaki manyetik alan hem seri bağlı bobine hemde paralel bağlı bobine akım verilerek sağlanabilir. Bu tür motorlar hassas hız gerektiren veya yüksek tork gerektiren yerlerde kullanılabilirler [15; 17].

Mıknatıs kullanan motorlara örnek olarak bilgisayarlarda soğutma işlemi için kullanılan fan motorlarını verebiliriz. Bu tip motorlarda manyetik alan bir mıknatıs ile elde edilir. Bu tür motorların dezavantajlarından biri de mıknatısın zamanla etkisini kaybetmesidir. Bu tür motorların gerilim-hız eğrileri oldukça lineerdir. Mıknatıs kullanan DC motorların başka bir özelliği de gerilim değişince hızın çok hızlı bir biçimde değişmesidir. [17]



Şekil 5.5 Mıknatıslı DC Motor [15]

Burada:

- V: Gerilim (V)
- R: Motor bobin direnci(Ω)
- L: Motor bobin indüktansı (H)
- J: Motora takılan yükün inertşiası ($\text{kg.m}^2/\text{s}^2$)

V_e : Motor dönüşünden meydana gelen geri emf (V)

θ : Motor şaftının dönüşü (rad)

w : Motor şaftının hızı (rad/s)

T : Motor torku (Nm)

Motorun üretmiş olduğu tork motora verilen akım ile doğru orantılıdır [15; 17].

$$T = K_t i \quad (5.5)$$

Burada K_t motor tork sabiti olarak bilinmektedir. Motorun dönüşünden dolayı elde edilen geri emf gerilimi motorun hızına bağlıdır [15; 17].

$$V_e = K_E w \quad (5.6)$$

Burada K_E geri emf sabitidir. Kirchoff kanunu kullanarak motor devresi için Denklem 5.7 yazılabilir [15; 17].

$$V - V_e = R_i + L di/dt \quad (5.7)$$

Veya

$$V = R_i + L di/dt + K_E w \quad (5.8)$$

Veya

$$T = J dw/dt \quad (5.9)$$

Denklem 5.5'den ve Denklem 5.9'dan dolayı Denklem 5.10'daki sonuca ulaşılır [15; 17].

$$i = j / K_t dw/dt \quad (5.10)$$

Denklem 5.8'den ve Denklem 5.10'dan Denklem 5.11 elde edilir [15; 17].

$$V = \frac{1}{s} \left(\frac{1}{s} \right) \quad (5.11)$$

Küçük motorlarda indüktans oldukça küçüktür. Denklem 5.10'dan indüktans çıkarılırsa Denklem 5.12'deki diferansiyel denklem elde edilir [15; 17].

$$V = \frac{1}{s} \left(\frac{1}{s} \right) \quad (5.12)$$

Laplas transformu kullanılarak Denklem 5.11, Denklem 5.13'deki gibi ifade edilebilir [15; 17]

$$\frac{1}{s} \left(\frac{1}{s} \right) \quad (5.13)$$

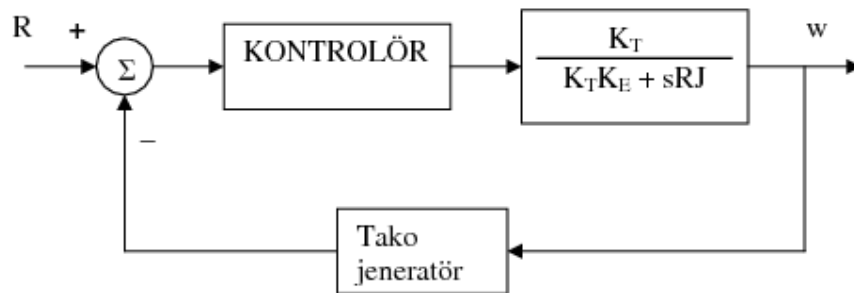
Veya,

$$\frac{1}{s} \left(\frac{1}{s} \right) \quad (5.14)$$

Motorun transfer fonksiyonu Denklem 5.15'deki ifade edilebilir [15; 17].

$$\frac{1}{s} \left(\frac{1}{s} \right) \quad (5.15)$$

DC motor hız ve pozisyon kontrolünde geri besleme kullanılmaktadır. Şekil 5.6'de kontrol geri besleme şeması verilmiştir [15; 17].



Şekil 5.6 DC Motor geri besleme kontrol blok şeması

Burada motor hızı takojeneratör veya optik enkoder kullanılarak ölçülür. Arzu edilen motor hızı (R) ile tespit edilen motor hızı (w) arasındaki fark alınır ve buna göre motor hızı istenildiği gibi kontrol edilir [15; 17].

Bazı motor uygulamarda şaftın pozisyonu kontrol edilmek istenmektedir. Pozisyon ' θ ' ile ifade edilirse ' w ' ve ' θ ' arasındaki bağıntı Denklem 5.16'deki gibidir [15; 17].

$$w = d\theta/dt \quad (5.16)$$

Bu sayede motora verilen gerilim ve motor şaftının pozisyonu arasında Denklem 5.17'deki diferansiyel denklem yazılabilir [15; 17].

$$V = \frac{RJ}{K_T} \frac{d^2\theta}{dt^2} + K_E \frac{d\theta}{dt} \quad (5.17)$$

Gerilim ile pozisyon arasındaki bağıntı Denklem 5.18'deki transfer fonksiyonu ile gösterilebilir [15; 17].

$$V(s) \longrightarrow \boxed{\frac{K_T}{RJs^2 + K_T K_E s}} \longrightarrow \theta(s) \quad (5.18)$$

5.6 Step Motor

Step (Adım) motorlar çok hassas hız ve konumlama gerektiren yerlerde kullanılır. Bu motorlar düşük hızlarda yüksek torklar üretebilirler. Step motorlar, motora gönderilen pulslar ile çalıştırılır. Her puls gönderildiğinde motor belirli bir miktar dönmektedir. Step motorlarda geri besleme yapılmamaktadır. Bu sebepten dolayı motor hızını ölçmek için takojeneratör veya optik enkodere gerek olmamaktadır. Kontrol alanında step motorların çok önemli bir yeri vardır. Bir step motorunun hassasiyeti adımın %3'ü ile %5'i arasındadır. Adım motorların avantajları aşağıdaki gibidir [12; 15; 17]:

- Geri besleme olmadan control devrelerinde rahatlıkla kullanılabilirler.
- Mevcut hata bir adımdan diğer bir adıma geçmez.
- Pulsar ile çalıştığı için mikrokontroler devrelerinde kolaylıkla kullanılabilirler.
- Adım motorların yapıları basit ve oldukça güvenilirdir.
- Düşük hızlarda yüksek tork vermektedirler.
- Çok düşük hızlarda (örnek olarak dakikada birkaç yüz defa döndürülecek uygulamalarda) kullanılabilirler.
- Adım kontrolü kullanılarak hassasiyetler artırılabilir.

Dezavantajları ise şöyledir:

- Motorun dönüşü adım adım olduğu için bazı uygulamalarda kullanılamaz.
- Sürtünmeden dolayı pozisyon kontrolünde hatalar oluşabilir.
- Çıkış güçleri sınırlıdır.
- Geri beslemeli hız ve pozisyon kontrolü uygulamalarında kullanılmaları zordur.
- Adım motoların maliyeti fazladır.

Adım motorların bir çok çeşidi mevcuttur. En çok kullanılan çeşidi ise sabit mıknatıs içerenidir. Adım motorlar ayrıca gövdesinin büyüklüğü ile de sınıflandırılır. Örnek olarak 'size 11' bir adım motorun gövde çapı 1.1 inç, 'size 45'in ise gövde çapı 4.5 inçtir [12; 15; 17].

Genellikle adım motorların iki fazı vardır, fakat üç veya beş fazlı motor bulmak mümkündür. Step motorlar unipolar ve bipolar olarak ikiye ayrılırlar. Unipolar Adım motorların 4, 5 veya 6 bacak bağlantıları bulunmaktadır. Ortak bacak genellikle '+V' gerilimine bağlanmaktadır. Bu durumda 6 bacaklı unipolar bir adım motoru kontrol etmek için sadece 4 bacağın kontrol edilmesi yeterlidir. Bipolar motorların ise sadece 4 bacak bağlantıları bulunmaktadır. Genel olarak unipolar motorları sürmek bipolar motorları sürmekten daha kolaydır [12; 15; 17].

Unipolar adım motorlarını tam adımlı veya yarım adımlı olarak sürebiliriz. En yaygın kombinasyonlar: Bir fazlı tam adım, 2 fazlı tam adım, 2 fazlı yarım adım olmak üzere üç değişik şekildedir. 5.4, 5.5 ve 5.6 numaralı çizelgelerde bu kombinasyonların tabloları mevcuttur. Çizelgelerdeki adım sayıları bir tam dönmeyi tamamlayacak kadar verilmiştir [12; 15; 17].

Çizelge 5.4 Bir fazlı tam adım [17]

Adım	A	B	C	D
1	1	0	0	0
2	0	1	0	0
3	0	0	1	0
4	0	0	0	1

Çizelge 5.5 İki fazlı tam adım [17]

Adım	A	B	c	D
1	1	0	0	1
2	1	1	0	0
3	0	1	1	0
4	0	0	1	1

Çizelge 5.6 İki fazlı yarım adım [17]

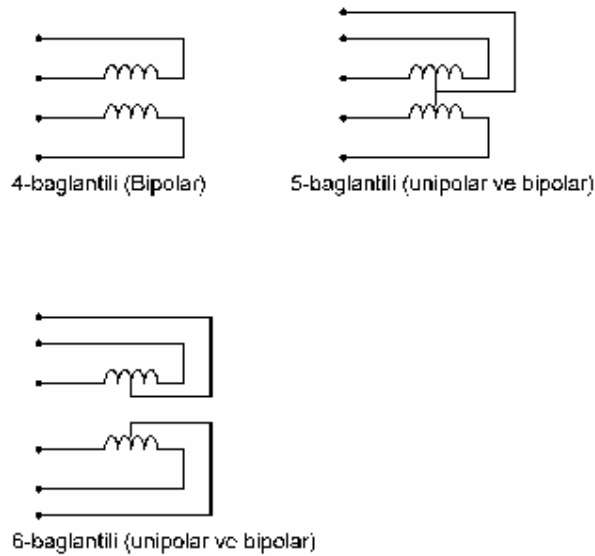
Adım	A	B	c	D
1	1	0	0	0
2	1	1	0	0
3	0	1	0	0
4	0	1	1	0
5	0	0	1	0
6	0	0	1	1
7	0	0	0	1
8	1	0	0	1

Bi-polar motorlarda ayrı bobinler olup bu bobinlere verilen gerilim yönünün değiştirilmesi gerekmektedir. Bu sebepten bu tür motorların sürüşü daha karmaşıktır. Bu tip motorları sürmek için genellikle çift H-Köprüsü içeren devreler kullanılmaktadır [12; 15; 17].

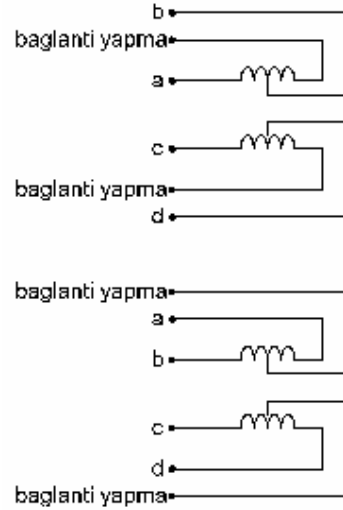
Çizelge 5.7 Bi-polar adım Motoru [17]

Adım	A	B	c	D
1	+	-	-	-
2	-	+	-	-
3	-	-	+	
4	-	-	-	+

Çizelge 5.7’de ‘+’ ve ‘-’ işaretler gerilimin yönünü göstermektedir. Bi-polar sürücü devresi ile unipolar motorlarda sürülebilir. Bu iki şekilde yapılabilir. Birinci yöntem olarak unipolar motorun orta bobinlerini kullanmamayı verebiliriz. Bu şekilde motorun sadece 4 bacağı kullanılmış olur. İkinci bir yöntemde motor bacaklarından orta bacaklar ve birer bacak kullanılır. Bu yöntem ile de toplam dört bacak kullanılır. İkinci yöntem ile daha yüksek hız elde edilebilir [12; 15; 17].



Şekil 5.7 Adım Motor Bağlantıları [17]



Şekil 5.8 Bi-polar sürücü ile unipolar motoru sürmek için yapılan bağlantılar [17]

Adım motorların hızları adım komutları arasında verilen zamana bağlıdır. Adım komutları arasındaki zaman ‘T’ olur ise ve adım sabiti β derece ise motor 1 saniyede β/T kadar adım döner. Motor bir tam devirde doğal olarak 360° kat ettiği için bir saniyedeki devir sayısını $\beta/360T$ olarak ifade edebiliriz. Bu durumda RPM’i Denklem 5.19’daki gibi ifade etmemiz gerekir [12; 15; 17].

$$\text{RPM} = 60\beta/360T \quad (5.19)$$

Burada:

RPM: Motorun bir dakikadaki devir sayısı (dakika^{-1})

β : Motorun adım sabiti (derece)

T: Adımlar arasındaki zaman (saniye)

Genellikle motor şaftının hangi pozisyonda olduğunu bulmak için doğal olarak Denklem 5.20’deki gibi kullanılır [12; 15; 17]:

$$n = v/\beta \quad (5.20)$$

Burada:

n: Motora uygulamamız gereken adım sayısı

v: Motorun dönmesi istenen derece

Motor şaftının kat ettiği mesafe içinde doğal olarak Denklem 5.21'deki denklemi kullanabiliriz [12; 15; 17]:

$$n = 180d / \pi r \beta \quad (5.21)$$

Burada:

d: Motor şaftının aldığı yol

r: Motor şaftının yarı çapı

Bazı adım motoru uygulamalarında motorun belirli bir zaman için dönmesini isteyebiliriz. Bunun için de motora uygulamamız gereken komut adım sayısını hesaplamak için Denklem 5.22'deki denklemi kullanabiliriz [12; 15; 17].

$$T_D = n \times T \quad (5.22)$$

Burada:

T: Motorun toplam dönme zamanı

Motor kontrol devrelerinde motor akımını sınırlamak için genellikle motor sarımları ile seri olarak bir direnç kullanılır. Bu direncin değeri motora verilen gerilime ve motorun maksimum güç harcama kapasitesine bağlı olarak seçilmelidir. Motor devresi için Denklem 5.23'deki denklem yazılabilir [12; 15; 17].

$$V = V_T + iR_{\text{motor}} + iR \quad (5.23)$$

Burada:

V_T : Transistördeki gerilim düşüşü (unipolar devrelerde 1V, bi-polar devrelerde 2V olarak alınabilir.)

i: Motor devresinden geçen akım

R_{motor} : Motor sarımının direnci

R: Motor devresi ile seri olarak kullanılacak olan dirençtir.

Adım motorları ve DC motorları şu şekilde karşılaştırabiliriz [17]:

- 1- Adım motorlar kolaylıkla mikrokontroler ile kontrol edilebilirler. DC motorların ise kontrolü daha komplektir.
 - 2- Adım motorlar geri beslemesiz olarak kullanılabilir. DC motorlar ise geri beslemeli olarak çalıştırılırlar.
 - 3- Adım motorlar fırçasız oldukları için DC motorlardan daha dayanıklıdır.
 - 4- Adım motorlarında kayma meydana gelebilir ve bundan dolayı hız ve pozisyon kontrolünde hata olabilir.
- DC motorlar geri besleme ile kontrol edildiğinden dolayı genellikle daha hızlı bir responsa sahiptirler.

5.7 Servo Motor

D.C servo motorları, genel olarak bir D.C. motoru olup, motora gerekli D.C. aşağıdaki metotlardan elde edilir [11; 12; 18].

- 1- Bir elektrik yükselteçten.
- 2- A.C. akımın doyumlu reaktörden geçirilmesinden.
- 3- A.C. akımın tristörden geçirilmesinden.
- 4- Amplidin, retotrol, regüleks gibi dönel yükselteçlerden elde edilir.

D.C. servo motorlar çok küçük güçlerden çok büyük güçlere kadar imal edilirler (0,05 Hp den 1000 Hp ye kadar). Bu motorlar klasik D.C. motorlar gibi imal edilirler. Bu motorlar küçük yapılıdır ve endüvileri (yükseklik, uzunluk / Çap oranıyla) kutup atalet momentini minimum yapacak şekilde tasarlanırlar. Küçük çaplı ve genellikle içerisinde kompanzasyon sargısı olan, kuvvetli manyetik alanı boyu uzun doğru akım motorlarına da servo motor denir [11; 12; 18].

D.C. servo motor çalışma prensibi açısından aslında, Statoru Daimi Mıknatıs bir D.C. motordur. Manyetik alan ile içinden akım geçirilen iletkenler arasındaki etkileşim nedeniyle bir döndürme momenti meydana gelir. Bu döndürme momenti manyetik alan vektörü ile sargı akım vektörü arasındaki açı 90° olduğunda

maksimum değeri alır. Bir D.C. servomotorda fırçaların konumları, her iki dönüş yönü için de döndürme momenti açısının 90° olmasını sağlayacak şekilde belirlenmiştir. Kolektör segmentlerinin fazla olması neticesinde momentin sıfır bir noktada rotorun hareketsiz kalması engellenmiş olur [11; 12; 18].

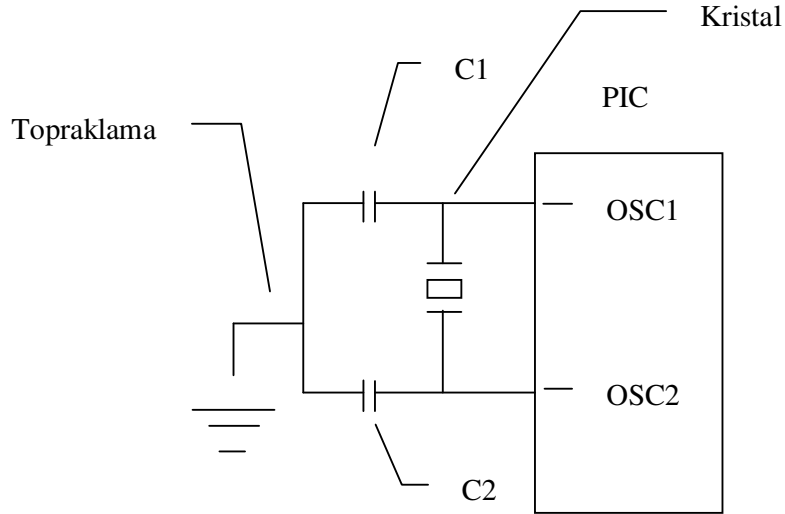
Sanayide kullanılan çeşitli doğru akım motorları vardır. Servo sistemlerde kullanılan doğru akım motorlarına ise D.C. servomotorlar adı verilir. D.C. servomotorlarda rotor eylemsizlik momenti çok küçüktür. Bu sebepten piyasada çıkış momentinin eylemsizlik momentine oranı çok büyük olan motorlar bulunur [11; 12; 18].

Bazı D.C. Servomotorların çok küçük zaman sabitleri vardır. Düşük güçlü D.C. servo motorlar piyasada genellikle bilgisayar kontrollü cihazlarda (disket sürücüler, teyp sürücüler, yazıcılar, kelime işlemciler, tarayıcılar vs.) kullanılırlar. Orta ve büyük güçlü servomotorlar ise sanayide genellikle robot sistemleri ile sayısal denetimli hassas diş açma tezgâhlarında kullanılır. D.C. motorlarda alan sargıları rotor sargılarına seri veya paralel bağlanır [11; 12; 18].

D.C. servomotorlar genellikle “elektronik hareketli denetleyiciler ” adı verilen servo sürücüler ile kontrol edilirler. Servo sürücüler servomotorun hareketini kontrol ederler. Kontrol edilen büyüklükler çoğu zaman noktadan noktaya konum kontrolü, hız kontrolü ve ivme programlamasıdır. PWM tekniği adı verilen darbe genişlik modülasyonu genellikle robot kontrol sistemlerinde, sayısal kontrol sistemlerinde ve diğer konum denetleyicilerinde kullanılırlar [11; 12; 18].

5.8 Kristal

Bir Osilatör çeşididir. Kristal mikrokontrolörün OSC1 ve OSC2 bacaklarına bağlanır. Ayrıca iki adet kapasitör kullanılır. Kapasitör seçimi kullanılan kristalin frekansına bağlıdır. Şekil 5.14’de osilatörün bağlanma şekli verilmiştir. Çizelge 5.8’de ise kristalin kullanımına göre kapasitör seçimi verilmiştir [12; 13; 19; 20].



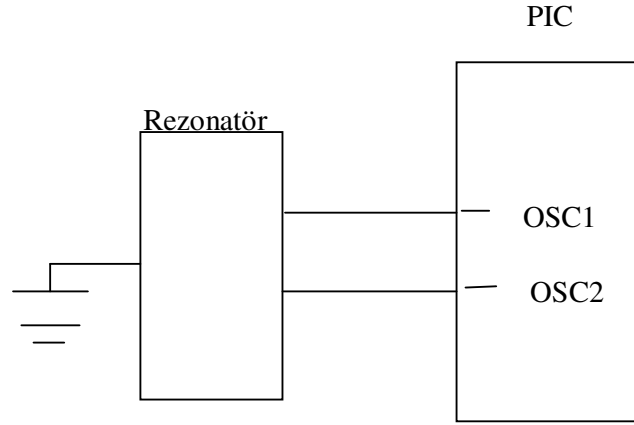
Şekil 5.9 Kristal Osilatör Devresi [12; 13]

Çizelge 5.8 Kristal Kullanımı İçin Kapasitör Seçimi [12; 13]

Osilatör Modu	Frekans	C1, C2
LP	32kHz	68 – 100 pF
LP	200kHz	15 – 33 pF
XT	100kHz	100 -150 pF
XT	2MHz	15 – 33 pF
XT	4MHz	15 – 33 pF
HS	4MHz	15 – 33 pF
HS	10MHz	15 – 33 pF

5.9 Rezonatör

Bir Osilatör çeşididir. Rezonatörler 4Mhz ile 10 MHz arası frekanslarda çalışırlar. Bir rezonatör osilatörü bir kristal osilatörü kadar hasas değildir. Şekil 5.15’de rezonatörün bağlanma şekli verilmiştir. [12; 13; 19; 20]



Şekil 5.10 Rezonatör Devresi [12]

5.10 Direnç

Dirençler yapımında kullanılan malzemeye göre ikiye ayrılır. Karbon dirençler ve Telli dirençler. Yine bu iki tip kendi arasında ikiye ayrılır. Bunlar sabit dirençler ve ayarlı dirençlerdir. Bunun yanında bazı özel dirençler de vardır (Fotodirenç, Termistör) [19; 20].

Bir direncin değerinin ne olduğu üzerine doğrudan yazılabileceği gibi en çok kullanılan yöntem olan renk kodları ile üzerine kodlanmasıdır. Genellikle dirençler üzerinde 4 veya 5 adet renk bandı bulunur. Bu renkler direnç üzerine kodlanırken renkler direncin bir tarafına daha yakın olarak yerleştirilir. Değer okuması yaparkende renk bandının kenara en yakın olanından başlamak gerekir. Okunan renkler direnç değer çizelgeside karşılaştırılarak direncin gerçek değeri tespit edilir.

[19; 20]

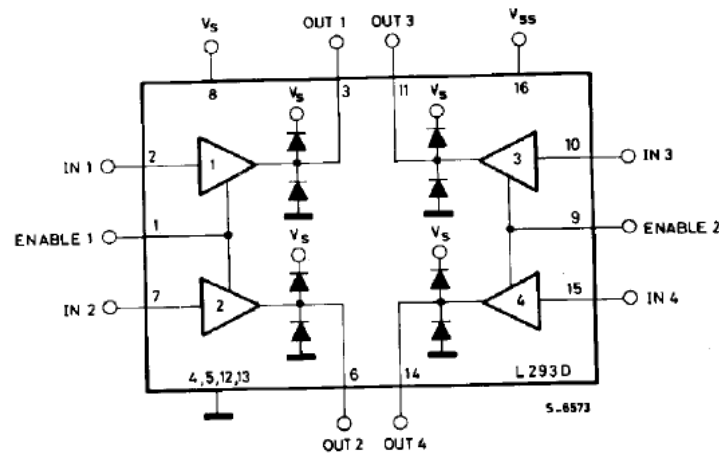
5.11 Led

Elektriksel olarak Led 'ler, bilinen normal diyotlar gibi davranırlar. Bu gerilim sabit olmayıp Led çeşidine göre değişmektedir. Kırmızı Led 'ler için 1.6 Volt olan iletim gerilimi, yeşil Led 'lerde 2,4 Volta kadar çıkmaktadır. Bu gerilim silisyum diyotlarda 0,65 - 0,70 Volt arasındadır [19; 20].

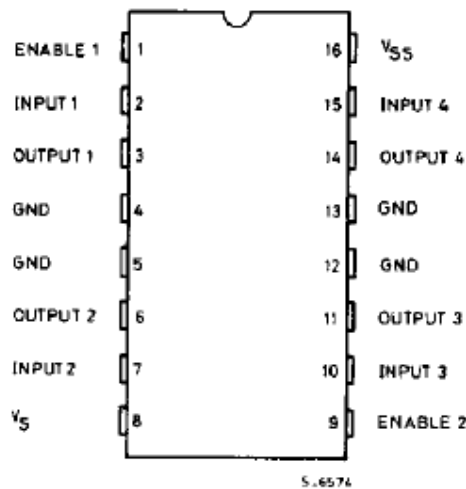
5.12 H-Köprüsü

H-Köprüsü entegre devresi belirli amperlere kadar çalışan motorları kontrol etmek için kullanılırlar. Bu çalışmada iki çeşit H Köprüsü kullanılmıştır. Bir tanesi L293D ötekisi ise LMD18200'dir [15; 17; 21; 22; 23].

Push-pull dört kanallı diyotlu driver olan L293D, her kanal başına 600mA çıktı akımı kapasitesindedir. Her kanal başına 1,2A pik yapan çıktı akımı mevcuttur. Aşırı ısıya karşı koruması mevcuttur. İçsel diyotları mevcuttur. Lojik "0" girdi voltajı 1,5V'a kadar çıkabilir. [15; 17; 21]

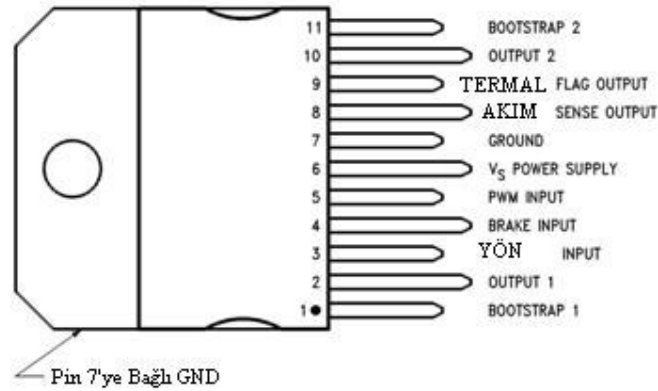


Şekil 5.11 L293D Bağlantı Şeması [21]

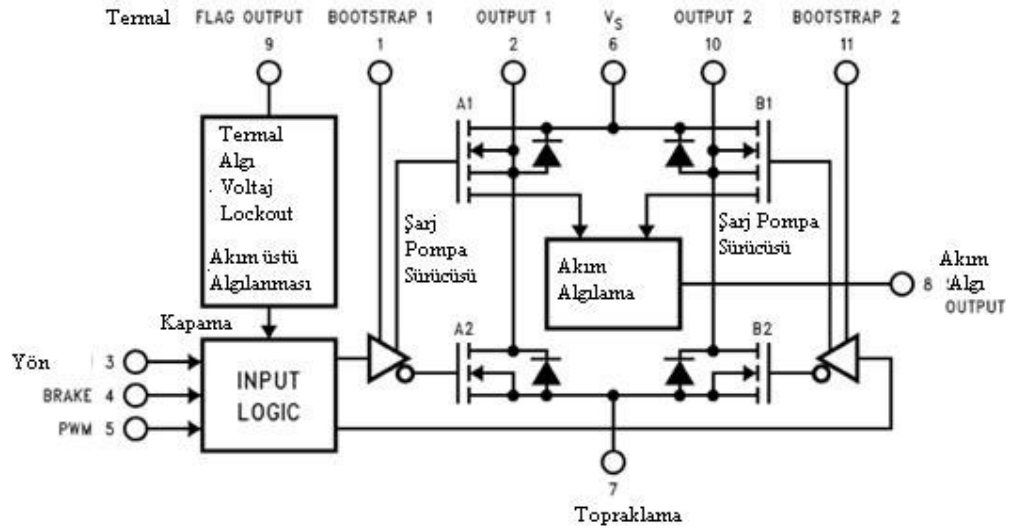


Şekil 5.12 L293D Pinleri [21]

LMD 18200, H-Köprüsünün iki çeşidini barındırmaktadır. LMD18200T ve LMD18200-2D. LMD18200T 11 bacaklıdır ve tek motoru kontrol etmek için kullanılır. Bu entegre devre TO-220 paketi içindedir. LMD18200-2D ise DIL paketi içerisinde. LMD 18200-2D 24 bacaklıdır ve iki ayrı, birbirinde ayrı motoru aynı anda kontrol etmek için kullanılır. Bu çalışmada LMD 18200T kullanılmıştır. [15; 17; 22]

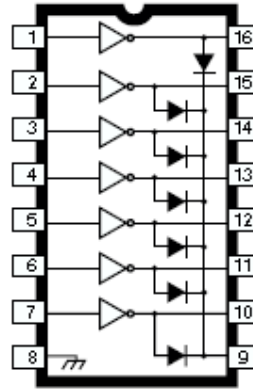


Şekil 5.13 LMD18200T Pinleri [22]



Şekil 5.14 LMD18200 Bağlantı Şeması [22]

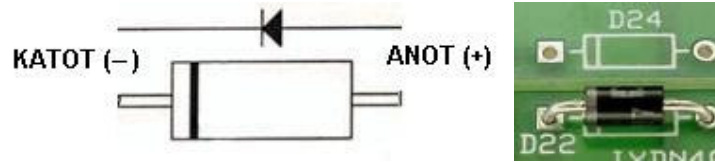
Yüksek voltaj, yüksek akım Darlington Transistör Array'dır. Bu çalışmada ULN2003'ün A serisi kullanılmıştır fakat aynı zamanda bu transistör çeşidinin L serisinde mevcuttur. [15; 17; 23]



Şekil 5.15 ULN2003 Serisi Bağlantı Şeması [23]

5.13 Diyot

Akımı bir yönde ileten, diğer yönde durduran devre elemanıdır. Devrelerde genel olarak doğrultma amaçlı kullanılırlar. Gösterimi ve simgesi Şekil 5.16'deki gibidir [19].



Şekil 5.16 Diyotun Şematik Gösterimi ve Resmi [15]

Diyolar lamba anahtarları gibi düşünülebilir. Anahtar kapatıldığında devre tamamlanır ve lamba yanar, ve anahtar açıldığında devre açılır ve akım geçmez dolayısıyla lamba söner. Bu durum Şekil 5.17'deki gibi gösterilebilir [19; 20].



Şekil 5.17 Diyotun Anahtar Şelinde Gösterimi [19; 20]

Diyotlar P ve N tipi iki yan iletken maddenin ekleminden oluşmuştur. Diyot'un, harici bir voltaj kaynağına bağlanarak kutuplandırılmasına, polarma denir. Anot voltajının katot'a göre daha büyük olması, doğru polarlama anot'un, katot'dan daha düşük olması durumu ise ters polarlama olarak ifade edilir. Bir diyot, normal olarak, doğru plorma durumunda 'iletim' 'de, ters polarmada ise 'kesim' ' de dedir [15].

5.14 Transistör

Transistör, yarı iletken malzemeden yapılmış elektronik devre elemanıdır. Her ne kadar diyodun yapısına benzese de çalışması vefonksiyonları diyottan çok farklıdır. Transistör yapısal bakımdan, yükselteç olarak çalışma özelliğine sahip bir devre elemanıdır [19; 20; 24].

Transistör iki eklemli üç bölgeli bir devre elemanı olup iki ana çeşittir: NPN, PNP.

Transistör, iki elektrodun arasındaki direnci, üçüncü elektroda uygulanan gerilim ile değişen bir devre elemanıdır. Transistör yan yana birleştirilmiş iki PN diyodundan oluşur. Birleşmesirasına göre NPN veya PNP tipi transistör oluşur [19; 20].

Transistörün başlıca çeşitleri şunlardır :

- Yüzey birleşmeli (Jonksiyon) transistor
- Nokta temaslı transistör
- Unijonksiyon transistör
- Alan etkili transistör
- Foto transistör

- Tetrot (dört uçlu) transistör
- Koaksiyal transistör

5.15 Kapasitör

Kapasitör, arası yalıtılmış iki iletken plakadan oluşan basit bir aygıttır. Bir pilin iki ucu arasına, bir kapasitör bağlandığında, elektronlar eksi kutuptan artı kutba doğru harekete geçer ve kapasitörün, pilin eksi kutbuna bakan plakasında birikmeye başlarlar. Diğer plakada ise, eşdeğer miktarda elektron eksikliği oluşur [19; 20].

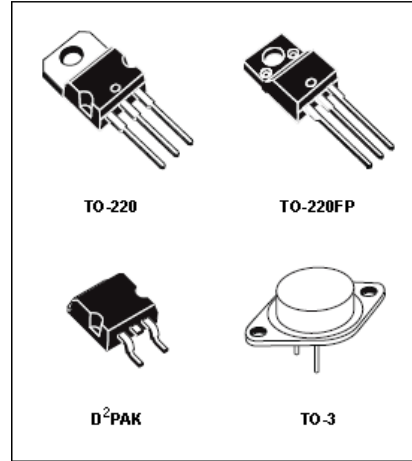
Başlangıçta, görece yüksek bir akım plakalara ulaşmış ve plakalardaki yük birikimi sonucu, aralarında bir gerilim oluşturmaya başlamıştır. Bu gerilim arttıkça, elektron akışına karşı koymaya başlar ve akım azalır. Plakalar arasındaki gerilim, pilin gerilimine ulaştığında, ikisi ters yönlü olduklarından, akım durur. Burada önemli olan bir husus; önce akımın plakalara ulaşıp artması, gerilimin ise sonradan artıp, bir bakıma ‘geriden’ inşasıdır. Bu yüzden, bir kapasitörde ‘gerilimin akımı izlediği, veya ‘akımın gerilime öncülük ettiği’ söylenir [19; 20].

Sonuç olarak, plakalar arasında bir elektrik alanı oluşmuş ve bu alana; kapasitörün karşı koymasına karşın elektron pompalamaya devam etmiş olan, yani kapasitör üzerinde iş yapmış bulunan pil vücut vermiştir. Ancak, bu elektrik alanının inşası için harcanmış olan enerji, dirençte harcananın aksine, ‘kaybolmuş’ olmayıp; depolanmış potansiyel enerjidir ve yeniden kullanılabilir. Nitekim, bu haliyle devreden pil çıkartılıp yerine bir ampul bağlanacak olsa; kapasitörün eksi plakasındaki elektronlar artı plakaya doğru akar ve plakaları nötr hale getirirken, bu arada ampulün yanmasına yol açarlar [19; 20].

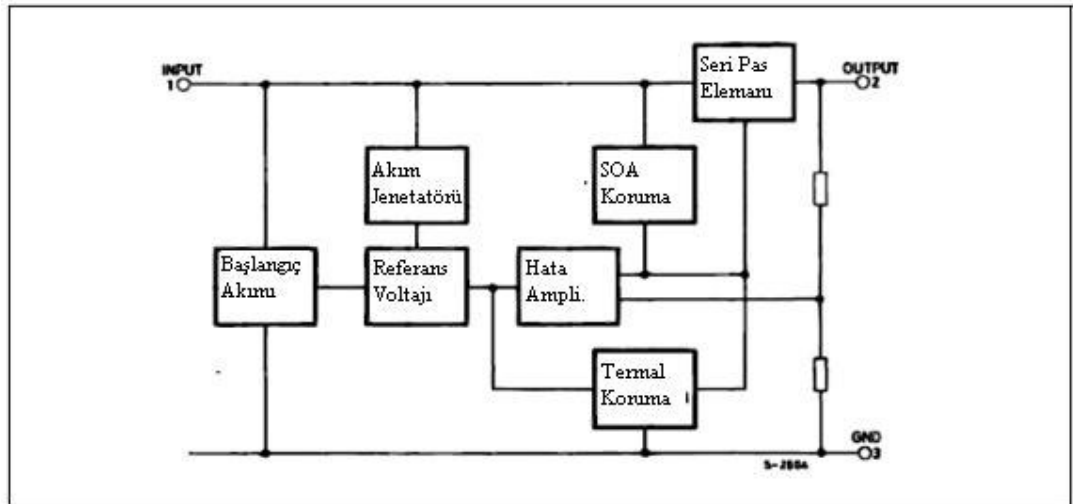
Pilin gerilimi V iken plakalarda oluşan en fazla yük miktarı Q ise, Q/V oranına kapasitans (C) denir [19; 20].

5.16 7805 Regülâtör Entegre Devre

Üç terminalli pozitif regülâtörlerdir. Bu çalışmada 7805'lerden L7805CV kullanılmıştır. L7805CV'lerden de TO-220 çeşidi kullanılmıştır. [24]



Şekil 5.18 L7805CV Çeşitleri [17]



Şekil 5.19 L7805 Bağlantı Şeması [24]

5.19 Devrelerin Eleman Listeleri

5.19.1 Adım Motor Devresi

- 2 Yönlü Konektör (4 adet)
 - ULN2003A (1 Adet)
 - 18 bacaklı DIL Soket (1 Adet)
 - 4MHz Rezonatör (1 Adet)
 - Switch (3 Adet)
 - 10K Ω Direnç (2 Adet)
 - 1,2K Ω Direnç (1 Adet)
 - LED 1mA – 3mA arası veya ile çalışan (1 Adet)
 - 22nF Kapasitör (1 Adet)
 - 220nF Kapasitör (1 Adet)
 - 4.7K Ω Direnç (1 Adet)
 - 7805 Regülatör Entegre Devre (1 Adet)
 - PIC 16F84A Mikrokontrolör (1 Adet)
- [17]

5.19.2 Servo Motor Devresi

- 5V RC Servo Motor (1 Adet)
- PIC 16F877A Mikrokontrolör (1 Adet)
- 20MHz (1 Adet)
- LED 1mA – 3mA arası veya ile çalışan (1 Adet)
- 470 Ω (1 Adet)
- 7805 Regülatör Entegre Devre (3 Adet)
- 10K Ω Direnç (1 Adet)
- Switch (3 Adet)

[11; 12]

5.19.3 DC Motor (L293D) Devresi

- PIC 16F84A Mikrokontrolör (1 Adet)
- 22pF Kapasitör (2 Adet)
- 4MHz Kristal (1 Adet)
- 4,7K Ω Direnç (1 Adet)
- 10K Ω Direnç (1 Adet)
- 1N4001 Diyot (4 Adet)
- 5V DC Motor (1 Adet)
- Switch (2 Adet)

[15]

5.19.4 DC Motor (LMD18200T) Devresi

- PIC 16F84A Mikrokontrolör (1 Adet)
- 22pF Kapasitör (2 Adet)
- 4MHz Kristal (1 Adet)
- 4,7K Ω Direnç (1 Adet)
- 10K Ω Direnç (2 Adet)
- Switch (2 Adet)
- 10nF Kapasitör (2 Adet)
- 12 V DC Motor (1 Adet)
- LMD18200T Entegre Devre (1 Adet)

[15]

5.20 Yazılım İçin Seçilen Programlama Dili

Mikro-denetleyici programlama hemen hemen tüm programlama dilleri ile yapılabilir fakat en çok kullanılan diller V.Basic, C ve Assembler dilleridir. Donanımın en verimli kullanılması için makinaya daha yakın dillerin kullanılması daha uygundur. Fakat kullanıcı için daha kolay sonuca ulaşılabilecek diller doğal olarak kullanıcıya daha yakın dillerdir. V.Basic kullanıcıya en yakın dillerdendir fakat kod optimizasyonunda en dezavantajlı dildir. Assembler makinaya en yakın

dildir fakat kullanıcı dostu bir dil değildir. Burada çözüme en uygun çözüm C dili ile mümkün olabilir. C dili kullanıcı açısından V.Basic'ten kayda değer miktarda daha zordur fakat kod optimizasyonu konusunda çok daha tatmin edicidir. Bu çalışmada kullanılan assembler komutları ekler bölümünde verilmiştir. Ayrıca çalışmada yapılan programlamada kullanılan yazmaçlar da ekler bölümünde mevcuttur [12; 13].

5.21 Yazılım İçin Seçilen Derleyiciler

Visual Basic, Java, C#, C++, C ve Assembler derleyeci ve linkerler ara moduller, .dll, .stl ve .exe dosyalar oluşturabilirler. Fakat mikrodnetleyici için program derlemek ve mikrodnetleyicinin anyalacağı koda dönüştürmek için o mikrodnetleyicilere ait derleyicilere ihtiyaç vardır. Bu denetleyicilere entegre mikrodnetleyici programlayıcıların olması da yapılacak işi kolaylaştıran fakat zorunluluk arz etmeyen bir konudur. Bu çalışmada kullanılan PIC mikrodnetleyiciler için son format HEX formatıdır. Bu format değişik dillerde yazılıp, o dillere ait PIC mikrodnetleyici derleyicileri ile derlenip HEX koda dönüştürüldüğü takdirde, aslında çok rahatlıkla programlayıcı donanımına ait programlayıcı yazılımlar ile PIC mikrodnetleyicilere aktarılabilirler [12; 13].

PIC mikrodnetleyici derleyecileri çeşitli olmakla birlikte C ve Assembler derleyecileri dillerin özelliklerinden dolayı düşük seviyeli derleyecilerdir. Düşük seviyeli diller makinaya yakın dillerdir. Yüksek seviyeli diller kullanıcıya yakın dillerdir. Düşük seviyeli diller yazılım optimizasyonu açısından daha uygundur. Ayrıca donanım seçimi açısından daha geniş bir yelpaze sunar [12; 13].

Ayrıca mikrodnetleyici programlama konusunda yardımcı olan programlarda mevcuttur. Bu tür programlar sayesinde mikrokntroler programlama çok daha kolaylaşmıştır. Ayrıca zaman açısından hem programlama, hem de devre oluşturma çok daha hızlanmıştır. Yazılıma yardımcı olan programlarda akım şeması simüle edilerek yazılım konusunda ön planlama, çalışmanın devamı açısından son

derece verimli ve kesin sonuçlar verecek hale gelmiştir. Donanıma yardımcı olan programlar ise zaman kazanımı olarak çok daha etkilidir. Bu tür programlarda devre şematik olarak çizilerek simüle edilir ve devre donanımsal olarak oluşturulunca ortaya çıkabilecek sorunları devre oluşturulmadan giderme olanağı verir [12; 13].

5.22 Yazılımların Algoritmalar

Bu bölümde verilmiş södökodların ait oldukları kodlar ekler bölümünde verilmiştir.

5.22.1 Servo2.asm

PIC 16F877 programlamak için başlangıç işlemleri yapılmıştır.

SAYAC, SAYAC1, SAYAC2, SAYAC3 değişkenleri tanımlanmıştır.

PORTB temizlenmiştir.

TRISA yazmacına 7 değeri yüklenmiştir. (RA0, RA1, RA2 = 1)

TRISB yazmacı sıfırlanır.

1 milisaniyelik sinyal üretimi PORTB'den D_1msn1 ve D_1msn2 döngüleri sayesinde zaman ayarlaması yapılarak sağlanmıştır.

1.2 milisaniyelik sinyal üretimi PORTB'den D_1.2msn1 ve D_1.2msn2 döngüleri sayesinde zaman ayarlaması yapılarak sağlanmıştır.

2 milisaniyelik sinyal üretimi PORTB'den D_2msn0, D_2msn1, D_2msn2 döngüleri sayesinde zaman ayarlaması yapılarak sağlanmıştır.

10 milisaniyelik gecikme D_10ms1 ve D_10ms2 ve s10 döngüleri ile sağlanmıştır.

15 milisaniyelik gecikme D_15ms1 ve D_15ms2 ve s15 döngüleri ile sağlanmıştır.

Button Kontrolleri (A0, A1 ve A2) “btfss” ve “goto” komutları ile sıraya konmuştur.

Ana Programda adımlar arasında ve yön değişimleri arasındaki gecikmeler button kontrolleri ile ilgili fonksiyonlar çağırılarak yapılmıştır.

5.22.2 STEP2.c

Pic.h ve delay.c dosyaları compile aşamasında STEP.c dosyasına eklenmiştir.

STEP, YON, KAPATMABUTTON, LED ifadeleri sırası ile RB0, RB1, RA4, RB7 için tanımlanmışlardır.

adim, gecikme, j değişkenleri tanımlanmıştır. RPM ise ilk değer verilerek tanımlanmıştır.

Adimat ve LedYak fonksiyonlarının prototipleri tanımlanmıştır.

adimat fonksiyonu tanımlanmıştır. Bu fonksiyonda A portlarına, j değişkeni atanmıştır. YON anahtarı 1’e çekilirse j 16 olana kadar sola iteler. j 16 olduğunda ise tekrar 1’e atanır. YON anahtarı 1 değilse j sağa itelenir.

LedYak fonksiyonu kapatma anahtarı kapalı ise led’e sinyal göndermek için sonuz döngü içermektedir.

TRISB yazmacına 0x7F değeri yüklenmiştir.

STEP’e 0 değeri yüklenerek RB0 sıfırlanmıştır.

j değişkenine sıfır değeri yüklenmiştir.

gecikme değişkeni hesaplanmıştır.

TRISA yazmacına 16 değeri yüklenmiştir.

Sonsuz döngüde Kapatma düğmesi kontrol edilmektedir. Eğer düğme açık ise adım düğmesi kontrol edilmektedir. Eğer düğmeye basılmış ise adimat fonksiyonu çağrılmaktadır ve bekletme fonksiyonu çağrılmaktadır. Kapatma düğmesi kapalı ise LedYak fonksiyonu çağrılarak led yakılmaktadır.

5.22.3 DCL293D.c

Pic.h dosyası compile aşamasında DCL293D.c dosyasına eklenmiştir.

ONOFF, YON, ON, SAAT_YONU, TERS_SAAT_YONU ifadeleri sırası ile RA0, RA1, 1, 5, 10 için tanımlanmışlardır.

TRISB, TRISA, PORTB yazmaçlarına sırası ile 0, 3, 0 değerleri yüklenmiştir. Bu sayede B portu çıkış yapılmış, A portu giriş yapılmış ve be portu temizlenmiştir.

Sonsuz döngüde; ONOFF düğmesinden sinyal geldiği zaman YON düğmesine basılmış ise PORTB, yani H-Köprüsüne bağlı ayaklardan ilgili olana sinyal gönderilir ve motor saat yönüne doğru döner. Eğer basılmamış ise motor saat yönünün tersine döner. ONOFF düğmesine basılmamış ise B portunu temizler.

5.22.4 LMD18.c

Pic.h dosyası compile aşamasında LMD18.c dosyasına eklenmiştir.

ONOFF, YON, ON, SAAT_YONU, TERS_SAAT_YONU ifadeleri sırası ile RA0, RA1, 1, 1, 3 için tanımlanmışlardır.

TRISB, TRISA, PORTB yazmaçlarına sırası ile 0, 3, 0 değerleri yüklenmiştir. Bu sayede B portu çıkış yapılmış, A portu giriş yapılmış ve be portu temizlenmiştir.

Sonsuz döngüde; ONOFF düğmesinden sinyal geldiği zaman YON düğmesine basılmış ise PORTB, yani H-Köprüsüne bağlı ayaklardan ilgili olana sinyal gönderilir ve motor saat yönüne doğru döner. Eğer basılmamış ise motor saat yönünün tersine döner. ONOFF düğmesine basılmamış ise B portunu temizler.

6. SONUÇ

Bu çalışmadaki amaç bir vana kontrol sistemi tasarlayıp incelemek olarak belirlenmiştir. Bu çalışma sonunda DC motorlar için olan sistemler dışındaki diğer sistemler doğrudan vana steminden bağlandıkları için daha kolay incelenebilmiştir. DC motorlar ise ayrıca kompleks bir dişli sistemine gereksinim duydukları için sadece motorun kontrolü açısından incelenebilmişlerdir. PIC16F84A kullanılan sistemlerdeki boş portlara daha başka devre elemanları ve mikrodnetleyiciler bağlanarak proseslerdeki diğer akış elemanları ve proses ana sistemi ile istenirse iletişim kurulabilir. PIC 16F877A mikrodnetleyicisi kullanılan sistemler için bu çok daha kolaydır çünkü bünyesinde A/D (Anolog dijital) çevirici bulundurur. Prensip olarak, her motor çeşidi kendi çeşidi için geçerli olan programlamada, değişik motor büyüklükleri için seçilen H-Köprülerinden dolayı oluşan ufak değişikliklerin haricinde aynıdır. Bu çalışmada da ortaya çıkan sonuca göre istenen akış kontrolü için motor çeşidi seçimi ve H-Köprüsü seçimi en az mikrodnetleyici seçimi kadar önemlidir. Örnek olarak L293D ve normal bir DC motor bir iğne uçlu glob vana için kullanılabilir veya kazan besı suyu kontrolü için LMD18200 ve Servo Motor kullanılabilir. Ya da daha büyük çapta vanalar için gerekli motor cinsi ile birlikte entegre değil de devre elemanları kullanılarak geliştirilmiş bir H-Köprüsü kullanılabilir.

Değişik tasarımlar üzerinden tartışma yürütülmesi sonucu seçilen vana çapı ile seçilmesi gerek H-Köprüsünün, steme uygulanması gereken kuvvetin değişmesi ve buna bağlı olarak motorun değişmesinden dolayı birbirini en çok etkileyen etmen olduğu görüldü. Bu da bu konuda yapılan çalışmaların sonuçlarını destekler nitelikte bir katkı sağlamaktadır. Bu çalışma, bunun gibi her tasarım sorununa özgü ele alınan çözüm seçeneklerine katkı niteliği taşımaktadır. Bu çalışma bir karşılaştırma tartışması olarak da nitelendirilebilir.

Bu tür çalışmalar yürütecekler için verilebilecek tavsiye gerek devre ve akış control elemanı seçiminde, gerekse proses control system tasarımlarında bilimsel olduğu kadar ekonomik olarak da bir optimizasyona gidilmesidir. Proses ihtiyacına göre öncelikle teknik daha sonra ekonomik olarak doğru mikrodnetleyiciyi ve buna

bağlı elemanları seçmek daha doğru sonuçlara götürür. Bu çalışmanın devamı niteliğinde olacak bir çalışma, vana kontrol sisteminin değişik proses kontrollere uygulanması şeklinde olabilir.

Bunun dışında impilimentasyon sonrası söylenebilecek şey şudur: İki sistemde yüksek teknolojik bilinç düzeyi gerektiren bakım ihtiyacını doğururlar bu da bakım ve personel giderini arttıracak gibi aynı zamanda da otomasyon sebebi ile personel sayısının azaltılması ve daha kesin akış kontrolü sebebi ile kayıpları azaltacağı için toplamda daha avantajlı görülmektedir.

Otomasyon ilk kuruluş maliyeti göz önünde tutularak genelde yapılmasına karar verilen bir sistem olduğu için genellikle en çok ihtiyaç olunan veya manuel olarak kontrol edilmesi neredeyse mümkün olmayan yerlerde tercih edilmektedir. Örnek olarak sürekli dozajlama gereken sistemlerde veya sürekli aynı seviyede kalması gereken kısımların giriş çıkışlarında vana otomasyonu kullanılmaktadır.

Endüstride proses otomasyonunun yaygın olmamasının sebebi teknolojinin kısa vadede pahalı olmasının ve bakımının daha yüksek bilinç düzeyi gerektiren personel tarafından yapılması gerekmesinin yanısıra sosyal ve politik sebeplere de dayanmaktadır. Dünyanın pek çok yerinde olduğu gibi özellikle kamu desteği ile kurulmuş olan endüstrilerde topluma sosyo-ekonomik fayda getirmek için otomasyondan vazgeçilen kısımlar olmuştur. Kamu desteği ile kurulmuş olan endüstriler boyut olarak çok büyük ve kompleks endüstriler oldukları için otomasyona doğrudan etkileri mevcuttur.

7. KAYNAKLAR

- 1- Smith, P. ve Zappe, R.W., Valve Selection Handbook (Fifth Edition), Elsever, 2004
- 2- Weiner, R. S., “Basic Criteria and Definitions for Zero Fluid Leakage”, Technical Report NO 32-926, Jet Propulsion Laboratory Cal. Instn. Of Techn., Aralık 1966
- 3- Wells, F. E., “A Survey of Leak Detection of Aerospace Hardware”, National Fall Conference for the American Society of Nondestructive Testing, Ekim 1968
- 4- Holmberg, E. G., “Valve Design: Special for Corrosives”, Chem Engg., 13 Haziran 1960
- 5- Robertson J. C., “Improving Valve Life and Operating Efficiency The Easy Way”, www.swspitcrew.com Strategic Work Systems, 1998
- 6- Darvas, L. A., “Cavitation in Closed Conduit Flow Control Systems”, Civil Engg. Trans. Instn. Engrs. Aust. Vol. CE12 No. 2, Ekim 1970
- 7- Yalçın, K., Hacimsel ve Santrifüj Pompalar, Çağlayan, 1998
- 8- Kobori, T., Yokohoma S. ve Miyashiro H., “Propagation Velocity of Pressure Wave in Pipe Line”, Hitachi Hyoron Vol. 37 No. 10, Ekim 1955
- 9- Boumann, H. D., “Universal Valve Noise Prediction Method”, ISA Handbook of Control Valves, 1976
- 10- Microchip Technology, PIC16F87X Data Sheet , 2001
- 11- Öztürk Ş. Ç., PIC ile Servo Motor Kontrolü www.scozturk.com, 15 haziran 2007
- 12- Dinçer, G., PIC Mikrokontrolör Uygulama Devreleri, Era, 2002
- 13- Bodur, Y., Adım Adım PICmicro Programlama, Era, 2001
- 14- Microchip Technology, PIC16F84A Data Sheet, 2001
- 15- İbrahim, D., PIC C Motor Kontrol Projeleri, Bileşim, Haziran 2004
- 16- İbrahim, D., C ile PIC Mikrokontrolör Programlama, Era, 2002
- 17- İbrahim, D., PIC Adım Motor Kontrolü, Bileşim, Eylül 2004
- 18- Coşkun, İ. ve Işık M. F., Servo Motoru Mikrodenetleyici ile Konum ve Hız Denetimi, G. Ü. Fen Bilimleri Dergisi, 08.03.2004

- 19- Boylestad, R. ve Nashelsky, L., Elektronik Elemanlar ve Devre Teorisi, M.E.B, 18.05.1994
- 20- IDC Technologies, Otomasyon Cep Klavuzu, Mart 2006
- 21- Regan, T., A DMOS 3A, 55V, H-Bridge: The LMD18200 National Semiconductor Application Note 694, Aralık 1999
- 22- SGS-Thomson Microelectronics, L293D Data Sheet, 1994
- 23- Allegro MicroSystems, 2003 -2024 High Voltage, High Current, Darlington Arrays Data Sheet 29304F, 1998
- 24- ST Microelectronics, L7800 Series Data Sheet, 2003

8. EKLER

8.1 PIC 16F84 ve 16F877’de Kullanılan Komut Seti

PIC mikrodenetleyicilerinde assembler dili ile programlama yapmak için kullanılan komut seti aşağıdaki gibidir:

ADDLW: Bir sabit and W’nin içeriğini topla

Sintaks: [etiket] ADDLW k

Operandlar: $0 \leq k \leq 255$

İşlem: $(W) + k \rightarrow (W)$

Etkilenen Status: C, DC, Z

Tanımı: W’nin içeriğini 8 bitlik k sabiti ile toplar ve sonucu W’ye aktarır.

ADDWF: Bir yazmaç içeriği ile W’nin içeriğini topla

Sintaks: [etiket] ADDWF f,d

Operandlar: $0 \leq f \leq 127$

$d \in [0,1]$

İşlem: $(W) + f \rightarrow (\text{hedef})$

Etkilenen Status: C, DC, Z

Tanımı: W’nin içeriğini 7 bitlik k sabiti ile toplar ve sonucu; d = 0 ise W’ye, d = 1 ise f’ye aktarır.

ANDLW Bir sabit ile W’nin içeriğine AND (ve) işlemi uygula

Sintaks: [etiket] ANDLW k

Operandlar: $0 \leq k \leq 255$

İşlem: $(W) .AND. (k) \rightarrow (W)$

Etkilenen Status: Z

Tanımı: W’nin içeriğini k ile AND işlemine tabi tutar ve sonucu W’ye aktarır.

ANDWF: Bir yazmaç içeriğine, W’nin içeriği ile AND işlemi uygula

Sintaks: [etiket] ANDWF f,d

Operandlar: $0 \leq f \leq 127$

$d \in [0,1]$

İşlem: $(W) .AND. (k) \rightarrow (hedef)$

Etkilenen Status: Z

Tanımı: W'nin içeriğini f yazmacının içeriği ile AND işlemine tabi tut ve sonucu; d = 0 ise W'ye, d = 1 ise f'ye yükler.

BCF: Yazmacın belirlenen bitini sıfırla

Sintaks: [etiket] BCF f,b

Operandlar: $0 \leq f \leq 127$

$0 \leq b \leq 7$

İşlem: $0 \rightarrow (f \ll b)$

Etkilenen Status: Yok

Tanımı: f yazmacının b. Bitini '0' yapar.

BSF: Yazmacın belirlenen bitini Set (1) et

Sintaks: [etiket] BSF f,b

Operandlar: $0 \leq f \leq 127$

$0 \leq b \leq 7$

İşlem: $0 \rightarrow (f \ll b)$

Etkilenen Status: Yok

Tanımı: f yazmacının b. Bitini '1' yapar.

BTFS Yazmacın belirlenen biti '1' ise, bundan sonraki komutu atla

Sintaks: [etiket] BTFS f,b

Operandlar: $0 \leq f \leq 127$

$0 \leq b \leq 7$

İşlem: $0 \rightarrow (f \ll b)$

Etkilenen Status: Yok

Tanımı: Yazmacın b. Bitinin '1' olup olmadığı kontrol edilir. Eğer '1' ise bu komutun altındaki komut işlenmez. Bir sonraki komuta atlanır. Aksi durumda sıradaki komut uygulanır.

BTFS Yazmacın belirlenen biti '0' ise, bundan sonraki komutu atla

Sintaks: [etiket] BTFSC f,b

Operandlar: $0 \leq f \leq 127$
 $0 \leq b \leq 7$

İşlem: $0 \rightarrow (f < b >)$

Etkilenen Status: Yok

Tanımı: Yazmacın b. Bitinin '0' olup olmadığı kontrol edilir. Eğer '0' ise bu komutun altındaki komut işlenmez. Bir sonraki komuta atlanır. Aksi durumda sıradaki komut uygulanır.

CALL: Altprogram çağır

Sintaks: [etiket] CALL k

Operandlar: $0 \leq k \leq 2047$

Operation: $(PC) + 1 \rightarrow TOS,$
 $k \rightarrow PC < 10:0 > ,$
 $(PCLATH < 4:3 >) \rightarrow PC < 12:11 >$

Etkilenen Status: Yok

Tanımı: Altprogramı çağırır. Önce PC'yi bir artırır ve yığının (stack) üstüne koyar. Sonra altprogram adresi PC'nin <10:0> bitlerine yüklenir. PCLATH'ın <4:3> bitlerindeki değerler, PC'nin üst bitleri olan <12:11> arasındaki bitlere yüklenir. CALL işlemi iki saat çevrimi sürede uygulanan bir dallanma komutudur.

CLRF: Yazmaç içeriğini sil (sıfırla)

Sintaks: [etiket] CLRF f

Operandlar: $0 \leq f \leq 127$

İşlem: $00h \rightarrow (f)$
 $1 \rightarrow Z$

Etkilenen Status: Z

Tanımı: f yazmacının içeriği sıfırlanır (temizlenir) ve değeri sıfır olduğu için status yazmacının zero biti set (1) edilir.

CLRW: W yazmacını temizle

Sintaks: [etiket] CLRW

Operandlar: Yok

İşlem: $00h \rightarrow (W)$

$$1 \rightarrow Z$$

Etkilenen Status: Z

Tanımı: W yazmacının içeriği sıfırlanır (temizlenir) ve değeri sıfır olduğu için status yazmacının zero biti set (1) edilir.

CLRWDI: Watchdog timer içeriğini sil (temizle)

Sintaks: [etiket] CLRWDI

Operandlar: Yok

İşlem: $00h \rightarrow WDI$

$0 \rightarrow WDI \text{ önbölücü sabit (prescaler)}$

$1 \rightarrow \overline{TO}$

$1 \rightarrow \overline{PD}$

Etkilenen Status: $\overline{TO}, \overline{PD}$

Tanımı: W yazmacının içeriği sıfırlanır (temizlenir) ve değeri sıfır olduğu için status yazmacının zero biti set (1) edilir.

COMF: Yazmaç değerinin tersini (komplement) al

Sintaks: [etiket] COMF f,d

Operandlar: $0 \leq f \leq 127$

$d \in [0,1]$

İşlem: $(f) \rightarrow (\text{hedef})$

Etkilenen Status: Z

Tanımı: f yazmacının içeriği terslenir ve d değeri sıfır ise sonuç W yazmacına, bir ise f yazmacına yüklenir.

DECF: Yazmaç içeriğini bir azalt

Sintaks: [etiket] DECF f,d

Operandlar: $0 \leq f \leq 127$

$d \in [0,1]$

İşlem: $(f) \rightarrow (\text{hedef})$

Etkilenen Status: Z

Tanımı: f yazmacının içeriği bir azaltılır ve d değeri sıfır ise sonuç W yazmacına, bir ise f yazmacına yüklenir.

DECFSZ: Yazmaç içeriğini bir azalt, '0' ise bir komut atla

Sintaks: [etiket] DECFSZ f,d

Operandlar: $0 \leq f \leq 127$

$d \in [0,1]$

İşlem: $(f) \rightarrow (\text{hedef})$

Sonuç '0' ise atla

Etkilenen Status: Yok

Tanımı: f yazmacının içeriği 1 azaltılır ve sonuç oluşan değer sıfır ise bu komutu izleyen komut atlanır. Sonuçta $d = 0$ is W'ye, 1 ise F'ye yüklenir. Komut atlamayla sonuçlanırsa, ikinci çevrim süresinde NOP uygulayarak, toplam iki saat çevrim süresinde işlenir. Atlama olmadığı durumda uygulanması bir saat çevrimi süredir.

GOTO: Adres git

Sintaks: [etiket] GOTO k

Operandlar: $0 \leq k \leq 2047$

İşlem: $k \rightarrow \text{PC}<10:0>$

$\text{PCLATH}<4:3> \rightarrow \text{PC}<12:11>$

Etkilenen Status: Yok

Tanımı: GOTO koşulsuz bir sapma komutudur. k'nın adresi neyse PC'ye <10:0> bitlerine yüklenir. Bellek sayfası neyse PCLATH<4:3> bitleri PC'nin üst bitlerine yüklenir ve adrese sapılır. Bu komut, iki saat çevrimi sürede uygulanır.

INCF: Yazmaç içeriğini bir arttır

Sintaks: [etiket] INCF f,d

Operandlar: $0 \leq f \leq 127$

$d \in [0,1]$

İşlem: $(f) + 1 \rightarrow (\text{hedef})$

Etkilenen Status: Z

Tanımı: f yazmacının içeriği bir arttırılır ve d değeri sıfır ise sonuç W yazmacına, bir ise f yazmacına yüklenir.

INCFSZ: Yazmaç içeriğini bir arttır, '0' ise bir komut atla

Sintaks: [etiket] INCFSZ f,d

Operandlar: $0 \leq f \leq 127$

$d \in [0,1]$

İşlem: $(f) + 1 \rightarrow (\text{hedef})$

Sonuç '0' ise atla

Etkilenen Status: Yok

Tanımı: f yazmacının içeriği bir arttırılır ve sonuçta oluşan değer sıfır ise bu komutu izleyen komut atlanır. Sonuç d değeri sıfır ise sonuç W yazmacına, bir ise f yazmacına yüklenir. Sonuç atlama ile sonuçlanırsa ikinci çevrim süresinde NOP uygulayarak toplam iki saat çevrim süresinde işlenir. Atlama olmadığı durumda uygulanması bir saat çevrim süredir.

IORLW: Bir sayı ile W'nin içeriğine IOR (veya) işlemini uygula

Sintaks: [etiket] IORLW k

Operandlar: $0 \leq k \leq 127$

$d \in [0,1]$

İşlem: $(W) .OR. k \rightarrow (W)$

Etkilenen Status: Z

Tanımı: W yazmacının içeriği k sabiti ile OR'lanır. Sonuç W'ya yüklenir.

Mantıksal işlem sonunda oluşan değer sıfır ise, status Z biti 1 yapılır.

IORWF: Yazmaç içeriği ile W'nin içeriğine IOR (veya) işlemi uygula

Sintaks: [etiket] IORWF f,d

Operandlar: $0 \leq f \leq 127$

$d \in [0,1]$

İşlem: $(W) .OR. f \rightarrow (\text{hedef})$

Etkilenen Status: Z

Tanımı: W yazmacının içeriği k sabiti ile OR'lanır ve sonuç $d = 0$ ise W'ye, $d = 1$ ise f'ye yüklenir. Mantıksal işlem sonunda oluşan değer '0' ise, status Z biti set (1) edilir.

MOVF: Yazmaç içeriğini hedefe taşı

Sintaks: [etiket] MOVF f,d

Operandlar: $0 \leq f \leq 127$

$$d \in [0,1]$$

İşlem: $(f) \rightarrow (\text{hedef})$

Etkilenen Status: Z

Tanımı: f yazmacının içeriği; $d = 0$ ise W, '1' ise kendisine yüklenir. Özellikle $d = 1$ olduğu durumda, f'nin içeriğinin '0' olup olmadığına bakılması yararlı olur. Çünkü bu durumda status'ün Z biti değişir.

MOVLW: W'ye bir sabit yükle

Sintaks: [etiket] MOVL W k

Operandlar: $0 \leq k \leq 255$

İşlem: $k \rightarrow (W)$

Etkilenen Status: Yok

Tanımı: Komut öncesi W'nin değeri ne olursa olsun, komuttan sonra k sabiti ile yüklenir.

MOVWF: W'nin içeriğini f yazmacına taşı

Sintaks: [etiket] MOVWF f

Operandlar: $0 \leq f \leq 127$

İşlem: $(W) \rightarrow (f)$

Etkilenen Status: Yok

Tanımı: W yazmacının içeriğini f yazmacına taşır.

NOP: İşlem yapma

Sintaks: [etiket] NOP

Operandlar: Yok

İşlem: Yok

Etkilenen Status: Yok

Tanımı: Hiçbir işlem yapmadan bir saat çevrimi zaman geçir.

RETFIE: Kesme (interrupt) altprogramından geri dön

Sintaks: [etiket] RETFIE

Operandlar: Yok

İşlem: $TOS \rightarrow PC,$
 $1 \rightarrow GIE$

Etkilenen Status: Yok

Tanımı: Kesme altprogramından dönmek için kullanılır. Dönüş hazırlığından başka bir işlem yapmaz. Dönüş yapacak adres TOS'de olduğu için, TOS değeri PC'ye yüklenir. INTCON kesme (interrupt) yazmacının, GIE (Global Interrupt Enable) biti set edilir. Komut iki saat çevriminde işlenir.

RETLW: Altprogramından W'ye bir sabit yükle ve geri dön

Sintaks: [etiket] RETLW k

Operandlar: $0 \leq k \leq 255$

İşlem: $k \rightarrow (W)$

Etkilenen Status: Yok

Tanımı: Altprogramından, W'ye k sabiti yüklenmiş olarak dönmek için kullanılır. Dönüş hazırlığı TOS değerinin PC'ye aktarılmasıyla yapılır. Komut iki saat çevrimi sürede işlenir.

RETURN: Altprogramdan geri dön

Sintaks: [etiket] RETURN

Operandlar: Yok

İşlem: $TOS \rightarrow PC,$

Etkilenen Status: Yok

Tanımı: Altprogramdan TOP'daki değere geri döner. Komut iki saat çevrimi sürer.

RLF: Yazmaç bitleri sola doğru döndür

Sintaks: [etiket] RLF f,d

Operandlar: $0 \leq f \leq 127$

$d \in [0,1]$

İşlem: Tanım kısmında açıklanmıştır.

Etkilenen Status: C

Tanımı: f yazmacındaki bitleri bir bit sola doğru yerleştirir. Yazmaç 8 bitlik olduğundan 7. bitin değeri status yazmacının carry bitine yerleştirilir. Daha sonra carry bitindeki değer f yazmacının 0. bitine aktarılır. Böylece hiçbir bit bozulmadan sola doğru kaymış olur. $d = 0$ ise sonucu W'ye aksi halde f'ye taşır. C biti f yazmacının en üst bitinin değerini taşır.

RRF: Yazmaç bitlerini birer bit sağa aktar

Sintaks: [etiket] RRF f,d

Operandlar: $0 \leq f \leq 127$

$d \in [0,1]$

İşlem: Tanım kısmında açıklanmıştır.

Etkilenen Status: C

Tanımı: f yazmacındaki bitleri bir bit sağa doğru yerleştir. Yazmaç 8 bitlik olduğundan 0. bitin değeri status yazmacının carry bitine yerleştirilir. Daha sonra carry bitindeki değer f yazmacının 7. bitine aktarılır. Böylece hiçbir bit bozulmadan sağa doğru kaymış olur. $d = 0$ ise sonucu W'ye aksi halde f'ye taşır. C biti f yazmacının en alt bitinin değerini taşır.

SLEEP: Uyu

Sintaks: [etiket] SLEEP

Operandlar: Yok

İşlem: $00h \rightarrow WDT$,

$0 \rightarrow WDT$ önbölücü sabit(prescaler),

$1 \rightarrow \overline{TO}$

$1 \rightarrow \overline{PD}$

Etkilenen Status: $\overline{TO}, \overline{PD}$

Tanımı: $\overline{PD}$, güç kesim (power down) biti temizlenir. $\overline{TO}$, süre aşımı (time-out) biti set (1) olur. WDT ve önbölücü sabit de sıfırlanır. Osilatörün durması ile işlemci uyuma moduna geçer. PIC bu durumda çok az güç harcar.

SUBLW Bir sabit W'nin içeriğinden çıkar

Sintaks: [etiket] SUBLW k

Operandlar: $0 \leq k \leq 255$

İşlem: $k - (W) \rightarrow (W)$

Etkilenen Status: C, DC, Z

Tanımı: f yazmacı içeriğinden, k çıkarılır. Sonuç W'ye yüklenir.

SUBWF: W'den f'yi çıkar

Sintaks: [etiket] SUBWF f,d

Operandlar: $0 \leq f \leq 127$

$d \in [0,1]$

İşlem: $(f) - (W) \rightarrow (\text{hedef})$

Etkilenen Status: C, DC, Z

Tanımı: f yazmacı içeriğinden, W çıkarılır. D = 0 ise sonuç W'ye, d = 1 ise f yazmacına yüklenir.

SWAPF: Yazmaç içeriğinde 4'lülerin (nibbles) yerini değiştir

Sintaks: [etiket] SWAPF f,d

Operandlar: $0 \leq f \leq 127$

$d \in [0,1]$

İşlem: $(f<3:0>) \rightarrow (\text{hedef}<7:4>),$

$(f<7:4>) \rightarrow (\text{hedef}<3:0>)$

Etkilenen Status: Yok

Tanımı: f yazmacının üst dörtlü biti ile alt dört biti yer değiştirirler. D = 0 ise sonuç W'ye, d = 1 ise f yazmacına yüklenir.

XORLW: Sayı ile W'nin içeriğini EXOR (Exclusive OR)'la

Sintaks: [etiket] XORLW k

Operandlar: $0 \leq k \leq 255$

İşlem: $(W) .XOR. k \rightarrow (W)$

Etkilenen Status: Z

Tanımı: W'nin içeriği ile f yazmacına mantıksal XOR işlemi uygulanır. Sonuç W yazmacına yüklenir.

XORWF: Yazmaç içeriği ile W'nin içeriğini EXOR(Exclusive OR)'la

Sintaks: [etiket] XORWF f,d

Operandlar: $0 \leq f \leq 127$

$d \in [0,1]$

İşlem: $(W) .XOR. (f) \rightarrow (\text{destination})$

Etkilenen Status: Z

Tanımı: W'nin içeriği ile f yazmacına mantıksal XOR işlemi uygulanır. D = 0 ise sonuç W'ye, d = 1 ise f yazmacına yüklenir. [12; 13]

8.2 Yazmaçlar

PIC16F84 mikrodnetleyicisinin yazmaç haritası Çizelge 8.1'deki gibidir.

Çizelge 8.1 PIC16F84 Yazmaç Haritası [12]

Dosya Adresi	BANK 0	BANK 1	Dosya Adresi
00h	INDF	INDF	80h
01h	TMR0	OPTION_REG	81h
02h	PCL	PCL	82h
03h	STATUS	STATUS	83h
04h	FSR	FSR	84h
05h	PORTA	TRISA	85h
06h	PORTB	TRISB	86h
07h	-	-	87h
08h	EEDATA	EECON1	88h
09h	EEADR	EECON2	89h
0Ah	PCLATH	PCLATH	8Ah
0Bh	INTCON	INTCON	8Bh
0Ch			8Ch
...	68 Genel Amaçlı Yazmaçlar (SRAM)	BANK 0 ile Ulaşılabilir.	...
4Fh			CFh
50h			D0h
...	'0' Olarak Okunan Görev Verilmemiş Bellek Alanı	'0' Olarak Okunan Görev Verilmemiş Bellek Alanı	...
7Fh			FFh

PIC 16F84A mikrodnetleyicilerini programlarken kullanılan belli başlı yazmaçların ayrıntılı açıklaması aşağıdaki gibidir. Yazmaç tanıtlarında kullanılan; R: Okunabilen bit, W: Yazılabilen bit, U: İşlenemez bit (sıfır olarak okunur), -n: POR'deki değer, '1' : Bit set edildi, '0': Bit sıfırlandı, x: Bit bilinmiyor anlamına gelmektedir [12].

Status (Durum) Yazmacı: (Adres: 03h, 83h)

R/W – 0 R/W - 0 R/W - 0 R – 1 R – 1 R/W – x R/W - x R/W – x

IRP	RP1	RP0	TO	PD	Z	DC	C
7	6	5	4	3	2	1	0

Bit 7: PIC16F84 veya 84A'da işlevsizdir. Her zaman "0" olarak tutulmalıdır.

Bit 6: PIC16F84 veya 84A'da işlevsizdir. Her zaman "0" olarak tutulmalıdır.

Bit 5: RP0: Yazmaç Bank'ı Seçme Biti (Direkt adresleme için kullanılır.)

01 = Bank 1 (80h – FFh)

00 = Bank 0 (00h – 7Fh)

Bit 4: $\overline{TO}$: Time Out Bitidir. (Zaman doldu.)

1 = Güç verildikten sonra CLRWDT komutu veya SLEEP komutu

0 = WDT time-out oluşmuştur.

Bit 3: $\overline{PD}$: Power-down bitidir. (Güç kesildi.)

1 = Güç verildikten sonra veya CLRWDT komutu ile

0 =SLEEP komutunun yerine getirilmesi ile birlikte

Bit2: Z: Sıfır Biti

1 = Bir aritmetik veya mantıksal işlemin sonucu '0' ise

0 = Bir aritmetik veya mantıksal işlemin sonucu '0' değil ise

Bit 1: DC: Digit Carry/ $\overline{\text{borrow}}$ (elde) Biti (ADDWF, ADDLW, SUBLW, SUBWF komutları) ($\overline{\text{borrow}}$, polarite tersinirdir.)

1 = Düşük dörtlü bit taşarsa 1 olur.

0 = Düşük dörtlü bit taşmazsa 0 olur.

Bit 0: C: Carry/ $\overline{\text{borrow}}$ (elde) Biti. (ADDWF, ADDLW, SUBLW, SUBWF komutları) ($\overline{\text{borrow}}$, polarite tersinirdir.)

1 = En duyarlı bit taşarsa 1 olur.

0 = En duyarlı bit taşmazsa 0 olur.

[12]

Option (Seçenek) Yazmacı: (Adres: 81h)

R/W - 1 R/W - 1 R/W - 1 R/W - 1 R/W - 1 R/W - 1 R/W - 1 R/W - 1

RBPU	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0
7	6	5	4	3	2	1	0

Bit 7: $\overline{\text{RBPU}}$: PORTB Çekme (Pull-up) Biti Aktifleştirme

1 = PORTB çekme dirençleri pasif olur.

0 = PORTB çekme dirençleri pasif olur.

Bit 6: INTEDG: Kesme (Interrupt) Kenarı Seçme Biti

1 = Kesme RB0/INT pininin yükselen kenarı ile olur.

0 = Kesme RB0/INT pininin düşen kenarı ile olur.

Bit 5: T0CS: TMR0 Saat Kaynağı Seçim Biti

1 = RA4/T0CKI bacağı saat kaynağı

0 = Saat kaynağı iç komut çevrimi olsun (CLKOUT)

Bit 4: T0SE: TMR0 Saat Kaynağı Kenar Seçme Biti

1 = RA4/T0CKI bacağının her düşen kenarı için arttır.

0 = RA4/T0CKI bacağıının her yükselen kenarı için arttır.

Bit 3: PSA: Ön bölücü Seçim Biti

1 = Ön bölücü WDT'ye ayrıldı

0 = Ön bölücü TMR0'a ayrıldı

Bit 2 – 0: PS2 – PS0: Önbölücü oran seçme bitleridir. Aşağıdaki anlamları ifade eder:

Bit Değeri	TMR0 Oranı	WDT Oranı
000	1:2	1:1
001	1:4	1:2
010	1:8	1:4
011	1:16	1:8
100	1:32	1:16
101	1:64	1:32
110	1:128	1:64
111	1:256	1:128

[12]

INTCON (Kesme) Yazmacı: (Adres: 0Bh, 8Bh)

R/W – 0 R/W - 0 R/W - 0 R/W – 0 R/W - 0 R/W - 0 R/W - 0 R/W - x

GIE	EEIE	TOIE	INTE	RBIE	TOIF	INTF	RBIF
7	6	5	4	3	2	1	0

Bit 7: GIE: Global Kesme Biti (Bütün kesmeleri aktif yapan bit)

1 = Maskeli olmayan kesmeleri aktif yap.

0 = Bütün kesmeleri kaldır.

Bit 6: EEIE: EE Yazma Bitişinde Kesme Aktif Yapma Biti

1 = EE yazma bitişinde kesmeyi aktif yap.

0 = EE yazma bitişinde kesmeyi kaldır.

Bit 5: T0IE: TMR0 Taşma Kesme Biti

1 = TMR0 Kesmesini aktif yap.

0 = TMR0 Kesmesini kaldır.

Bit 4: INTE: RB0/INT Kesme Aktif Biti

1 = RB0/INT Kesmesini aktif yap.

0 = RB0/INT Kesmesini aktif kaldır.

Bit 3: RBIE: RB Port Değişirme Kesme Biti

1 = RB port değişimi kesmesini aktif yap.

0 = RB port değişimi kesmesini kaldır.

Bit 2 : T0IF: TMR0 Taşma Uyarısı Bayrak Biti

1 = TMR0 Taştı (Yazılımda temizlenmelidir.)

0 = TMR0 Taşmadı.

Bit 1: INTF: RB0/INT Kesme Bayrak Biti

1 = RB0/INT Kesmesi oldu.

0 = RB0/INT Kesmesi olmadı.

Bit 0: RBIF: RB Port Değişikliği Uyarı Kesme Biti

1 = RB7 – RB4 bitlerinden en az biri durum değiştirdi (Yazılımda temizlenmelidir).

0 = RB7 – RB4 bitlerinden en az biri durum değiştirmedir.

[12]

EECON1 (EEPROM) Yazmacı: (Adres 88h)

U - 0 U - 0 U - 0 R/W - 0 R/W - x R/W - 0 R/W - 0 R/S - 0

----	----	----	EEIF	WRERR	WREN	WR	RD
7	6	5	4	3	2	1	0

Bit 7: Kullanılmaz. '0' olarak okunur.

Bit 4: EEIF: EEPROM Yazma İşlemi Kesme Biti

1 = Yazma işlemi Biti(Yazılımda temizlenmelidir).

0 = Yazma işlemi başlamadı veya bitmedi.

Bit 3: WRERR: EEPROM Hata Bayrak Biti

1 = Yazma işlemi bitmeden kesildi.

0 = Yazma işlemi bitti.

Bit 2 : WREN EEPROM Yazma Biti

1 = Yazmaya izin ver.

0 = EEPROM'a yazmayı durdur.

Bit 1: WR: Yazma Kontrol Biti

1 = Yazmayı başlatır. (Yazma bitince bu bit donanım tarafından temizlenir.

WR biti sadece yazılım tarafından '1' yapılabilir.)

0 = EEPROM'a yazma bitmiştir.

Bit 0: RD: EEPROM Okuma Kontrol Biti

1 = EEPROM okumayı başlatır(RD donanım tarafından temizlenir. Bu bit sadece yazılım tarafından '1' yapılabilir.) .

0 = EEPROM okumayı başlatmaz.

[12]

PIC16F877 mikrodenetleyicisinin yazmaç haritası Çizelge 8.2'deki gibidir.

Çizelge 8.2 PIC16F877 Yazmaç Haritası [13]

Dosya Adresi	BANK 0	Dosya Adresi	BANK 1	Dosya Adresi	BANK 0	Dosya Adresi	BANK 0
00h	INDF	80h	INDF	100h	INDF	180h	INDF
01h	TMR0	81h	OPTION_REG	101h	TMR0	181h	OPTION_REG
02h	PCL	82h	PCL	102h	PCL	182h	PCL
03h	STATUS	83h	STATUS	103h	STATUS	183h	STATUS
04h	FSR	84h	FSR	104h	FSR	184h	FSR
05h	PORTA	85h	TRISA	105h	-----	185h	-----
06h	PORTB	86h	TRISB	106h	PORTB	186h	TRISB
07h	PORTC	87h	TRISC	107h	-----	187h	-----
08h	PORTD	88h	TRISD	108h	-----	188h	-----
09h	PORTE	89h	TRISE	109h	-----	189h	-----
0Ah	PCLATH	8Ah	PCLATH	10Ah	PCLATH	18Ah	PCLATH
0Bh	INTCON	8Bh	INTCON	10Bh	INTCON	18Bh	INTCON
0Ch	PIR1	8Ch	PIE1	10Ch	EEDATA	18Ch	EECON1
0Dh	PIR2	8Dh	PIE2	10Dh	EEADR	18Dh	EECON2
0Eh	TMR1L	8Eh	PCON	10Eh	EEADRH	18Eh	Rezerve
0Fh	TMR1H	8Fh	-----	10Fh	EEDATH	18Fh	Rezerve
10h	T1CON	90h	-----	110h	EEADRH	190h	Genel Amaçlı Yazmaçlar 16 Byte
11h	TMR2	91h	SSPCON2	111h	Genel Amaçlı Yazmaçlar 16 Byte	191h	
12h	T2CON	92h	PR2	112h		192h	
13h	SSPBUF	93h	SSPADDD	113h		193h	
14h	SSPCON	94h	SSPSTAT	114h		194h	
15h	CCPR1L	95h	-----	115h		195h	
16h	CCPR1H	96h	-----	116h		196h	
17h	CCP1CON	97h	-----	117h		197h	
18h	RCSTA	98h	TXSTA	118h		198h	
19h	TXREG	99h	SPBRG	119h		199h	
1Ah	RCREG	9Ah	-----	11Ah		19Ah	
1Bh	CCPR2L	9Bh	-----	11Bh		19Bh	
1Ch	CCPR2H	9Ch	-----	11Ch		19Ch	

1Dh	CCP2CON	9Dh	-----	11Dh		19Dh	
1Eh	ADRESH	9Eh	ADRESL	11Eh		19Eh	
1Fh	ADCON0	9Fh	ADCON1	11Fh		19Fh	
20h	Genel Amaçlı Yazmaçlar 96 Byte	A0h	Genel	120h	Genel	1A0h	Genel
...		...	Amaçlı	...	Amaçlı	...	Amaçlı
		EFh	Yazmaçlar 80 Byte	16Fh	Yazmaçlar 80 Byte	1EFh	Yazmaçlar 80 Byte
		F0h	70h – 7Fh	170h	70h – 7Fh	1F0h	70h – 7Fh
		...		...		...	
7Fh		FFh	Erişim	17Fh	Erişim	1FFh	Erişim

PIC 16F877A mikrodnetleyicilerini programlarken kullanılan belli başlı yazmaçların ayrıntılı açıklaması aşağıdaki gibidir. Yazmaç tanıtımlarında kullanılan; R: Okunabilen bit, W: Yazılabilen bit, U: İşlenemez bit (sıfır olarak okunur), -n: POR'deki değer, '1' : Bit set edildi, '0': Bit sıfırlandı, x: Bit bilinmiyor anlamına gelmektedir [13].

Status (Durum) Yazmacı: (Adres: 03h, 83h, 103h, 183h)

R/W – 0 R/W - 0 R/W – 0 R – 1 R – 1 R/W - x R/W - x R/W – x

IRP	RP1	RP0	TO	PD	Z	DC	C
7	6	5	4	3	2	1	0

Bit 7: IRP: Yazmaç Bank Adresleme (Dolaylı Adresleme).

0 = Bank 0, 1 (00h – FFh)

1 = Bank 2, 3 (100h – 1FFh)

Bit 6-5: RP0: Yazmaç Bank'ı Seçme Biti (Direkt adresleme için kullanılır.)

11 = Bank 3 (180h -1FFh)

10 = Bank 2 (100h – 17Fh)

01 = Bank 1 (80h – FFh)

00 = Bank 0 (00h – 7Fh)

Bit 4: $\overline{TO}$: Time Out Bitidir. (Zaman doldu.)

1 = Güç verildikten sonra CLRWDT komutu veya SLEEP komutu

0 = WDT time-out oluşmuştur.

Bit 3: $\overline{PD}$: Power-down bitidir. (Güç kesildi.)

1 = Güç verildikten sonra veya CLRWDT komutu ile

0 = SLEEP komutunun yerine getirilmesi ile birlikte

Bit2: Z: Sıfır Biti

1 = Bir aritmetik veya mantıksal işlemin sonucu '0' ise

0 = Bir aritmetik veya mantıksal işlemin sonucu '0' değil ise

Bit 1: DC: Digit Carry/ $\overline{\text{borrow}}$ (elde) Biti (ADDWF, ADDLW, SUBLW, SUBWF komutları) ($\overline{\text{borrow}}$, polarite tersinirdir.)

1 = Düşük dörtlü bit taşarsa 1 olur.

0 = Düşük dörtlü bit taşmazsa 0 olur.

Bit 0: C: Carry/ $\overline{\text{borrow}}$ (elde) Biti. (ADDWF, ADDLW, SUBLW, SUBWF komutları) ($\overline{\text{borrow}}$, polarite tersinirdir.)

1 = En duyarlı bit taşarsa 1 olur.

0 = En duyarlı bit taşmazsa 0 olur.

[13]

Option (Seenek) Yazmacı: (Adres: 81h, 181h)

R/W - 1	R/W - 1	R/W - 1	R/W - 1	R/W - 1	R/W - 1	R/W - 1	R/W - 1
RBPU	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0
7	6	5	4	3	2	1	0

Option yazmacının bitleri PIC16F84 ile aynıdır. [4]

INTCON (Kesme) Yazmacı: (Adres: 0Bh, 8Bh, 10Bh, 18Bh)

R/W - 0	R/W - 0	R/W - 0	R/W - 0	R/W - 0	R/W - 0	R/W - 0	R/W - x
GIE	EEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF
7	6	5	4	3	2	1	0

INTCON yazmacının bitleri PIC16F84 ile aynıdır. [13]

EECON1 (EEPROM) Yazmacı: (Adres 18Ch)

R/W - x	U - 0	U - 0	U - 0	R/W - x	R/W - 0	R/W - 0	R/S - 0
EEPGD	-----	-----	-----	WRERR	WREN	WR	RD
7	6	5	4	3	2	1	0

Bit 7: EEGD: Program /Data EEPROM Seme Biti.

Bit 6-4: Kullanılmaz. ‘0’ olarak okunur.

Bit 3: WRERR: EEPROM Hata Bayrak Biti

1 = Yazma işlemi bitmeden kesildi.

0 = Yazma işlemi bitti.

Bit 2 : WREN EEPROM Yazma Biti

1 = Yazmaya izin ver.

0 = EEPROM'a yazmayı durdur.

Bit 1: WR: Yazma Kontrol Biti

1 = Yazmayı başlatır. (Yazma bitince bu bit donanım tarafından temizlenir.

WR biti sadece yazılım tarafından '1' yapılabilir.)

0 = EEPROM'a yazma bitmiştir.

Bit 0: RD: EEPROM Okuma Kontrol Biti

1 = EEPROM okumayı başlatır(RD donanım tarafından temizlenir. Bu bit sadece yazılım tarafından '1' yapılabilir.) .

0 = EEPROM okumayı başlatmaz.

[13]

8.3 Yazılım ve Açıklaması

Bu çalışmada C ve Assembler dilleri kullanılmıştır. C ile yazılmış olan programlarda kodda da görüleceği üzere dahil edilen dosyalar da verilmiştir.

8.3.1 Delay.c

Bazı uygulamalarda "Delay.c" dosyası kullanılmıştır. Dosyanın içeriği aşağıdaki gibidir:

```
#include    "delay.h"

void
DelayMs(unsigned char cnt)
{
    #if    XTAL_FREQ <= 2MHZ
        do {
            DelayUs(996);
        } while(--cnt);
    #endif
}
```

```

#endif

#if XTAL_FREQ > 2MHZ
    unsigned char    i;
    do {
        i = 4;
        do {
            DelayUs(250);
        } while(--i);
    } while(--cnt);
#endif
}

```

Açıklama:

Doğal olarak delay.h dosyası C programlama diliyle yazılmıştır. Amacı C ile yazılan motor kontrol programlarındaki bekleme sürelerini regüle etmektir.

Kristal seçeneklerine göre iki durum için tanımlanmıştır. Bu durumlar karşısında yapılacaklar **#if#endif** bloklarında tanımlanmıştır. Buna göre Kristal frekansı eğer 2MHz’ten düşükse veya eşitse DelayUs fonksiyonuna ‘996’ değeri girilerek veya kristal frekansı 2 MHz’den büyükse DelayUs fonksiyonuna ‘250’ değeri girilip 4 defa döngüye sokularak ‘1’ milisaniyelik bekleme süresi üreten DelayMs(unsigned char cnt) fonksiyonu tanımlanmıştır.

8.3.2 Delay.h

Bu dosyada dahil edilen “delay.h” dosyası aşağıdaki gibidir:

```

#ifndef XTAL_FREQ
#define XTAL_FREQ 4MHZ /* Crystal frequency in MHz */
#endif

#define MHZ *1000L /* number of kHz in a MHz
*/
#define KHZ *1 /* number of kHz in a kHz */

#if XTAL_FREQ >= 12MHZ

```

```

#define      DelayUs(x)  { unsigned char _dcnt; \
                          _dcnt = (x)*((XTAL_FREQ)/(12MHZ)); \
                          while(--_dcnt != 0) \
                              continue; }

#else

#define      DelayUs(x)  { unsigned char _dcnt; \
                          _dcnt = (x)/((12MHZ)/(XTAL_FREQ))+1; \
                          while(--_dcnt != 0) \
                              continue; }

#endif

extern void DelayMs(unsigned char);

```

Açıklama:

Bu dosyada Delay.c dosyasının header dosyasıdır. İçinde DelayUs(x) tanımlıdır. Bu işlemler ile mikrosaniye kristal frekansı sayesinde hesaplanır. Bu sayede DelayMs fonksiyonunun 1 milisaniye bekleme üretmesi sağlanır.

8.3.3 Pic.h

Tüm C dili ile yapılmış uygulamalarda “pic.h” dosyası dahil edilmiştir. Bu dosyanın içeriği aşağıdaki gibidir:

```

#ifndef _PIC_H
#define _PIC_H

#if defined (_12F629)      ||      defined (_12F675)
    #include      <pic12f6x.h>
#endif

#if defined (_16F627)      ||      defined (_16F628)
    #include      <pic16f6x.h>
#endif

```

```

#if defined (_16C84) || \
    Defined (_16F84) || defined (_16F84A)
    #include <pic1684.h>
#endif

#if defined (_16F877)
    #include <pic1687x.h>
#endif

#if defined (_16F877A)
    #include <pic168xa.h>
#endif

#define CLRWDT()    asm(" clrwdt")
#define SLEEP()      asm(" sleep")
#define NOP()        asm("nop")

#define __mkstr1(x)    #x
#define __mkstr(x)      __mkstr1(x)
#define __CONFIG(x)    asm("\tpsect config,class=CONFIG,delta=2");\
                        asm("\tglobal\tconfig_word"); \
                        asm("config_word"); \
                        asm("\tdw " __mkstr(x))

#define __IDLOC(w)    asm("\tpsect idloc,class=IDLOC,delta=2");\
                        asm("\tglobal\tidloc_word"); \
                        asm("idloc_word"); \
                        asm("\tirpc\t__arg," __mkstr(w)); \
                        asm("\tdw 0&__arg&h"); \
                        asm("\tendm")

#if defined (EEPROM_SIZE)
#define __EEPROM_DATA(a, b, c, d, e, f, g, h) \

```

```

asm("\tpsect eeprom_data,class=EEDATA,delta=2"); \
asm("\t\t__mkstr(a) "," __mkstr(b) "," __mkstr(c) ","
\ __mkstr(d) "," __mkstr(e) "," __mkstr(f) "," __mkstr(g)
"," __mkstr(h))

#endif

// global interrupt enable macros

#ifndef ei
#define ei() (GIE = 1) // interrupt enable bit
#endif

#ifndef di
#define di() { do { GIE = 0; } while ( GIE == 1 ); } // disable
interrupt bit
#endif

#endif /* _PIC_H */

```

Açıklama:

Bu header dosyası, başlangıç ayarları ile ilgilidir. Çalışılan PIC mikrokontrolörüne göre dahil edilecek header dosyalarını belirtir. Ayrıca mikrokontroleri düzenleyen başlangıç işlemlerinin C içine asm implimentasyonu olarak yapılmasını sağlayan kodlar içerir.

8.3.4 Pic1684.h

Pic.h dosyasında eğer mantığı ile dahil edilen dosyalardan, bu çalışmada yer verilen uygulamalarda kullanılacak olan “pic1684.h” dosyasının içeriği şöyledir:

Pic1684.h:

```
/*
```

```

*   Header file for the Microchip
*   PIC 16CR83 chip
*   PIC 16F83 chip
*   PIC 16C84 chip
*   PIC 16F84 chip
*   PIC 16F84A chip
*   PIC 16CR84 chip
*   Midrange Microcontrollers
*/

static volatile unsigned char RTCC   @ 0x01;
static volatile unsigned char TMR0   @ 0x01;
static volatile unsigned char PCL    @ 0x02;
static volatile unsigned char STATUS @ 0x03;
static          unsigned char FSR    @ 0x04;
static volatile unsigned char PORTA  @ 0x05;
static volatile unsigned char PORTB  @ 0x06;
static volatile unsigned char EEDATA  @ 0x08;
static volatile unsigned char EEADR  @ 0x09;
static          unsigned char PCLATH @ 0x0A;
static volatile unsigned char INTCON @ 0x0B;

static          unsigned char bank1 OPTION @ 0x81;
static volatile unsigned char bank1 TRISA @ 0x85;
static volatile unsigned char bank1 TRISB @ 0x86;
static volatile unsigned char bank1 EECON1 @ 0x88;
static volatile unsigned char bank1 EECON2 @ 0x89;

/*   STATUS bits */
static volatile bit RP0 @ (unsigned)&STATUS*8+5;
static volatile bit TO  @ (unsigned)&STATUS*8+4;
static volatile bit PD  @ (unsigned)&STATUS*8+3;
static volatile bit ZERO @ (unsigned)&STATUS*8+2;
static volatile bit DC   @ (unsigned)&STATUS*8+1;
static volatile bit CARRY @ (unsigned)&STATUS*8+0;

/*   PORTA bits */
static volatile bit RA4 @ (unsigned)&PORTA*8+4;
static volatile bit RA3 @ (unsigned)&PORTA*8+3;
static volatile bit RA2 @ (unsigned)&PORTA*8+2;
static volatile bit RA1 @ (unsigned)&PORTA*8+1;

```

```

static volatile bit    RA0    @ (unsigned) &PORTA*8+0;

/*      PORTB bits      */
static volatile bit    RB7    @ (unsigned) &PORTB*8+7;
static volatile bit    RB6    @ (unsigned) &PORTB*8+6;
static volatile bit    RB5    @ (unsigned) &PORTB*8+5;
static volatile bit    RB4    @ (unsigned) &PORTB*8+4;
static volatile bit    RB3    @ (unsigned) &PORTB*8+3;
static volatile bit    RB2    @ (unsigned) &PORTB*8+2;
static volatile bit    RB1    @ (unsigned) &PORTB*8+1;
static volatile bit    RB0    @ (unsigned) &PORTB*8+0;
static volatile bit    INT    @ (unsigned) &PORTB*8+0;

/*      INTCON bits */
static volatile bit    GIE    @ (unsigned) &INTCON*8+7;
static volatile bit    EEIE   @ (unsigned) &INTCON*8+6;
static volatile bit    TOIE   @ (unsigned) &INTCON*8+5;
static volatile bit    INTE   @ (unsigned) &INTCON*8+4;
static volatile bit    RBIE   @ (unsigned) &INTCON*8+3;
static volatile bit    T0IF   @ (unsigned) &INTCON*8+2;
static volatile bit    INTF   @ (unsigned) &INTCON*8+1;
static volatile bit    RBIF   @ (unsigned) &INTCON*8+0;

/*      OPTION bits */
static bank1 bit    RBPU    @ (unsigned) &OPTION*8+7;
static bank1 bit    INTEDG    @ (unsigned) &OPTION*8+6;
static bank1 bit    T0CS    @ (unsigned) &OPTION*8+5;
static bank1 bit    T0SE    @ (unsigned) &OPTION*8+4;
static bank1 bit    PSA     @ (unsigned) &OPTION*8+3;
static bank1 bit    PS2     @ (unsigned) &OPTION*8+2;
static bank1 bit    PS1     @ (unsigned) &OPTION*8+1;
static bank1 bit    PS0     @ (unsigned) &OPTION*8+0;

/*      TRISA bits      */
static volatile bank1 bit    TRISA4    @ (unsigned) &TRISA*8+4;
static volatile bank1 bit    TRISA3    @ (unsigned) &TRISA*8+3;
static volatile bank1 bit    TRISA2    @ (unsigned) &TRISA*8+2;
static volatile bank1 bit    TRISA1    @ (unsigned) &TRISA*8+1;
static volatile bank1 bit    TRISA0    @ (unsigned) &TRISA*8+0;

/*      TRISB bits      */

```

```

static volatile bank1 bit    TRISB7    @ (unsigned)&TRISB*8+7;
static volatile bank1 bit    TRISB6    @ (unsigned)&TRISB*8+6;
static volatile bank1 bit    TRISB5    @ (unsigned)&TRISB*8+5;
static volatile bank1 bit    TRISB4    @ (unsigned)&TRISB*8+4;
static volatile bank1 bit    TRISB3    @ (unsigned)&TRISB*8+3;
static volatile bank1 bit    TRISB2    @ (unsigned)&TRISB*8+2;
static volatile bank1 bit    TRISB1    @ (unsigned)&TRISB*8+1;
static volatile bank1 bit    TRISB0    @ (unsigned)&TRISB*8+0;

/*    EECON1 bits */
static volatile bank1 bit    EEIF    @ (unsigned)&EECON1*8+4;
static volatile bank1 bit    WRERR    @ (unsigned)&EECON1*8+3;
static volatile bank1 bit    WREN    @ (unsigned)&EECON1*8+2;
static volatile bank1 bit    WR    @ (unsigned)&EECON1*8+1;
static volatile bank1 bit    RD    @ (unsigned)&EECON1*8+0;

#define EEPROM_SIZE    64

/* macro versions of EEPROM write and read */
#define    EEPROM_WRITE(addr, value)
while(WR) continue;EEADR=(addr);EEDATA=(value); \
        CARRY=0;if(GIE)CARRY=1;GIE=0; \

        WREN=1;EECON2=0x55;EECON2=0xaa;WR=1;WREN=0; \
        if (CARRY)GIE=1

#define    EEPROM_READ(addr) ((EEADR=(addr)),(RD=1),EEDATA)

#if defined(_16F83) || defined(_16F84) || defined(_16F84A)
/* macro versions of Flash read and write */
#define FLASH_WRITE(addr, value)    EEADR=((addr)&0xff); \
        EEADRH=((addr)>>8); \
        EEDATH=((value)>>8); \
        EEDATA=((value)&0xff); \
        WREN=1; \
        EECON2 = 0x55; \
        EECON2 = 0xaa; \
        WR=1; \
        asm("nop"); \
        asm("nop"); \
        WREN=0

```

```

#define FLASH_READ(addr)          EEADR=((addr)&0xff);    \
                                  EEADRH=((addr)>>8);      \
                                  WREN=0;                  \
                                  RD=1;                    \
                                  asm("nop");              \
                                  asm("nop");              \
                                  ((EEDATH << 8) | EEDATA)

#endif

/* library function versions */

extern void eeprom_write(unsigned char addr, unsigned char value);
extern unsigned char eeprom_read(unsigned char addr);

#define CONFIG_ADDR      0x2007

/*osc configurations*/
#define RC      0x3FFF      // resistor/capacitor
#define HS      0x3FFE      // high speed crystal/resonator
#define XT      0x3FFD      // crystal/resonator
#define LP      0x3FFC      // low power crystal/resonator

/*watchdog*/
#define WDTEN      0x3FFF      // enable watchdog timer
#define WDTDIS      0x3FFB      // disable watchdog timer

/*power up timer*/
#if defined (_16C84)
#define PWRTEN      0x3FFF      /* enable power up timer*/
#define PWRTDIS      0x3FF7      /* disable power up timer
*/
#else
#define PWRTDIS      0x3FFF      /* disable power up timer
*/
#define PWRTEN      0x3FF7      /* enable power up timer
*/
#endif

#if defined (_16CR83) || defined(_16CR84)
#define DP      0x3F7F      // data code is protected
#define PROTECT      0x008F      // program code is
protected

```

```
// alternative definition
#define DATPROT          0x3F7F          // use DP
#define DATUNPROT 0x3FFF          // use UNPROTECT
#elif defined (_16C84)
#define PROTECT          0x3FEF          // program code is
protected
#elif defined (_16F83) || defined(_16F84) || defined(_16F84A)
#define PROTECT          0x000F          // program code is
protected
#endif

#define UNPROTECT 0x3FFF          // do not protect the code
```

Açıklama:

Pic.h header dosyası tarafından da çağırılan bu header dosyası, PIC 16F84 için program yazmak istiyorsak doğal olarak dahil etmemiz gereken bir dosyadır. Bu header dosyasında, mikrokontrolörü programlarken kullandığımız terimlerin tanımları mevcuttur. Kullandığımız komutların mikrokontrolör hafızasındaki adresleri tanımlanmıştır. Bu header dosyası sayesinde direk adresler yerine o adreslere tanımlanmış değişkenler ile okunabilirliği çok daha iyileştirilmiş kodlar oluşturabilmekteyiz. Ayrıca hafızaya yazma ve hafızadan okuma yapan fonksiyonlarda bu dosyada tanımlanmıştır.

8.3.5 Pic1687x.h

Pic.h dosyasında eğer mantığı ile dahil edilen dosyalardan, bu çalışmada yer verilen uygulamalarda kullanılacak olan “pic1687x.h” dosyasının içeriği şöyledir:

Pic1687x.h:

```
/*
 *   Header file for the Microchip
 *   PIC 16F870
 *   PIC 16F871
 *   PIC 16F872 chip
 *   PIC 16F873 chip
 *   PIC 16F874 chip
 *   PIC 16F876 chip
```

```

*    PIC 16F877 chip
*    Midrange Microcontroller
*/

#if defined(_16F874)    || defined(_16F877) || defined(_16F871)
#define    __PINS_40
#endif

static volatile unsigned char INDF    @ 0x00;
static volatile unsigned char TMR0    @ 0x01;
static volatile unsigned char PCL     @ 0x02;
static volatile unsigned char STATUS  @ 0x03;
static          unsigned char FSR     @ 0x04;
static volatile unsigned char PORTA   @ 0x05;
static volatile unsigned char PORTB   @ 0x06;
static volatile unsigned char PORTC   @ 0x07;
#ifdef __PINS_40
static volatile unsigned char PORTD   @ 0x08;
static volatile unsigned char PORTE   @ 0x09;
#endif
static          unsigned char PCLATH   @ 0x0A;
static volatile unsigned char INTCON   @ 0x0B;
static volatile unsigned char PIR1    @ 0x0C;
static volatile unsigned char PIR2    @ 0x0D;
static volatile unsigned char TMR1L   @ 0x0E;
static volatile unsigned char TMR1H   @ 0x0F;
static volatile unsigned char T1CON   @ 0x10;
static volatile unsigned char TMR2    @ 0x11;
static volatile unsigned char T2CON   @ 0x12;
#if !defined(_16F870) && !defined(_16F871)
static volatile unsigned char SSPBUF   @ 0x13;
static volatile unsigned char SSPCON   @ 0x14;
#endif
static volatile unsigned char CCPR1L   @ 0x15;
static volatile unsigned char CCPR1H   @ 0x16;
static volatile unsigned char CCP1CON   @ 0x17;
#ifdef _16F872
static volatile unsigned char RCSTA    @ 0x18;
static volatile unsigned char TXREG    @ 0x19;
static volatile unsigned char RCREG    @ 0x1A;
#endif
#if !defined(_16F870) && !defined(_16F871)

```

```

static volatile unsigned char CCP2L      @ 0x1B;
static volatile unsigned char CCP2H      @ 0x1C;
static volatile unsigned char CCP2CON    @ 0x1D;
#endif
#endif

static volatile unsigned char ADRESH     @ 0x1E;
static volatile unsigned char ADCON0     @ 0x1F;

/*    bank 1 registers */
static          unsigned char bank1 OPTION      @ 0x81;
static volatile unsigned char bank1 TRISA @ 0x85;
static volatile unsigned char bank1 TRISB @ 0x86;
static volatile unsigned char bank1 TRISC @ 0x87;
#ifdef    __PINS_40
static volatile unsigned char bank1 TRISD @ 0x88;
static volatile unsigned char bank1 TRISE @ 0x89;
#endif

static volatile unsigned char bank1 PIE1 @ 0x8C;
static volatile unsigned char bank1 PIE2 @ 0x8D;
static volatile unsigned char bank1 PCON @ 0x8E;
#if !defined(_16F870) && !defined(_16F871)
static volatile unsigned char bank1 SSPCON2      @ 0x91;
#endif

static volatile unsigned char bank1 PR2 @ 0x92;
#if !defined(_16F870) && !defined(_16F871)
static volatile unsigned char bank1 SSPADD      @ 0x93;
static volatile unsigned char bank1 SSPSTAT      @ 0x94;
#endif
#endif

static volatile unsigned char bank1 TXSTA @ 0x98;
static volatile unsigned char bank1 SPBRG @ 0x99;
#endif

static volatile unsigned char bank1 ADRESL      @ 0x9E;
static volatile unsigned char bank1 ADCON1      @ 0x9F;

/*    bank 2 registers */
static volatile unsigned char bank2 EEDATA      @ 0x10C;
static volatile unsigned char bank2 EEADR @ 0x10D;
static volatile unsigned char bank2 EEDATH      @ 0x10E;
static volatile unsigned char bank2 EEADRH      @ 0x10F;

```

```

/*    bank 3 registers */
static volatile unsigned char bank3 EECON1      @ 0x18C;
static volatile unsigned char bank3 EECON2      @ 0x18D;

/*    STATUS bits */
static volatile bit        IRP      @ (unsigned)&STATUS*8+7;
static volatile bit        RP1      @ (unsigned)&STATUS*8+6;
static volatile bit        RP0      @ (unsigned)&STATUS*8+5;
static volatile bit        TO       @ (unsigned)&STATUS*8+4;
static volatile bit        PD       @ (unsigned)&STATUS*8+3;
static volatile bit        ZERO     @ (unsigned)&STATUS*8+2;
static volatile bit        DC       @ (unsigned)&STATUS*8+1;
static volatile bit        CARRY    @ (unsigned)&STATUS*8+0;

/*    PORTA bits */
static volatile bit        RA5      @ (unsigned)&PORTA*8+5;
static volatile bit        RA4      @ (unsigned)&PORTA*8+4;
static volatile bit        RA3      @ (unsigned)&PORTA*8+3;
static volatile bit        RA2      @ (unsigned)&PORTA*8+2;
static volatile bit        RA1      @ (unsigned)&PORTA*8+1;
static volatile bit        RA0      @ (unsigned)&PORTA*8+0;

/*    PORTB bits */
static volatile bit        RB7      @ (unsigned)&PORTB*8+7;
static volatile bit        RB6      @ (unsigned)&PORTB*8+6;
static volatile bit        RB5      @ (unsigned)&PORTB*8+5;
static volatile bit        RB4      @ (unsigned)&PORTB*8+4;
static volatile bit        RB3      @ (unsigned)&PORTB*8+3;
static volatile bit        RB2      @ (unsigned)&PORTB*8+2;
static volatile bit        RB1      @ (unsigned)&PORTB*8+1;
static volatile bit        RB0      @ (unsigned)&PORTB*8+0;

/*    PORTC bits */
static volatile bit        RC7      @ (unsigned)&PORTC*8+7;
static volatile bit        RC6      @ (unsigned)&PORTC*8+6;
static volatile bit        RC5      @ (unsigned)&PORTC*8+5;
static volatile bit        RC4      @ (unsigned)&PORTC*8+4;
static volatile bit        RC3      @ (unsigned)&PORTC*8+3;
static volatile bit        RC2      @ (unsigned)&PORTC*8+2;
static volatile bit        RC1      @ (unsigned)&PORTC*8+1;
static volatile bit        RC0      @ (unsigned)&PORTC*8+0;

```

```

/*      PORTD bits      */
#ifdef      __PINS_40
static volatile bit    RD7    @ (unsigned)&PORTD*8+7;
static volatile bit    RD6    @ (unsigned)&PORTD*8+6;
static volatile bit    RD5    @ (unsigned)&PORTD*8+5;
static volatile bit    RD4    @ (unsigned)&PORTD*8+4;
static volatile bit    RD3    @ (unsigned)&PORTD*8+3;
static volatile bit    RD2    @ (unsigned)&PORTD*8+2;
static volatile bit    RD1    @ (unsigned)&PORTD*8+1;
static volatile bit    RD0    @ (unsigned)&PORTD*8+0;

/*      PORTE bits      */
static volatile bit    RE2    @ (unsigned)&PORTE*8+2;
static volatile bit    RE1    @ (unsigned)&PORTE*8+1;
static volatile bit    RE0    @ (unsigned)&PORTE*8+0;
#endif

/*      INTCON bits */
static volatile bit    GIE    @ (unsigned)&INTCON*8+7;
static volatile bit    PEIE   @ (unsigned)&INTCON*8+6;
static volatile bit    T0IE   @ (unsigned)&INTCON*8+5;
static volatile bit    INTE   @ (unsigned)&INTCON*8+4;
static volatile bit    RBIE   @ (unsigned)&INTCON*8+3;
static volatile bit    T0IF   @ (unsigned)&INTCON*8+2;
static volatile bit    INTF   @ (unsigned)&INTCON*8+1;
static volatile bit    RBIF   @ (unsigned)&INTCON*8+0;
// alternate definitions
static volatile bit    TMR0IE    @ (unsigned)&INTCON*8+5;
static volatile bit    TMR0IF    @ (unsigned)&INTCON*8+2;

/*      PIR1 bits */
#ifdef      __PINS_40
static volatile bit    PSPIF @ (unsigned)&PIR1*8+7;
#endif
static volatile bit    ADIF   @ (unsigned)&PIR1*8+6;
#ifdef _16F872
static volatile bit    RCIF   @ (unsigned)&PIR1*8+5;
static volatile bit    TXIF   @ (unsigned)&PIR1*8+4;
#endif
#endif
#if !defined(_16F870) && !defined(_16F871)

```

```

static volatile bit    SSPIF @ (unsigned)&PIR1*8+3;
#endif

static volatile bit    CCP1IF @ (unsigned)&PIR1*8+2;
static volatile bit    TMR2IF @ (unsigned)&PIR1*8+1;
static volatile bit    TMR1IF @ (unsigned)&PIR1*8+0;

/*    PIR2 bits    */
static volatile bit    EEIF @ (unsigned)&PIR2*8+4;
#if !defined(_16F870) && !defined(_16F871)
static volatile bit    BCLIF @ (unsigned)&PIR2*8+3;
#endif
static volatile bit    CCP2IF @ (unsigned)&PIR2*8+0;
#endif
#endif

/*    T1CON bits    */
static volatile bit    T1CKPS1 @ (unsigned)&T1CON*8+5;
static volatile bit    T1CKPS0 @ (unsigned)&T1CON*8+4;
static volatile bit    T1OSCEN @ (unsigned)&T1CON*8+3;
static volatile bit    T1SYNC @ (unsigned)&T1CON*8+2;
static volatile bit    TMR1CS @ (unsigned)&T1CON*8+1;
static volatile bit    TMR1ON @ (unsigned)&T1CON*8+0;

/*    T2CON bits    */
static volatile bit    TOUTPS3 @ (unsigned)&T2CON*8+6;
static volatile bit    TOUTPS2 @ (unsigned)&T2CON*8+5;
static volatile bit    TOUTPS1 @ (unsigned)&T2CON*8+4;
static volatile bit    TOUTPS0 @ (unsigned)&T2CON*8+3;
static volatile bit    TMR2ON @ (unsigned)&T2CON*8+2;
static volatile bit    T2CKPS1 @ (unsigned)&T2CON*8+1;
static volatile bit    T2CKPS0 @ (unsigned)&T2CON*8+0;

/*    SSPCON bits    */
#if !defined(_16F870) && !defined(_16F871)
static volatile bit    WCOL @ (unsigned)&SSPCON*8+7;
static volatile bit    SSPOV @ (unsigned)&SSPCON*8+6;
static volatile bit    SSPEN @ (unsigned)&SSPCON*8+5;
static volatile bit    CKP @ (unsigned)&SSPCON*8+4;
static volatile bit    SSPM3 @ (unsigned)&SSPCON*8+3;
static volatile bit    SSPM2 @ (unsigned)&SSPCON*8+2;
static volatile bit    SSPM1 @ (unsigned)&SSPCON*8+1;

```

```

static volatile bit    SSPM0 @ (unsigned)&SSPCON*8+0;
#endif

/*    CCP1CON bits    */
static volatile bit    CCP1X @ (unsigned)&CCP1CON*8+5;
static volatile bit    CCP1Y @ (unsigned)&CCP1CON*8+4;
static volatile bit    CCP1M3 @ (unsigned)&CCP1CON*8+3;
static volatile bit    CCP1M2 @ (unsigned)&CCP1CON*8+2;
static volatile bit    CCP1M1 @ (unsigned)&CCP1CON*8+1;
static volatile bit    CCP1M0 @ (unsigned)&CCP1CON*8+0;

/*    RCSTA bits    */
#ifndef _16F872
static volatile bit    SPEN @ (unsigned)&RCSTA*8+7;
static volatile bit    RX9 @ (unsigned)&RCSTA*8+6;
static volatile bit    SREN @ (unsigned)&RCSTA*8+5;
static volatile bit    CREN @ (unsigned)&RCSTA*8+4;
static volatile bit    ADDEN @ (unsigned)&RCSTA*8+3;
static volatile bit    FERR @ (unsigned)&RCSTA*8+2;
static volatile bit    OERR @ (unsigned)&RCSTA*8+1;
static volatile bit    RX9D @ (unsigned)&RCSTA*8+0;
#endif

/*    CCP2CON bits    */
#if !defined(_16F870) && !defined(_16F871) && !defined(_16F872)
static volatile bit    CCP2X @ (unsigned)&CCP2CON*8+5;
static volatile bit    CCP2Y @ (unsigned)&CCP2CON*8+4;
static volatile bit    CCP2M3 @ (unsigned)&CCP2CON*8+3;
static volatile bit    CCP2M2 @ (unsigned)&CCP2CON*8+2;
static volatile bit    CCP2M1 @ (unsigned)&CCP2CON*8+1;
static volatile bit    CCP2M0 @ (unsigned)&CCP2CON*8+0;
#endif

/*    ADCON0 bits    */
static volatile bit    ADCS1 @ (unsigned)&ADCON0*8+7;
static volatile bit    ADCS0 @ (unsigned)&ADCON0*8+6;
static volatile bit    CHS2 @ (unsigned)&ADCON0*8+5;
static volatile bit    CHS1 @ (unsigned)&ADCON0*8+4;
static volatile bit    CHS0 @ (unsigned)&ADCON0*8+3;
static volatile bit    ADGO @ (unsigned)&ADCON0*8+2;
static volatile bit    ADON @ (unsigned)&ADCON0*8+0;

```

```

/*    OPTION bits */
static bank1 bit  RBPU  @ (unsigned)&OPTION*8+7;
static bank1 bit  INTEDG      @ (unsigned)&OPTION*8+6;
static bank1 bit  T0CS  @ (unsigned)&OPTION*8+5;
static bank1 bit  T0SE  @ (unsigned)&OPTION*8+4;
static bank1 bit  PSA   @ (unsigned)&OPTION*8+3;
static bank1 bit  PS2   @ (unsigned)&OPTION*8+2;
static bank1 bit  PS1   @ (unsigned)&OPTION*8+1;
static bank1 bit  PS0   @ (unsigned)&OPTION*8+0;

/*    TRISA bits    */
static volatile bank1 bit  TRISA5      @ (unsigned)&TRISA*8+5;
static volatile bank1 bit  TRISA4      @ (unsigned)&TRISA*8+4;
static volatile bank1 bit  TRISA3      @ (unsigned)&TRISA*8+3;
static volatile bank1 bit  TRISA2      @ (unsigned)&TRISA*8+2;
static volatile bank1 bit  TRISA1      @ (unsigned)&TRISA*8+1;
static volatile bank1 bit  TRISA0      @ (unsigned)&TRISA*8+0;

/*    TRISB bits    */
static volatile bank1 bit  TRISB7      @ (unsigned)&TRISB*8+7;
static volatile bank1 bit  TRISB6      @ (unsigned)&TRISB*8+6;
static volatile bank1 bit  TRISB5      @ (unsigned)&TRISB*8+5;
static volatile bank1 bit  TRISB4      @ (unsigned)&TRISB*8+4;
static volatile bank1 bit  TRISB3      @ (unsigned)&TRISB*8+3;
static volatile bank1 bit  TRISB2      @ (unsigned)&TRISB*8+2;
static volatile bank1 bit  TRISB1      @ (unsigned)&TRISB*8+1;
static volatile bank1 bit  TRISB0      @ (unsigned)&TRISB*8+0;

/*    TRISC bits    */
static volatile bank1 bit  TRISC7      @ (unsigned)&TRISC*8+7;
static volatile bank1 bit  TRISC6      @ (unsigned)&TRISC*8+6;
static volatile bank1 bit  TRISC5      @ (unsigned)&TRISC*8+5;
static volatile bank1 bit  TRISC4      @ (unsigned)&TRISC*8+4;
static volatile bank1 bit  TRISC3      @ (unsigned)&TRISC*8+3;
static volatile bank1 bit  TRISC2      @ (unsigned)&TRISC*8+2;
static volatile bank1 bit  TRISC1      @ (unsigned)&TRISC*8+1;
static volatile bank1 bit  TRISC0      @ (unsigned)&TRISC*8+0;

#ifdef    __PINS_40
/*    TRISD bits    */
static volatile bank1 bit  TRISD7      @ (unsigned)&TRISD*8+7;

```

```

static volatile bank1 bit TRISD6 @ (unsigned)&TRISD*8+6;
static volatile bank1 bit TRISD5 @ (unsigned)&TRISD*8+5;
static volatile bank1 bit TRISD4 @ (unsigned)&TRISD*8+4;
static volatile bank1 bit TRISD3 @ (unsigned)&TRISD*8+3;
static volatile bank1 bit TRISD2 @ (unsigned)&TRISD*8+2;
static volatile bank1 bit TRISD1 @ (unsigned)&TRISD*8+1;
static volatile bank1 bit TRISD0 @ (unsigned)&TRISD*8+0;

/*      TRISE bits      */
static volatile bank1 bit IBF @ (unsigned)&TRISE*8+7;
static volatile bank1 bit OBF @ (unsigned)&TRISE*8+6;
static volatile bank1 bit IBOV @ (unsigned)&TRISE*8+5;
static volatile bank1 bit PSPMODE @ (unsigned)&TRISE*8+4;

static volatile bank1 bit TRISE2 @ (unsigned)&TRISE*8+2;
static volatile bank1 bit TRISE1 @ (unsigned)&TRISE*8+1;
static volatile bank1 bit TRISE0 @ (unsigned)&TRISE*8+0;
#endif

/*      PIE1 bits      */
#ifdef __PINS_40
static volatile bank1 bit PSPIE @ (unsigned)&PIE1*8+7;
#endif
static volatile bank1 bit ADIE @ (unsigned)&PIE1*8+6;
#ifdef _16F872
static volatile bank1 bit RCIE @ (unsigned)&PIE1*8+5;
static volatile bank1 bit TXIE @ (unsigned)&PIE1*8+4;
#endif
static volatile bank1 bit SSPIE @ (unsigned)&PIE1*8+3;
static volatile bank1 bit CCP1IE @ (unsigned)&PIE1*8+2;
static volatile bank1 bit TMR2IE @ (unsigned)&PIE1*8+1;
static volatile bank1 bit TMR1IE @ (unsigned)&PIE1*8+0;

/*      PIE2 bits      */
static volatile bank1 bit EEIE @ (unsigned)&PIE2*8+4;
static volatile bank1 bit BCLIE @ (unsigned)&PIE2*8+3;
#ifdef _16F872
static volatile bank1 bit CCP2IE @ (unsigned)&PIE2*8+0;
#endif

/*      PCON bits      */

```

```

static volatile bank1 bit    POR    @ (unsigned)&PCON*8+1;
static volatile bank1 bit    BOR    @ (unsigned)&PCON*8+0;

/*    SSPCON2 bits */
#if !defined(_16F870) && !defined(_16F871)
static volatile bank1 bit    GCEN   @ (unsigned)&SSPCON2*8+7;
static volatile bank1 bit    ACKSTAT @ (unsigned)&SSPCON2*8+6;
static volatile bank1 bit    ACKDT  @ (unsigned)&SSPCON2*8+5;
static volatile bank1 bit    ACKEN  @ (unsigned)&SSPCON2*8+4;
static volatile bank1 bit    RCEN   @ (unsigned)&SSPCON2*8+3;
static volatile bank1 bit    PEN    @ (unsigned)&SSPCON2*8+2;
static volatile bank1 bit    RSEN   @ (unsigned)&SSPCON2*8+1;
static volatile bank1 bit    SEN    @ (unsigned)&SSPCON2*8+0;
#endif

/*    SSPSTAT bits    */
#if !defined(_16F870) && !defined(_16F871)
static volatile bank1 bit    STAT_SMP @ (unsigned)&SSPSTAT*8+7;
static volatile bank1 bit    STAT_CKE @ (unsigned)&SSPSTAT*8+6;
static volatile bank1 bit    STAT_DA   @
(unsigned)&SSPSTAT*8+5;
static volatile bank1 bit    STAT_P    @
(unsigned)&SSPSTAT*8+4;
static volatile bank1 bit    STAT_S    @
(unsigned)&SSPSTAT*8+3;
static volatile bank1 bit    STAT_RW   @
(unsigned)&SSPSTAT*8+2;
static volatile bank1 bit    STAT_UA   @
(unsigned)&SSPSTAT*8+1;
static volatile bank1 bit    STAT_BF   @
(unsigned)&SSPSTAT*8+0;
#endif

/*    TXSTA bits    */
#ifndef _16F872
static volatile bank1 bit    CSRC  @ (unsigned)&TXSTA*8+7;
static volatile bank1 bit    TX9   @ (unsigned)&TXSTA*8+6;
static volatile bank1 bit    TXEN  @ (unsigned)&TXSTA*8+5;
static volatile bank1 bit    SYNC  @ (unsigned)&TXSTA*8+4;
static volatile bank1 bit    BRGH  @ (unsigned)&TXSTA*8+2;
static volatile bank1 bit    TRMT  @ (unsigned)&TXSTA*8+1;

```

```

static volatile bank1 bit    TX9D    @ (unsigned)&TXSTA*8+0;
#endif

/*    ADCON1 bits */
static volatile bank1 bit    ADFM    @ (unsigned)&ADCON1*8+7;
static volatile bank1 bit    PCFG3    @ (unsigned)&ADCON1*8+3;
static volatile bank1 bit    PCFG2    @ (unsigned)&ADCON1*8+2;
static volatile bank1 bit    PCFG1    @ (unsigned)&ADCON1*8+1;
static volatile bank1 bit    PCFG0    @ (unsigned)&ADCON1*8+0;

/*    EECON1 bits */
static volatile bank3 bit    EEPGD    @ (unsigned)&EECON1*8+7;
static volatile bank3 bit    WRERR    @ (unsigned)&EECON1*8+3;
static volatile bank3 bit    WREN     @ (unsigned)&EECON1*8+2;
static volatile bank3 bit    WR       @ (unsigned)&EECON1*8+1;
static volatile bank3 bit    RD       @ (unsigned)&EECON1*8+0;

#if defined(_16F870) || defined(_16F871) || defined(_16F872)
#define EEPROM_SIZE        64
#elif defined(_16F873) || defined(_16F874)
#define EEPROM_SIZE        128
#else
#define EEPROM_SIZE        256
#endif

/* macro versions of EEPROM write and read */
#define      EEPROM_WRITE(addr, value)
while(WR)continue;EEADR=(addr);EEDATA=(value); \

        EECON1&=0x7F;CARRY=0;if (GIE)CARRY=1;GIE=0; \

        WREN=1;EECON2=0x55;EECON2=0xAA;WR=1;WREN=0; \
        if (CARRY)GIE=1

#define      EEPROM_READ(addr)
((EEADR=(addr)), (EECON1&=0x7F), (RD=1), EEDATA)

/* macro versions of Flash read and write */
#define FLASH_WRITE(addr, value)    EEADR=((addr)&0xff); \
        EEADRH=((addr)>>8); \
        EEDATH=((value)>>8); \

```

```

        EEDATA=((value)&0xff); \
        EEPGD = 1; \
        WREN=1; \
        EECON2 = 0x55; \
        EECON2 = 0xaa; \
        WR=1; \
        asm("nop"); \
        asm("nop"); \
        WREN=0

#define FLASH_READ(addr)          EEADR=((addr)&0xff); \
        EEADRH=((addr)>>8); \
        WREN=0; \
        EEPGD = 1; \
        RD=1; \
        asm("nop"); \
        asm("nop"); \
        ((EEDATH << 8) | EEDATA)

/* library function versions */

extern void eeprom_write(unsigned char addr, unsigned char value);
extern unsigned char eeprom_read(unsigned char addr);

#define CONFIG_ADDR      0x2007

/*osc configurations*/
#define RC      0x3FFF      // resistor/capacitor
#define HS      0x3FFE      // high speed crystal/resonator
#define XT      0x3FFD      // crystal/resonator
#define LP      0x3FFC      // low power crystal/resonator

/*watchdog*/
#define WDTEN      0x3FFF      // enable watchdog timer
#define WDTDIS      0x3FFB      // disable watchdog timer

/*power up timer*/
#define PWRTEN      0x3FF7      // enable power up timer
#define PWRTDIS      0x3FFF      // disable power up timer

/*brown out reset*/

```

```

#define BOREN            0x3FFF        // enable brown out reset
#define BORDIS           0x3FBF        // disable brown out reset

/*Low Voltage Programmable*/
#define LVPEN            0x3FFF        // low voltage programming
enabled
#define LVPDIS           0x3F7F        // low voltage programming
disabled

/*data code protected*/
#define DP               0x3EFF        // protect data code
// alternately
#define DPROT            0x3EFF        // use DP
#define DUNPROT          0x3FFF        // use UNPROTECT

/* Flash memory write enable/protect */
#define WRTEN            0x3FFF        /* flash memory write enabled */
#define WRTDIS           0x3DFF        /* flash memory write
protected/disabled */

/*debug option*/
#define DEBUGEN          0x37FF        // debugger enabled
#define DEBUGDIS 0x3FFF        // debugger disabled

/*code protection*/
#define PROTECT          0x0FCF        /* protect program code */
#define UNPROTECT 0x3FFF        /* do not protect the code */

#if !defined(_16F870) && !defined(_16F871)
#define PROTECT_TOP 0x2FEF        /* protects top 256 bytes, 1F00h
to 1FFFh */
#define PROTECT50 0x1FDF        /* protects upper half of address
space */
/* 1000h to 1FFFh (pic16f887/876) */
/* 0800h to 0FFFh (pic16f874/873) */
#endif
#endif

```

Açıklama:

Pic.h header dosyası tarafından da çağırılan bu header dosyası, PIC 16F877 için program yazmak istiyorsak doğal olarak dahil etmemiz gereken bir dosyadır. Bu

header dosyasında, mikrokontrolörü programlarken kullandığımız terimlerin tanımları mevcuttur. Kullandığımız komutların mikrokontrolör hafızasındaki adresleri tanımlanmıştır. Bu header dosyası sayesinde direk adresler yerine o adreslere tanımlanmış değişkenler ile okunabilirliği çok daha iyileştirilmiş kodlar oluşturabilmekteyiz. Ayrıca hafızaya yazma ve hafızadan okuma yapan fonksiyonlarda bu dosyada tanımlanmıştır.

8.3.6 P16877A.inc

Assembler dili ile yazılmış olan uygulama dosyalarının içine dâhil edilmiş dosyalardan P16877A.inc'in içeriği şöyledir:

LIST

; P16F877A.INC Standard Header File, Version 1.00 Microchip Technology, Inc.

NOLIST

; This header file defines configurations, registers, and other useful bits of
; information for the PIC16F877A microcontroller. These names are taken to match
; the data sheets as closely as possible.

; Note that the processor must be selected before this file is
; included. The processor may be selected the following ways:

- ; 1. Command line switch:
- ; C:\MPASM MYFILE.ASM /PIC16F877A
- ; 2. LIST directive in the source file
- ; LIST P=PIC16F877A
- ; 3. Processor Type entry in the MPASM full-screen interface

=====

;

; Revision History

```

;
;=====

;Rev:  Date:  Reason:
;1.02  05/28/02 Corrected values for _CP_ALL and _CP_OFF in Configuration Bits
section.
;1.01  09/13/01 Added the PIR2 bit CMIF and the PIE2 bit CMIE
;1.00  04/19/01 Initial Release (BD - generated from PIC16F877.inc)

;=====
;
;   Verify Processor
;
;=====

    IFNDEF __16F877A
        MESSG "Processor-header file mismatch.  Verify selected processor."
    ENDIF

;=====
;
;   Register Definitions
;
;=====

W          EQU   H'0000'
F          EQU   H'0001'

;----- Register Files-----

INDF          EQU   H'0000'
TMR0          EQU   H'0001'
PCL           EQU   H'0002'
STATUS        EQU   H'0003'

```

FSR	EQU	H'0004'
PORTA	EQU	H'0005'
PORTB	EQU	H'0006'
PORTC	EQU	H'0007'
PORTD	EQU	H'0008'
PORTE	EQU	H'0009'
PCLATH	EQU	H'000A'
INTCON	EQU	H'000B'
PIR1	EQU	H'000C'
PIR2	EQU	H'000D'
TMR1L	EQU	H'000E'
TMR1H	EQU	H'000F'
T1CON	EQU	H'0010'
TMR2	EQU	H'0011'
T2CON	EQU	H'0012'
SSPBUF	EQU	H'0013'
SSPCON	EQU	H'0014'
CCPR1L	EQU	H'0015'
CCPR1H	EQU	H'0016'
CCP1CON	EQU	H'0017'
RCSTA	EQU	H'0018'
TXREG	EQU	H'0019'
RCREG	EQU	H'001A'
CCPR2L	EQU	H'001B'
CCPR2H	EQU	H'001C'
CCP2CON	EQU	H'001D'
ADRESH	EQU	H'001E'
ADCON0	EQU	H'001F'
OPTION_REG	EQU	H'0081'
TRISA	EQU	H'0085'
TRISB	EQU	H'0086'
TRISC	EQU	H'0087'
TRISD	EQU	H'0088'

TRISE	EQU	H'0089'
PIE1	EQU	H'008C'
PIE2	EQU	H'008D'
PCON	EQU	H'008E'
SSPCON2	EQU	H'0091'
PR2	EQU	H'0092'
SSPADD	EQU	H'0093'
SSPSTAT	EQU	H'0094'
TXSTA	EQU	H'0098'
SPBRG	EQU	H'0099'
CMCON	EQU	H'009C'
CVRCON	EQU	H'009D'
ADRESL	EQU	H'009E'
ADCON1	EQU	H'009F'
EEDATA	EQU	H'010C'
EEADR	EQU	H'010D'
EEDATH	EQU	H'010E'
EEADRH	EQU	H'010F'
EECON1	EQU	H'018C'
EECON2	EQU	H'018D'

;----- STATUS Bits -----

IRP	EQU	H'0007'
RP1	EQU	H'0006'
RP0	EQU	H'0005'
NOT_TO	EQU	H'0004'
NOT_PD	EQU	H'0003'
Z	EQU	H'0002'
DC	EQU	H'0001'
C	EQU	H'0000'

;----- INTCON Bits -----

GIE	EQU	H'0007'
PEIE	EQU	H'0006'
T0IE	EQU	H'0005'
TMR0IE	EQU	H'0005'
INTE	EQU	H'0004'
RBIE	EQU	H'0003'
T0IF	EQU	H'0002'
TMR0IF	EQU	H'0002'
INTF	EQU	H'0001'
RBIF	EQU	H'0000'

;----- PIR1 Bits -----

PSPIF	EQU	H'0007'
ADIF	EQU	H'0006'
RCIF	EQU	H'0005'
TXIF	EQU	H'0004'
SSPIF	EQU	H'0003'
CCP1IF	EQU	H'0002'
TMR2IF	EQU	H'0001'
TMR1IF	EQU	H'0000'

;----- PIR2 Bits -----

CMIF	EQU	H'0006'
EEIF	EQU	H'0004'
BCLIF	EQU	H'0003'
CCP2IF	EQU	H'0000'

;----- T1CON Bits -----

T1CKPS1	EQU	H'0005'
---------	-----	---------

```

T1CKPS0      EQU   H'0004'
T1OSCEN      EQU   H'0003'
NOT_T1SYNC   EQU   H'0002'
T1INSYNC     EQU   H'0002' ; Backward compatibility only
T1SYNC       EQU   H'0002'
TMR1CS       EQU   H'0001'
TMR1ON       EQU   H'0000'

```

```

;----- T2CON Bits -----

```

```

TOUTPS3      EQU   H'0006'
TOUTPS2      EQU   H'0005'
TOUTPS1      EQU   H'0004'
TOUTPS0      EQU   H'0003'
TMR2ON       EQU   H'0002'
T2CKPS1      EQU   H'0001'
T2CKPS0      EQU   H'0000'

```

```

;----- SSPCON Bits -----

```

```

WCOL         EQU   H'0007'
SSPOV        EQU   H'0006'
SSPEN        EQU   H'0005'
CKP          EQU   H'0004'
SSPM3        EQU   H'0003'
SSPM2        EQU   H'0002'
SSPM1        EQU   H'0001'
SSPM0        EQU   H'0000'

```

```

;----- CCP1CON Bits -----

```

```

CCP1X        EQU   H'0005'
CCP1Y        EQU   H'0004'
CCP1M3       EQU   H'0003'

```

```

CCP1M2      EQU   H'0002'
CCP1M1      EQU   H'0001'
CCP1M0      EQU   H'0000'

```

```

;----- RCSTA Bits -----

```

```

SPEN        EQU   H'0007'
RX9         EQU   H'0006'
RC9         EQU   H'0006' ; Backward compatibility only
NOT_RC8     EQU   H'0006' ; Backward compatibility only
RC8_9       EQU   H'0006' ; Backward compatibility only
SREN        EQU   H'0005'
CREN        EQU   H'0004'
ADDEN       EQU   H'0003'
FERR        EQU   H'0002'
OERR        EQU   H'0001'
RX9D        EQU   H'0000'
RCD8        EQU   H'0000' ; Backward compatibility only

```

```

;----- CCP2CON Bits -----

```

```

CCP2X       EQU   H'0005'
CCP2Y       EQU   H'0004'
CCP2M3      EQU   H'0003'
CCP2M2      EQU   H'0002'
CCP2M1      EQU   H'0001'
CCP2M0      EQU   H'0000'

```

```

;----- ADCON0 Bits -----

```

```

ADCS1       EQU   H'0007'
ADCS0       EQU   H'0006'
CHS2        EQU   H'0005'
CHS1        EQU   H'0004'

```

```

CHS0          EQU  H'0003'
GO            EQU  H'0002'
NOT_DONE      EQU  H'0002'
GO_DONE       EQU  H'0002'
ADON          EQU  H'0000'

```

```

;----- OPTION_REG Bits -----

```

```

NOT_RBPU      EQU  H'0007'
INTEDG        EQU  H'0006'
T0CS          EQU  H'0005'
T0SE          EQU  H'0004'
PSA           EQU  H'0003'
PS2           EQU  H'0002'
PS1           EQU  H'0001'
PS0           EQU  H'0000'

```

```

;----- TRISE Bits -----

```

```

IBF           EQU  H'0007'
OBF           EQU  H'0006'
IBOV          EQU  H'0005'
PSPMODE       EQU  H'0004'
TRISE2        EQU  H'0002'
TRISE1        EQU  H'0001'
TRISE0        EQU  H'0000'

```

```

;----- PIE1 Bits -----

```

```

PSPIE         EQU  H'0007'
ADIE          EQU  H'0006'
RCIE          EQU  H'0005'
TXIE          EQU  H'0004'
SSPIE         EQU  H'0003'

```

```

CCP1IE          EQU   H'0002'
TMR2IE          EQU   H'0001'
TMR1IE          EQU   H'0000'

```

```

;----- PIE2 Bits -----

```

```

CMIE            EQU   H'0006'
EEIE            EQU   H'0004'
BCLIE           EQU   H'0003'
CCP2IE          EQU   H'0000'

```

```

;----- PCON Bits -----

```

```

NOT_POR         EQU   H'0001'
NOT_BO          EQU   H'0000'
NOT_BOR         EQU   H'0000'

```

```

;----- SSPCON2 Bits -----

```

```

GCEN            EQU   H'0007'
ACKSTAT         EQU   H'0006'
ACKDT           EQU   H'0005'
ACKEN           EQU   H'0004'
RCEN            EQU   H'0003'
PEN             EQU   H'0002'
RSEN            EQU   H'0001'
SEN             EQU   H'0000'

```

```

;----- SSPSTAT Bits -----

```

```

SMP             EQU   H'0007'
CKE             EQU   H'0006'
D               EQU   H'0005'
I2C_DATA        EQU   H'0005'

```

```

NOT_A          EQU  H'0005'
NOT_ADDRESS    EQU  H'0005'
D_A           EQU  H'0005'
DATA_ADDRESS   EQU  H'0005'
P             EQU  H'0004'
I2C_STOP      EQU  H'0004'
S            EQU  H'0003'
I2C_START     EQU  H'0003'
R            EQU  H'0002'
I2C_READ      EQU  H'0002'
NOT_W         EQU  H'0002'
NOT_WRITE     EQU  H'0002'
R_W          EQU  H'0002'
READ_WRITE    EQU  H'0002'
UA           EQU  H'0001'
BF           EQU  H'0000'

```

;----- TXSTA Bits -----

```

CSRC          EQU  H'0007'
TX9           EQU  H'0006'
NOT_TX8       EQU  H'0006' ; Backward compatibility only
TX8_9        EQU  H'0006' ; Backward compatibility only
TXEN         EQU  H'0005'
SYNC         EQU  H'0004'
BRGH         EQU  H'0002'
TRMT         EQU  H'0001'
TX9D         EQU  H'0000'
TXD8         EQU  H'0000' ; Backward compatibility only

```

;----- CMCON Bits -----

```

C2OUT        EQU  H'0007'
C1OUT        EQU  H'0006'

```

C2INV	EQU	H'0005'
C1INV	EQU	H'0004'
CIS	EQU	H'0003'
CM2	EQU	H'0002'
CM1	EQU	H'0001'
CM0	EQU	H'0000'

;----- CVRCON Bits -----

CVREN	EQU	H'0007'
CVROE	EQU	H'0006'
CVRR	EQU	H'0005'
CVR3	EQU	H'0003'
CVR2	EQU	H'0002'
CVR1	EQU	H'0001'
CVR0	EQU	H'0000'

;----- ADCON1 Bits -----

ADFM	EQU	H'0007'
PCFG3	EQU	H'0003'
PCFG2	EQU	H'0002'
PCFG1	EQU	H'0001'
PCFG0	EQU	H'0000'

;----- EECON1 Bits -----

EEPGD	EQU	H'0007'
WRERR	EQU	H'0003'
WREN	EQU	H'0002'
WR	EQU	H'0001'
RD	EQU	H'0000'

;=====

==

```

;
;   RAM Definition
;
;=====
==

__MAXRAM H'1FF'
__BADRAM H'8F'-H'90', H'95'-H'97', H'9A'-H'9B'
__BADRAM H'105', H'107'-H'109'
__BADRAM H'185', H'187'-H'189', H'18E'-H'18F'

;=====
==

;
;   Configuration Bits
;
;=====
==

_CP_ALL          EQU   H'1FFF'
_CP_OFF          EQU   H'3FFF'
_DEBUG_OFF       EQU   H'3FFF'
_DEBUG_ON        EQU   H'37FF'
_WRT_OFF         EQU   H'3FFF' ; No prog memmory write protection
_WRT_256         EQU   H'3DFF' ; First 256 prog memmory write
protected
_WRT_1FOURTH     EQU   H'3BFF' ; First quarter prog memmory write
protected
_WRT_HALF        EQU   H'39FF' ; First half memmory write protected
_CPD_OFF         EQU   H'3FFF'
_CPD_ON          EQU   H'3EFF'
_LVP_ON          EQU   H'3FFF'
_LVP_OFF         EQU   H'3F7F'
_BODEN_ON        EQU   H'3FFF'

```

```

_BODEN_OFF      EQU   H'3FBF'
_PWRTE_OFF      EQU   H'3FFF'
_PWRTE_ON       EQU   H'3FF7'
_WDT_ON         EQU   H'3FFF'
_WDT_OFF        EQU   H'3FFB'
_RC_OSC         EQU   H'3FFF'
_HS_OSC         EQU   H'3FFE'
_XT_OSC         EQU   H'3FFD'
_LP_OSC         EQU   H'3FFC'

```

LIST

Açıklama:

Assembler dilini kullanarak yazdığımız programlarda bulunan bazı sembollerin hafıza üzerindeki yerleri tanımlanmıştır. Bu include dosyasında PIC 16F877A mikrodenetleyicisini programlarken kullanacağımız tüm tamınlar mevcuttur. Bu header dosyası bizi PIC 16F877A programlarken hafıza adreslerini direk olarak kodda kullanma zorunululuğunu kaldırmaktadır. Bu sebeple kodun okunabilirliği çok önemli bir miktarda artmaktadır. Özellikle yazmaç bitlerinin ayrı ayrı tanımlanması kod yazmayıda oldukça kolaylaştırmıştır.

8.3.7 P16F84A.inc

“P16F84A.inc” Dosyasının içeriği şöyledir:

LIST

```
; P16F84A.INC Standard Header File, Version 2.00 Microchip Technology, Inc.
```

NOLIST

```
; This header file defines configurations, registers, and other useful bits of
; information for the PIC16F84 microcontroller. These names are taken to match
; the data sheets as closely as possible.
```

; Note that the processor must be selected before this file is
 ; included. The processor may be selected the following ways:

- ; 1. Command line switch:
- ; C:\MPASM MYFILE.ASM /PIC16F84A
- ; 2. LIST directive in the source file
- ; LIST P=PIC16F84A
- ; 3. Processor Type entry in the MPASM full-screen interface

```

=====
;
; Revision History
;
=====

```

;Rev: Date: Reason:

;1.00 2/15/99 Initial Release

```

=====
;
; Verify Processor
;
=====

```

IFNDEF __16F84A

 MESSG "Processor-header file mismatch. Verify selected processor."

ENDIF

```

=====
;
; Register Definitions
;

```

```
=====
```

```
W          EQU  H'0000'
```

```
F          EQU  H'0001'
```

```
----- Register Files-----
```

```
INDF          EQU  H'0000'
```

```
TMR0          EQU  H'0001'
```

```
PCL           EQU  H'0002'
```

```
STATUS        EQU  H'0003'
```

```
FSR           EQU  H'0004'
```

```
PORTA         EQU  H'0005'
```

```
PORTB         EQU  H'0006'
```

```
EEDATA        EQU  H'0008'
```

```
EEADR         EQU  H'0009'
```

```
PCLATH        EQU  H'000A'
```

```
INTCON        EQU  H'000B'
```

```
OPTION_REG    EQU  H'0081'
```

```
TRISA         EQU  H'0085'
```

```
TRISB         EQU  H'0086'
```

```
EECON1        EQU  H'0088'
```

```
EECON2        EQU  H'0089'
```

```
----- STATUS Bits -----
```

```
IRP           EQU  H'0007'
```

```
RP1           EQU  H'0006'
```

```
RP0           EQU  H'0005'
```

```
NOT_TO        EQU  H'0004'
```

```
NOT_PD        EQU  H'0003'
```

```
Z             EQU  H'0002'
```

```
DC            EQU  H'0001'
```

C EQU H'0000'

;----- INTCON Bits -----

GIE EQU H'0007'
 EEIE EQU H'0006'
 T0IE EQU H'0005'
 INTE EQU H'0004'
 RBIE EQU H'0003'
 T0IF EQU H'0002'
 INTF EQU H'0001'
 RBIF EQU H'0000'

;----- OPTION_REG Bits -----

NOT_RBPU EQU H'0007'
 INTEDG EQU H'0006'
 T0CS EQU H'0005'
 T0SE EQU H'0004'
 PSA EQU H'0003'
 PS2 EQU H'0002'
 PS1 EQU H'0001'
 PS0 EQU H'0000'

;----- EECON1 Bits -----

EEIF EQU H'0004'
 WRERR EQU H'0003'
 WREN EQU H'0002'
 WR EQU H'0001'
 RD EQU H'0000'

;=====

;

```

;   RAM Definition
;
;=====

__MAXRAM H'CF'
__BADRAM H'07', H'50'-H'7F', H'87'

;=====
;
;   Configuration Bits
;
;=====

_CP_ON      EQU  H'000F'
_CP_OFF     EQU  H'3FFF'
_PWRTE_ON   EQU  H'3FF7'
_PWRTE_OFF  EQU  H'3FFF'
_WDT_ON     EQU  H'3FFF'
_WDT_OFF    EQU  H'3FFB'
_LP_OSC     EQU  H'3FFC'
_XT_OSC     EQU  H'3FFD'
_HS_OSC     EQU  H'3FFE'
_RC_OSC     EQU  H'3FFF'

LIST

```

Açıklama:

Assembler dilini kullanarak yazdığımız programlarda blunan bazı sembollerin hafıza üzerindeki yerleri tanımlanmıştır. Bu include dosyasında PIC 16F84A mikrokontrolörlerini programlar kullanacağımız tüm tamınlar mevcuttur. Bu header dosyası bizi PIC 16F84A programlarken hafıza adreslerini direk olarak kodda kullanma zorunululuğunu kaldırmaktadır. Bu sebeple kodun okunabilirliği çok

önemli bir miktarda artmaktadır. Özellikle yazmaç bitlerinin ayrı ayrı tanımlanması kod yazmayıda oldukça kolaylaştırmıştır.

8.3.8 Servo2.asm

Bu motorun kontrolü assembler programlama diliyle yazılmıştır. İçeriği verilmiş olan dosya “servo2.asm” ‘dir.

```
;*****
;
;          SERVO-MOTOR
;    PIC 16F877 ile RC Servo Motor Kontrolu
;
;*****

list    p = 16F877, R = DEC

#include    <P16F877.inc>

__CONFIG    (_WDT_OFF & _XT_OSC & _PWRTE_ON & _CP_OFF)

SAYAC equ    0x21    ; Dongu degiskenleri (
SAYAC1    equ    0x22
SAYAC2    equ    0x23
SAYAC3    equ    0x24

org        0x000
goto    Basla

Ilk_Islemler            ;BASLANGIC ISLEMLERI

;+++++++Sonradan Eklenen Bolum+++++++
    clrf    PORTA        ;PortA temizlendi
;+++++++
    clrf    PORTB        ; PORTB yazmaci temizlendi
    bcf     STATUS, RP1 ; Bank1'e gecme islemi
    bsf     STATUS, RP0 ; Bank1'e gecme islemi
;+++++++Sonrdan Eklenen Bolum+++++++
    clrf    TRISA
    movlw   B'00000111'        ;PortA 0,1,2 artik giris
```

```

        movwf TRISA
;+++++
        clrf  TRISB      ; B portu cikis yapmak icin sifirlandi

        bcf      STATUS, RP0 ; Bank0'a gecme islemi

        return          ;ilk islemler fonksiyonu bitisi
;+++++
Sinyal_1msn          ;1 MSN'LIK SINYAL URETIMI (Tum Fonksiyon
Darbenin Tamami)
        Altyordam

        movlw 40          ;SAYAC1'e 40 dec sayisinin aktarilmasi
        movwf SAYAC1
        bsf      PORTB, 1    ; RB-1 portunun 1'e cekilerek servoya
sinyal goderilmesi

D_1msn1
        movlw      40
        movwf      SAYAC2

D_1msn2

        decfsz      SAYAC2, F
        goto        D_1msn2
        decfsz      SAYAC1, F
        goto        D_1msn1          ;5000x0.2 = 1msn(0.2 = 0.05 x 4)

        bcf      PORTB, 1

        return
;+++++Sonradan Eklendi+++++
Sinyal_1.2msn          ;1.2 MSN'LIK SINYAL URETIMI (Tum Fonksiyon
Darbenin Tamami)
        Altyordam1.2

        movlw 48          ;SAYAC1'e 48 dec sayisinin aktarilmasi
        movwf SAYAC1
        bsf      PORTB, 1    ; B-1 portunun 1'e cekilerek servoya
sinyal goderilmesi

```

```

nop          ;1.2 msn'ye tamamlamak icin
nop
nop
nop
nop
nop
nop
nop
nop
nop
nop

D_1.2msn1
movlw        40
movwf        SAYAC2

D_1.2msn2

decfsz        SAYAC2, F
goto         D_1.2msn2
decfsz        SAYAC1, F
goto         D_1.2msn1          ;5000x0.2 = 1 msn

bcf          PORTB, 1

return

;+++++++Sonradan Eklendi+++++++
Sinyal_2msn          ;2 MSN'LIK SINYAL URETIMI (Tum Fonksiyon
Darbenin Tamami)
    Altyordam2

    movlw 40          ;SAYAC1'e 40 dec sayisinin aktarilmasi
    movwf SAYAC1
    bsf    PORTB, 1    ; B-1 portunun 1'e cekilerek servoya
sinyal goderilmesi

D_2msn0
movlw        2
movwf        SAYAC3

```

```

D_2msn1
    movlw    40
    movwf    SAYAC2

D_2msn2

    decfsz    SAYAC2, F
    goto     D_2msn2
    decfsz    SAYAC1, F
    goto     D_2msn1        ;5000x0.2 = 1 msn
    decfsz    SAYAC3, F
    goto     D_2msn0

    bcf      PORTB, 1

    return

;+++++
Gecikme_10msn        ;Servonun Durmasi icin gerekli islem
gecikmesi

    movlw    128
    movwf    SAYAC1

D_10ms1

    movlw    128
    movwf    SAYAC2

    nop                        ;49920x0.2=9984msn

D_10ms2

    decfsz    SAYAC2, F
    goto     D_10ms2
    decfsz    SAYAC1, F
    goto     D_10ms1        ;49926x0.2 =9985.2msn

    movlw    25
    movwf    SAYAC1

```

```

s10                                     ; 10 msn gecikme eklentisi

    decfsz      SAYAC1, F
    goto  s10
    return

;+++++
Gecikme_15msn                         ;Servonun Kontrolu icin
gerekli islem gecikmesi

    movlw 157
    movwf SAYAC1

D_15ms1

    movlw 157
    movwf SAYAC2

D_15ms2

    decfsz      SAYAC2, f
    goto  D_15ms2
    decfsz      SAYAC1, f
    goto  D_15ms1

    movlw 36
    movwf SAYAC1

s15

    decfsz      SAYAC1, f
    goto  s15

    return

;+++++
;+++++Sonradan Eklenen Kisim+++++
;+++++Button Kontrolleri+++++
A0_Kontrol

    btfss PORTA, 0
    goto  A0_Kontrol

```

A1_Kontrol

```

    btfss PORTA, 1
    goto  A2_Kontrol
    ;----->call  Sinyal_1.2msn

```

A2_Kontrol

```

    btfss PORTA, 2
    goto  Konumla
    ;----->call  Sinyal_2msn

```

;+++++++ANA PROGRAM ++++++

Basla

```

    call  Ilk_Islemler
    ;+++++++Sonradan Eklenen Bolum+++++++

```

```

    call  A0_Kontrol

```

;+++++++

Konumla

```

    movlw 20
    movwf SAYAC

```

Saga

;Saat yonunde hareket

```

    bcf      PORTB, 1

```

```

    call  Sinyal_1msn      ;sinyal
    call  Gecikme_15msn    ;15 msn de 1
    decfsz SAYAC, F        ;20 defa
    goto  Saga

```

```

    movlw 25
    movwf SAYAC

```

Dur_1

```

    call  Gecikme_10msn      ;500 msnlik ara GEREKLI
    decfsz      SAYAC, f
    goto  Dur_1

;+++++++Sonrdan Eklenen Bolum+++++++
    call  A1_Kontrol

    bcf      PORTB, 1

    call  Sinyal_1.2msn      ;sinyal
    call  Gecikme_15msn      ;15 msn de 1
    decfsz      SAYAC, F      ;20 defa
    goto  Saga

    movlw 25
    movwf SAYAC

Dur_2

    call  Gecikme_10msn      ;500 msnlik ara GEREKLI
    decfsz      SAYAC, f
    goto  Dur_2
;#####

    call  A2_Kontrol

    bcf      PORTB, 1

    call  Sinyal_2msn      ;sinyal
    call  Gecikme_15msn      ;15 msn de 1
    decfsz      SAYAC, F      ;20 defa
    goto  Saga

    movlw 25
    movwf SAYAC

Dur_3

    call  Gecikme_10msn      ;500 msnlik ara GEREKLI
    decfsz      SAYAC, f

```

```

        goto   Dur_3
;+++++
;        goto   Konumla (yeri degistirildi)
        end

```

Açıklaması:

İlk olarak PIC 16F877A'nın konfigürasyonu yapıldıktan sonra bekleme süreleri üretmek için kullanılacak olan değişkenler tanımlanmıştır. Daha sonra PORTA anahtarları kullanmak için giriş, PORTB'de motoru pulslamak için çıkış yapılmıştır. Bu işlemler ilk önce iki port temizlendikten sonra yapılmıştır.

Sinyal_1msn, Sinyal_1.2msn, Sinyal_2msn ;1 ms, 1,2 ms ve 2 ms'lik sinyal fonksiyonlarıdır. Bu fonksiyonlar motora verilecek sinyalin uzunluğunu belirleyen fonksiyonlardır. Sinyalin uzunluğuna göre motor değişik tepkiler verir. 1 milisaniyelik pulslar sero motorun yerinde kalmasını sağlar, 1.2 milisaniyelik pulslar adım atmasını sağlar. 2 milisaniyelik pulslar ise ters yönde adım atmasını sağlar.

Gecikme_10msn, Gecikme_15msn; 10ms'lik ve 15 ms'lik gecikmeler üretmek için yazılmış fonksiyonlardır.

A0_Kontrol, A1_Kontrol, A2_Kontrol; 1. , 2. ve 3. anahtarlarla ilgili fonksiyonlardır. Bu anahtarlara basıldığında yapılacak işlemleri belirtmektedir.

Ana fonksiyonda ise bütün bu fonksiyonlar birbirlerini kontrol edecek şekilde motorun çalışmasını sağlama amaçlı kullanılmıştır.

8.3.9 Step2.c

Step (adım) motor kontrol programı C programlama dili ile yazılmıştır. Aşağıda içeriği verilmiş olan dosya "STEP2.c" 'dir.

```

#include <pic.h>
#include <delay.c>

#define STEP RB0
#define YON RB1

#define KAPATMABUTTON RA4
#define LED RB7

const unsigned int RPM = 100;
unsigned int adim;
unsigned int gecikme;
unsigned char j;
void adimat(void);
void LedYak(void);

void adimat(void)
{
    PORTA = j;

    if(YON == 1)
    {
        j = j << 1;
        if (j == 16)
            j == 1;
    }
    else
    {
        j = j >> 1;
        if(j == 0)
            j = 8;
    }
}

void LedYak(void)
{
    for(;;)
    {
        if(KAPATMABUTTON == 1 | YON == 0 | STEP == 0)
            LED = 1;
    }
}

```

```

        else
            LED = 0;
    }
}

void main(void)
{

    TRISB = 0;
    TRISB = 0x7F;

    STEP = 0;
    j = 0;
    gecikme = 3000/RPM;

    TRISA = 0;
    TRISA = 16;

    for(;;)
    {
        if (KAPATMABUTTON == 0){
            j = 1;
            for(;;)
            {
                while(STEP == 1);
                while(STEP == 0);
                adimat();
                DelayMs(gecikme);
            }
        }
        else
            LedYak();
    }
}

```

Açıklaması:

Adimat(), step motora adım attırmak için yazılmış bir fonksiyondur. Main() fonksiyonunda başlangıç şartları belirlenip, gecikmede kullanılacak değerler değişkenlere atandıktan sonra anahtarın (YON) durumuna göre adimat fonksiyonu “if...else” komutları aracılığı ile motora yön vererek, çalıştırır. LedYak(void) fonksiyonu ise anahtarın durumuna göre bizi uyaran Led için çalışan bir fonksiyondur.

8.3.10 DCL293D.c

L293D entegre devreli DC motor kontrolü için gerekli olan kod hem C programlama dili ile hem de Assembler programlama dili ile yazılmıştır.

İlk olarak, aşağıda içeriği verilmiş olan dosya “DCL293D.c” ‘dir.

```
#include <pic.h>

#define ONOFF RA0
#define YON RA1

#define ON          1
#define SAAT_YONU   5
#define TERS_SAAT_YONU 10

void main (void)
{
    TRISB = 0;
    TRISA = 3;

    for(;;)
    {
        if (ONOFF == ON)
        {
            if(YON == 0)
                PORTB = SAAT_YONU;
            else
                PORTB = TERS_SAAT_YONU;
        }
    }
}
```

```

    }
    else
        PORTB = 0;
}
}

```

Açıklaması:

Görüldüğü üzere L293D Entegre Devresi halindeki H-Köprüsü ile yapılmış olan DC motor kontrol devresinin PIC 16F84A mikrokontrolörü için C diliyle yazılan programın LMD18200 Entegre Devresi halindeki H-Köprüsü ile yapılmış devredeki PIC16F84A mikrodenetleyicisi için C diliyle yazılmış olan programdan yapıcı hiç farkı yoktur.

Bu kodda gayet kolayca görülebileceği gibi “main” fonksiyonunda YON, RA1 olarak tanımlanmıştır. Yani PIC 16F84A mikrokontrolörünün 18. Bacağına bağlı olan anahtara basıldığı zaman B portuna bağlı ayaklardan 9. ayak uyarılır ve motor saat yönün tersinde döner. Basılmaz ise 7. Ayaktan uyarılan motor saat yönünde çalışmaya devam eder. Ayrıca header dosyalarından da takip edilebileceği üzere “main” fonksiyonun başında ‘A’ portu giriş ‘B’ portu çıkış olarak tanımlanmıştır.

8.3.11 DCL293D.asm

Son olarak da L293D entegre devreli DC motor kontrolü için gerekli kodun Assembler programlama dili ile yazılmış olan versiyonu aşağıdaki gibidir.

Aşağıda içeriği verilmiş olan dosya “DCL293D.asm” ‘dir.

List p = 16F84

```
#include <p16F84A.inc>
```

```
#define onoff PORTA, 0
```

```
#define yon PORTA, 1
```

```
#define saat_yonu    5
#define ters_saat_yonu    d'10'

    org    0x00
    bsf          STATUS, RP0
    clrf    TRISB
    movlw  3
    movwf  TRISA
    bcf          STATUS, RP0

tekrar

    btfss  onoff
    goto   m_off
    btfss  yon
    goto   saat
    movlw  ters_saat_yonu
    movwf  PORTB
    goto   tekrar

saat

    movlw  saat_yonu
    movwf  PORTB
    goto   tekrar

m_off

    clrf    PORTB
    goto   tekrar

end
```

Açıklaması:

Bu program PIC16F84A için Assembler dili ile yazılmıştır. Bir önceki kod ile aynı işlevdedir. Yazılmış olan C kodunun assembler örneği olarak kabul edilebilir. Başlangıç işlemlerinde Port B sıfırlanmış daha sonra RB3 çıkış yapılmıştır. Daha sonra PortA sıfırlanmış ve giriş için ayrılmıştır. “Tekrar” işleminde anahtara basılıp basılmadığı “btfss” komutu ile kontrol edilir. Basılmış ise ter saat yönüne döndüren koda atlama yapılır. Basılmamış ise saat yönüne döndüren koda atlama yapılır. Her iki durumdada anahtarın konumunu devamlı kontrol eden “tekrar” kemsine dönüş yapılır. Sonsuz döngü içinde bu işlemler devam eder.

8.3.12 DCLMD18.c

LMD18200 Entegre devreli DC motor kontrolü için C ile gerçekleştirilmiş yazılım şöyledir:

Aşağıda içeriği verilmiş olan dosya “DCLMD18.c” ‘dir.

```
#include <pic.h>

#define ONOFF      RA0
#define YON RA1

#define ON          1
#define SAAT_YONU  1
#define TERS_SAAT_YONU  3

void main(void)
{
    TRISB = 0;
    TRISA = 3;
    PORTB = 0;

    for(;;)
    {
        if (ONOFF == 0)
```

```

    {
        if(YON == 0)
            PORTB = SAAT_YONU;
        else
            PORTB = TERS_SAAT_YONU;
    }
    else
        PORTB = 0;
}
}

```

Açıklaması:

Görüldüğü üzere LMD18200 Entegre Devresi halindeki H-Köprüsü ile yapılmış olan DC motor kontrol devresinin PIC 16F84A mikrokontrolörü için C diliyle yazılan programın L293D Entegre Devresi halindeki H-Köprüsü ile yapılmış devredeki PIC16F84A mikrodnetleyicisi için C diliyle yazılmış olan programdan yapıcı hiç farkı yoktur.

Bu kodda gayet kolayca görülebileceği gibi “main” fonksiyonunda YON, RA1 olarak tanımlanmıştır. Yani PIC 16F84A mikrokontrolörünün 18. Bacağına bağlı olan anahtara basıldığı zaman B portuna bağlı ayaklardan 9. ayak uyarılır ve motor saat yönün tersinde döner. Basılmaz ise 7. Ayaktan uyarılan motor saat yönünde çalışmaya devam eder. Ayrıca header dosyalarından da takip edilebileceği üzere “main” fonksiyonun başında ‘A’ portu giriş ‘B’ portu çıkış olarak tanımlanmıştır.

ORÇUN DURSUN

Kişisel Bilgiler

Doğum Tarihi: 06-07-1979

Adres: Adnan Menderes Bul. Tutkun Apt. No: 74 AYDIN

Cep Telefonu: 0-532-3309713

E-Mail: orcundursun@hotmail.com

Öğrenim Durumu

Lise: İzmir Amerikan Koleji / 1997

Üniversite: İstanbul Üniversitesi, Mühendislik Fakültesi,
Makina Mühendisliği Bölümü / 2004

Yüksek Lisans: Gebze Yüksek Teknoloji Enstitüsü,
Tasarım ve İmalat Bölümü / Devam Etmekte

Dil

İngilizce / Çok İyi

Almanca / Orta

Tecrübe

T.Ü.P.R.A.Ş. (İzmir Rafinerisi) (Staj) 2001

İPRAGAZ (İzmir Dolum Tesisi) (Staj) 2002

H.U.M. (TARİŞ-AYDIN Pamuk Yağı Kombinesi Montajı ve İşletmeye
Alınması) 2004-2006

TARİŞ (TARİŞ-AYDIN Pamuk Yağı Kombinesi Ham Yağ Şefliği) 2006-2007