Y. DURNA, 2007

T.C.
NİĞDE ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ
ELEKTRİK - ELEKTRONİK MÜHENDİSLİĞİ ANABİLİM DALI

MİKRODENETLEYİCİ TABANLI RF'Lİ KONTROL UYGULAMASI

YÜKSEK LİSANS TEZİ

YILMAZ DURNA

Ağustos 2007

T.C.
NİĞDE ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ
ELEKTRİK - ELEKTRONİK MÜHENDİSLİĞİ ANABİLİM DALI

MİKRODENETLEYİCİ TABANLI RF'Lİ KONTROL UYGULAMASI

YILMAZ DURNA

Yüksek Lisans Tezi

Danışman

Doç. Dr. Murat UZAM

Ağustos 2007

Doç.Dr.Murat UZAM danışmanlığında Yılmaz DURNA tarafından hazırlanan "Mikrodenetleyici Tabanlı RF'li Kontrol Uygulaması" adlı bu çalışma jürimiz tarafından Niğde Üniversitesi Fen Bilimleri Enstitüsü Elektrik-Elektronik Mühendisliği Anabilim Dalında Yüksek Lisans tezi olarak kabul edilmiştir.

Başkan : Doç.Dr.Murat UZAM (Danışman, Niğde Üniversitesi)

(Ünvan, Adı ve Soyadı) (Üniversite)

Üye : Yrd.Doç.Dr.Halis ALTUN (Niğde Üniversitesi)

(Ünvan, Adı ve Soyadı) (Üniversite)

Üye : Yrd.Doç.Dr.Mutlu AVCI (Çukurova Üniversitesi)

(Ünvan, Adı ve Soyadı) (Üniversite)

ONAY:

Bu tez, Fen Bilimleri Enstitüsü Yönetim Kurulunca belirlenmiş olan yukarıdaki jüri üyeleri tarafından …./…./20.... tarihinde uygun görülmüş ve Enstitü Yönetim Kurulu'nun …./…./20.... tarih ve …..... sayılı kararıyla kabul edilmiştir.

...../...../200..

Prof. Dr. Meysun İBRAHİM

Enstitü Müdürü

ÖZET

MİKRODENETLEYİCİ TABANLI RF'Lİ KONTROL UYGULAMASI

DURNA, Yılmaz
Niğde Üniversitesi
Fen Bilimleri Enstitüsü
Elektrik - Elektronik Mühendisliği Anabilim Dalı

Danışman : Doç. Dr. Murat UZAM

Ağustos 2007, 163 sayfa

Bu çalışmanın amacı 18 serisi PIC mikrodenetleyicileri kullanarak RF ile uzaktan kontrol işleminin gerçekleştirilmesidir.

Bu çalışmada 18 serisi PIC olarak 18F452 mikrodenetleyicileri ve üretici firmaya ait MCC18 C derleyicisi kullanılmıştır. Yazılım standart C dili ile yapılmıştır. Uzaktan kontrol işlemini gerçekleştirmek amacıyla ABACOM firmasının RF alıcı ve verici modülleri seçilmiştir. Çalışmanın temelinde yer alan kontrol kartı olarak, sanayide yoğun olarak kullanılan PLC sistemlerinin modellemesi yapılmıştır.

PLC sistemlerinin uzaktan RF ile kontrolüne imkân veren bu çalışma ile otomasyon sistemlerinin kablosuz olarak gerçekleştirilmesine yönelik bir uygulama sunulmuştur.

Anahtar sözcükler: RF, PLC, PIC, 18F452 mikrodenetleyici,

SUMMARY

MICROCONTROLLER BASED CONTROL APPLICATION WITH RF

DURNA, Yılmaz

Nigde University

Graduate School of Natural and Applied Sciences

Department of Electrical and Electronics Engineering

Supervisor : Assoc.Prof. Dr. Murat UZAM

August 2007, 163 pages

The aim of this study is to implement an RF control application by using 18 series PIC microcontrollers.

In this study, 18 series PIC 18F452 microcontrollers and MCC18 C compiler which belongs the same producer are used. The software utilized is the standard C language. ABACOM RF receiver and transmitter modules are chosen to implement the remote control application. The basic control cards PLC systems which are widely used in todays' industry.

This study leads the way towards controlling PLC systems with RF. An example is provided to demonstrate the applicability of the proposed wireless automation systems.

Keywords: RF, PLC, PIC, 18F452 microcontroller,

TEŞEKKÜR

Niğde Üniversitesi Mühendislik-Mimarlık Fakültesi Elektrik-Elektronik Mühendisliği Bölümünde yüksek lisans tezi olarak hazırlanan bu çalışmamda, bana yön veren ve uğraşlarıyla bilime önemli katkıda bulunan danışmanım Doç. Dr. Murat UZAM'a, yüksek lisans çalışmam süresince bana her konuda yardımcı olan Elektrik-Elektronik Mühendisliği bölüm hocalarıma ve her türlü maddi manevi desteğini benden esirgemeyen aileme en içten teşekkürlerimi sunarım. Otomasyon sanayinde yaygın olarak kullanılan PLC'lerin uzaktan kontrol edilmesine yönelik bir uygulama olan bu çalışmanın sanayimize ve üniversitelerde bu konu ile ilgili yapılan çalışmalara katkıda bulunmasını dilerim.

İÇİNDEKİLER

ÖZET iii
SUMMARY iv
TEŞEKKÜR v
İÇİNDEKİLER DİZİNİ vi
ÇİZELGELER DİZİNİ x
ŞEKİLLER DİZİNİ xi
FOTOĞRAF VB.MALZEMELER DİZİNİ xiii
KISALTMA VE SİMGELER xiv
BÖLÜM I. GİRİŞ 1
BÖLÜM II. MİKRODENETLEYİCİLER 6
2.1 Giriş 6
2.2 Mikroişlemciler 7
2.2.1 Merkezi işlem birimi 7
2.2.1.1 Kaydediciler 8
2.2.1.2 Aritmetik mantık birimi (ALU) 8
2.2.1.3 Zamanlama-kontrol birimi 8
2.2.2 Giriş-çıkış birimi (G/Ç) 8
2.2.3 Bellek 9
2.3 Mikrodenetleyiciler 9
2.3.1 EEPROM bellekli mikrodenetleyiciler 11
2.3.2 Flash (EPROM) bellekli mikrodenetleyiciler 11
2.3.3 OTP bellekli mikrodenetleyiciler 11
2.4 Mikrodenetleyicinin Mikroişlemciye Olan Üstünlükleri 12
2.5 PIC Mikrodenetleyicisi 12
2.5.1 Özellikleri 13
2.5.2 Kod verimliliği 13
2.5.3 Güvenilirliliği 13
2.5.4 Komut seti 13
2.5.5 Hızı 14

2.5.6 Statik işlem…….. 14
2.6 PIC 18F452…… 14
2.6.1 PIC18F452'nin besleme uçları ve beslenmesi…….. 15
2.6.2 PIC18F452'nin reset uçları…….. 15
2.6.3 I/O portları…… 16
2.6.3.1 Port A……. 16
2.6.3.2 Port B……. 16
2.6.3.3 Port C……. 17
2.6.3.4 Port D……. 17
2.6.3.5 Port E…… 17
2.6.4 Yakalama / karşılaştırma ve PWM birimi…….. 17
2.6.5 PIC 18F452 mimarisi.....…….. 18
BÖLÜM III. MPLAB IDE VE MPLAB C18 C COMPILER …...…… 20
3.1 MPLAB IDE……..... 20
3.2 MPLAB C18 Kurulumu…..……..... 20
3.2.1 Proje oluşturulması……..... 23
3.3 MPLAB C18 Derleyici Kütüphanesi ve Özellikleri…….. 32
3.4 MPLAB C18 Derleyici……. 32
3.4.1 MPLAB C18 veri çeşitleri…… 33
3.4.1.1 bit…….. 33
3.4.1.2 unsigned char…… 33
3.4.1.3 signed char…… 34
3.4.1.4 unsigned int…….. 34
3.4.1.5 signed int…….. 34
3.4.1.6 long……... 35
3.4.1.7 unsigned long……... 35
3.4.1.8 float…….. 35
3.4.1.9 double……... 35
3.4.2 Değişkenlere tanım esnasında değer vermek……... 36
3.4.3 Programda açıklayıcı yazılar…… 36
3.4.4 Değişkenlerin bellekte saklanması…… 37
3.4.5 Diziler……... 37
3.4.6 Program değişken isimleri…… 39

3.4.7 Static değişkenler……. 39
3.4.8 Volatile değişkenler…….. 39
3.4.9 Persistent değişkenler……... 39
3.4.10 Mutlak adres değişkenleri…….. 40
3.4.11 Operatörler…….. 40
3.4.12 Program akış kontrolü…… 42
3.4.12.1 if – else……. 42
3.4.12.2 for……. 44
3.4.12.3 while……. 45
3.4.12.4 do…….. 46
3.4.12.5 break…… 46
3.4.12.6 switch – case……. 47
3.4.13 Kullanıcı fonksiyonları……... 48
3.4.14 Ön-işlemci direktifleri…… 49
3.4.14.1 #define…….. 49
3.4.14.2 #asm ve #endasm…….. 50
3.4.14.3 #include…… 51
3.4.15 Programda değişik sayı tabanı kullanımı……... 51
3.4.16 Yapılar (Structures)…… 52
3.4.17 İşaretçiler (Pointers)……... 53
3.4.18 A/D çevirici fonksiyonlar…..…… 54
3.4.19 USART fonksiyonları……..…… 55
BÖLÜM IV. DONANIM YAPISI VE ÇALIŞMASI ……. 56
4.1 Çalışma Modları…… 58
4.1.1 Otomatik çalışma modu…… 59
4.1.1.1 Aynı program……... 59
4.1.1.1.2 Farklı program……... 62
4.1.1.1.3 Sonuçları iste……. 63
4.1.2 Manuel çalışma modu……. 63
4.2 Sistemde Kullanılan RF Alıcılar……... 63
4.3 Sistemde Kullanılan RF Vericiler……. 64
4.4 Sistemde Kullanılan RF Güç Yükselteci……... 64
4.5 Sistem Anteni…… 65

4.6 PLC Kartı Devresi……..…….... 67
4.7 Kumanda Cihazı Devreleri……..…….. 69
4.8 Sistemin Alıcı ve Verici Devreleri…….... 72
4.9 ICPROG Programı ve ICSP Programlama Devreleri..... 77
BÖLÜM V. AKIŞ DİYAGRAMLARI …….…… 79
BÖLÜM VI. SONUÇ VE ÖNERİLER ……...…… 88
KAYNAKLAR ..…...…… 89
EK-A SİSTEM YAZILIMI …….….. 90
EK-B PLC KARTINA AİT HEX KODLAR ……. 144
EK-C KUMANDA BİRİMİNE AİT HEX KODLAR …… 149
EK-D KATALOG BİLGİLERİ ……..……... 157

ÇİZELGELER DİZİNİ

Çizelge 3.1 C dilinin operatörleri..... 41
Çizelge 3.2 A/D çevirici fonksiyonları..... 54
Çizelge 3.3 USART fonksiyonları..... 55
Çizelge 5.1 İletişim veri paketi..... 87

ŞEKİLLER DİZİNİ

Şekil 1.1	Mikrodenetleyici tabanlı RF'li kontrol uygulaması blok diyagramı.	4
Şekil 2.1	Mikrodenetleyici yapısı.…...……...	6
Şekil 2.2	PIC 18F452 iç yapısı…….....	18
Şekil 2.3	18F452 bacak yapısı…….....	19
Şekil 3.1	MPLAB C18 kurulum başlangıcı.....	21
Şekil 3.2	Lisans anlaşması……	21
Şekil 3.3	Okuma dosyası……..	22
Şekil 3.4	Kurulum dizini……..	22
Şekil 3.5	MPLAB IDE ve MPLAB C18 entegrasyonu için gerekli dosyalar..	23
Şekil 3.6	Kullanılacak cihazı seçin……...	24
Şekil 3.7	Seçilmiş cihaz……	25
Şekil 3.8	Projede kullanılacak dil aracı……	25
Şekil 3.9	İşletilebilir C18 derleyicisi ve kurulum dizini……...	26
Şekil 3.10	C18 derleyicisi bağlayıcısı ve kurulum dizini……...	26
Şekil 3.11	C18 Assembly derleyicisi ve kurulum dizini……...	27
Şekil 3.12	Projeyi isimlendirme…….	27
Şekil 3.13	Projeye dosya ekleme……	28
Şekil 3.14	Derleme özelliklerinin ayarlanması……..	28
Şekil 3.15	Derlenecek projeye başlık, kütüphane ve bağlayıcı dosyalarının dahil edilmesi.....	29
Şekil 3.16	Projenin derlenmesi……...	29
Şekil 3.17	Derleme sonucu…….	30
Şekil 3.18	C18 derleyicisi akış şeması……...	31
Şekil 4.1	RFRQ2 alıcısı……	64
Şekil 4.2	RFTQ2 vericisi……..	64
Şekil 4.3	PLA-05W-433 yükselteci……..	65
Şekil 4.4	Anten……..	65
Şekil 4.5	Anten SWR değeri…….	66
Şekil 4.6	PLC kartı devre şeması……..	68

Şekil 4.7	Tuş takımı devre şeması……	69
Şekil 4.8	Kumanda birimi devre şemasının tamamı…….	70
Şekil 4.9	Kumanda birimi akü şarj devresi……....	71
Şekil 4.10	Verici devre şeması……	72
Şekil 4.11	Alıcı devre şeması……..	73
Şekil 4.12	ICPROG programı ilk defa çalıştırıldığında ekran görüntüsü.....	77
Şekil 4.13	ICPROG programının ekran görüntüsü.....	78
Şekil 4.14	PLC kartına ait PIC 18F452 ICSP bağlantı uçları.....	78
Şekil 5.1	Kumanda birimi ana menüsüne ait akış diyagramı…....……..	79
Şekil 5.2	Manüel kontrol alt menüsüne ait akış diyagramı.....……..…..	80
Şekil 5.3	Otomatik kontrol alt menüsüne ait akış diyagramı…....…….....	81
Şekil 5.4	Manüel kontrol-aynı program alt menüsüne ait akış diyagramı.....	82
Şekil 5.5	Otomatik kontrol-aynı program alt menüsüne ait akış diyagramı.....	83
Şekil 5.6	Manüel kontrol-farklı program alt menüsüne ait akış diyagramı.....…….	84
Şekil 5.7	Otomatik kontrol-farklı program alt menüsüne ait akış diyagramı.....…….	85
Şekil 5.8	PLC kartına ait akış diyagramı.....	86

FOTOĞRAF VB. MALZEMELER DİZİNİ

Fotoğraf 2.1	PIC 18F452.....	19
Fotoğraf 4.1	Mikrodenetleyici tabanlı RF'li kontrol uygulaması.....	56
Fotoğraf 4.2	Kumanda birimi……	57
Fotoğraf 4.3	PLC kartı……...	57
Fotoğraf 4.4	PLC kartına ait alıcı ve verici……...	58
Fotoğraf 4.5	Kumanda birimi menü……..	58
Fotoğraf 4.6	Otomatik kontrol menü…….	59
Fotoğraf 4.7	Aynı program menü……..	59
Fotoğraf 4.8	İlk dört program örneği…….	60
Fotoğraf 4.9	İkinci dört program örneği……	60
Fotoğraf 4.10	Gönderme tamamlandı ekranı……..	61
Fotoğraf 4.11	Sonuçlar ekranı…….	61
Fotoğraf 4.12	Çalıştırılacak PLC kartının seçimi……	62
Fotoğraf 4.13	Uygun SWR değerlerini sağlayan kutular……	66
Fotoğraf 4.14	Kumanda birimi alıcı ve verici kartları…….	74
Fotoğraf 4.15	Kumanda birimi kontrol kartı……...	74
Fotoğraf 4.16	Kumanda birimi……	75
Fotoğraf 4.17	Alıcı kartı……..	75
Fotoğraf 4.18	Verici kartı……	76
Fotoğraf 4.19	Kumanda birimi akü şarj kartı……..	76
Fotoğraf 4.20	PIC 18F452 ICSP programlayıcı devre.....	78

KISALTMA VE SİMGELER

KISALTMA/SİMGE

RF	Radio Frequency (Radyo Frekans)
PLC	Programmable Logic Controller (Programlanabilir Lojik Denetleyici)
PIC	Peripheral Interface Controller (Çevresel Arayüz Denetleyicisi)
RISC	Reduced Instruction Set Controller (Azaltılmış Komut Kümeli Denetleyici)
ICSP	In Circuit Serial Programming (Devre İçi Seri Programlama)
CPU	Central Processing Unit (Merkezi İşlem Ünitesi)
RAM	Rasgele Erişimli Bellek
ROM	Sadece Okunur Bellek
SWR	Standing Wave Ratio (Duran Dalga Oranı)

BÖLÜM I

GİRİŞ

Marconi tarafından 12 Aralık 1901 tarihinde Avrupa ile Amerika kıtası arasında yapılan ilk okyanus aşırı radyo haberleşme denemesinde gönderilen bilgi Mors kodunda sadece 3 noktadan oluşan "s" harfi idi. İyonosferden yansıyarak 3.500 km yol alan elektromanyetik dalgalar İngiltere-Cornwall'da bulunan vericiden Kanada-Newfoundland'de bulunan alıcıya ulaşmış ve Marconi, 1895 yılında İtalya'da yaptığı ilk denemeye göre çok büyük bir başarı elde etmişti. Bu başarılı deneme Marconi'ye 1909 yılında Nobel Fizik Ödülünü de kazandırmıştı. Özellikle kıtalararası haberleşmede ve mobil haberleşmede büyük imkanlar sağlayan bu teknoloji ülkemize yaklaşık 13 yıl gecikmeyle gelmiş ve 1. Dünya Savaşının başladığı günlerde kullanılmaya başlanmıştır. Uzun yıllar sonra, gazete sayfalarında, İstanbul Atatürk Havalimanı Dış Hatlar Terminalinde kamunun kullanımına açık, kablosuz yüksek hızlı internet ve VPN (Virtual Private Network) erişim sisteminin kurulduğu ve deneme amacıyla hizmete verildiği haberi yer almaktadır. Aradan geçen 100 yılda, bazen birkaç yıl, bazen de yıllar süren gecikmelerle ülkemize gelen yeni radyo teknolojileri, bu defa fazla gecikmeden gelmiş, koltuklarına oturduklarında ilk işi internete bağlanmak olan Internet kullanıcılarına havalimanında beklerken yüksek hızda internete erişme imkânını sağlamıştır [1].

Başlangıçta, sadece "s" harfinin kodlanarak iletebilmesi için büyük vericilere ve anten alanlarına ihtiyaç duyan radyo teknolojisi bugün, taşınabilir ve avuç içi bilgisayarlarına takılan özel kartlarla 54 Mbit/s hızında İnternet erişimine imkân sağlayan bir duruma gelmiştir. 2.4 GHZ bandında ve IEEE 802.11g protokolünde çalışan bu sistemler, üniversite yerleşkelerinde, havalimanlarında, alış-veriş merkezlerinde ve hatta simit saraylarında kullanılmaya başlanmıştır.

Kablosuz veri iletişiminde yaşanan bu gelişmeler, doğal olarak sadece internet erişimini kolaylaştırmakla sınırlı kalmamış, bazen farkında bile olmadan kullandığımız birçok uygulamalarla, günlük hayatımızın bir parçası haline gelmiştir. Bu gelişmelerin bir ürünü de SRD (Short Range Device/ Kısa Menzilli Cihaz) veya ISM (Industrial, Scientific and Medical/ Endüstriyel, Bilimsel ve Tıbbi) olarak tanımlanan frekans

bantlarında çalışan cihazlardır (RF Modüller). Bu cihazlar kullanılarak geliştirilen birçok uygulama vardır. Bunlardan birkaç tanesini şöyle sıralayabiliriz:

- Ev Otomasyonu Sistemleri
- RKE (Remote Keyless Entry/ Uzaktan Anahtarsız Giriş)
- Kablosuz Alarm ve Güvenlik Sistemleri
- Uzaktan Ölçüm Okuma Sistemleri (Telemetry Systems)
- Kablosuz Kumanda Sistemleri

ISM/SRD uygulamalarında kullanılan RF Modüllerin veri aktarım kapasiteleri, yukarıda bahsedilen IEEE 802.11 a/b/g standartlarında çalışan cihazlar kadar yüksek değildir. Ancak, kullanım amaçları açısından bakıldığında buna gerek olmadığı görülecektir. ISM ve SRD uygulamalarında amaç "Komuta ve Kontrol" dür. Amaç bir yerdeki bilgi kütlesini bir başka yere aktarmak değildir. Bilginin iletilmesi "Komuta ve Kontrol" yaklaşımı içerisinde sadece araçtır, amaç değildir. Bu açıdan bakıldığında, değil megabit/saniye seviyelerinde veri aktarımı, kilobit/saniye seviyesindeki hızlar bile fazlasıyla yeterli olmakta, ihtiyaçları fazlasıyla karşılamaktadır. Burada bahsedilen ISM/SRD uygulamalarından, günlük hayatımızda kullanılan birkaç örneği vererek daha iyi anlayabiliriz.

Genel olarak güvenlik sistemlerinin mimarisi aynıdır ve iki ana unsurdan oluşurlar;

- Merkezi Kontrol Birimi (Santral)
- Çevresel Donanımlar (Algılayıcılar, Sirenler, Telefon Arayıcıları vb.)

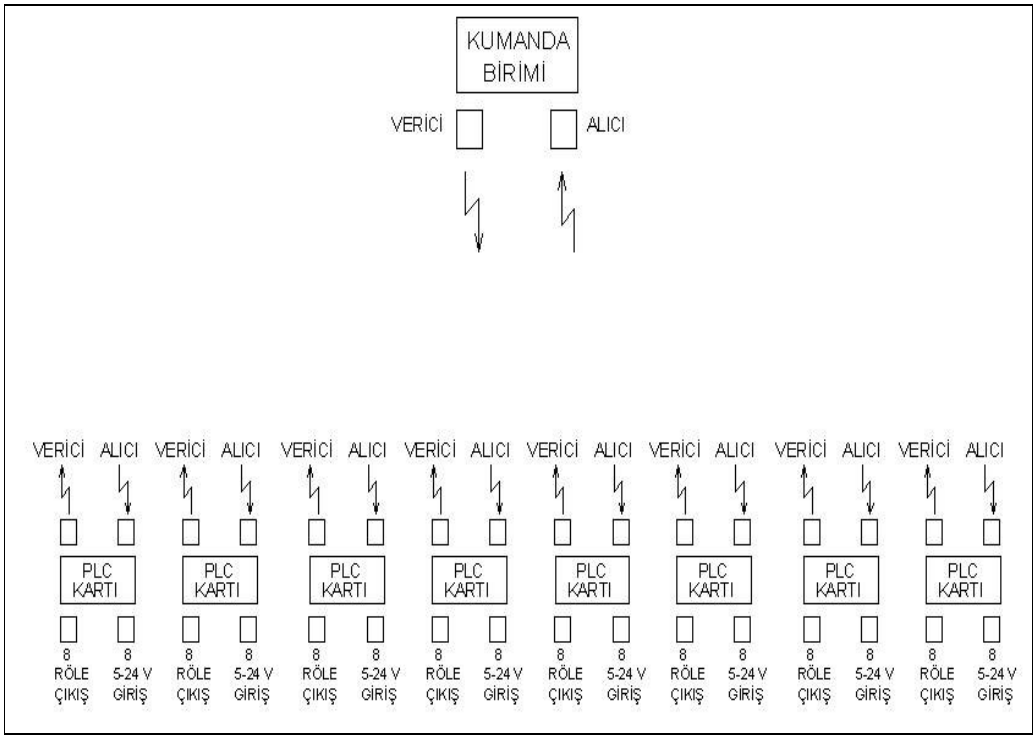
Merkezi kontrol birimi, çevresel donanımlar ile haberleşerek durumlarını öğrenir ve aldığı bilgiler doğrultusunda sistemi işletir, bilgileri kullanıcı ara yüzünde görüntüler, gerekirse alarm durumuna geçerek çeşitli ikazlar üretir, özet olarak komuta-kontrol faaliyetlerini icra eder. Bu noktada kablolu ve kablosuz sistemleri birbirinden ayıran en temel fark, merkezi ünite ile çevresel donanımlar arasındaki haberleşmenin sağlandığı iletim ortamıdır. Bu fark doğrultusunda sistemler, kablolu ve kablosuz olarak sınıflandırılırlar.

Komuta-Kontrol faaliyetleri çerçevesinde, merkezi kontrol biriminin çevresel donanımlardan öncelikle durum bilgilerini alması gerekmektedir. Bu bilgiler, donanımın cinsine göre farklılıklar arz edecektir, ancak temel olarak "Var" veya "Yok" şeklinde, yani "1" veya "0" anlamında olacaklardır. Kullanılan donanım hareket algılayıcısı ise ürettiği bilgi "Hareket Var = (1)" veya "Hareket Yok = (0)" olacaktır. Eğer donanım kapı kontağı ise ürettiği bilgi "Kapı Açık= (1)" veya "Kapı Kapalı = (0)" olacaktır. Diğer algılayıcılar için de durum bundan çok farklı değildir. Merkezi kontrol birimi bu bilgileri değerlendirecek, gerekirse ikaz sistemlerini devreye sokacaktır, "Alarm Var = (1)" veya Alarm Yok = (0)". Bu bilgiler, merkezi kontrol biriminin ihtiyaç duyduğu durum bilgileridir ve kablosuz olarak aktarılacak veri içerisinde yer almaları gerekmektedir. Bunun yanında böyle bir sistemin çalışabilmesi için, sistemi oluşturan cihazların mantıksal olarak adreslenmesi gerekmektedir. Merkezi kontrol birimi ile çevresel donanımları arasında bir Komuta-Kontrol Çevrimi oluşturulmalı, haberleşme için bir protokol geliştirilmelidir. İşte bu protokol çerçevesinde oluşturulacak veri paketlerinin içerisinde, sistemi meydana getiren unsurların adreslerinin, ürettikleri bilgilerin veya komutların ve hata/güvenlik tedbirleri için ilave bilgilerin yer alması gerekmektedir. Yukarıda anlatıldığı üzere, sistem içerisinde oluşturulan bilgiler genellikle "1" veya "0" dan yani birkaç bitten ibaret olduğundan, oluşturulacak veri paketlerinin boyutları birkaç on byte'ı geçmeyeceği aşikardır. Dolayısıyla kilobit/saniye mertebelerindeki veri iletim hızları, bu tip sistemler için fazlasıyla yeterli olmaktadır.

Uzaktan ölçüm sistemlerindeki genel amaç da aynıdır. Bu sistemler, çeşitli ölçüm noktalarındaki değerlerin yerinde ölçülüp, elde edilen bilgilerin başka bir noktaya iletilmesi amacıyla geliştirilmişlerdir ve çok geniş bir kulanım yelpazesine sahiptirler. Sanayi, meteoroloji, uzay araştırmaları, araba yarışları, su altı araştırmaları, tıp vb. sayılamayacak kadar çok miktarda kullanım alanları vardır. Burada asıl olan, yapılan ölçümümün neticesinde elde edilen verinin, bir yerden başka bir yere aktarılmasıdır.
Bir başka örnek uygulama ise gittikçe yaygınlaşan Kablosuz Taksi Çağrı Sistemidir ki bu uygulamada da durum, diğer örneklerden farklı değildir.

Bu çalışmada, yukarıda bahsedilen uygulamaların gerçekleştirilmesinde kullanılan, kablosuz veri aktarımı yapmak üzere, alıcı-verici tasarımı üzerinde durulacaktır. Çalışmanın başlangıcındaki amaç, kablosuz veri aktarımı yapabilen bir alıcı-verici

tasarımı ve örnek bir uygulamanın gerçekleştirilmesi olmasına rağmen, ilerleyen aşamalarda bunun yeterli olmayacağı kanaatine varılmıştır. Netice itibariyle, alıcı-verici tasarımı ve uygulaması olarak başlanan çalışma, "Kablosuz Veri İletimi İçin Alıcı-Verici Kartı Tasarımı ve Örnek Bir Uygulama" ya dönüşmüştür. Bu uygulama da kontrol kartı bölümüne PLC örnek alınmıştır. Uzaktan kontrollü bir otomasyon çalışması oluşturulmaya çalışılmıştır. Bu henüz yeni bir kavramdır. PLC sistemlerinin RF ile kontrolüne uygulamada pek rastlanılmamaktadır. Bu çalışma bu anlamda kontrol sistemlerinin RF ile uygulamasına model olarak önemli bir katkıda bulunabilecektir. Şekil 1.1'de gerçekleştirilen sisteme ait blok diyagram görülmektedir. Bu çalışmanın bundan sonraki bölümleri şu şekilde düzenlenmiştir.



Şekil 1.1 Mikrodenetleyici tabanlı RF'li kontrol uygulaması blok diyagramı

BÖLÜM II'de mikro denetleyiciler ve PIC mikrodenetleyicisine ait temel bilgiler verilmiş özellikleri hakkında detaylı açıklamalar verilmiştir. PIC 18F452 mikrodenetleyicisinin port özellikleri ve karakteristik özelliklerinden detaylıca bahsedilmiştir.

BÖLÜM III'te PIC programını yazmak ve derlemek için kullanılan MPLAB IDE ve MPLAB C18 C derleyicisi anlatılmaktadır. Program yazımında kullanılan C diline ait fonksiyonlardan, değişkenlerden, operatörlerden, dizilerden, yapılardan ve döngü fonksiyonlarından bahsedilmiştir. Derleyiciye ait özel tanımlı fonksiyonlara sınırlı da olsa yer verilmiştir. Derleyici kurulumu ve proje oluşturulması yeni kullanıcılara kullanım kolaylığı açısından detaylıca anlatılmıştır.

BÖLÜM IV'te çalışmaya ait donanım ve yapısı anlatılmıştır. Sistemin çalışma modları ile çalışma modlarına ait alt menülerin işlevlerinden bahsedilmiştir. Çalışmada gerçekleştirilen devre kartlarının şemaları verilmiş ve çalışmaları incelenmiştir.

BÖLÜM V'te çalışmaya ait yazılımın akış diyagramlarına yer verilmiş ve yazılımın çalışması anlatılmıştır.

BÖLÜM VI'da ise sonuç ve öneriler bulunmaktadır.

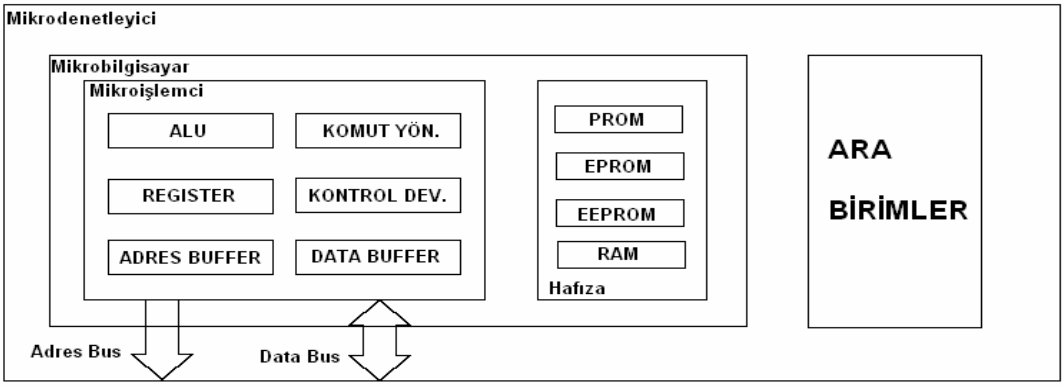
BÖLÜM II

MİKRODENETLEYİCİLER

2.1 Giriş

Mikrodenetleyiciler günümüzde hemen hemen tüm elektronik ürünler içerisinde kullanılmaktadır. Bu bölümde mikrodenetleyiciler ve özel olarak bu çalışmada kullanılan PIC18F452 mikrodenetleyicisi ile ilgili açıklamalara yer verilmiştir.

Mikrodenetleyicileri incelemeden önce mikroişlemci ve mikrobilgisayar terimlerini açıklamak gerekir. Şekil 2.1'de mikroişlemci ve mikrobilgisayarı kapsamakta olan mikrodenetleyiciye ait yapı görülmektedir.



Şekil 2.1 Mikrodenetleyici yapısı

Mikroişlemci, verileri işler ve sistemi oluşturan çeşitli birimler arasında bilgi akışı kontrolünü gerçekleştirir. Veri işlemenin büyük çoğunluğunu merkezi işlem biriminde bulunan Aritmetik Lojik Birimi (ALU) üzerinden gerçekleştirir. Ancak bu işlemlerin gerçekleştirilmesi sırasında Kod Çözme Kontrol Birimleri ile çeşitli Saklayıcılar (Registers) da çok yoğun olarak kullanılır. Merkezi işlem birimini oluşturan çeşitli alt birimlerin tek bir entegre devre üzerinde gerçekleştirilmiş, üretilmiş haline mikroişlemci adı verilir [2].

Bir mikroişlemci kullanarak hazırlanmış bilgisayarlara mikrobilgisayar denilmektedir. Hafıza ve giriş çıkış birimlerinin miktarı, türü ve kapasitesi uygulamaya bağlı olarak değişir.

Mikrobilgisayara Paralel Data Portları, Usart, ADC, DAC vb. gibi arabirimlerin eklenmesiyle mikrodenetleyici elde edilir. Bu temel bileşenler tek bir yonga içerisinde üretilerek mikrodenetleyici entegreleri elde edilmiştir. Mikrodenetleyicilerin sahip olduğu kabiliyetler, sağladığı imkanlar, fiyatları, donanım özellikleri ve kullanım alanları farklıdır. Bu projede mikrodenetleyici olarak Microchip firmasının (www.microchip.com) 18FXX2 adıyla ürettiği PIC ailesinin en gelişmiş üyelerinden olan PIC 18F452 kullanılmıştır.

2.2 Mikroişlemciler

Mikroişlemci herhangi bir sistemde merkezi işlem birimidir ve bulunduğu sistemde aritmetik ve mantıksal işlemleri yürütür. Bir klavyeden gelen verilerin, bir basınç algılayıcısından gelen sayısallaştırılmış çıkışın veya bir başka verinin bilgisayara alınması ve bu bilgilerin çıkış olarak sağlanması mikroişlemci tarafından kontrol edilir. Mikroişlemcinin bu tür işlemleri giriş olarak algılayıp, çıkışa yansıtması için mikroişlemci programlanır. Bir mikroişlemci, belleğinde saklı bulunan programı her bir komutu sıra ile okuyarak yürütür. Her komut önce, onu yürütmek için gerekecek işlemleri belirlemek üzere mikroişlemcinin anlayabileceği makina koduna çevrilir ve sonra gereken işlemler yapılır. Mikroişlemci entegre devresi, yazılan programları meydana getiren makina komutlarını yorumlamak ve yerine getirmek için gerekli olan tüm mantıksal devreleri içerir [2].

2.2.1 Merkezi işlem birimi

Bilgisayarın kalbi veya beyni olarak adlandırılan mikroişlemciler aynı zamanda merkezi işlem birimi olarak da anılırlar. Merkezi işlem birimi genel olarak aşağıdaki işlemleri yapar:

- Sistemdeki bütün elemanlar ve birimlere zamanlama ve kontrol sinyali sağlar,
- Bellekten komut veya veri alıp getirir ve işler,
- Veriyi giriş/çıkış birimlerine ya da oradan kendisine aktarır,
- Komutların kodunu çözer,
- Komutla birlikte aritmetik ve mantık işlemlerini gerçekleştirir,

Intel ve Motorola firmalarının üretmiş olduğu mikroişlemcilerde yukarıda bahsedilen işlemleri gerçekleştirmek üzere üç ana bölüm vardır. Bunlar; Kaydediciler, Aritmetik Mantık Birimi (ALU) ve Zamanlama-Kontrol Birimidir.

2.2.1.1 Kaydediciler

Mikroişlemcinin içinde işlem yaparken geçici olarak verilerin saklandıkları saklayıcılardır. Mikroişlemci içerisinde değişik işlemleri gerçekleştirmek üzere akümülatör, indis kaydedicileri, stack pointer, program counter gibi kaydediciler mevcuttur. Mikroişlemciler içerisinde işlem yapılırken bu kaydedicilere veri atılabilir, toplama, karşılaştırma ve kaydırma gibi bazı işlemler gerçekleştirilebilir [2].

2.2.1.2 Aritmetik mantık birimi (ALU)

ALU, yürütülmekte olan komutta belirtilen iki değer üzerinde aritmetik ve lojik işlemleri yerine getirir. Bu iki değerden biri daima özel bir mikroişlemci kaydedicisi olan ve A kaydedicisi olarak adlandırılan kaydedicinin o anki içeriğidir. Diğer ALU girişi ise; mikroişlemcideki genel amaçlı kaydedici grubundan bir başka kaydedicidir [2].

2.2.1.3 Zamanlama-kontrol birimi

Mikroişlemcinin kontrol birimi, mikroişlemcinin içinde ve dışında olan bütün veri aktarımlarını ve ALU işlemlerini kontrol eder ve çevre birimlerle eş zamanlama için gerekli sinyallemeyi sağlar. Ayrıca, kontrol birimi ALU'da en son yapılan aritmetik ve mantıksal işlemin sonucunu yansıtan bayrakları giriş olarak alır [2].

2.2.2 Giriş-çıkış birimi (G/Ç)

Mikroişlemcinin dış dünya ile ilişkisinin sağlandığı ünitedir. Mikroişlemciye verilen bilgiler bu ünite yolu ile işlemci içerisine alınırken, CPU ve diğer birimlerde işlenen veriler yine bu ünite sayesinde dış ortama aktarılırlar [2].

2.2.3 Bellek

Bellek bir mikrobilgisayar sistemi içerisinde bilgileri saklar. Bellek iki bölüme ayrılmıştır. Bir bölümü programı saklar, diğer bölüm ise mikro bilgisayara gerek duyduğu bilgi ya da veriyi tutmak için kullanılır.

Genel amaçlı bir mikrobilgisayar sisteminde program sıklıkla değiştirilir. Çamaşır makinası denetleyicisi gibi bir mikroişlemci tabanlı sistemde program, fabrikada üretilirken yüklenmiştir ve asla değişmez. Bu tür bir uygulama için program ROM adı verilen özel bir bellek türünde saklanır.

ROM (Sadece okunur bellek): Program deyimleri, veri vb. sadece ROM bellekten kopya edilebilir. Yapımcı veya kullanıcı tarafından ROM belleğe kaydedilen hiçbir bilgi değiştirilemez ve bu tür belleklere yeni kayıt yapılamaz. ROM belleklerden başka bir de RAM bellekler vardır.

RAM (Rastgele erişimli bellek): Kural olarak RAM'lar, üzerinde yazma ve okuma işlemleri yapılabilen bellekler olarak tanımlanmıştır. Bir başka deyişle RAM, değişken veri saklayabilen bir rastgele erişimli bellektir. RAM, bir elektronik devre olduğu için, güce gereksinim duyar. Güç kaynağı kesildiği anda çok anlık olsa bile saklanan bir veri yitirilir. Sistem anahtarı kapatıldığı anda, belleğin içeriği yitirilmiş olur. Bu nedenle bu belleğe kalıcı olmayan saklama ortamı adı verilir. Başka bellek türleri de vardır. Örneğin, EPROM, silinebilir programlanabilir ROM'dur. EPROM, programcıya bir yonga satın alıp bunu programlama imkanı verir.

2.3 Mikrodenetleyiciler

Bir yazılım olmadan hiçbir işe yaramayan, ancak içerisine yazılan program vasıtasıyla istenilen bir işlemi gerçekleştiren kontrol elemanıdır. Mikrodenetleyici yazılım olması halinde neredeyse sınırsız bir kullanım alanına sahiptir.

Aslında mikrodenetleyici bir bakıma bir bilgisayardır. Her nekadar bir klavyesi, monitörü, kasası ve bunun gibi çevre birimleri olmasa da bir bilgisayarın yaptığı herşeyi yapabilir. Örneğin her bilgisayarın bir merkezi işlem ünitesi (CPU:Central

Processing Unit) vardır ve bu ünite makine kodlarını bizim anlayabileceğimiz karakterlere dönüştürür, programları yorumlar, işler, düzenler, bilgisayarın çeşitli birimleriyle irtibat kurar. Ve bu işlemleri yaparken bazı değişkenleri ve geçici olarak elde ettiği bilgileri sakladığı bir rastgele erişimli hafızaya (RAM:Random Access Memory) ihtiyaç duyar. Ayrıca bilgisayarların dış dünyayla bilgi alış ve verişlerinde kullandıkları bazı giriş ve çıkış üniteleri bulunmaktadır. Örnek olarak fare ve klavye giriş yaptığımız elemanlara, monitör ve yazıcı çıkış aldığımız elemanlara birer örnektir. Bilgilerimizi kaydettiğimiz harddiskler ise hem giriş hem de çıkış elemanı olarak çalışmaktadırlar. Aynen bilgisayarda olduğu gibi mikrodenetleyiciye de fare ve klavye gibi çevre elemanlarının işlemlerini nispeten de olsa yerine getirecek elemanlar ekleyerek küçük bir bilgisayar gibi kullanmamız mümkündür [2].

Bir mikrodenetleyici genel olarak aşağıdaki birimlerden oluşur:

- CPU (Merkezi işlem ünitesi - Central Processing Unit)
- RAM (Rastgele erişimli bellek-Random Access Memory)
- EPROM/PROM/ROM (Silinir, yazılır sadece okunur bellek-Erasable Programmable Read Only Memory)
- I/O (Giriş/çıkış - Input/Output) - seri ve paralel
- Timers (Zamanlayıcılar)
- Interrupt controller (Kesmeler)

Sadece kullanılacak işe uygun özellikleri bulunan bir mikrodenetleyici seçildiğinde maliyet nispeten düşmektedir. Mikrodenetleyicilerde işlemler ve komutlar bit bit kontrol edilebildiğinden giriş ve çıkış birimleri ve kesmeler çok etkin bir şekilde kullanılabilmektedir. Şu an kullandığımız masaüstü veya dizüstü bilgisayarlar genel amaçlı bilgisayarlardır ve binlerce programı çalıştırabilirler. Mikrodenetleyiciler ise özel amaçlı bilgisayarlardır ve programlandıkları işi en iyi şekilde yaparlar. Bunun dışında;

- Mikrodenetleyiciler sadece bir iş için programlanmışlardır ve bu program içlerindeki ROM'da değişmemek üzere saklı bulunur.
- Mikrodenetleyiciler düşük güçte çalışan yongalardır. Bir bilgisayar 50W

civarı güç harcarken mikrodenetleyiciler sadece 50 miliWatt civarında güç harcarlar.

- Mikrodenetleyicilere sadece girdi yapılmaz aynı zamanda çıktı da alınabilir. LED göstergelerle, sıvı kristal göstergelerle, ikaz sesleriyle vb. Örneğin televizyo uzaktan kumandasında bulunan mikrodenetleyicileri incelersek: Kumandada bulunan mikrodenetleyici tuşlara basıldığında girdisini almış olur ve bunu televizyondaki alıcı mikrodenetleyiciye gönderir. Ve o mikrodenetleyici de gelen sinyale göre çıkış vererek ya kanal seçer, ya ses ayarı yapar ya da televizyonla ilgili işlemleri yapar.
- Mikrodenetleyiciler genelde küçük ve düşük fiyatlı yongalardır. Bir çok parçadan oluşan karmaşık bir devrenin kolayca küçük boyutlara ve maliyete indirilmesini sağlar.

Ayrıca mikrodenetleyiciler belleklerine göre de çeşitlilik gösterirler;

2.3.1 EEPROM bellekli mikrodenetleyiciler

Elektriksel olarak silinebilen ve yazılabilen belleklerdir. Çoğu mikrodenetleyicilerde sınırlı sayıda bulunan EEPROM'lar, bir defadan fazla yazılıp silinebildikleri için oldukça kullanışlıdır.

2.3.2 Flash (EPROM) bellekli mikrodenetleyiciler

Flash bellekler EEPROM'lardan daha hızlı ve daha çok yazma silme işlemine izin vermeleri yönünden üstündürler.

2.3.3 OTP bellekli mikrodenetleyiciler

Bir kez programlanabilen mikrodenetleyicilerdir. OTP bir kez programlanabilen bir ROM'dur. Program bir EPROM programlayıcı ile bir kez yazdıktan sonra silinemez veya değiştirilemez. Bu yöntem program artık tamamen hazır olduğunda ve bütün hatalardan arındırıldıktan sonra kullanılır.

2.4 Mikrodenetleyicinin Mikroişlemciye Olan Üstünlükleri

Mikroişlemcinin kullanımı ve mikroişlemcili sistemin tasarımı mikrodenetleyicili sisteme göre hem daha masraflı hem de daha karmaşıktır. Mikrodenetleyicili bir sistemin çalışması için elemanın kendisi ve bir osilasyon kaynağının olması yeterlidir. Mikrodenetleyicinin ihtiyaç duyduğu önbellek ve giriş çıkış birimi bir yonga içerisinde bulunmaktadır. Ancak mikroişlemcili bir sistemde önbellek harici olarak bulunur.

2.5 PIC Mikrodenetleyicisi

Harvard mimarisindeki ilk mikrodenetleyici ünitesi, General Instruments firması tarafından 1970'lerin ortalarında üretilen Signetics 8X300 modeliydi. Bu 16 bitlik CP1600 MPU için programlanabilen giriş/çıkış portu olmak üzere Peripheral Interface Controller (Çevrebirim ara yüz denetleyicisi - PIC) olarak tasarlandı.

General Instruments firması mikroelektronik bölümünü sattı ve bu bölüm 1988 yılında Arizona Microchip Technology adıyla yeni bir firmaya dönüştü. Microchip'in (www.microchip.com) ana ürünü, bugün de hala öyle olan, PIC serisi mikrodenetleyicilerdir. 1989'da ilk piyasaya sürülen aile PIC16C5X serisiydi. Bu Harvard mikrodenetleyiciler 33 komutluydu. Bütün komutlar 12-bit word olarak kodlanıyordu. Azaltılmış Komut Kümesi (Reduced Instruction Set Computer - RISC) temelli olan komut seti hızlı, etkili ve ucuz işlemci üretimini sağladı. PIC16C5XX 12-bit çekirdekli ailede 512 ve 2048 komutluk tek sefer programlanabilen (One Time Programmable (OTP)) EEPROM Program belleği, 25-73 bayt veri belleği, 18-28 pinli paketlerde 12 veya 20 giriş/çıkış pini ve 8-bit zamanlayıcı gibi özellikler bulunmaktaydı. PIC12CXXX ailesi bunların 8-pinlik eşdeğerleridir [3].

1992 yılında 14-bitlik çekirdeğe sahip PIC16CXXX ailesi daha fazla program alanının ve kesme (interrupt) işlemleri yanında A/D çeviriciler, 16 bit sayıcılar gibi çevre birimlerinin kullanımına olanak sağladı. Bu ailedeki RISC komut seti de 12-bit çekirdektekilerle hemen hemen aynıydı ve 35 komuttan oluşuyordu. 1997'de çarpma yapabilen bir ALU ve ileri arabirim yeteneklerine sahip 16-bit PIC17CXXX ailesi piyasaya sunuldu. Ardından 1999 yılında da genişletilmiş 16-bit çekirdekli

PIC18CXXX ailesi sunuldu. Bu ailedeki işlemcilerde komut sayısı 77 idi ve bu yüksek-seviye dillerin derleyicilerin ihtiyaçlarını daha fazla karşılıyordu [3].

2.5.1 Özellikleri

Bilgisayar denetimi gerektiren bir uygulamayı geliştirirken seçilecek mikrodenetleyicinin ilk olarak tüm istenilen özellikleri yerine getirip getirmediğine, daha sonra da maliyet düşüklüğüne bakılmalıdır. Ayrıca, yapılacak uygulamanın devresini kurmadan önce seçilen mikrodenetleyicinin desteklediği bir yazılım üzerinde simülasyon yapılıp yapılamayacağı da dikkate alınmalıdır.

2.5.2 Kod verimliliği

PIC, Harvard mimarisi temelli 8 bit'lik bir mikrodenetleyicidir. Bu, bellek ve veri için ayrı yerleşik bus'ların bulunduğu anlamına gelir. Böylelikle akış miktarı veriye ve program belleğine eşzamanlı erişim sayesinde arttırılmış olur. Geleneksel mikrodenetleyicilerde veri ve programı taşıyan bir tek yerleşik bus bulunur. Bu, PIC'le karşılaştırıldığında işlem hızını en az iki kat yavaşlatır.

2.5.3 Güvenilirliliği

Tüm komutlar 12 veya 14 bit'lik bir program bellek sözcüğüne sığar. Yazılımın, programın VERİ kısmına atlamaya ve VERİ'yi komut gibi çalıştırmasına gerek yoktur. Bu işlem 8 bit'lik bus kullanan ve Harvard mimarisi temelli olmayan mikrodenetleyicilerde gerçekleşmektedir.

2.5.4 Komut seti

PIC ailesinde yazılım gerçekleştirmek için çok az komut öğrenmek yeterlidir. PIC tarafından kullanılan komutların hepsi kaydedici (register) temellidir CALL, GOTO ve bit test eden BTFSS, INCFSZ gibi komutlar dışında, her bir komut, tek bir çevrimde çalışır. Mikrodenetleyicinin çalışmasını ve işletmesini sağlayan bilgidir. Başarılı bir uygulama veya ürün isteniyorsa yazılım hatasız olmalıdır. Yazılım C ve Pascal gibi çeşitli yüksek seviyeli dillerde veya düşük seviyeli assembly dilinde yazılabilir.

2.5.5 Hızı

PIC, osilatör ve yerleşik saat yolu (clock bus) arasına bağlı yerleşik bir (divide by 4) 4'lü bölünme'ye sahiptir. Bu, özellikle 4 MHz'lik kristal kullanıldığında komut sürelerinin hesaplanmasında kolaylık sağlar. Her bir komut döngüsü 1 ms'dir. PIC oldukça hızlı bir mikrodenetleyicidir. Örneğin 5 milyon komutluk bir programın, 20 MHz'lik bir kristalle adımlanması yalnız 1 saniye sürer. Bu süre 386 SX 33 hızının neredeyse 2 katıdır.

2.5.6 Statik işlem

PIC tamamıyla statik bir mikrodenetleyicidir. Başka bir deyişle sistem saati durdurulduğunda, tüm kaydedici içerikleri korunur. Pratikte bunu tam olarak gerçekleştirmek mümkün değildir. PIC uyutma moduna getirildiğinde, saat durur ve PIC uyutma işleminden önce hangi durumda olduğunu ifade eden çeşitli bayrakları kurar. PIC uyuma modunda yalnızca 1 µA'den küçük bir değere sahip bekleme (standby) akımı çeker.

2.6 PIC 18F452

PIC 18F452 mikrodenetleyicisi, Microchip'in 18FXX2 adıyla ürettiği PIC ailesinin en gelişmiş üyesidir. 40 bacaklı bir mikrodenetleyicidir. A, B, C, D, E, olmak üzere toplam 5 portu vardır. Bütün portlar dijital I\O amaçlı kullanılabilir. Bunu yanında A ve E portları istenildiğinde A\D çevrim için analog giriş olarak da kullanılabilir. 18FXX2 serisi PIC sekiz değişik osilatör çeşidi ile kullanılabilir. Kullanıcı FOSC2, FOSC1 ve FOSC0'ı aşağıdaki sekiz modu kullanarak programlayabilir [4].

- LP Low Power Crystal
- XT Crystal/Resonator
- HS High Speed Crystal/Resonator
- HS+PLL High Speed Crystal/Resonator with PLL enabled
- RC External Resistor/Capacitor
- RCIO External Resistor/Capacitor with I/O pin enabled

- EC Exterrnal Clock
- ECIO Exterrnal Clock with I/O pin enabled

XT, LP , HS veya HS+PLL osilatör modlarında kristal veya seramik rezonatör OSC1 ve OSC2 pinlerine bağlanır.

PIC18F452 işlemcisinin başlıca özellikleri:

- 32 Kbyte Flash hafıza
- 256 byte EEprom
- 40 MHz'e kadar osilatör girişi
- 3 adet harici kesme (interrupt) girişi
- 8 bit ve 16 bit 4 adet zamanlayıcı
- 2 adet Capture & compare girişi
- RS 232 ve RS 485 destekli Usart bilgi aktarma modülü
- 10 bit Analog / Dijital dönüştürücü
- Düşük güç tüketimli ve yüksek hızlı " FLASH / EEPROM " teknolojisi

2.6.1 PIC18F452'nin besleme uçları ve beslenmesi

PIC18F452'nin besleme gerilimi 11, 12 ve 31, 32 numaralı bacaklardan uygulanmaktadır. 11 ve 32 numaralı Vdd ucu +5 V'a ve 12, 31 numaralı Vss ucu toprağa bağlanır. PIC'e ilk defa enerji verildiği anda meydana gelebilecek gerilim dalgalanmaları nedeniyle, oluşabilecek istenmeyen arızaları önlemek amacıyla 100nF'lık dekuplaj kondansatörünün devreye bağlanması gerekmektedir. PIC'ler CMOS teknolojisi ile üretildiklerinden 2 ile 6 Volt arasında çalışabilmektedirler. +5 V'luk bir gerilim ise ideal bir değer olmaktadır.

2.6.2 PIC18F452'nin reset uçları

Kullanıcının programı kasti olarak kesip başlangıca döndürebilmesi için PIC'in 1 numaralı ucu MCLR olarak kullanılmaktadır. MCLR ucuna 0 Volt uygulandığında

programın çalışması başlangıç adresine döner. Programın ilk başlangıç adresinden itibaren tekrar çalışabilmesi için, aynı uca +5 Volt gerilim uygulanmalıdır.

2.6.3 I/O portları

PIC18F452 ayrı ayrı beş port bulunmaktadır. I/O pinlerinin bazıları çevresel arabirimlerden alternatif fonksiyonları için de kullanılabilir. Pin eğer çevresel olarak kullanılırsa I/O pini olarak kullanılamaz.

2.6.3.1 Port A

Her bir biti bağımsız olarak giriş veya çıkış olarak tanımlanabilmektedir. 6 bit genişliğindedir (PICF84'de 5 bittir). RA0, RA1, RA2, RA3, RA4 ve RA5 bitleri analog/dijital çevirici olarak konfigüre edilebilmektedir. Buna ek olarak RA2 ve RA3 gerilim referansı olarak da konfigüre edilebilmektedir (bu durumda bu bitler aynı anda A/D çevirici olarak kullanılamamaktadır). İşlemciye ilk defa gerilim uygulandığında RA4 hariç diğer beş PORTA biti A/D çeviricidir. Eğer RA portunun bazı bitlerini sayısal giriş/çıkış olarak kullanmak istersek ADCON1 kaydedicisinde değişiklik yapmamız gerekmektedir. Yazmaçlardaki değişikliklerin nasıl yapılacağı, tüm devrenin katalogunda bulunmaktadır.

2.6.3.2 Port B

Her bir biti bağımsız olarak sayısal giriş veya çıkış olarak tanımlanabilmektedir. 8 bit genişliğindedir. B portunun her bacağı dahili bir dirençle Vdd'ye bağlıdır. Bu özellik varsayılan olarak etkin değildir. Ancak OPTION kaydedicisinin 7. bitini 0 yaparak B portunun bu özelliğini etkinleştirilebilir. RB4 - RB7 bacakları aynı zamanda bacakların sayısal durumlarında bir değişiklik olduğunda INTCON kaydedicisinin 0. biti olan RBIF bayrağını 1 yaparak kesme oluşturmaktadır. Bu özelliği, işlemci uyuma (sleep) konumundayken, devreye bağlı tuş takımında her hangi bir tuşa basıldığında işlemcinin yeniden etkinleşmesi için kullanılabilir. Bütün bunların yanı sıra RB6 ve RB7 yüksek gerilim programlama, RB3 ise düşük gerilim programlama modlarında da kullanılmaktadır [4].

2.6.3.3 Port C

Her bir biti bağımsız olarak sayısal giriş veya çıkış olarak tanımlanabilmektedir. 8 bit genişliğindedir. Tüm port bacakları Schmitt Trigger girişlidir.

2.6.3.4 Port D

Her bir biti bağımsız olarak sayısal giriş veya çıkış olarak tanımlanabilmektedir. 8 bit genişliğindedir. Tüm port bacakları Schmitt Trigger girişlidir. TRISE yazmacının 4.biti olan PSPMODE bitini 1 yaparak paralel slave modda kullanılabilir. Bu fonksiyon aracılığıyla 8 bit genişliğindeki herhangi bir mikroişlemci bus'ına bağlanabilir.

2.6.3.5 Port E

Her bir biti bağımsız olarak giriş veya çıkış olarak tanımlanabilmektedir. 3 bit genişliğindedir. RE0, RE1 ve RE2 bacaklarında Schmitt Trigger giriş tamponları vardır. Her bir bacak analog/dijital çevirici olarak konfigüre edilebilmektedir. Eğer port D, paralel slave port olarak konfigüre edilirse, RE0, RE1 ve RE2 bacakları port D'nin bağlandığı mikroişlemci bus'ına sırasıyla READ, WRITE ve CHIP SELECT kontrol girişleri olarak kullanılabilmektedir. Bunun için TRISE uygun biçimde ayarlanmalıdır. İşlemciye ilk defa gerilim uygulandığında port E'nin üç biti de A/D çeviricidir. Eğer E portunun bazı bitleri sayısal giriş/çıkış olarak kullanılmak istenirse ADCON1 kaydedicisinde değişiklik yapılması gerekecektir [4].

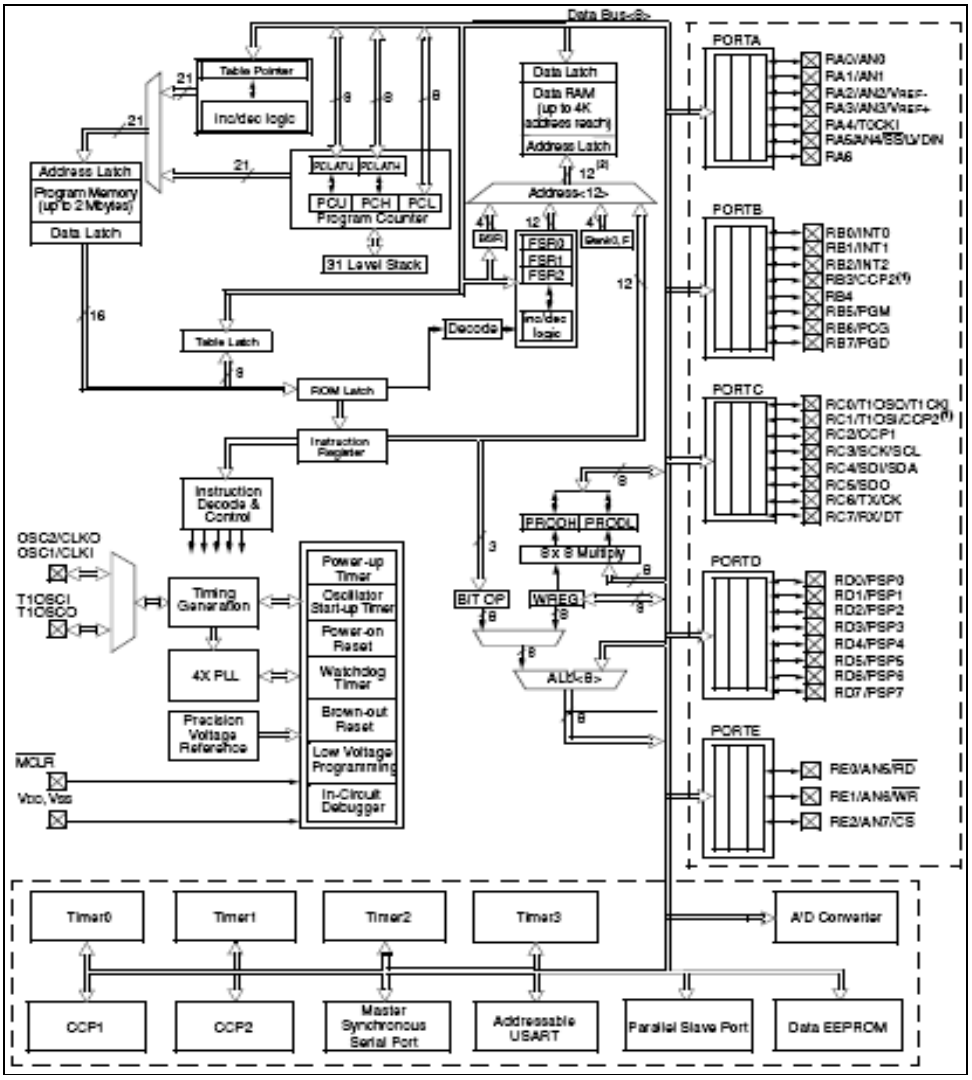
2.6.4 Yakalama / karşılaştırma ve PWM birimi

Her capture/compare ve pwm birimi 16 bitlik yakalama kaydedicisi, 16 bitlik karşılaştırma kaydedicisi veya 16 bitlik PWM (darbe genişlik modülasyonu) kaydedicisi olarak kullanılmaktadır. Yakalama modunda, TMR1 kaydedicisinin değeri, RC2/CCP1 bacağının durumunda bir gelişme olduğunda CCPR1H:CCPR1L yazmaçlarına yazılmakta ve PIR1 kaydedicisinin 2. biti olan CCP1IF kesme bayrağı 1 olmaktadır. RC2 bacağının durumu, her düşen kenarda, her yükselen kenarda, her yükselen 4. veya 16. kenarda kontrol edilecek şekilde CCP1CON kaydedicisi aracılığıyla ayarlanarak konfigüre edilebilir.

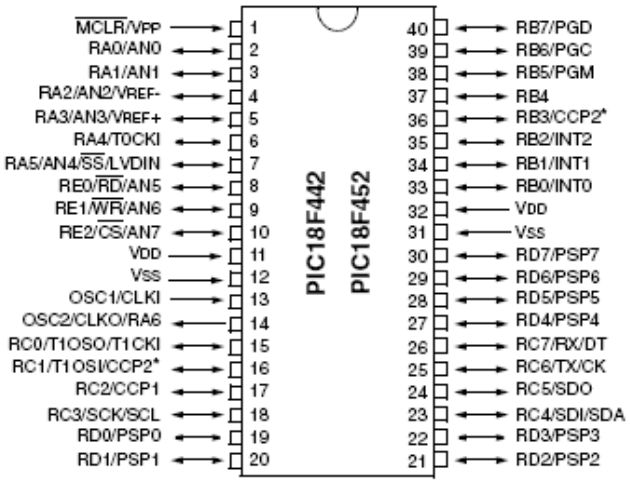
Karşılaştırma modu ise CCPR1 kaydedicisindeki 16 bitlik değer düzenli olarak TMR1 kaydedici değeriyle karşılaştır ve bir eşitlik olduğunda RC2/CCP1 bacağı CCP1CON kaydedicisinde yaptığımız ayara göre 1, 0 olur veya durumunu korur. PWM modunda ise RC2/CCP1 bacağı 10 bit çözünürlükte darbe genişlik modülasyonlu bir işaret üretecek şekilde konfigüre edilebilir. PR2 yazmacı darbe genişlik periyodunun tayininde kullanılmaktadır [4].

2.6.5 PIC 18F452 mimarisi

Şekil 2.2'de PIC18F452 entegresi iç yapısı, Şekil 2.3'de bacak yapısı ve Fotoğraf 2.1'de görünümü verilmiştir [4].



Şekil 2.2 PIC 18F452 iç yapısı



Şekil 2.3 18F452 bacak yapısı



Fotoğraf 2.1. PIC 18F452

BÖLÜM III

MPLAB IDE VE MPLAB C18 C COMPILER

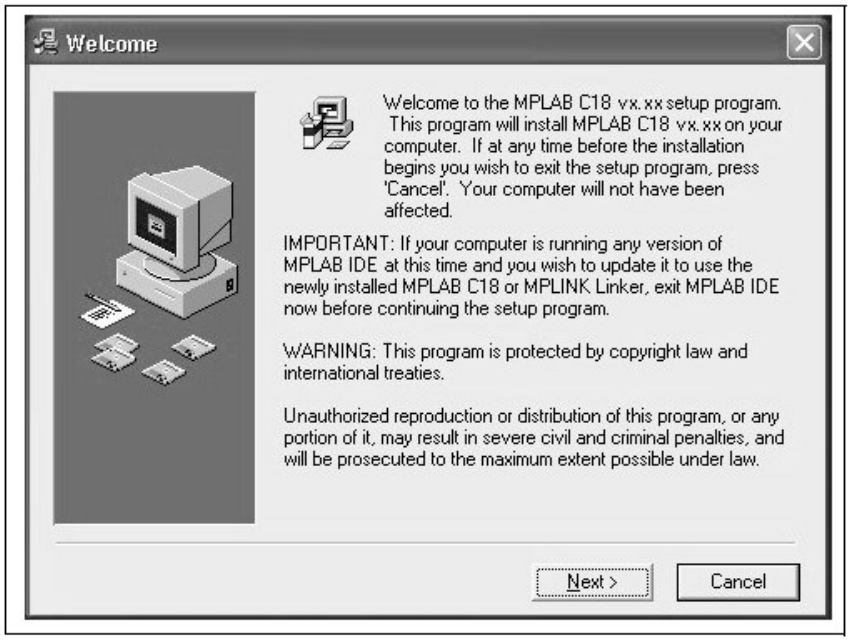
Bu bölümde, projeye ait PIC programını yazmak ve derlemek için kullanılan MPLAB ve MPLAB C18 C derleyicisi anlatılmaktadır.

3.1 MPLAB IDE

MPLAB, Microchip firması tarafından geliştirilmiş entegre tasarım ortamıdır. MPASM derleyicisi, assembly kodlarını yazmak için geliştirilmiş metin editörü, MPSIM simülatörü ve proje yöneticisi içeren MPLAB, tasarımcılar için oldukça yararlı bir araçtır. MPLAB tasarım ortamında yukarıda sözü edilen olanaklardan yararlanabilmek için assembly kodlarıyla program yazmak gerekir. Ancak diğer dillerin derleyicilerinin de son günlerde çok popüler oluşu, yeni versiyonlarında diğer firmaların derleyicilerinin de kullanılabilmesini olanaklı hale getirmiştir. Microchip firması Basic, Pascal ve C programlama derleyicilerini MPLAB ortamından kullanabilme olanağı sağlayarak metin editörü ve derleyicilerin aynı ortamda çalışmasına olanak sağlamıştır. HI-TECH, IAR, Byte Craft, B. Knudsen, CCS ve microEngineering Labs firmalarının derleyicileri için MPLAB entegrasyon ortamı sağlamaktadır. Bu çalışmada MPLAB C18 C derleyicisi kullanıldığından MPLAB IDE metin editörü simülasyon amaçlı kullanılmıştır. Bu bölümde MPLAB IDE programının kurulumuna yer verilmemiştir. Ancak Projenin kaynak kodlarının yazıldığı MPLAB C18 derleyisinin kurulumu, MPLAB IDE entegrasyon ortamında proje oluşturulması ve derlenmesine yer verilmiştir. C derleyicisi kurulmadan önce MPLAB IDE programının kurulduğu kabul edilmiştir.

3.2 MPLAB C18 Kurulumu

Program Microchip firmasının internet sitesinden indirilen 60 günlük deneme sürümü veya lisanslı olarak satılan CD'den kurulabilir. Her ikisinin de kurulumu aşağıda bahsedildiği şekliyle aynıdır.

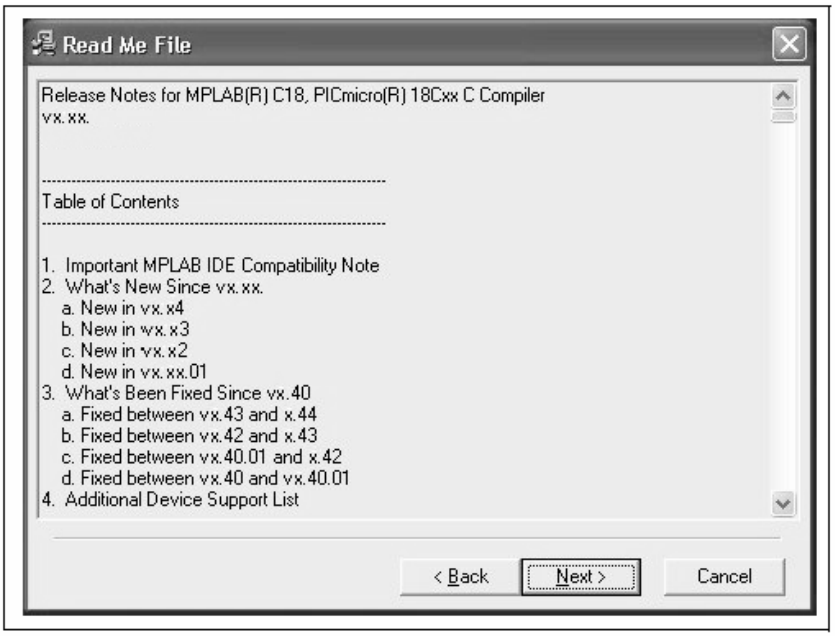


Şekil 3.1 MPLAB C18 kurulum başlangıcı

Şekil 3.1'de MPLAB C18 kurulum başlangıcı görülmektedir. Kurulum CD'si çalıştırıldıktan sonra ekrana birkaç diyalog adımı gelmektedir. Bunlardan sırası ile Welcome ekranı kurulumu yapılacak programın versiyon numarasını göstermektedir. Şekil 3.2'de kurulum başlangıcındaki lisans anlaşması ekranı görülmektedir.

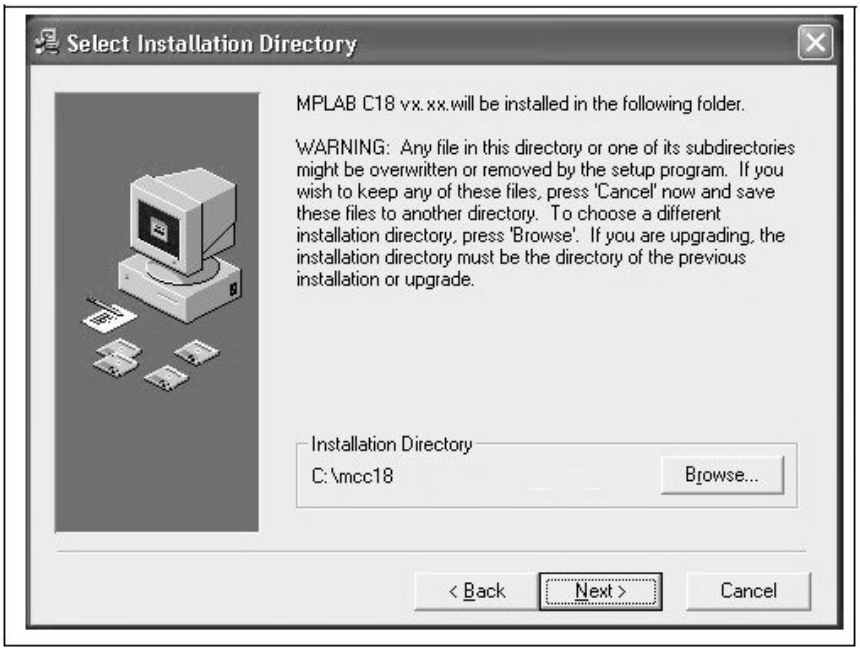


Şekil 3.2 Lisans anlaşması



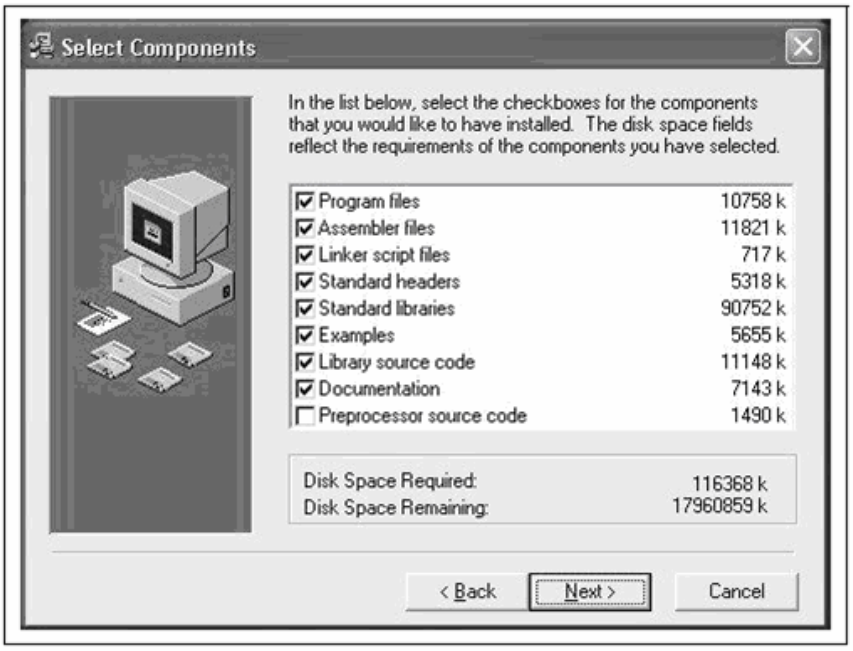
Şekil 3.3 Okuma dosyası

Şekil 3.3'te yazılıma ait okuma dosyası görülmektedir. Bu dosya yazılım versiyonunun desteklediği cihazları ve yeni özellikleri hakkında önemli bilgi vermektedir. Normal olarak değişen her yazılım versiyonu ile birlikte kurulum okuma dosyası da değişmektedir. Okuma işlemi yapılır ve sonraki adıma geçilir.



Şekil 3.4 Kurulum dizini

Kurulum dizini Şekil 3.4'te gösterildiği gibi olmalıdır. Aksi takdirde derleme sırasında bir takım hata mesajları ile karşılaşılabilir.



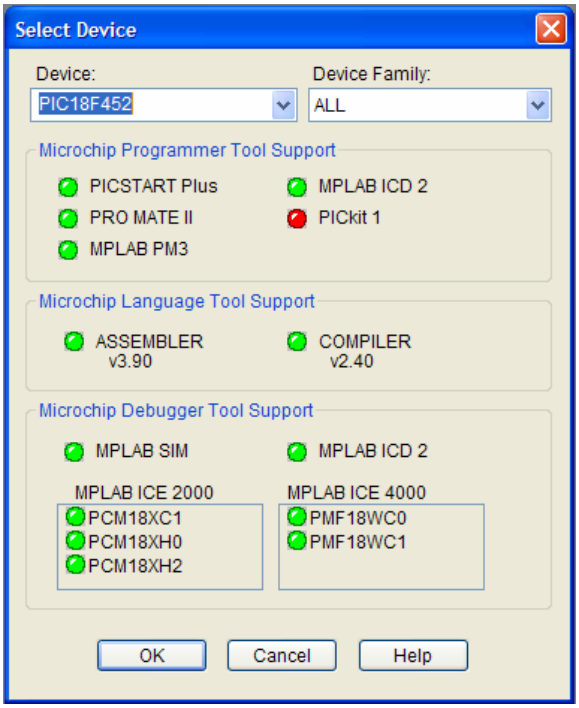
Şekil 3.5 MPLAB IDE ve MPLAB C18 entegrasyonu için gerekli dosyalar

Kurulumun bu bölümünde C derleyicisinin entegrasyon ortamında doğru çalışması için gerekli olan dosyalar ile kullanıcıya kolaylık sağlayan örnek ve doküman dosyaları da kurulmaktadır. Yukarıda anlatılan kurulum basamaklarından sonra proje kodlarının yazılacağı ortam hazır duruma gelmiştir. Şekil 3.5'te kurulumda gerekli dosyalar görülmektedir.

3.2.1 Proje oluşturulması

MPLAB IDE'nin Project Manager aracı adım adım yazıp derlemeyi sağlar. Bir MPLAB projesi sırayla şu adımlarla oluşturulur [5].

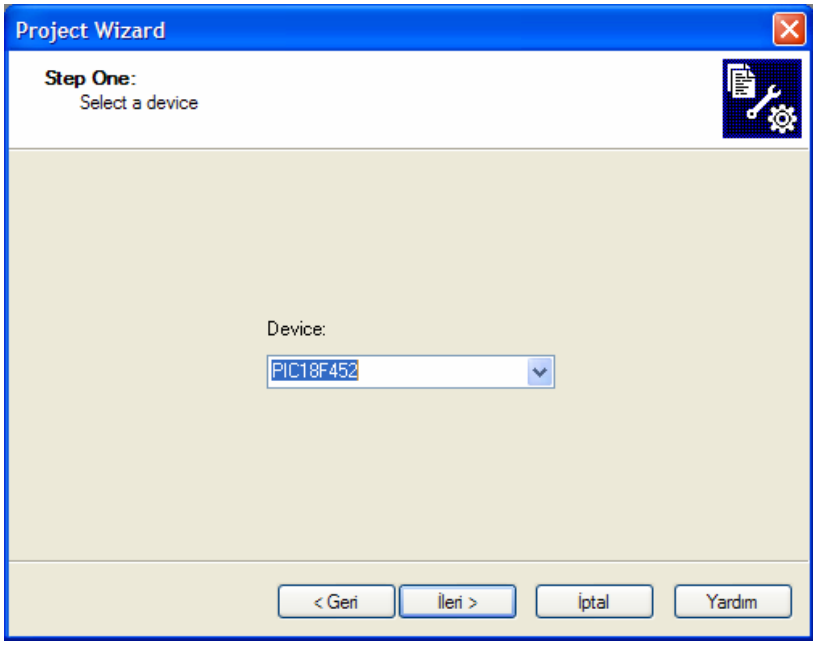
1. MPLAB IDE ekranındaki Configure menüsünden "Select Device" sekmesine tıklanır.
2. Ekrana gelen listeden projede kullanılacak PIC'in model numarası girilir ve OK tuşuna basılır.



Şekil 3.6 Kullanılacak cihazı seçin

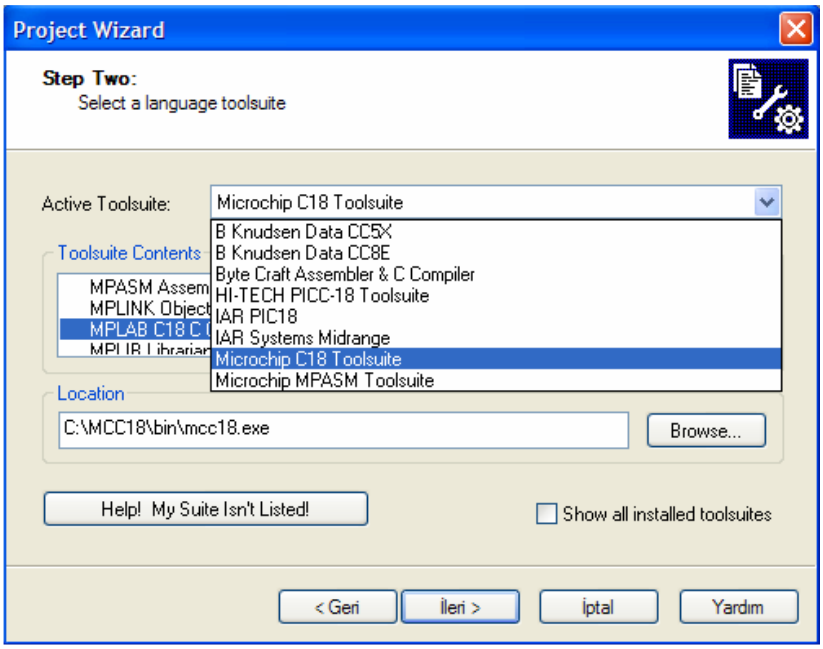
Şekil 3.6'da gözükmekte olan küçük baloncuklar MPLAB IDE'nin bu cihaz için hangi bileşenleri desteklediğini gösterir. Yeşil ışık tam olarak desteklendiğini gösterir. Sarı ışık MPLAB yazılımına yapılacak eklentilerle gerekli desteğin sağlanılabileceğini gösterir. Kırmızı ışık bileşenin desteklenmediğini gösterir.

Sıradaki adım proje sihirbazını kullanarak bir proje oluşturmaktır. Proje derlenecek olan dosyaların organize edilmesini sağlar. Project menüsünden Project Wizard seçilir. Bundan sonra ekrana gelen pencereden PIC seçimi yapılır. Menü daha önce seçilmiş cihazı gösterecektir. İlgili cihaz seçilerek ileri tuşuna basılır. Şekil 3.7'de seçilen ilgili cihaz görülmektedir. Şekil 3.8'de projenin derlenmesi için seçilen dil aracı görülmektedir. MPLAB IDE Şekil 3.8'de görüldüğü gibi birden fazla derleyiciyi destekler.

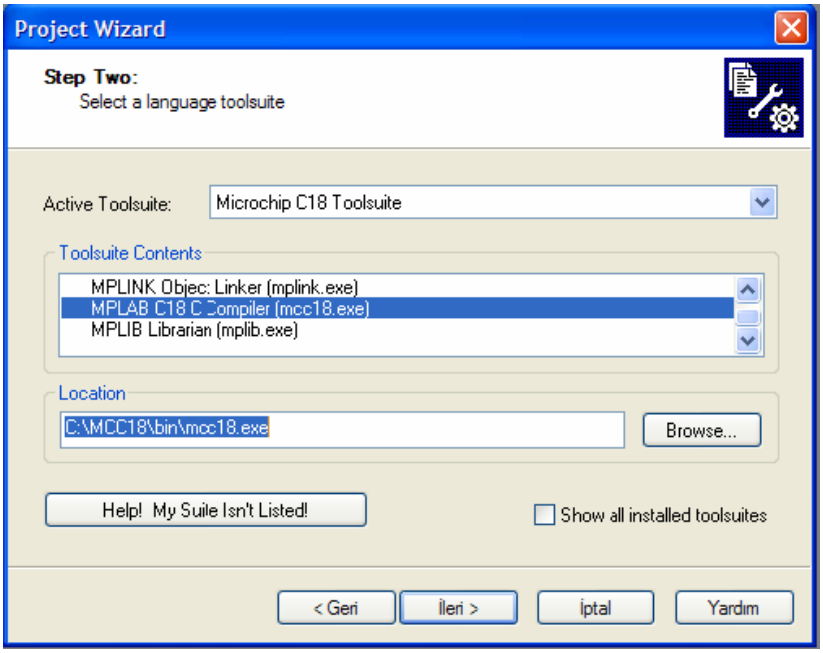


Şekil 3.7 Seçilmiş cihaz

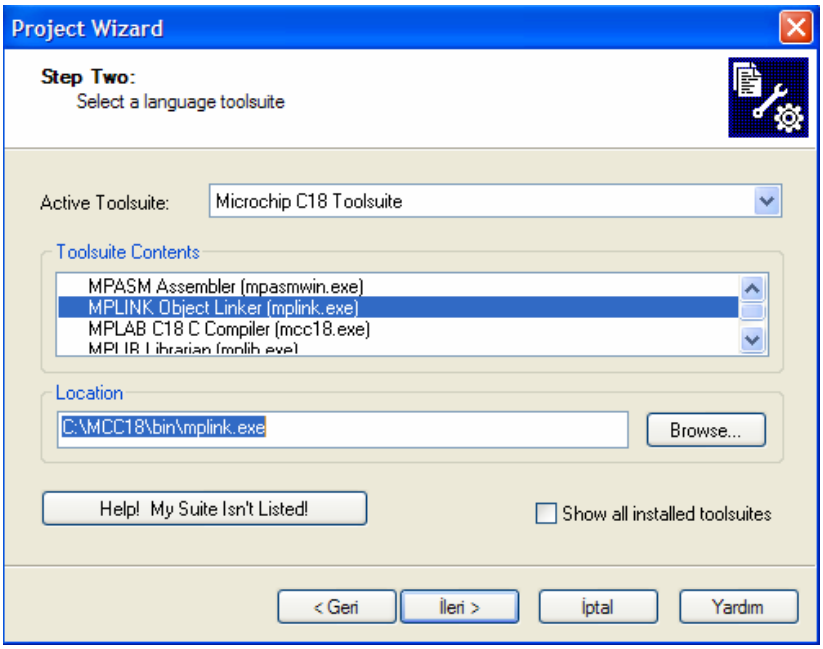
Daha sonra projenin derlemesini sağlayacak araç ve aracın çalışması için gerekli işletilebilir olan Şekil 3.9, Şekil 3.10 ve Şekil 3.11'deki programlar seçilerek kurulum yerleri proje yürütücüsüne gösterilir.



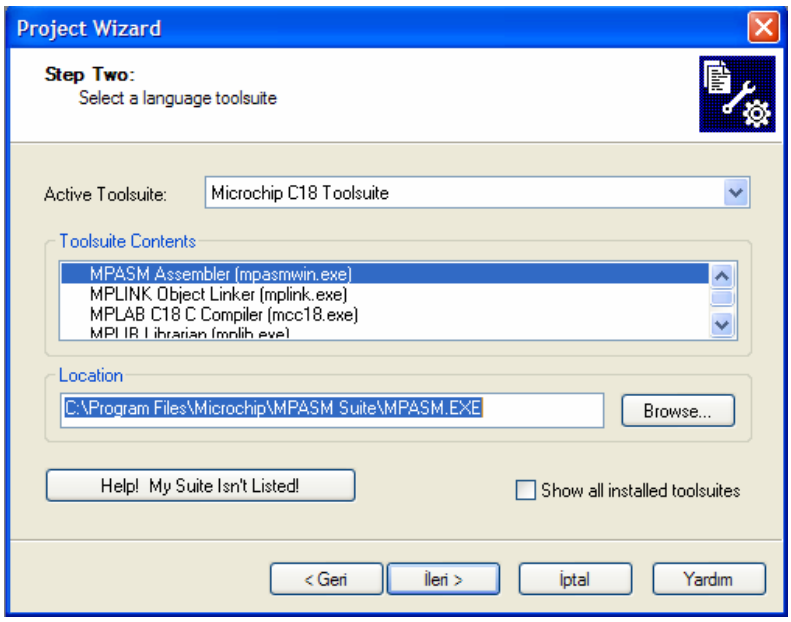
Şekil 3.8 Projede kullanılacak dil aracı



Şekil 3.9 İşletilebilir C18 derleyicisi ve kurulum dizini

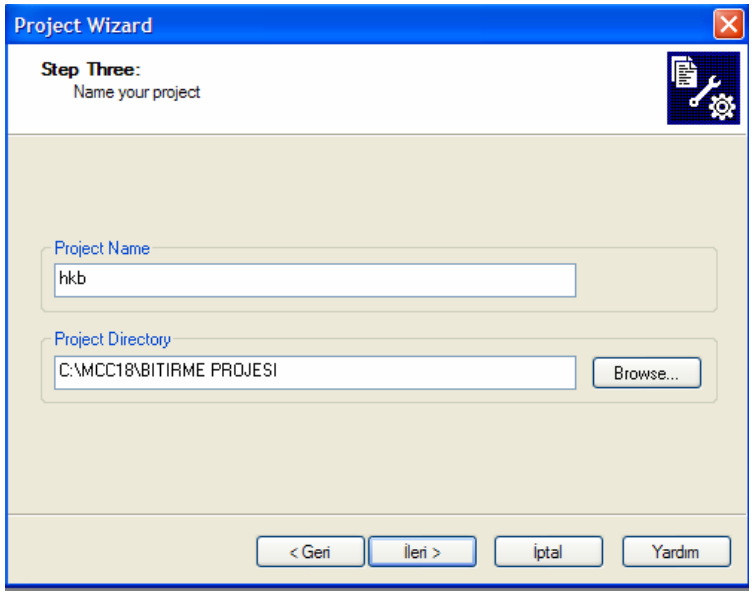


Şekil 3.10 C18 derleyicisi bağlayıcısı ve kurulum dizini



Şekil 3.11 C18 Assembly derleyicisi ve kurulum dizini

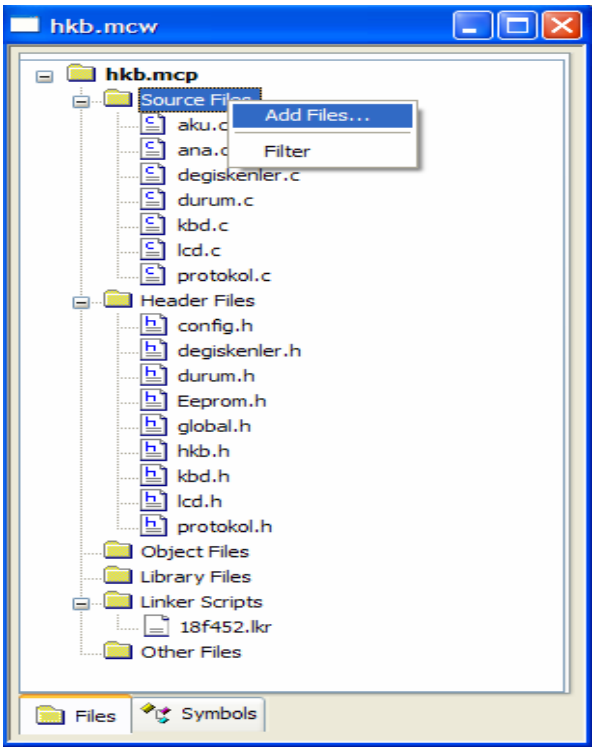
Sonraki adımda proje isimlendirip kaydedilir. Şekil 3.12'de proje isimlendirmesi görülmektedir.



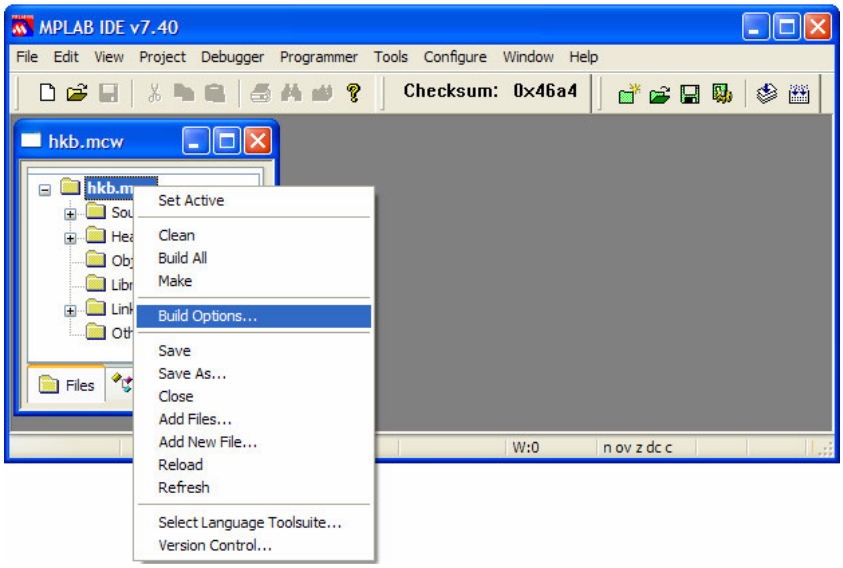
Şekil 3.12 Projeyi isimlendirme

Bu aşamadan sonra proje sihirbazı proje için dosya seçimine izin verir. İlgili dosya Add Files butonuna basılarak eklenir ve yanındaki kutucuk işaretlenir. Şekil 3.13'de projeye dosya ekleme işlemi görülmektedir.

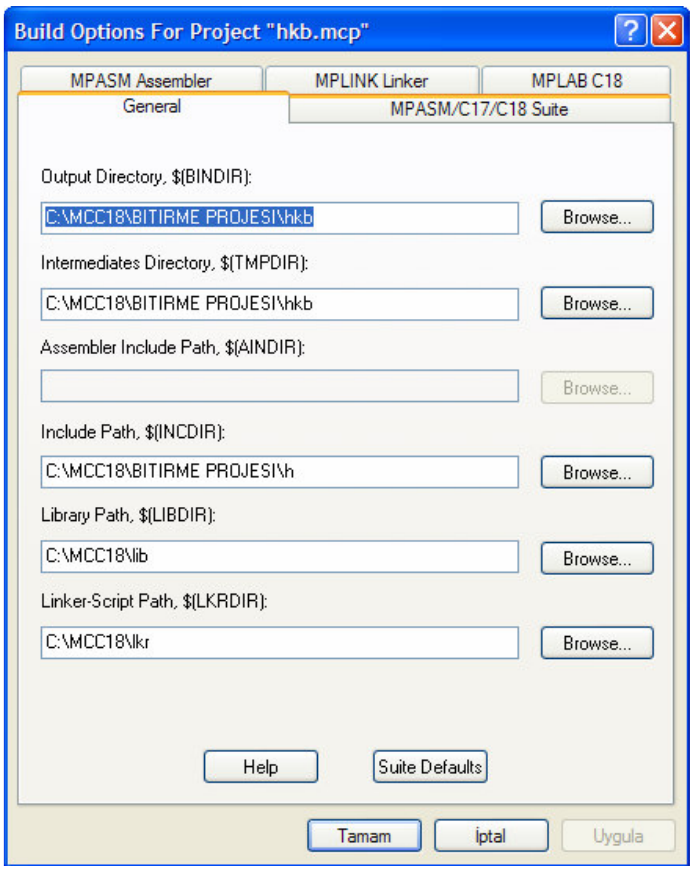
Şimdi sıra linker scripti eklemeye geldi. Linker script (bağlayıcı) dosyaları için de her PIC'e göre özel script dosyaları vardır. Örnek olarak bu projede 18F452 kullanıldığından 18F452.lkr dosyası seçilir. (Dosya yolu: C:\Program Files\Microchip\MPASM Suite\LKR)



Şekil 3.13 Projeye dosya ekleme

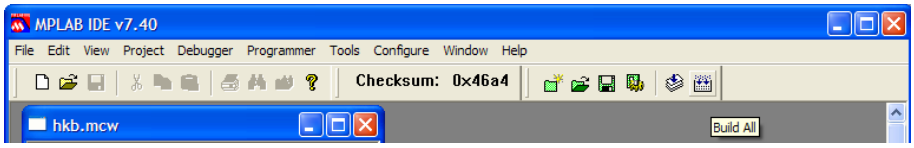


Şekil 3.14 Derleme özelliklerinin ayarlanması



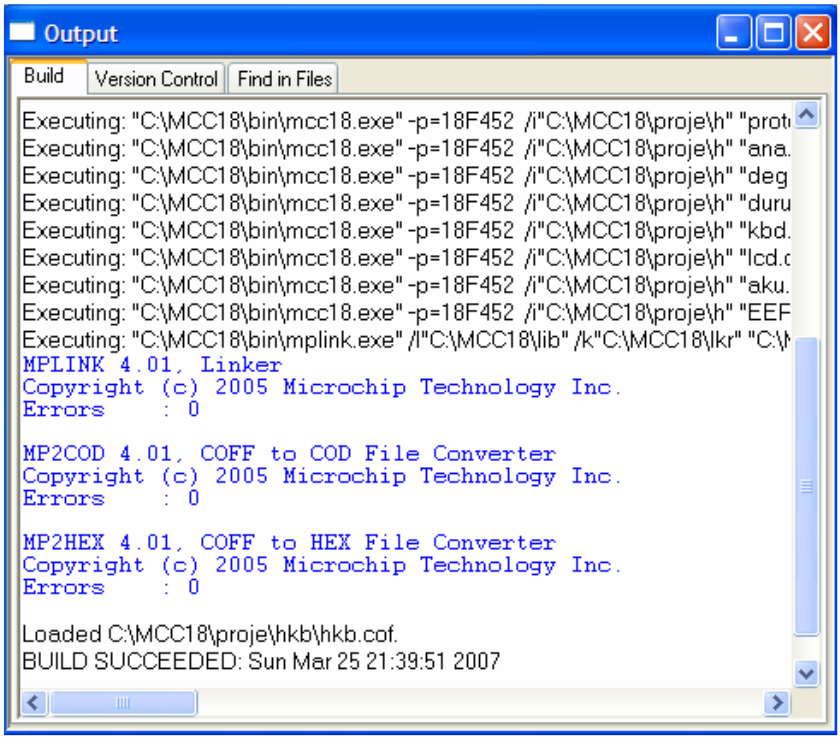
Şekil 3.15 Derlenecek projeye başlık, kütüphane ve bağlayıcı dosyalarının dahil edilmesi

Projeye kütüphane dosyaları, başlık dosyası dizini ve bağlayıcı dizini yeri Şekil 3.14 ve Şekil 15'te gösterildiği gibi derleme öncesinde tanıtılarak ayar yapılır.



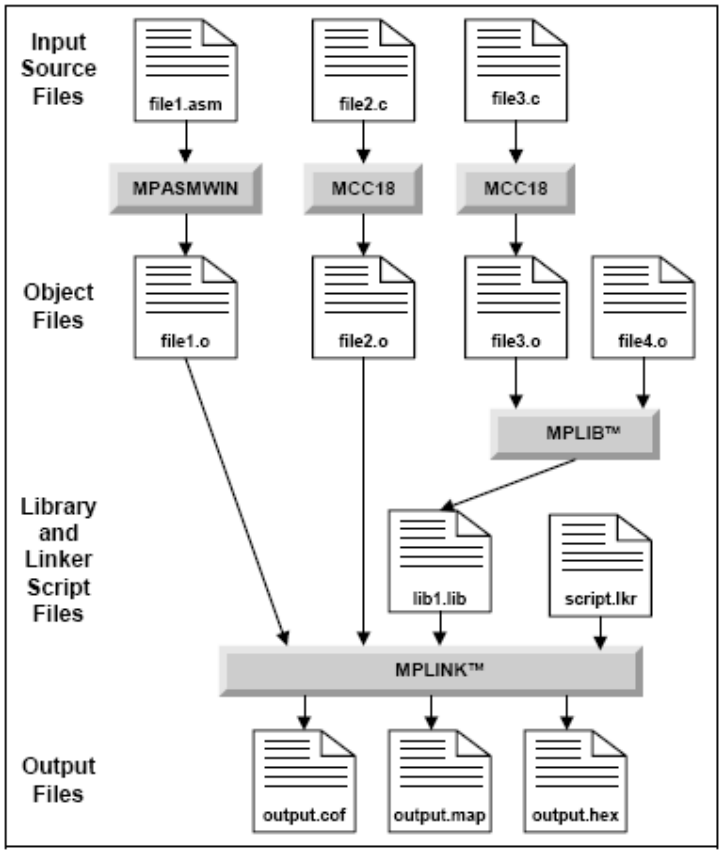
Şekil 3.16 Projenin derlenmesi

Ana menüden Project seçilip "Build All" seçeneğine tıklanır. Eğer kodda hiç hata yoksa derleme sonucu Şekil 3.17'deki gibi görüntülenecektir.



Şekil 3.17 Derleme sonucu

Eğer hata varsa output penceresinde bulunan hatalar görüntülenecektir. Hata durumunda output penceresinin uyarısına göre kullanılan kaynak dosyalar veya koddaki yazım hataları gözden geçirilmeli ya da uygun derleyicinin seçilip seçilmediğine dikkat edilmelidir.



Şekil 3.18 C18 derleyicisi akış şeması

Şekil 3.18'de C18 derleyicisinin çalışmasına ait akış diyagramı verilmiştir. C ve asm uzantılı yazılan kodlar MCC18 ve MPASMWIN işletim programlarınca object uzantılı dosyalara dönüştürülür. Gerekli kütüphane dosyaları ile birleştirme ve bağlama işlemi yapılarak hex, map ve cof uzantılı çıkış dosyaları elde edilir [5].

3.3 MPLAB C18 Derleyici Kütüphanesi ve Özellikleri

C programlama dili günümüzde en yaygın olarak kullanılan programlama dillerinden biridir. Bir PC ve bir mikrodenetleyici için yazılmış olan C derleyicileri her ne kadar birbirlerine benzeseler de aralarında bazı önemli farklar vardır. PC için yazılmış olan bir C derleyicisi genellikle daha büyük olup programcıya çok çeşitli fonksiyonlar sunmaktadır (örneğin dosya açıp kapama, dosya okuma vs). Bunun yanında, mikrodenetleyici için yazılmış olan bir C derleyicisi daha küçük olup programcıya sunulan fonksiyon sayısı da oldukça sınırlıdır. Bununla birlikte, standart C dilini öğrenmiş olan bir programcı, kullanmış olduğu mikrodenetleyicinin mimarisini de bildiği takdirde çok kolaylıkla C dilinde program geliştirebilir. Bu kısım içerisinde C18 derleyicisinin kullandığı standart C dili bildirimlerinden ve 18 serisi PIC mikrodenetleyiciler için sağladığı özel fonksiyonlardan bahsedilecektir.

3.4 MPLAB C18 Derleyici

Microchip firması tarafından geliştirilmiş olan ve ANSI C standardını izleyen bu C derleyicisi pek çok özelliklere sahip olup birçok endüstriyel uygulamada, öğretim ve öğrenim amaçlı olarak veya elektronik hobi olarak kullanılmaktadır.

MPLAB C18 derleyicisi, yine Hi-Tech firmasının ürettiği ve pazarladığı PICC derleyicisine çok benzemektedir. MPLAB C18 ile sadece Pic 18xxxx serisi mikro denetleyiciler programlanabilmektedir. MPLAB C18 kullanarak çok karmaşık PIC uygulamaları geliştirmek mümkündür. MPLAB C18 derleyicisini Hi-Tech gibi derleyicilerden farklılaştıran ve tercih nedeni yapan en önemli özellik kesme fonksiyonu içerisine başka bir fonksiyonu çağırabilmesidir.

Bu bölümde, MPLAB C18 derleyicisinin özelliklerini ve bu derleyici ile C programı geliştirilmesi incelenmektedir.

3.4.1 MPLAB C18 veri çeşitleri

MPLAB C18 C derleyicisi aşağıda belirtilen veri çeşitlerini desteklemektedir [6,7]. Burada küçük harf kullanıldığına dikkat edilmelidir.

bit
unsigned char
signed char
unsigned int
signed int
long
unsigned long
float
double

Şimdi bu veri çeşitlerini örneklerle daha yakından inceleyelim.

3.4.1.1 bit

Bu veri çeşidinde sayılar Boolean (0 veya 1) olabilir. Örneğin, aşağıdaki tanımda değişken flag sadece 0 veya 1 değerlerini alabilir.

bit flag;

3.4.1.2 unsigned char

Bu veri çeşidi ile 8 bitlik sayılar tanımlanır ve bu sayılar 0 ve 255 arasında değer alabilirler. Örneğin, aşağıdaki veri tanımında değişken **q**, 0 ve 255 arasında herhangi bir değer alabilir ve burada **q** 180 olarak atanmıştır. Aynı zamanda, değişken **a** karakter R'ye eşitlenmiştir. **unsigned char** tanımı ile hem 8 bitlik bir integer hem de bir karakter tanımlanabilir:

unsigned char q,a;
q = 180;

a= ‘R’;

Burada dikkat edileceği gibi, C dilinde her satır noktalı-virgül (“;”) işareti ile bitmektedir. Satır sonuna bu işaret konulmadığı takdirde derleyici hata verecektir.

3.4.1.3 signed char

İşaret içeren bu veri tanımı, -128 ve +127 arasındaki sayıları tanımlamak için kullanılır. Aşağıdaki örnekte değişken **p**’ye -50 değeri ve değişken **z**’ye 28 değeri atanmıştır.

signed char p,z;
p = -50;
z = 28;

3.4.1.4 unsigned int

Bu tanım ile 16 bit olan ve 0 ile 65535 arasında değişen sayılar tanımlanabilir. Aşağıdaki örnekte, değişken **q**’ya 915 ve değişken **x**’e 3000 değerleri atanmıştır.

unsigned int q;
q = 915;
x = 3000;

3.4.1.5 signed int

İşaret içeren ve 16 bitlik sayılar için kullanılan bu tanımda, sayının -32768 ve +32767 arasında olması gerekir. Aşağıdaki örnekte değişken **t**’ye -500 değeri atanmıştır.

signed int t;
t = -500;

3.4.1.6 long

32 bit ve işaret içeren sayılar için kullanılan bu tanımda sayının -2147483648 ve +2147483647 arasında olması gerekir. Örneğin, aşağıdaki örnekte değişken **m long** olarak tanımlanmış ve değeri 200000 yapılmıştır.

long m;
m = 200000;

3.4.1.7 unsigned long

32 bit işaretsiz sayılar için kullanılan bu tanımda sayıların 0 ve 4294967295 arasında olması gerekir. Aşağıdaki örnekte değişken **k unsigned long** olarak tanımlanmış ve değeri 2000000 yapılmıştır.

unsigned long k;
k =2000000;

3.4.1.8 float

Bu tanım 24 veya 32 bitlik kayan nokta değişkenler için kullanılır. Kayan nokta değişkenler matematik işlemlerinde çok önem taşımaktadırlar. Aşağıdaki örnekte değişken **temp**'e 1.23 değeri atanmıştır.

float temp;
temp = 1.23;

3.4.1.9 double

Bu tanımda sayılar 24 veya 32 bit olabilmektedir. Aşağıdaki örnekte değişken **sum** 12.45 değerini almaktadır.

double sum;
sum = 12.45;

3.4.2 Değişkenlere tanım esnasında değer vermek

C programlama dilinde değişkenlere tanım esnasında değer vermek mümkündür. Örneğin, aşağıda değişken **temp**'e tanım esnasında 10 değeri ve değişken **x**'e ise karakter 'A' değeri atanmıştır.

unsigned int temp = 10;
unsigned char x = 'A';

3.4.3 Programda açıklayıcı yazılar

İki tane öne eğik çizgiden ("//") sonra yazılanlar derleyici tarafından öneme alınmamakta ve programın çalışmasını kolay anlamak için bu tip açıklayıcı yazılar tavsiye edilmektedir.

i = 0; //i değişkenini sıfır yap
j = j + 2; //j değişkenine 2 ilave et
//şimdi iki sayıyı çarp
// ve neticeyi sakla

Açıklamaları bir diğer şekilde yazma formatı ise, açıklamayı "/*" karakterleri ile başlatmak ve "*/" karakterleri ile bitirmektir. Aşağıdaki örnekte bu açıklama şekli gösterilmiştir:

/* Bu program iki tane sayıyı toplar.
Sayılardan biri x, diğeri ise y dir.
Netice z de saklanır.
*/

z = x + y;

Herhangi bir programda her iki tanım şeklini de kullanmak mümkündür.

3.4.4 Değişkenlerin bellekte saklanması

MPLAB C18 C derleyicisinde değişkenler normal olarak RAM bellekte (genel maksatlı saklayıcılarda) saklanmaktadır. Eğer bir değişkenin değeri sabitse ve program boyunca değişmeyecekse, o değişkeni program belleğinde (EPROM veya Flash) saklamak mümkündür. Bu şekilde, kapasitesi sınırlı olan RAM belleği çok kullanılmamış olur.

Sabit bir değişkeni program belleğinde saklamak için önüne **const** komutu ilave edilmeli ve değişkenin değeri tanım esnasında belirtilmelidir. Aşağıdaki örnekte **temp1** değişkeni RAM belleğinde saklanır, **temp2** değişkeni ise 12 değerini alır ve program belleğinde bir sabit olarak saklanır. Burada **temp2** değişkeninin değeri program boyunca sabit olup program içerisinde değiştirilemez. **temp1** değişkeni ise istenildiği gibi ve istenilen an değiştirilebilir.

int temp1 ;
const int temp2 = 12;

3.4.5 Diziler

Diziler (arrays) benzeri bir takım değişkenleri bir grup altında toplamak için kullanılırlar. Örneğin,

int sonuc[6];

Bu komut, 6 tane tam sayı tanımlar ve bu tam sayıların ilki **sonuc[0]**, ikincisi **sonuc[1]**, ve sonuncusu ise **sonuc[5]** olmaktadır.

Örneğin, **sonuc[3]** = 18 komutu sonuc dizisinin dördüncü elemanını 18 sayısına eşitler. Aynı şekilde, **sonuc[2]** = **sonuc[1]** komutu **sonuc** dizisinin ikinci elemanını üçüncü elemanına eşitler.

Bir dizinin elemanları RAM bellekte veya program belleğinde saklanabilirler. Eğer elemanların değerleri sabitse ve program boyunca değişmeyecekse, **const** komutu kullanarak bu elemanlar program bellekte saklanabilirler. Bu şekilde, kapasitesi kısıtlı

olan RAM bellek de kullanılmamış olur. Aşağıdaki örnekte 5 elemanlı **temp** dizisinin elemanları program bellekte saklanırlar. Bu dizi bellekte 10 bayt tutar (her integer 2 bayt yer tutar).

const int temp[] = {1, 5, 9, 12, 45};

Aşağıdaki örnekte ise **temp** dizisi RAM bellekte 10 bayt tutar.

int temp[] = {1, 5, 9, 12, 45};

C programlama dilinde **string** değişkenleri ASCII 0 ile sonlandırılan ve bir takım karakterler ihtiva eden dizilerdir. Bu durumda,

char name[ ] = "Suzan";

Komutunda derleyici otomatik olarak name **string**'i için sonuncu karakteri ASCII 0 olan 6 karakter ayırır. Bu örnekte dizi RAM bellekte saklanmıştır. Aynı dizinin başına **const** komutu koyarak diziyi program bellekte saklayabiliriz.

Diziler birden çok boyutlu olabilirler. Aşağıdaki örnekte 2 boyutlu bir dizi tanımlanmıştır:

int temp[3][5];

Burada **temp**, 3 satırı ve 5 sütunu olan bir dizi olarak düşünülebilir. Dizinin ilk elemanı **temp[0][0]**, ikinci elemanı **temp[0][1]**, üçüncü elemanı **temp[0][2]** vs. dir.

Örneğin, **temp[0][1]** = 10 komutu dizinin ikinci elemanını 10 sayısına eşitler.

Dizi elemanlarını tanım esnasında belirtmek mümkündür. Aşağıdaki örnekte test dizisi RAM bellekte saklanmış ve elemanları **test[0]**=2, **test[1]**=5, ve **test[2]**=8 olarak belirlenmiştir:

int test[3] = {2,5,8};

Daha önce bahsedildiği gibi, **const** komutunu kullanarak dizi elemanlarını program bellekte sabit olarak saklayabiliriz.

3.4.6 Program değişken isimleri

Program değişkenleri bir karakter veya alt çizgi ("_") ile başlamalı ve bunu takiben istenilen miktarda karakter veya 0-9 arasında sayı kullanılmalıdır. Aşağıdakiler geçerli olan bazı program değişken isimleridir.

size
toplam_miktar
max5
_ortalama
j
ge2ci23ci
a_b_c_d

3.4.7 Static değişkenler

Static değişkenler genel olarak bir fonksiyonda bulunan ve fonksiyonu her kullanışta değerleri değişmeyen değişkenlerdir. Bu değişkenleri tanımlarken isimleri başına **static** yazılır. Örneğin, bir fonksiyona girişte, fonksiyonda daha önce kullanılan bir değişkenin değeri belirsizdir. Fakat değişkeni **static** yapmakla değerini muhafaza etmiş oluruz.

3.4.8 Volatile değişkenler

Volatile değişkenler, bir değişkenin her kullanıldığında eski değerini muhafaza edemeyeceği durumlarda kullanılır. Bütün giriş-çıkış portları ve kesme rutinlerinin değiştirdiği değişkenler **volatile** yapılmalıdır.

3.4.9 Persistent değişkenler

Genel olarak PIC mikrodenetleyici reset yapıldığında programda kullanılan bütün değişkenler ilk olarak sıfır değerini alırlar. Yalnız bazı durumlarda, birtakım

değişkenlerin reset esnasında değerlerini kaybetmelerini istemeyebiliriz. Herhangi bir değişkenin ismi önüne **persistent** kelimesini yazmakla o değişkenin reset esnasında sıfırlanmasını istemediğimizi belirtiriz.

3.4.10 Mutlak adres değişkenleri

Bir değişkenin ismi sonuna "@" karakteri konarak o değişkene bir mutlak adres değeri verilebilir. Örneğin,

unsigned char Portbit @ 0x06;

Yukarıdaki satırda belleğin 6ncı adresinde **Portbit** diye bir değişken tanımlanmıştır. Burada derleyici **Portbit** değişkeni için bir yer ayırmaz, fakat sadece **Portbit** ismini mutlak adres 6 ya eşitler. Bu tanımın assembler eşiti şudur.

Portbit EQU 06h

Bit veri çeşidi ve mutlak adres birleştirilip istenilen bir adresteki herhangi bir bit okunup değiştirilebilir. Örneğin, **STATUS** kaydedicisi bellekte adres 3'tedir. Bu kaydedicinin üçüncü bitini istiyorsak, **STATUS** yazmacını adres 3'te tanımlayıp 3x8 + 3 = 27nci biti şu şekilde tanımlayabiliriz.

static unsigned char STATUS @ 0x03;
static bit PD @ (unsigned)&STATUS*8+3;

Burada **PD** değişkeni bellekteki 27nci biti (**STATUS** yazmacının 3ncü biti) tanımlamıştır.

3.4.11 Operatörler

Herhangi bir programlama dilinde operatörler son derece önem taşımaktadırlar. Bütün aritmetik ve lojik işlemler bu operatörler sayesinde yapılmaktadır. MPLAB C18 derleyicisi Çizelge 3.1'de görülen operatörleri desteklemektedir.

Çizelge 3.1 C dilinin operatörleri

Seviye	**Operatör**	**Tanım**	**Öncelik Yönü (associativity)**
1	( ) [ ] . ->	öncelik kazandırma ve fonksiyon çağırma index operatörü (subscript) yapı elemanına ulaşım (structure access) yapı elemanına gösterici ile ulaşım	soldan sağa
2	+ - ++ -- ~ ! * & sizeof (tür)	işaret operatörü (unary) işaret operatörü (unary) 1 artırma (increment) 1 eksiltme (decrement) bitsel değil (bitwise not) mantıksal değil (logical not) içerik operatörü (indirection) adres operatörü (address of) sizeof operatörü tür dönüştürme (type cast operator)	sağdan sola
3	* / %	çarpma (multiplication) bölme (division) modulus (bölümden kalan)	soldan sağa
4	+ -	toplama (addition) çıkarma (suntraction)	soldan sağa
5	<< >>	bitsel sola kaydırma (bitwise shift left) bitsel sağa kaydırma (bitwise shift right)	soldan sağa
6	< > <= >=	küçüktür (less than) büyüktür (greater than) küçük eşittir (less than or equal) büyük eşittir (greater than or equal)	soldan sağa
7	== !=	eşittir (equal) eşit değildir (not equal to)	soldan sağa
8	&	bitsel VE (bitwise AND)	soldan sağa
9	^	bitsel EXOR (bitwise EXOR)	soldan sağa
10	|	bitsel VEYA (bitwise OR)	soldan sağa
11	&&	mantıksal VE (logical AND)	soldan sağa
12	||	mantıksal VEYA (logical OR)	soldan sağa
13	?:	koşul operatörü (conditional operator)	sağdan sola

3.4.12 Program akış kontrolü

Program akışını değiştiren komutlar assembler dili de olmak üzere her programlama dilinde son derece önemlidirler. Bu komutlar sayesinde döngü yapabiliriz veya bir değişkenin değerine bağlı olarak değişik işlemler yapabiliriz. MPLAB C18 aşağıdaki program akış kontrol komutlarını destekler [6,7].

if-else
for
while
do
goto
break
switch - case

Şimdi bu komutların kullanımlarını daha yakından inceleyelim.

3.4.12.1 if - else

Koşula bağlı olarak program akışını değiştiren bu komut genel olarak şu şekillerde kullanılabilir.

```
if(koşul)komut;

veya,

if(koşul)
{
komut;
komut;
…….
…….
komut;
```

```
}

veya,

if(koşul)
{
komut;
komut;
…….
}
else
{
komut;
komut;
…….
}
```

sadece bir komut içeren durumlarda şu şekilde de kullanılabilir:

```
if(koşul)
komut;
else
komut;
```

Burada, komutlar sonunda noktalı-virgül işaretlerine dikkat ediniz.

Aşağıdaki örnekte, **p** değişkeni 0 ise **b** değişkeni 1 olarak arttırılır, aksi halde **b** değişkenine 2 ilave edilir.

```
if(p = = 0)
b=b+1;
else
b=b+2;
```

Yukarıdaki örnekte eşittir operatörü olan “= =” karakterlerinin kullanıldığına dikkat ediniz.

3.4.12.2 for

for komutu program içerisinde döngü yapmak için kullanılır. Döngü bir veya birtakım komutu birden fazla tekrarlamak maksadıyla kullanılır. Bu komut şu şekillerde olabilir.

```
for(başla; koşul; artış)komut;
```

veya,

```
for(başla; koşul; artış)
{
komut;
komut;
…….
}
```

burada, başla döngü değişkeninin ilk değerini, artış ise son değerini belirtmektedirler.

Aşağıdaki örnekte döngü değişkeni i, 1’den 10’a kadar gitmekte ve **sayac** değişkeni 10 defa arttırılmaktadır.

```
for(i = 1 ; k = 10; i + + )
sayac+ +;
```

for komutunu kullanarak iç içe döngü yapmak mümkündür. Aşağıdaki örnekte iç döngü 5 defa ve dış döngü ise 10 defa tekrarlanmaktadırlar.

```
for(i=0; i<10; i+ +)
{
for(j=0; j<5; j+ +)
{
```

```
komut;
.....
}
}
```

3.4.12.3 while

Döngü yapmak için kullanılan bir başka komut da **while** komutu olup bu komutun genel kullanım şekli şu şekildedir:

```
while(koşul)komut;
```

veya,

```
while(koşul)
{
komut;
komut;
.....
}
```

Belirtilen koşul doğru olduğu müddetçe döngü devam etmektedir. Eğer döngü başında seçenek doğru değilse döngü içerisindeki komutlar işlem görmezler. Aşağıdaki örnekte döngü 10 defa tekrarlanmaktadır.

```
i=0;
while(i < 10)
{
komut;
komut;
i++;
}
```

while komutunu kullanırken döngü içerisinde koşulun sağlanmasına dikkat etmemiz gerekir, aksi halde sonsuz bir döngü elde etmiş oluruz.

3.4.12.4 do

Döngü yapmak için kullanılan bir diğer komut ise **do** komutudur. Burada seçenek döngünün sonunda kontrol edilir ve bundan dolayı koşul doğru olmasa bile döngü en az bir defa çalışmış olur. Bu komutun genel kullanımı şu şekildedir.

```
do
{
komut;
komut;
……..
}while(koşul);
```

Aşağıdaki örnekte döngü değişkeni **j**, 0'dan 5'e kadar gitmekte ve döngü 5 defa tekrarlanmaktadır.

```
j = 0;
do
{
komut;
komut;
j++;
}while < 5);
```

while komutunu kullanırken döngü içerisinde koşulun sağlanmasına dikkat etmemiz gerekir, aksi halde sonsuz bir döngü elde etmiş oluruz. Ayni zamanda, koşul sağlansa bile döngü en az bir defa yapılmaktadır.

3.4.12.5 break

break komutu, bir döngü içerisinde ve döngü bitmeden, döngü dışına çıkmak için kullanılmaktadır. Aşağıdaki örnekte **j** değişkeni 10 olunca döngü durur ve döngü sonrasındaki komutlar işlem görürler. Bu komut bir sonra göreceğimiz **switch-case** komutunda çok kullanılmaktadır.

```
while(i < 100)
{
komut;
komut;
if(j = = 10)break;
}
```

3.4.12.6 switch - case

Bu komut, çoklu **if-else** komutu gibi görev yapmaktadır. Bir değişken alınır ve bu değişkenin değerine bağlı olarak çeşitli komutlar işlem görürler.

Aşağıdaki örnekte, **sec** değişkeninde bulunan ve 'A' ve 'F' arasında olan 1-digit bir hexadecimal sayının decimal sayıya dönüştürülmesi gösterilmiştir. Dönüştürülen sayı **hex** gibi bir değişkende saklanmıştır:

```
Switch(sec)
{
Case'A': hex = 65;
break;
Case'B': hex = 66;
break;
Case'C': hex = 67;
break;
Case'D': hex = 68;
break;
Case'E': hex = 69;
break;
Case'F': hex = 70;
break;
default: hex = 0;
break;
}
```

Burada, **sec** değişkeninin değeri 'A' ise **hex** değişkeni 65 olur, 'B'' ise hex değişkeni 66

olur, ‘C’ ise **hex** değişkeni 67 olur v.s. Eğer **sec** değişkeni ‘A’ ve ‘F’ arasında değilse **default** komutu çalıştırılır ve bu örnekte **hex** değişkeni 0’a eşitlenir. **switch-case** komutunda **break** komutunun çok sık olarak kullanıldığına dikkat ediniz (eğer **break** kullanılmazsa program bir sonraki koşula devam etmektedir).

3.4.13 Kullanıcı fonksiyonları

Fonksiyonlar genel olarak bir program içerisinde kullanılan ve ana programdan bağımsız programdırlar. Her fonksiyonun bir ismi olup istenilirse bir fonksiyon ana programa bilgi aktarabilir. Başında **void** komutu ile başlayan fonksiyonlar ana programa bilgi aktaramazlar. Örneğin, aşağıdaki fonksiyonun ismi **ledon** olup bu fonksiyon ana programa bilgi aktarmaz.

```
void ledon()
{
led = 1;
}
```

Aşağıdaki **ledon** fonksiyonu ise ana programa **integer** bir bilgi aktarır:

```
int ledon()
{
komut;
komut;
return(değişken);
}
```

Fonksiyonlar ana programdan bilgi alıp bu bilgiyi işler ve tekrar ana programa bilgi aktarabilirler. Aşağıdaki örnekte kare fonksiyonu ana programdan bir **integer** değer alır (parameter) ve bunun karesini alıp ana programa aktarır:

```
int kare(int y)
{
int w;
w = y*y;
return (w) ;
}
```

Bu fonksiyon ana program tarafından şu şekilde kullanılabilir:

```
s = kare(p);
```

Kare fonksiyonu ana programdan **p** değişkeninin değerini alır ve bu sayının karesini alıp geri ana programa aktarır. Fonksiyon başındaki **int** tanımı fonksiyonun ana programa **integer** (tam sayı) aktaracağını gösterir. Bu durumda, ana programdaki **s** değişkeni **p**'nin karesine eşitlenmiş olur.

Burada önemli olan bir nokta, fonksiyonda kullanılan **w** ve **y** değişkenleri sadece fonksiyona ait olup ana programda olan bir **w** veya **y** değişkeni ile hiçbir ilgileri yoktur. Aynı zamanda, ana programda kullanılan bir değişken ile fonksiyonda kullanılan ve aynı ismi taşıyan herhangi bir değişken arasında bir bağıntı yoktur.

3.4.14 Ön-işlemci direktifleri

MPLAB C18 derleyici, diğer standart C derleyicilerde bulunan çeşitli ön-işlemci direktiflerini destekler. Yaygın olarak kullanılan ön-işlemci direktifleri aşağıda açıklanmıştır.

3.4.14.1 #define

Bu direktif program başında kullanılır ve bir makro gibi çalışıp semboller yerine tanımlanan eşitlerini koyar.

Örneğin;

```
#define size 10
```

```
#define max 100
#define min 0
```

program içerisinde bu semboller kullanıldığı zaman yerlerine otomatik olarak değerleri konur.

```
a = 1; // a değişkenini 1 yap
a = a + size; // a değişkeni 11 değerini alır
```

define direktifi çok kompleks de olabilmektedir. Aşağıdaki makro tanımında 2*(a+b) işlemi mult(a,b) olarak tanımlanmıştır:

```
#define mult(a,b) 2*(a + b)
```

Şimdi bu **makronun** bir program içerisinde kullanımına bakalım.

```
a = 2; // a değişkenini 2 yap
b = 5; // b değişkenini 5 yap
c = mult(a,b); // c değişkeni 2*(2+5)=14 olur
```

3.4.14.2 #asm ve #endasm

Bu direktifler sayesinde bir assembler programı veya bir veya birkaç assembler komutu C programına ilave edilebilir:

```
.....
.....
a=2;
#asm
movlw 10h
#endasm
.....
.....
```

Yukarıdaki programda görüleceği gibi assembler komutları **#asm** ve **#endasm** direktifleri arasına yerleştirilmelidir.

3.4.14.3 #include

Bu direktifi kullanarak başka bir text dosyasını programımız içerisine alabiliriz. Örneğin, "myproject.h" dosyası programımıza şu şekilde ilave edilebilir:

#include "myproject.h"

veya,

#include <myproject.h>

PIC yazılımı geliştirirken <pic.h> dosyası programımıza ilave edilmelidir. Bu dosya içerik olarak PIC mikrodenetleyiciye ait çeşitli değişkenleri tanımlamaktadır.

3.4.15 Programda değişik sayı tabanı kullanımı

MPLAB C18 derleyicisinin normal sayı tabanı 10 (desimal) olup birçok mikroişlemci ve mikrodenetleyici uygulamalarında kullanılan sayı tabanının zaman zaman değiştirilmesi gerekmektedir. Örneğin, mikrodenetleyici tabanlı kontrol uygulamalarında genellikle 16 tabanı (hexadecimal) veya 8 tabanı (octal) kullanılmaktadır.

Bir sayıyı hexadecimal olarak yazmak için önüne %x (veya %0X), octal olarak yazmak için önüne %o ve ikili tabanda yazmak için ise önüne 0b koymak gerekir. Aşağıda değişik tabanlarda bazı örnekler verilmiştir.

```
toplam = %0x1f; // toplam = 0001 1111
sum = %o37; // sum = 011 111
a = 0b00001111 // a = 00001111
```

3.4.16 Yapılar (Structures)

Yapılar, birbirleriyle ilgili olan çeşitli değişkenleri yeni bir değişken altında toplamak için kullanılır. Örneğin, bir şahıs hakkındaki, bir takım bilgiler normal olarak şu şekilde tanımlanabilir.

```
unsigned char adi[80];
unsigned char soyadi[80];
unsigned int yasi;
unsigned char adresi[80];
```

Bir yapı (structure) kullanarak yukarıdaki tanımları şu şekilde yazabiliriz:

```
struct sahsi
{
unsigned char adi[80];
unsigned char soyadi[80];
unsigned int yasi;
urisigned char adresi[80];
}
```

Bu durumda, **sahsi** yapısı bir şahıs hakkında olan bütün tanımları bir çatı altında toplamış olur. Bir yapı yukarıdaki gibi tanımlandığı zaman bellekte hiçbir yer tutmaz. Ancak bu yapı bir sonraki örnekte gösterildiği gibi kullanıldığı zaman bellekte yer tutar.

Şimdi, aşağıdaki yeni tanımı yapabiliriz:

struct sahsi benim_bilgiler;

Yukarıdaki tanım ile benim_bilgiler yeni bir değişken olup bu değişkenin çeşidi **sahsi**'dir ve bu değişken bellekte yer tutar. Şimdi, bu yeni değişkeni kullanarak şu komutu yazabiliriz.

```
Benim_bilgiler.yasi = 25;
```

Yapıları kullanarak bir değişkene istenilen sayıda bit ayırabiliriz. Aşağıdaki örnekte **x** ve **y** 1 bitlik değişkendirler ve **z** ise 6 bitlik bir değişkendir. Değişken **temp flags** çeşidi bir yapıya sahip olup bellekte toplam 1 bayt tutmaktadır.

```
struct flags
{
unsigned char x:1;
unsigned char y:1;
unsigned char z:6;
}

struct flags temp;
```

3.4.17 İşaretçiler (Pointers)

C programlama dilinde işaretçi kavramı çok geniş olarak kullanılmaktadır. Genel olarak bir işaretçi bir değişkenin bellekteki adresini gösterir. Örneğin, **x** bir değişken ise, **x**'i gösteren bir işaretçi x'in bellekteki adresini tutar. İşaretçiler, değişken ismi önüne "*" karakteri koyarak tanımlanırlar. Değişken çeşidine göre işaretçi çeşidi de değişmektedir. Örneğin, bir karakter işaretçisi bir integer için kullanılamaz. Aşağıdaki örnekte, **p** bir karakter işaretçisi olarak tanımlanmıştır.

char *p;

Burada **p** bir karakter işaretçisi olarak tanımlandığı halde şu an herhangi bir yerde kullanılmış değildir. Herhangi bir değişken önüne "&" karakterini koyarak o değişkenin adresini elde edebiliriz. Daha sonra da işaretçi değişkenini kullanabiliriz. Örneğin, işaretçi **p**'ye, z değişkeninin adresini şu şekilde yükleyebiliriz.

p = &z;

p işaretçisi şimdi **z** değişkeninin adresini tutmaktadır. "*" operatörünü kullanarak, adresi bilinen bir değişkenin asıl değerini okuyabiliriz. Aşağıdaki örnekte, daha önce tanımlanan **z** değişkeninin değeri 3 yapılmıştır.

*p = 3;

Herhangi bir değişkenin adresine erişebilme birçok uygulamalarda çok kullanışlı olmaktadır. Bu şekilde daha verimli ve daha kısa programlar geliştirmek mümkün olabilmektedir.

İşaretçiler dizilerde çok kullanışlı olmaktadırlar. Bir dizinin ismi o dizinin adresini tutmaktadır. Bu durumda, bir işaretçi kullanarak dizi işlemlerini kolaylıkla yapabiliriz. Bir örnek aşağıda verilmiştir.

```
char buffer[10]; // 10 elemanı olan bir buffer tanımlama
char *p; // p bir karakter işaretçisidir
p = buffer; // p işaretçiye buffer adresi yüklendi
*p = 0; // buffer dizisinin ilk elemanını 0 yap
p++; // p'yi arttır (sonraki elemanı adresle)
*p = 1; // buffer'in ikinci elemanını 1 yap
```

Yukarıda da görüleceği gibi işaretçiler ile çeşitli aritmetik işlemler yapmak ve değişik adreslere gitmek mümkündür.

3.4.18 A/D Çevirici Fonksiyonları

Çizelge 3.2'de MCC 18 C derleyicisinin A/D Çevirici fonksiyonları yer almaktadır.

Çizelge 3.2 A/D Çevirici fonksiyonları

Fonksiyon	Tanımı
BusyADC	A/D çevirici kullanımda mı?
CloseADC	A/D çeviriciyi pasif yap.
ConvertADC	A/D çevir.
OpenADC	A/D çeviriciyi ayarla.
ReadADC	A/D çevirme işleminin sonucunu oku.
SetChanADC	Kullanılan A/D kanalını seç.

3.4.19 USART Fonksiyonları

Çizelge 3.3'te MCC18 C derleyicisine ait USART fonksiyonları yer almaktadır.

Çizelge 3.3 USART fonksiyonları

Fonksiyon	Tanımı
BusyUSART	USART kullanımda mı?
CloseUSART	USART'ı pasif yap.
DataRdyUSART	USART okuma tamponundaki veri erişilebilir mi?
getcUSART	USART'tan bir byte'ı oku.
getsUSART	USART'tan diziyi oku.
OpenUSART	USART'ı ayarla.
putcUSART	USART'a bir byte yaz.
putsUSART	USART'a veri belleğine dizi yaz.
putrsUSART	USART'a program belleğine dizi yaz.
ReadUSART	USART'tan bir byte oku.
WriteUSART	USART'a bir byte yaz
baudUSART	Artırılmış USART'ın veri iletim hızı konfigürasyon bitlerini ayarla.

BÖLÜM IV

DONANIM YAPISI VE ÇALIŞMASI

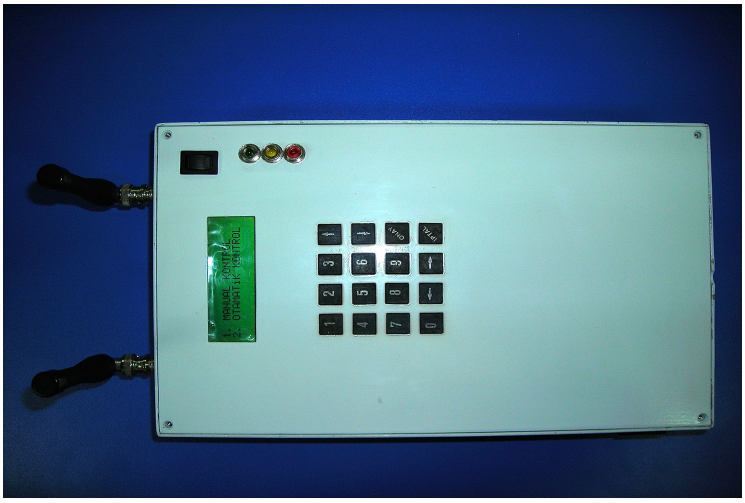
Bu bölümde gerçekleştirilmiş olan sisteme ait donanım incelenmektedir.

Tasarlanan sistem bir adet kumanda birimi, sekiz adet PLC kartı ve her PLC kartına ait alıcı ve vericilerden oluşmaktadır. Kumanda biriminden gönderilen komutlar ile PLC kartlarında bulunan röleler çalıştırılmakta PLC kartına yapılan 5V veya 24V girişler PLC kartına bağlı alıcı ve vericiler üzerinden kumanda birimine bildirilmektedir. Bu kısımda sistemin elektriksel özelliklerine geçmeden önce sistemin anlaşılmasını kolaylaştırmak amacıyla öncelikle çalışma modları anlatılacaktır.

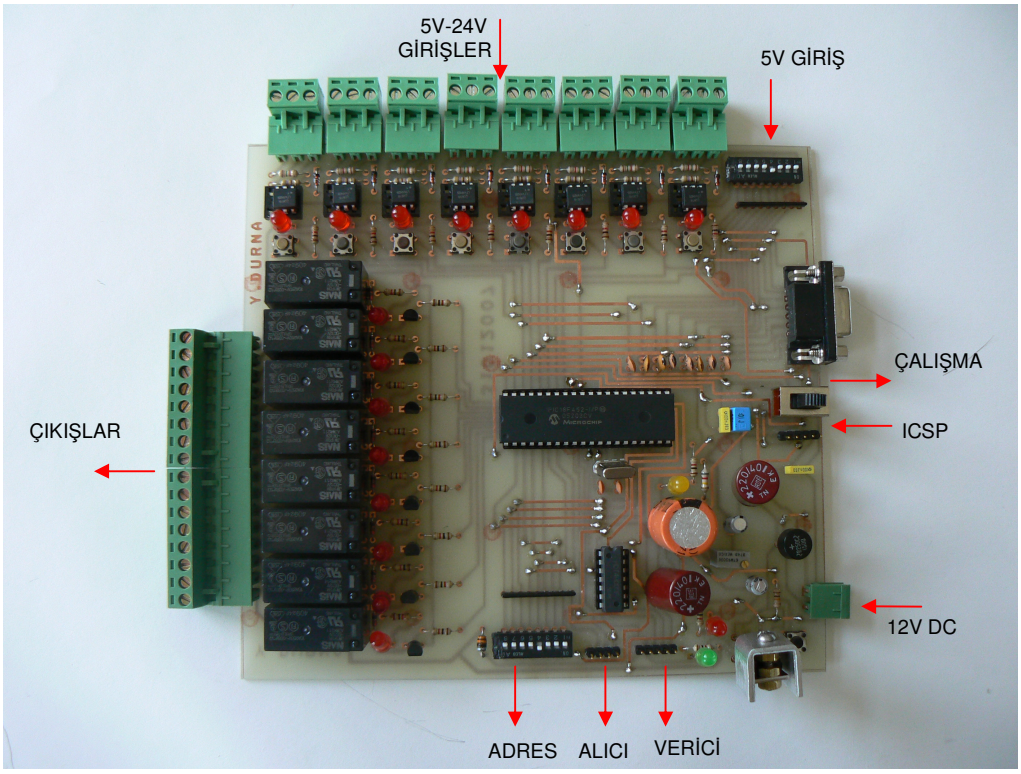
Fotoğraf 4.1'de mikrodenetleyici tabanlı RF'li kontrol uygulaması, Fotoğraf 4.2'de uygulamaya ait kontrol birimi, Fotoğraf 4.3'de PLC kartı ve Fotoğraf 4.4'te PLC kartına ait alıcı ve verici görülmektedir.



Fotoğraf 4.1 Mikrodenetleyici tabanlı RF'li kontrol uygulaması



Fotoğraf 4.2 Kumanda birimi



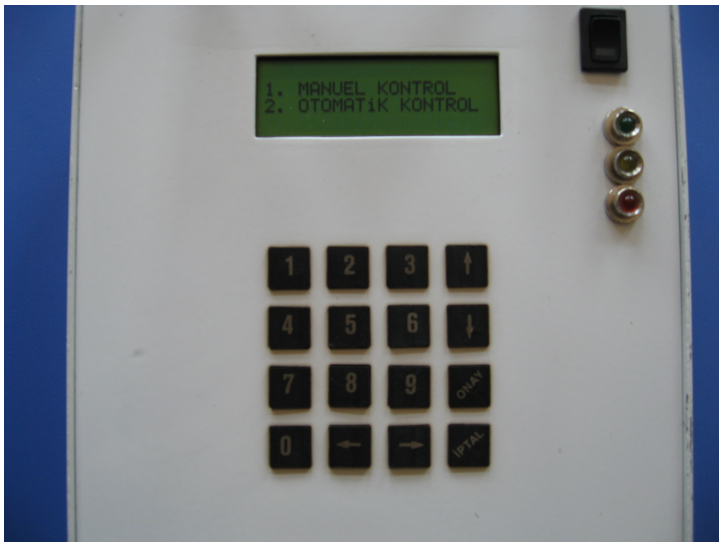
Fotoğraf 4.3 PLC kartı



Fotoğraf 4.4 PLC kartına ait alıcı ve verici

4.1 Çalışma Modları

Fotoğraf 4.5'te kumanda birimine ait ana menü görülmektedir. Sistem istenildiğinde kumanda birimi üzerindeki tuş takımı ile göstergeden seçilerek manüel veya otomatik olmak üzere iki farklı çalışma modunda kullanılabilecek şekilde tasarlanmıştır.



Fotoğraf 4.5 Kumanda birimi menü

4.1.1 Otomatik çalışma modu

Fotoğraf 4.6'da otomatik çalışma moduna ait otomatik kontrol menüsü görülmektedir. Kumanda üzerindeki tuş takımı ile otomatik kontrol seçilir. Göstergede sırası ile aynı program, farklı program ve sonuçları göster seçenekleri görülmektedir.



Fotoğraf 4.6 Otomatik kontrol menü

4.1.1.1 Aynı program

Fotoğraf 4.7'de PLC'ler için program seçilen aynı program menüsü görülmektedir. Aynı program seçeneği ile tüm alıcı PLC kartlardaki mikrodenetleyicilere önceden yüklenmiş sekiz adet programdan birinin çalıştırılması istenir.



Fotoğraf 4.7 Aynı program menü

Fotoğraf 4.8'de ilk dört program örneği görülmektedir. Tuş takımından aşağı ok tuşuna basılarak diğer dört program da görülebilmektedir. Bu sekiz adet programdan birinin liste numarası girilerek onay tuşuna basılır. Fotoğraf 4.9'da ikinci dört program örneği görülmektedir.



Fotoğraf 4.8 İlk dört program örneği



Fotoğraf 4.9 İkinci dört program örneği

Fotoğraf 4.10'da kumanda biriminden gönderilen bilgilere ait gönderme tamamlandı ekranı görülmektedir. Ekranda görülen gönderme tamamlandı yazısından sonra tüm PLC kartları seçilen programı alır ve çalıştırır.



Fotoğraf 4.10 Gönderme tamamlandı ekranı

Gönderme işleminin tamamlanmasının ardından sonuçları göster ekranı göstergede belirmektedir. Sonuçları göster ekranı her bir hedef PLC kartına yapılan girişlerin durumunu göstermektedir. Otomatik modda çalışırken kumanda birimi her on saniyede tüm hedef PLC kartlarına sorgu yapmaktadır. Bu sorguya verilen cevap sonuçları ekranda gözlemlenmektedir. Fotoğraf 4.11'de sonuçları göster ekranı görülmektedir.



Fotoğraf 4.11 Sonuçlar ekranı

Sistem sorgulamadan sonra PLC kartlarının girişlerini okur. Girişlerinde mevcut önceki duruma göre farklılık olanlara giriş değerlerine göre hangi programı çalıştırması gerektiğini her 10 saniyede yaptığı kontroller sonucunda kumanda birimi üzerinden gönderir ve ilgili PLC'ler yeni programı çalıştırır. PLC'lerin çalıştırdığı programlar daha önceden mikrodenetleyicilere yüklenmiş olan programlardır.

4.1.1.1.2 Farklı program

Bu çalışma şekli aynı program kısmı ile hemen hemen aynıdır. Sadece çalıştırılacak PLC'lere ana menüdeki sekiz programdan istenilen program seçilerek çalıştırılması istenen PLC kartına gönderilir. Fotoğraf 4.12'de çalıştırılacak PLC kartlarının seçimi görülmektedir.



Fotoğraf 4.12 Çalıştırılacak PLC kartının seçimi

Bu işlem her PLC kartı için ayrı ayrı yapılır. Gönderme işlemi tamamlandıktan sonra PLC'lere ait girişleri gösteren ekrana otomatik olarak geçilir ve her 10 saniyede PLC'lere yapılan sorgu işlemine başlanır. Sorgu işlemi sonucunda aynı program

modunda olduğu gibi girişlerinde mevcut duruma göre farklılık olan PLC'lere çalıştırması gerekli program bildirilir.

4.1.1.1.3 Sonuçları iste

Bu menü iptal tuşu ile sorgu ekranından çıkılması durumunda PLC'lerin giriş durumlarını öğrenmek maksadıyla istenildiğinde kullanılmak üzere tasarlamıştır.

4.1.2 Manüel çalışma modu

Otomatik çalışma modu ile aynı menü ve özelliklere sahiptir. Aynı program, "farklı program" ve "sonuçları göster" özelliklerine sahiptir. Çalıştırma şekli otomatik modda bahsedildiği gibidir. Bu mod kullanılırken bilinmesi gereken şey PLC kartlarına kumanda biriminden çalıştırılacak programın gönderme yapılmasından sonra kumanda birimi her on saniyede bir sorgu işlemi gerçekleştirir. Bu sorgu işlemine cevap veren PLC kartlarından girişlerinde değişiklik olanlar olabilir. Manuel modda girişlerindeki değer değişmiş olsa da kumanda birimi üzerindeki tuş takımından girilmediği sürece sorgu esnasında yeni bir program çalıştırılması PLC kartlarından otomatik olarak istenmez.

4.2 Sistemde Kullanılan RF Alıcılar

Şekil 4.1'de sistemde kullanılan RFRQ2 alıcısı görülmektedir. Abacom (www.abacom.com) firmasının üretmiş olduğu RFRQ2 radyo frekanslı alıcı devresi, 433.92 MHz frekansında çalışmaktadır. Küçük fiziksel boyutu ve düşük güç tüketimi sayesinde uzaktan kontrol sistemleri için kullanıma uygun olan devre, 17.3 cm'lik bir antenle birlikte en başarılı biçimde çalışabilmektedir. Besleme gerilimi olarak 4.5 Volt ile 5.5 Volt arası bir gerilim seçilmelidir. 5.5 Volt'tan yüksek bir besleme gerilim uygulanması durumunda devrenin bozulma olasılığı çok yüksektir. Ayrıca besleme gerilimi üzerinde 100 mV'dan daha fazla parazit gerilim olmaması gerekir. 300 baud ile 9600 baud hızları arasında veri transferi yapabilen devre, -25 °C ile +80 °C arası ortam sıcaklığında çalışmaktadır. RFRQ2 6.8 mA besleme akımı çekmektedir. Hem sayısal hem de analog çıkışa sahiptir. Hassasiyet olarak -120 dBm seviyesindeki sinyallere kadar çıkış üretebilmektedir [8].



Şekil 4.1 RFRQ2 alıcısı

4.3 Sistemde Kullanılan RF Vericiler

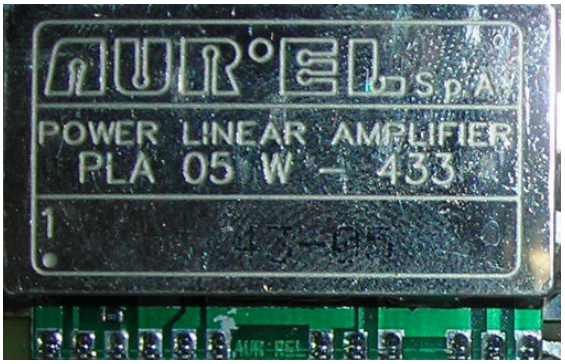
Şekil 4.2'de sistemde kullanılan RFTQ2 vericisi görülmektedir. Abacom firmasının üretmiş olduğu RFTQ2 radyo frekanslı verici devresi, 433.92 MHz frekansında çalışmaktadır. Besleme gerilimi olarak 3 Volt ile 12 Volt arası bir gerilim seçilmelidir. 12 Volt'tan yüksek bir besleme gerilim uygulanması durumunda devrenin bozulma olasılığı çok yüksektir. Ayrıca besleme gerilimi üzerinde 100 mV'dan daha fazla parazit gerilim olmaması gerekir. 300 baud ile 9600 baud hızları arasında veri transferi yapabilen devre, -25 °C ile +80 °C arası ortam sıcaklığında çalışmaktadır. RFTQ2 8 mA besleme akımı çekmektedir. Kapalı alanlarda 75 m açık alanda ise 250 m mesafeye kadar alıcılar ile haberleşebilmektedir [8].



Şekil 4.2 RFTQ2 vericisi

4.4 Sistemde Kullanılan RF Güç Yükselteci

Şekil 4.3'de sistemde kullanılan güç yükselteci görülmektedir. 0.5 W çıkış gücü üreten PLA-05W-433 yükselteci veri iletişim mesafesinin arttırılmasında önemli rol oynamaktadır. 430-435 MHz aralığında çalışmakta ve 12-15 Volt arası DC ile beslenen yükseltece RFTQ2 vericisinin anten çıkışı giriş olarak bağlanmaktadır. Açık alanda 1 km ve üzeri mesafede veri iletişimi sağlamaktadır. Yükselteç gönderme esnasında 200 mA civarında akım çekmektedir [8].



Şekil 4.3 PLA-05W-433 yükselteci

4.5 Sistem Anteni

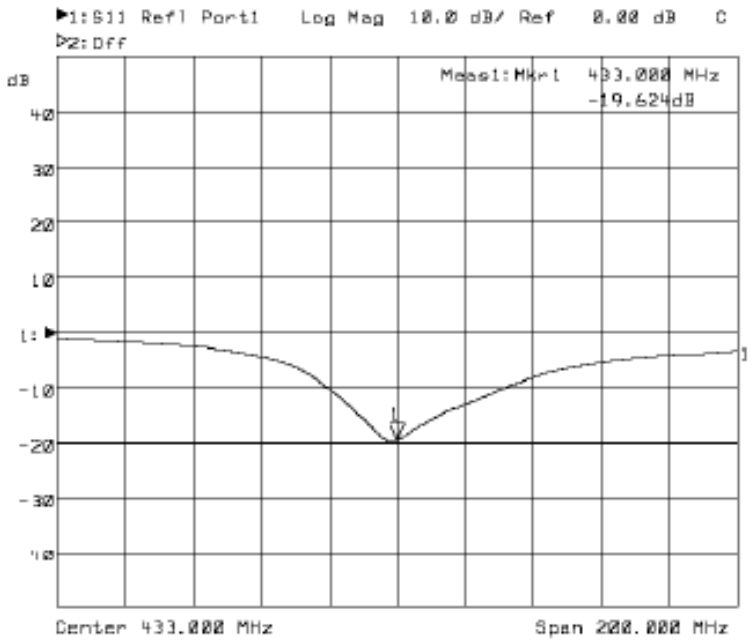
Şekil 4.4'te sistem de kullanılan anten görülmektedir. Abacom firmasının kendi modülleri için üretmiş olduğu anten, RFRQ2 alıcı ve RFTQ2 verici devreleriyle uyumlu çalışmaktadır. 434 MHz frekansında çalışan anten, 4MHz bant genişliğine, 50 Ω çıkış empedansına, ve 1.5 gerilim duran dalga oranına sahiptir. Kısaca gerilim duran dalga oranından bahsetmek gerekirse; VSWR genel olarak iletim hattının yük empedansı ile karakteristik empedansı arasındaki uygunsuz eşlemenin bir ölçüsüdür. Eğer verici uygun bir antenle sonlandırılmamışsa (yani uygun yük) yansıma artacaktır. Dolayısıyla SWR de artacaktır ve giden gücün büyük bir kısmı yansıyacaktır. Bu durumda maksimum güç iletilmemiş olacaktır. Bunun için verici uygun empedans değeri olan bir antenle sonlandırılmalıdır, empedans değeri 50 Ohm olmalıdır [8].



Şekil 4.4 Anten

Yapmış olduğumuz çalışmada anten SWR değerini yaklaşık olarak -20dB yani yansıma katsayısı 1 değerine ayarlayabilmek için alıcı ve vericilerin içine yerleştirildiği ve antenimize şase olan alüminyum kutu kullanılmıştır. Antenin düzgün çalışması için

SWR değerleri network analizör cihazında incelenmiştir. Şekil 4.5'de anten SWR değeri görülmektedir. Fotoğraf 4.13'te uygun SWR değerini sağlayan alüminyum kutular görülmektedir.



Şekil 4.5 Anten SWR değeri



Fotoğraf 4.13 Uygun SWR değerlerini sağlayan kutular

4.6 PLC Kartı Devresi

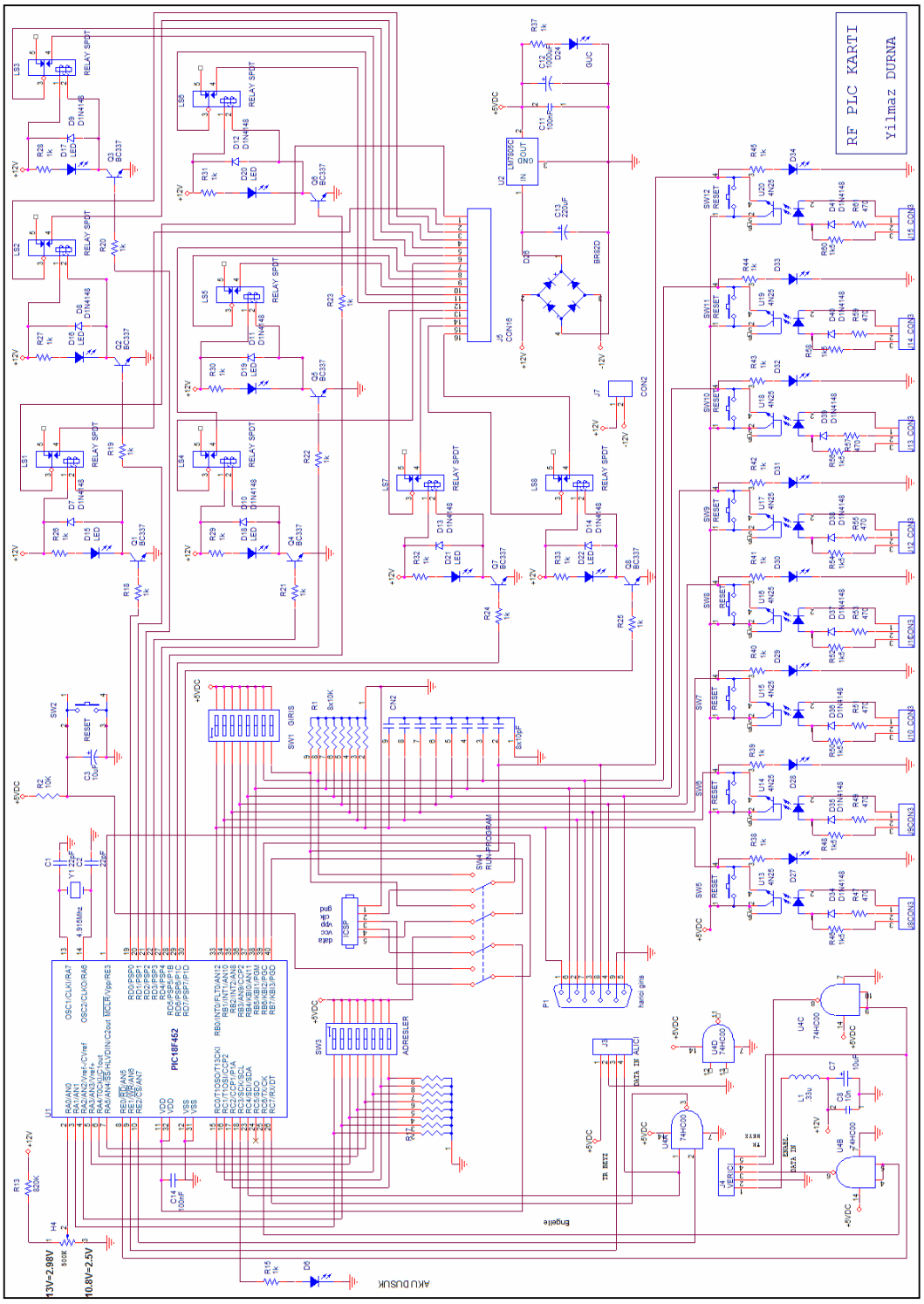
Şekil 4.6'da PLC kartına ait devre şeması görülmektedir. PLC kartı AC veya DC 12 Volt ile çalışmaktadır. 7805 regülatörü ile devredeki 5V gereksinimi sağlanmaktadır. Devreye 1k5 dirençleri üzerinden 24V, 470 ohm dirençleri üzerinden 5V harici giriş bağlanabilmektedir. 4N25 optokuplörü harici girişlerin mikrodenetleyiciye zarar vermeden algılanması amacıyla kullanılmıştır.

Ayrıca devrede bulunan anahtarlar ile de harici sekiz adet 5V girişi dışarıdan besleme almadan yapılabilmektedir. Devre üzerinde mikrodenetleyicinin sökülmeden programlanmasını sağlayan ICSP bağlantıları yapılmış ve bu iş için bir adet 4PDT anahtar tahsis edilmiştir. Anahtarın konumuna göre PIC programlanmakta veya normal olarak çalışmaktadır. ICSP girişine ait data, Vcc, Vpp, clk ve gnd pinleri harici bağlantı için devre kartı üzerine çıkarılmıştır.

PLC kartında çıkış olarak 12V DC ile çalışan kontak anma değerleri 250VAC 10A olan sekiz adet röle kullanılmaktadır. Rölelerin çalışma esnasında çektikleri akım 40 mA civarındadır. Sekiz adet rölenin çalışması 1 amper kadar akım verebilen 7812 regülatörü ile sorunsuz sağlanabilmektedir.

Kontrol kartı üzerinde mikrodenetleyicinin adres bilgilerini aldığı sekiz bitlik anahtar yerleştirilmiştir. Bu anahtar üzerinden ikili sisteme göre girilen değer kontrol kartının kimlik bilgisi olarak mikrodenetleyici tarafından değerlendirilerek buna göre işleme tabi tutulmaktadır

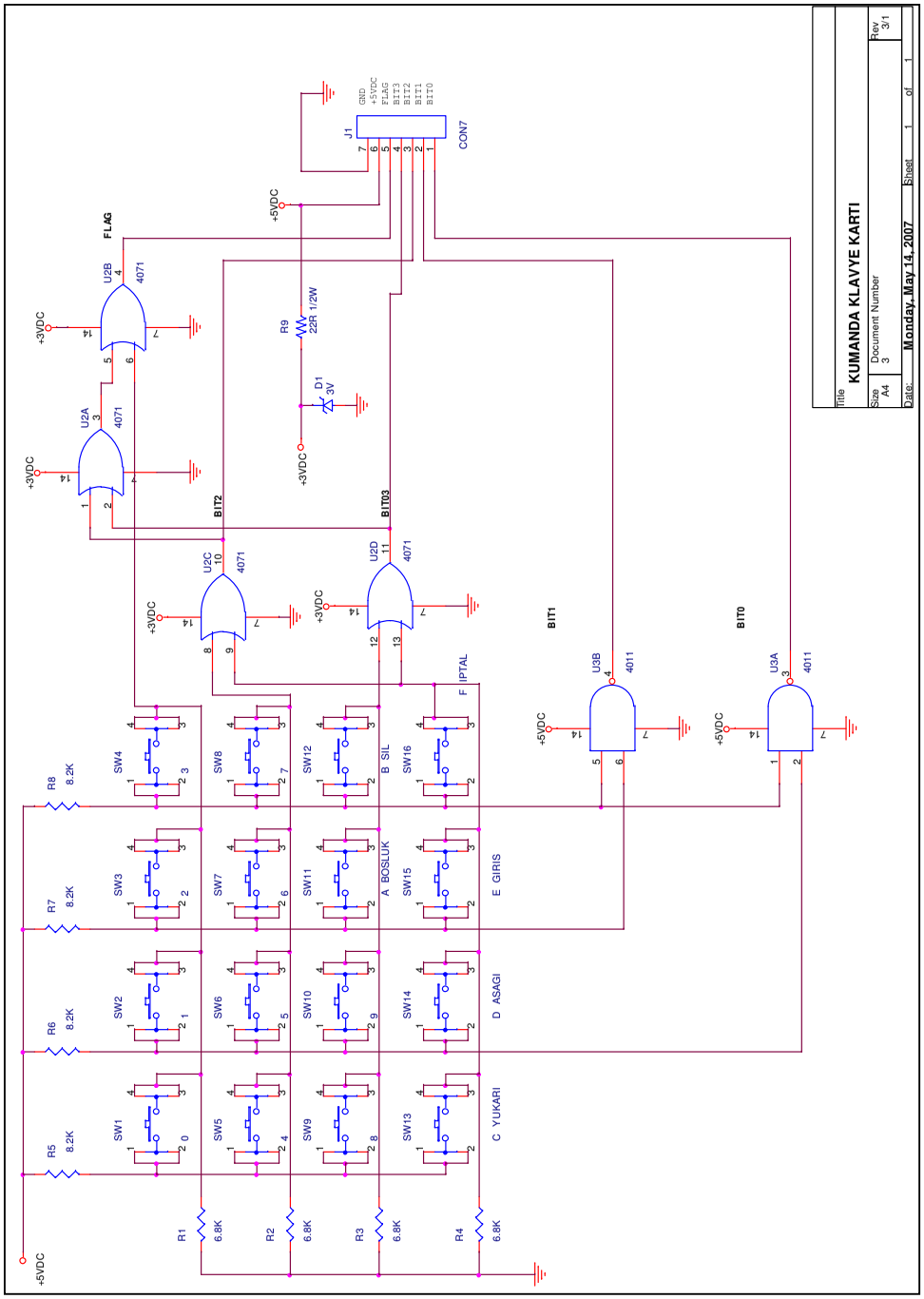
Alıcı ve verici modüllerin kontrol kartına bağlantıları devre üzerine sıralı pimler ile yapılmıştır. Devrede kullanılan 74HC00 entegresi ile mikrodenetleyicinin TX bacağı lojik olarak terslenerek vericinin data bacağına bağlanmaktadır. Bunun nedeni TX bacağı data yokken normalde 5V seviyesindedir. Bu seviye terslenerek vericinin data göndermesi engellenmektedir. Aynı şekilde alıcının çıkışı da mikrodenetleyicinin RX bacağına terslenip bağlanmaktadır. Çünkü gönderdiğimiz data'yı tersleyip göndermiştik.



Şekil 4.6 PLC kartı devre şeması

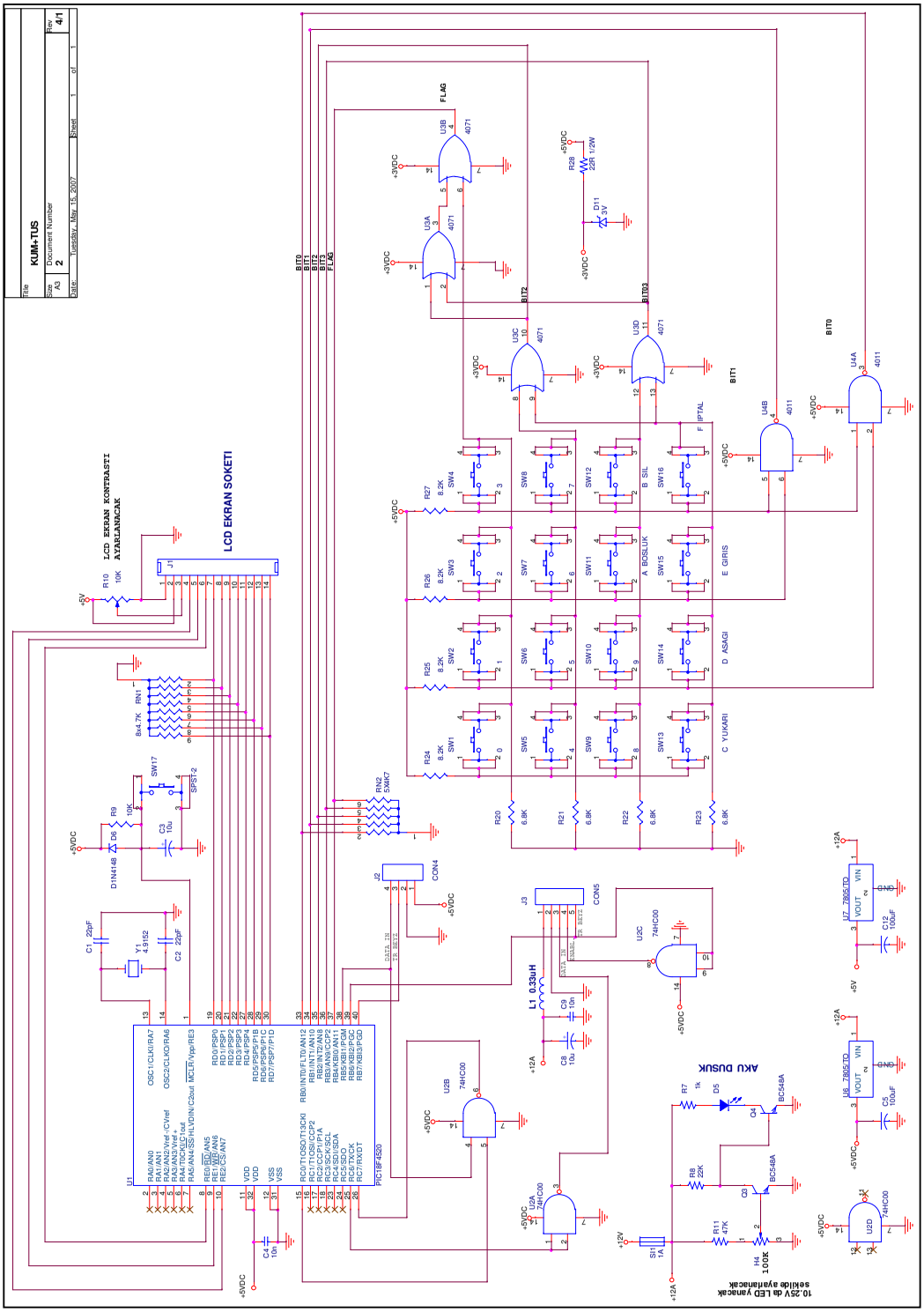
4.7 Kumanda Cihazı Devreleri

Şekil 4.7'de kumanda biriminde kullanılan tuş takımının devre şeması görülmektedir. Kumanda birimi tuş takımı tasarımında tuşlara basılınca dört bit hexadecimal çıkış ve flag datası üreten aşağıdaki devre kullanılmıştır. İşlemci tarafından okunan bu girişler yazılım ile decimal sisteme çevrilmektedir.

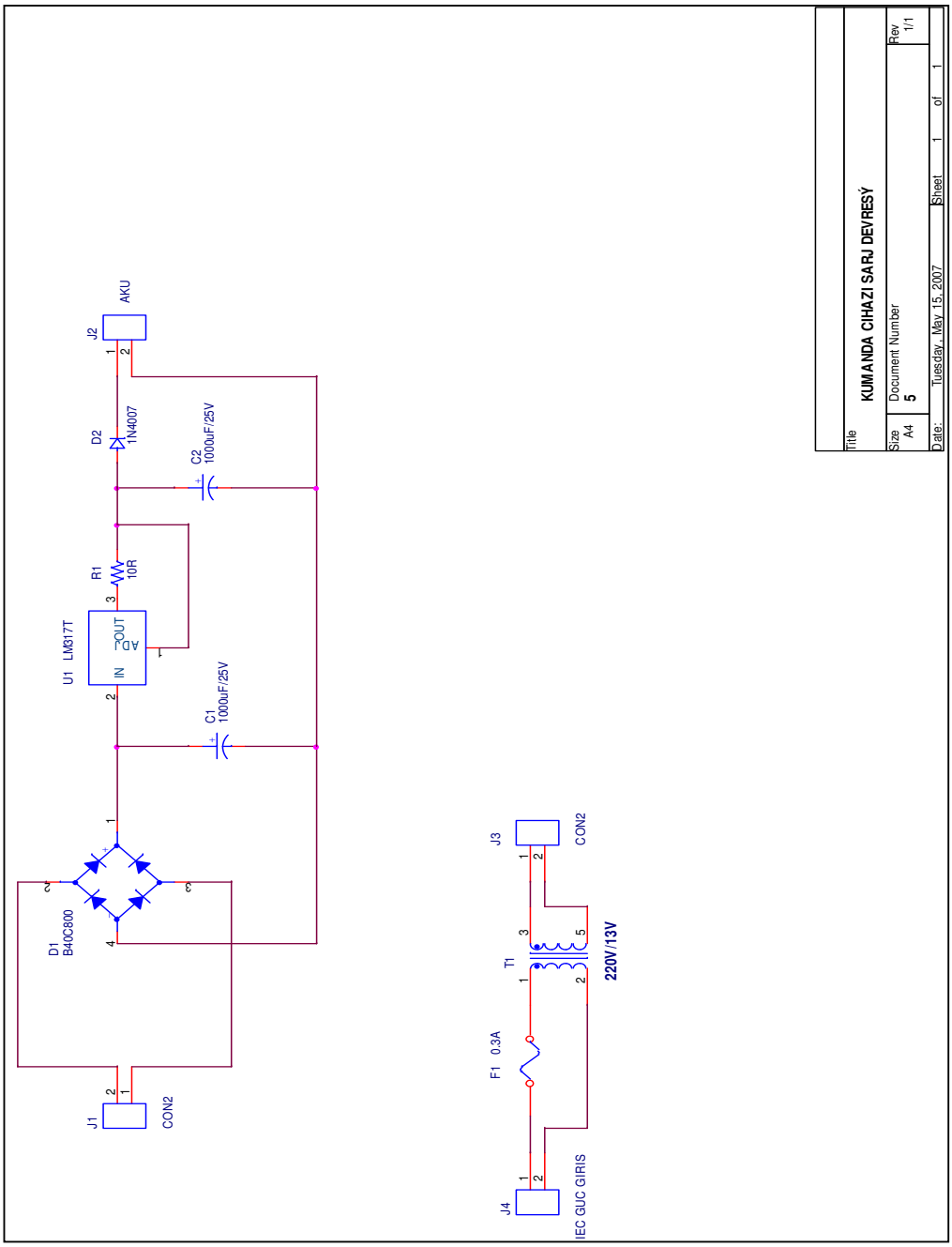


Şekil 4.7 Tuş takımı devre şeması

Şekil 4.8'de kumanda biriminde kullanılan mikrodenetleyiciye bağlı kontrol devresi görülmektedir. Tuş takımı devresinin bağlantısı, LCD ekran kontrol uçları ile RF alıcı ve vericiyi kontrol eden bağlantı noktaları açıkça görülmektedir.



Şekil 4.8 Kumanda birimi devre şemasının tamamı

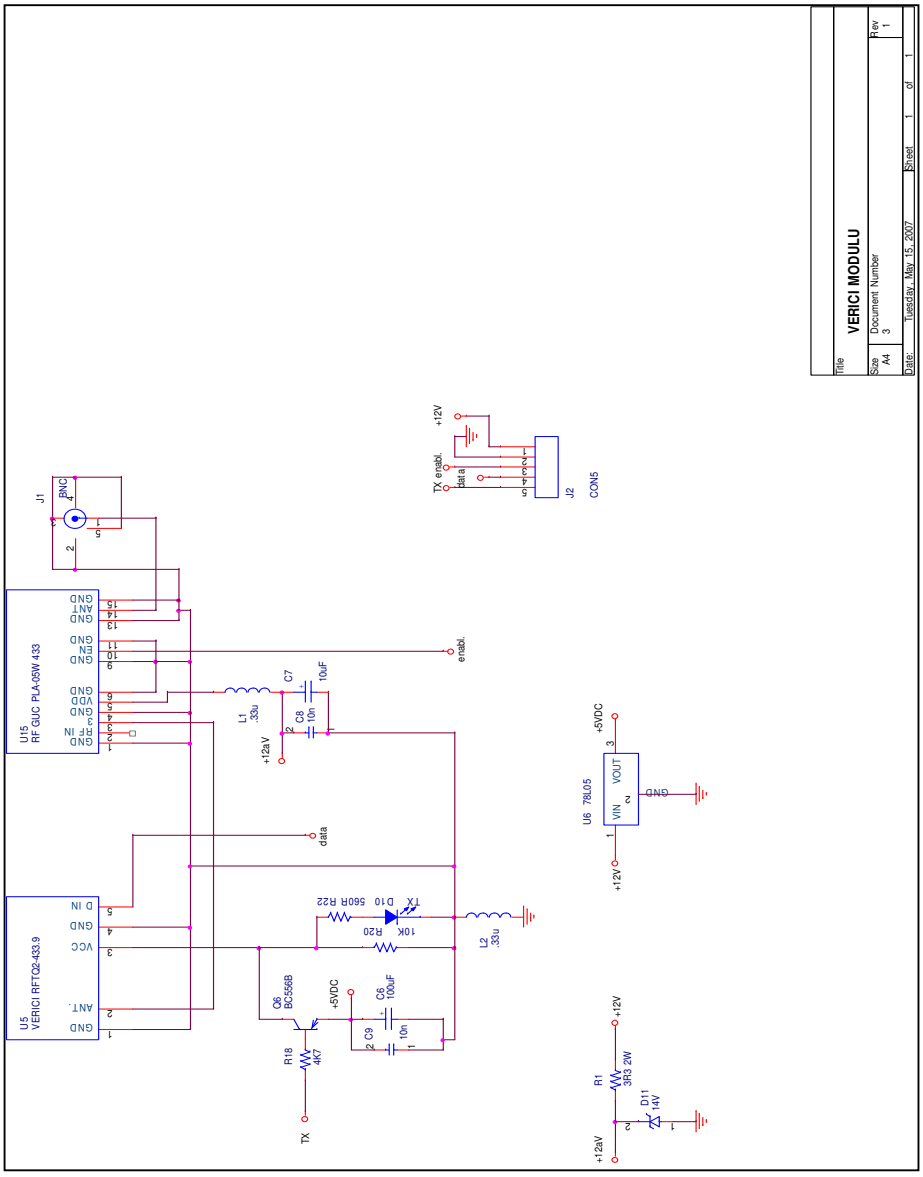


Şekil 4.9 Kumanda birimi akü şarj devresi

Şekil 4.9'da kumanda biriminde kullanılan bakımsız tip 12V aküye ait şarj devresi görülmektedir. Kumanda biriminde kullanılan 12V aküyü boşaldığında şarj etmek amacıyla LM317 entegresi kullanılarak 220VAC den şarj devresi yapılmıştır. Akünün düşük voltaj değerini kumanda biriminde bulunan akü düşük devresi ve devrede bulunan led ile tespit etmek mümkündür.

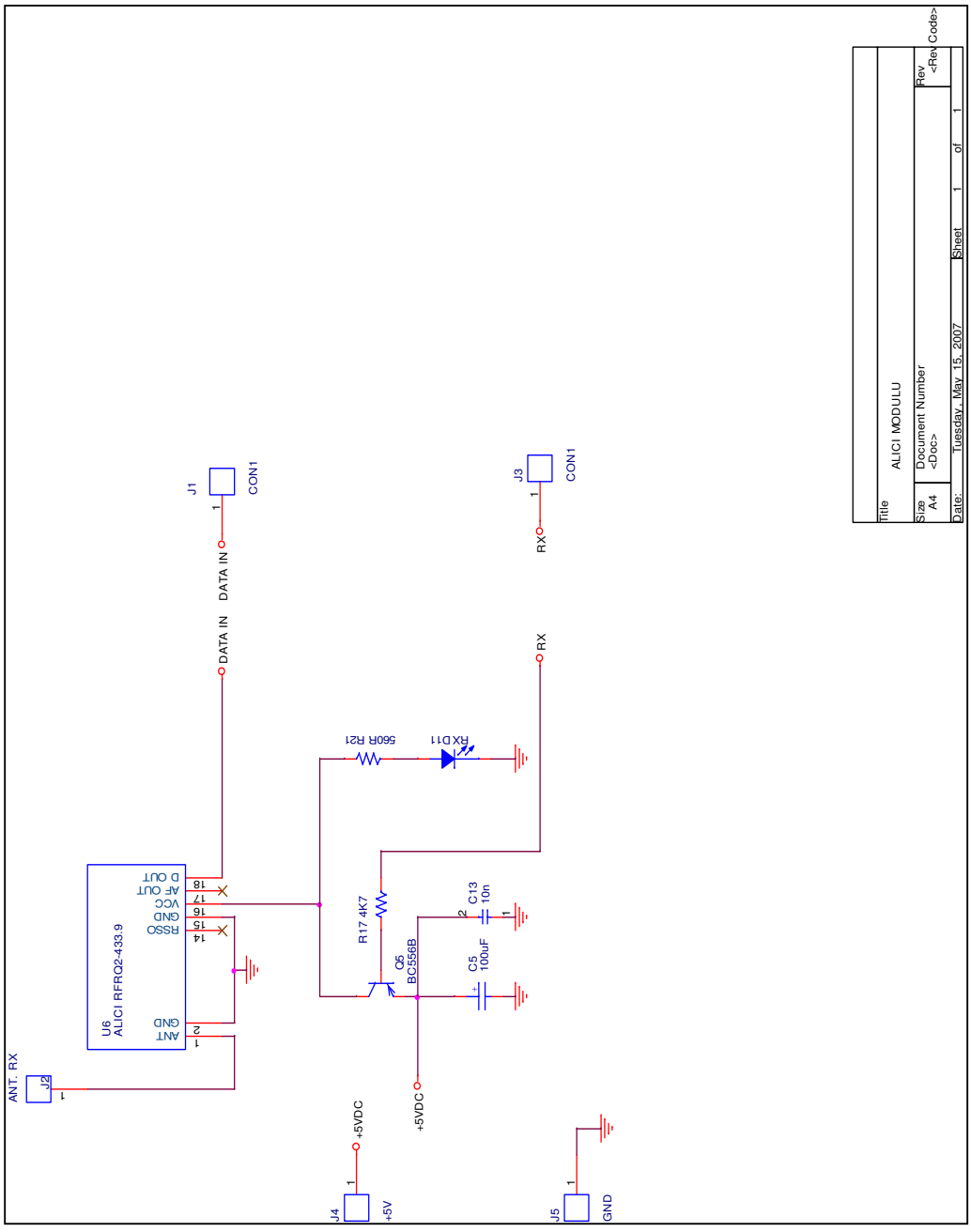
4.8 Sistemin Alıcı ve Verici Devreleri

Sistemin RF vericisi aşağıdaki devrede TX ucuna verilecek 0V ile aktif duruma gelmektedir. RF vericinin data ucuna çalışma esnasında mikrodenetleyiciden gönderme bilgileri gelmektedir. RF vericinin anten çıkışı güçlendirilmek üzere RF güç yükseltecinin 3 numaralı bacağına bağlanır. Yükseltecin EN ucuna verilecek 0V ile yükselteç giriş yapılan RF sinyali 0.5W güçlendirerek anten çıkışına verir. Şekil 4.10'da sistem verici devresi görülmektedir.

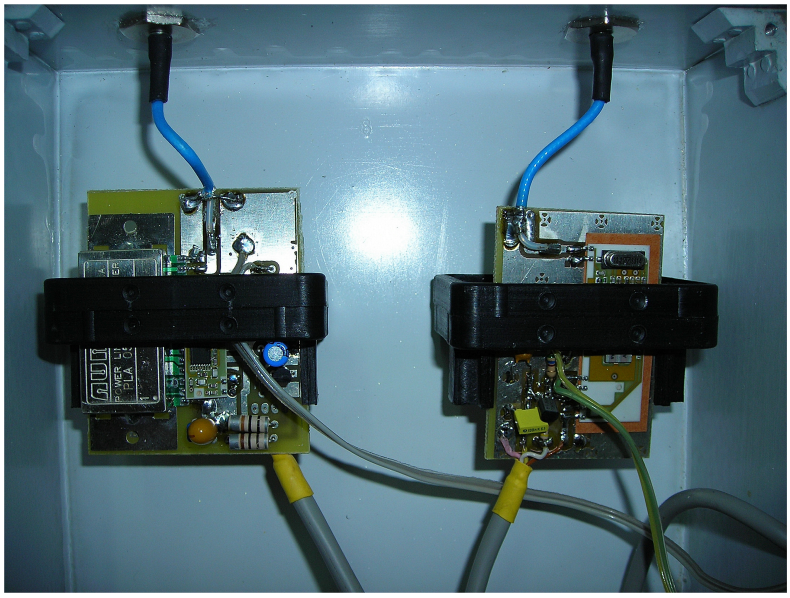


Şekil 4.10 Verici devre şeması

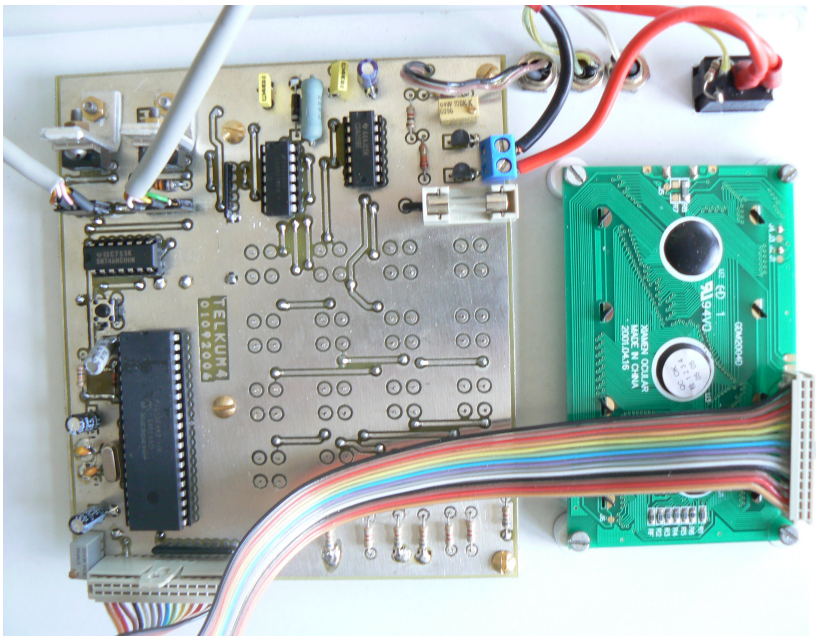
Sistemin RF alıcı devre şeması Şekil 4.11'de görülmektedir. Alıcı devresinde RX bağlantı ucuna 0V verilmesi ile alıcı aktif olur. Antenden aldığı sinyalleri data ucu ile mikrodenetleyiciye gönderir.



Şekil 4.11 Alıcı devre şeması

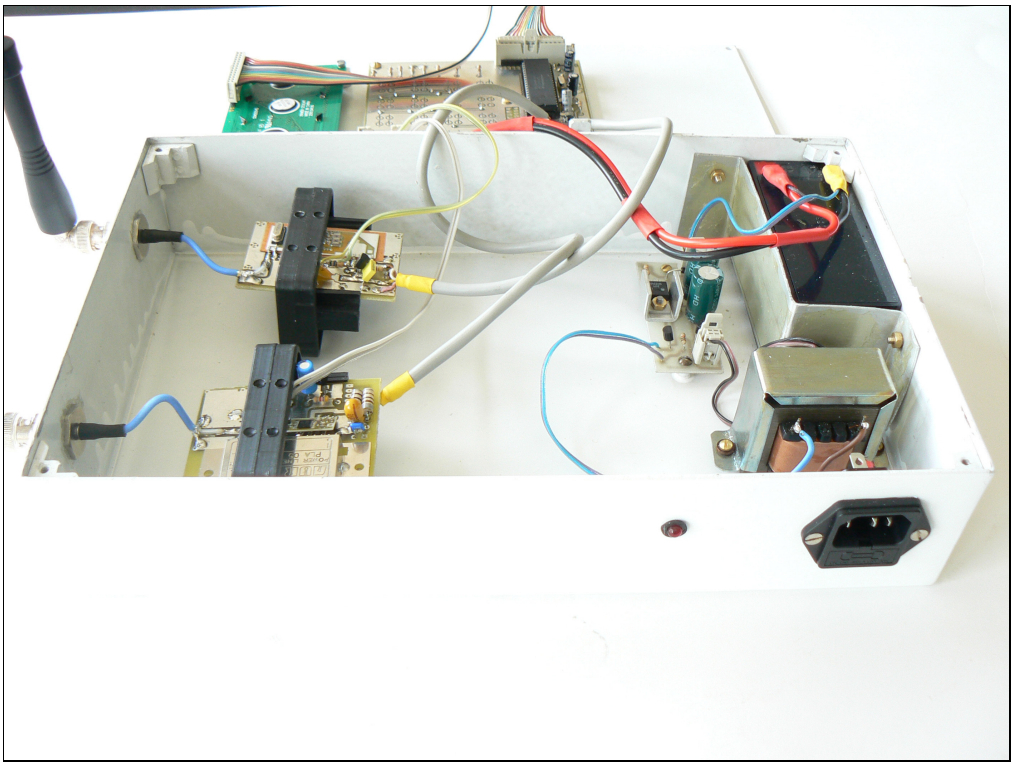


Fotoğraf 4.14 Kumanda birimi alıcı ve verici kartları

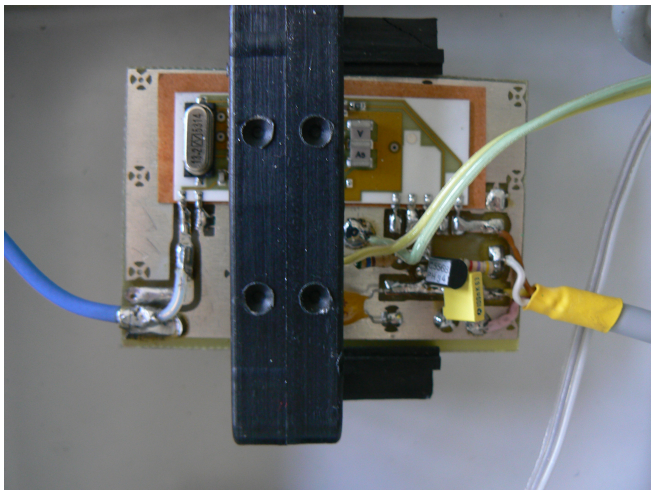


Fotoğraf 4.15 Kumanda birimi kontrol kartı

Fotoğraf 4.14'te kumanda birimi kutusu içinde kullanılan RF alıcı ve verici kartlarının montajı ve bağlantıları görülmektedir. Fotoğraf 4.15'te kumanda birimi kontrol kartının yerleşimi, bağlantıları ve basılmış devresi görülmektedir.

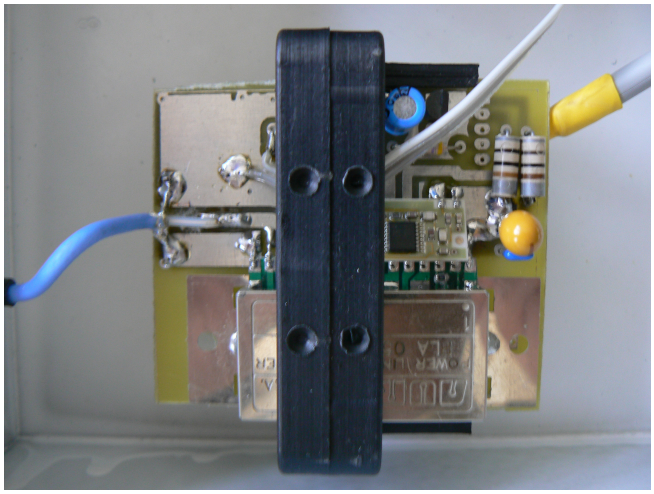


Fotoğraf 4.16 Kumanda birimi

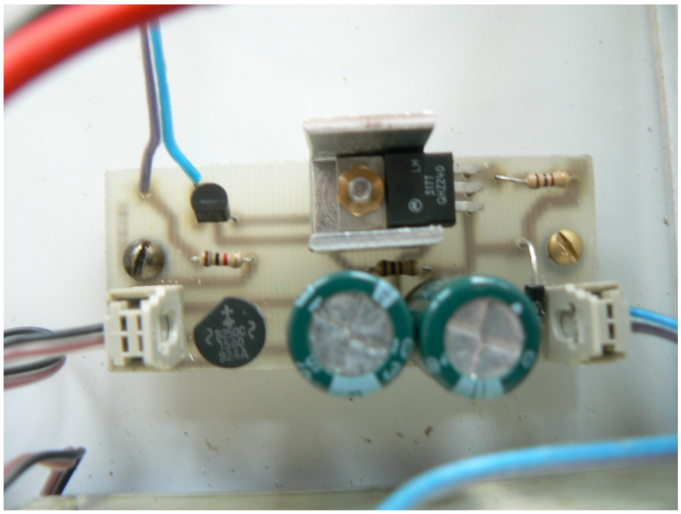


Fotoğraf 4.17 Alıcı kartı

Fotoğraf 4.16'da kumanda biriminin akü bağlantıları görülmektedir. Fotoğraf 4.17'de sistemin RF alıcısına ait montaj gösterilmektedir.



Fotoğraf 4.18 Verici kartı

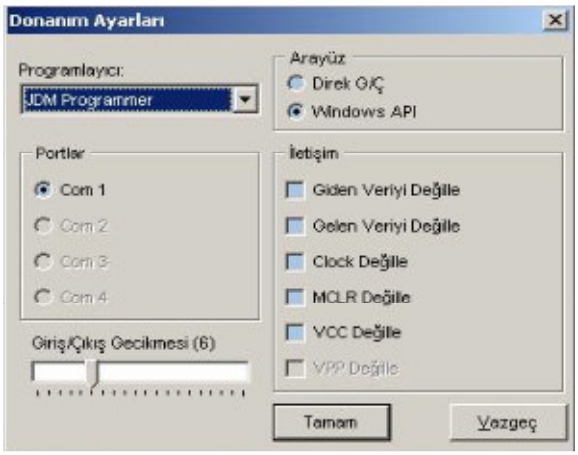


Fotoğraf 4.19 Kumanda birimi akü şarj kartı

Fotoğraf 4.18'de sistemin vericisine ait montaj ve fotoğraf 4.19'da kumanda biriminde kullanılan bakımsız tip 12V akünün şarj devresine ait yerleşim ve montaj görülmektedir.

4.9 ICPROG Programı ve ICSP Programlama Devreleri

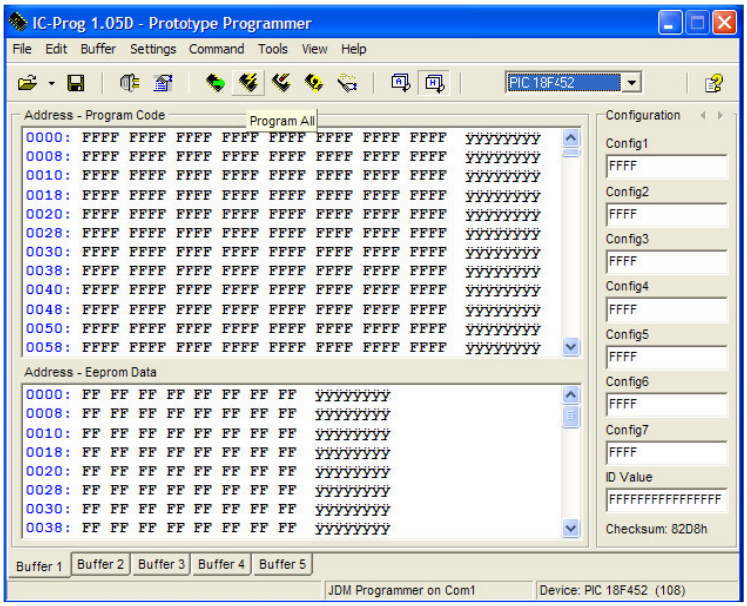
Derlenmiş olan C kodları ICPROG programı ve bilgisayarın seri portuna bağlanmış bir donanım yoluyla mikrodenetleyiciye yüklenir. Kullanılan mikrodenetleyici PIC 18F452 tarafından desteklenen ICSP sayesinde uygulama kartından entegre sökülmeden kodlar yüklenebilmektedir. ICPROG programı ilk defa çalıştırıldığında Şekil 4.12'de görülen pencere ekrana gelir.



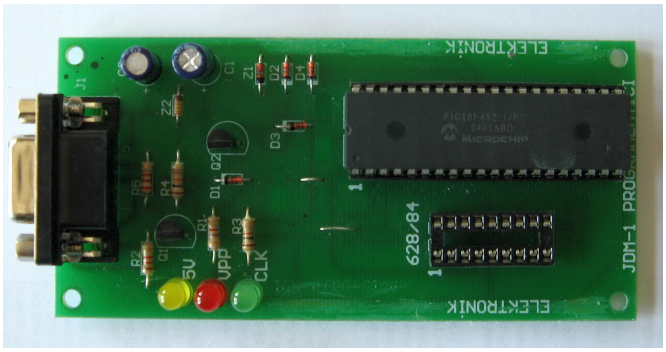
Şekil 4.12 ICPROG programı ilk defa çalıştırıldığında ekran görüntüsü

Bu pencerede görülen arayüz kısmında işletim sistemine bağlı olarak bir değişiklik yapmak gereklidir. Eğer NT tabanlı bir işletim sistemi kullanılıyorsa (Windows NT, 2000, XP) Windows API seçilmelidir. Programlayıcı kısmından kullanılan programlayıcı kartı seçilmelidir. Portlar kısmında ise programlayıcı donanım hangi porta bağlı ise o port seçilmelidir. Giriş/Çıkış Gecikmesi diye adlandırılan kısımda ise kullanılan mikrodenetleyicinin desteklediği bir gecikme seçilmelidir. Bu çalışmada kullanılan mikrodenetleyici olan PIC 18F452 için 10 gecikme seviyesi seçilerek çalışma yapılmıştır. Belirtilen ayarlar yapılıp tamam tuşuna basıldıktan sonra ICPROG programının Şekil 4.13'te gösterilen ana penceresi ekrana gelir. Bu ekranın sağ üst bölgesinde bulunan seçim kutusundan programlamak istenen entegre numarası seçilir. Daha sonra Dosya menüsünden aç seçeneğine tıklanarak entegreye yüklenmek istenen hex dosya açılır ve "Program All" butonuna basılır. PIC 18F452 mikrodenetleyicisini devre üzeri seri programlamak için clock ve veri için iki hat, besleme, toprak ve programlama voltajı için üç hat olmak üzere toplam beş hat gereklidir. Bu hatlar sırası ile PGD, PGCLK, VDD, VSS ve MCLR pinlerine karşılık gelmektedir. Bu çalışmada

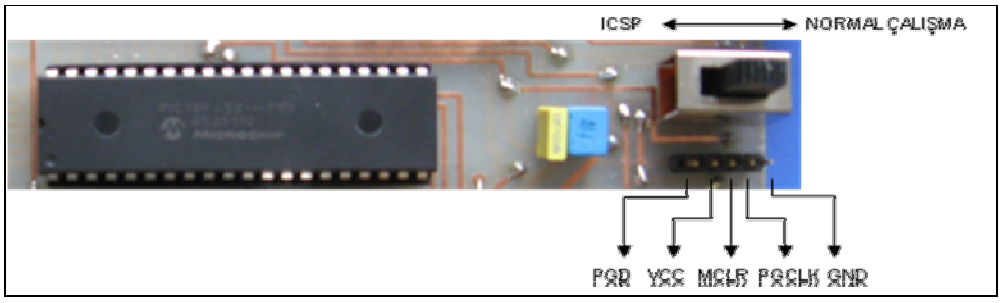
kullanılan PIC 18F452 mikrodenetleyicisini ICSP kullanılarak programlamak için Fotoğraf 4.20'de verilen JDM-1 Programlayıcı devresi kullanılmıştır. Programlayıcı devresinin PLC kartına bağlanacağı ICSP uçları Şekil 4.14'te verilmiştir.



Şekil 4.13 ICPROG programının ekran görüntüsü



Fotoğraf 4.20 PIC 18F452 ICSP programlayıcı devre

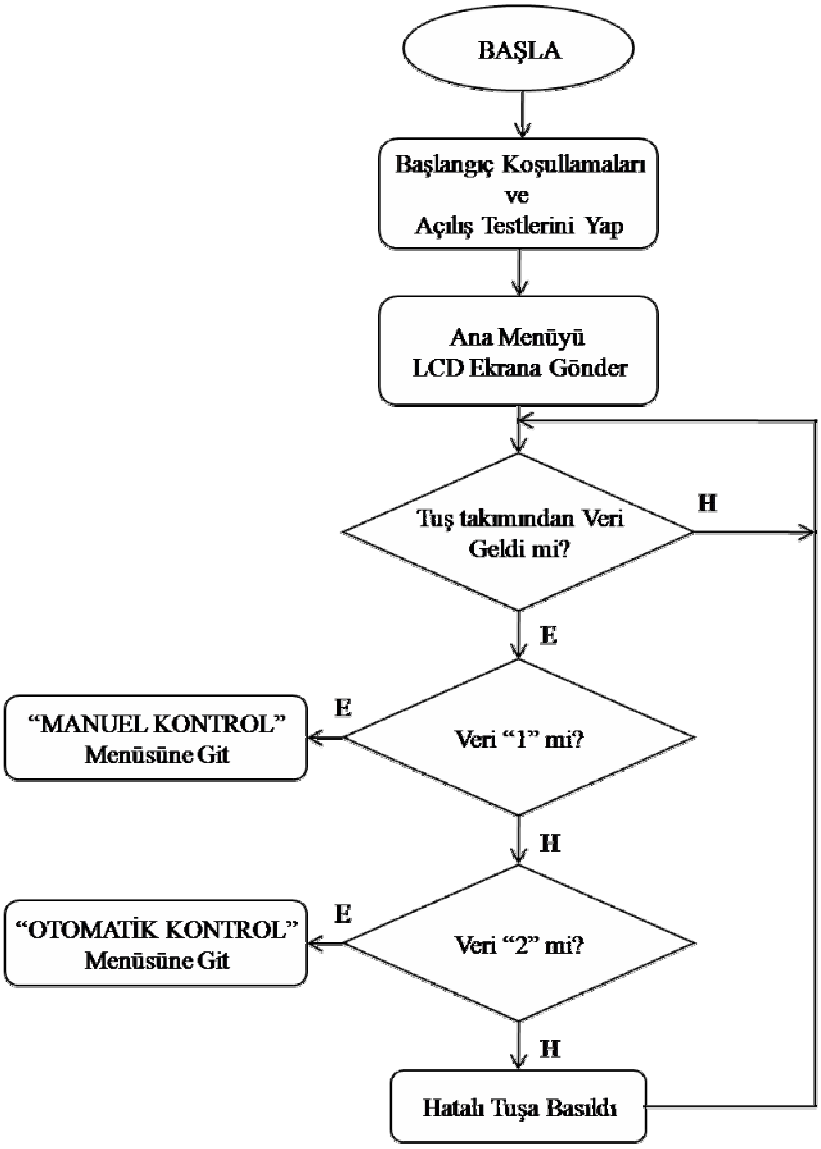


Şekil 4.14 PLC kartına ait PIC 18F452 ICSP bağlantı uçları

BÖLÜM V

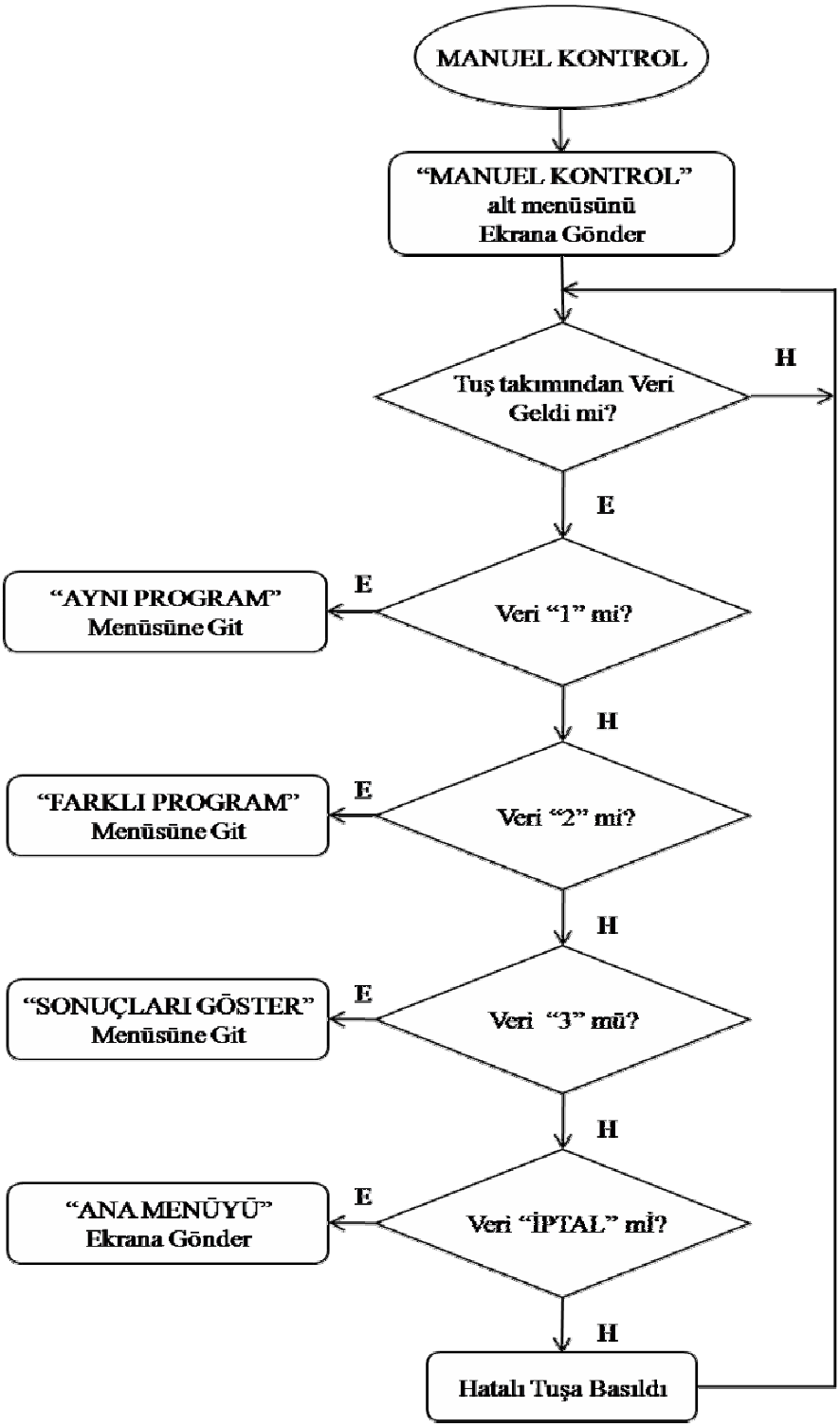
AKIŞ DİYAGRAMLARI

Gerçekleştirilmiş olan sisteme ait yazılımı ifade eden akış şemaları bu bölümde incelenmektedir. Sistem yazılımı EK-A'da görülmektedir. Akış diyagramları mikrodenetleyicinin çalıştırmakta olduğu programın akış sırasını gösterir. Akış diyagramlarından programın nasıl işlediği izlenebilir.



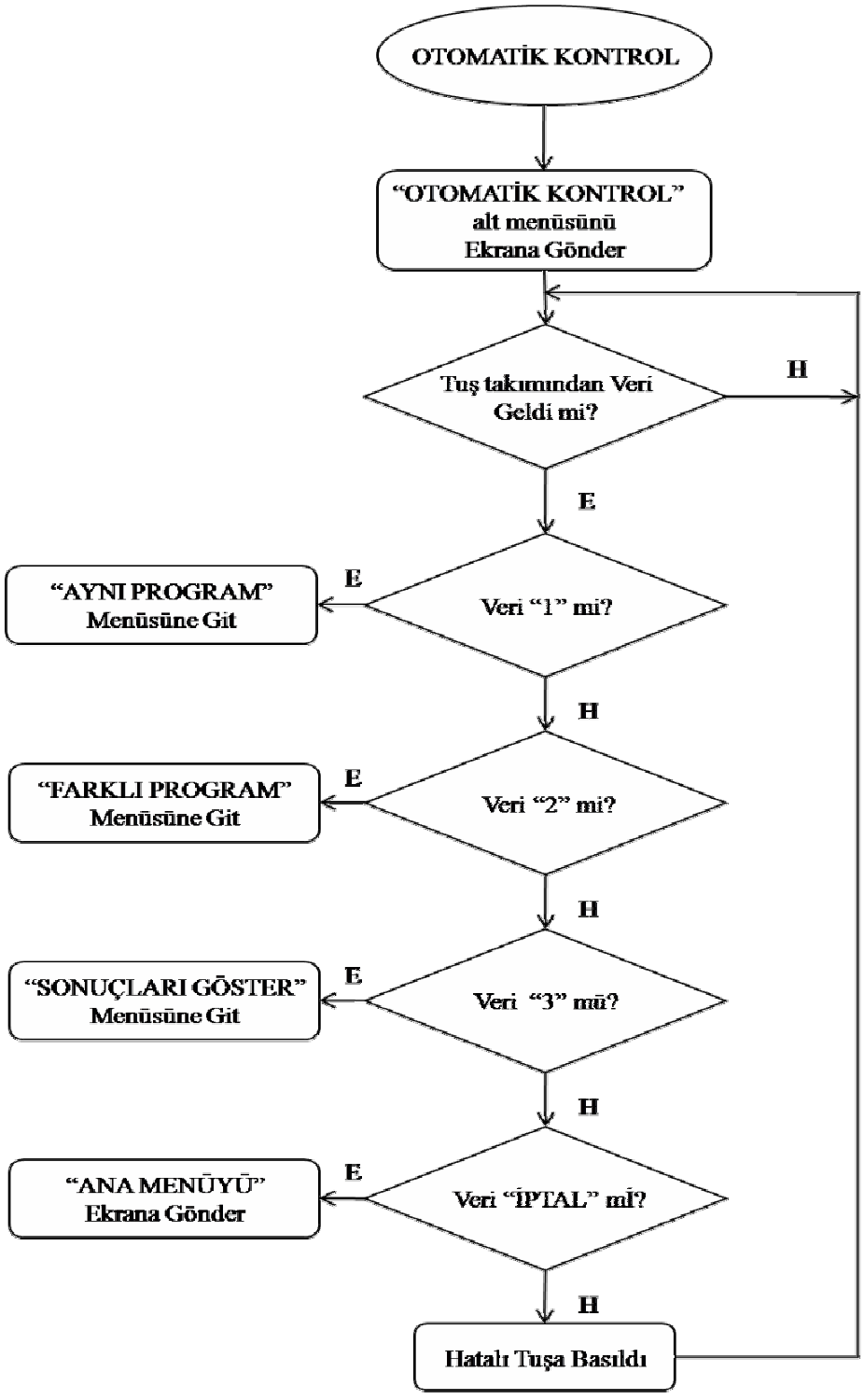
Şekil 5.1 Kumanda birimi ana menüsüne ait akış diyagramı

Şekil 5.1'de görülen akış diyagramı manüel veya otomatik çalışma modlarına geçilmesini sağlar. Tuş takımından 1 değeri girilirse manüel, 2 değeri girilirse otomatik çalışma moduna geçer.



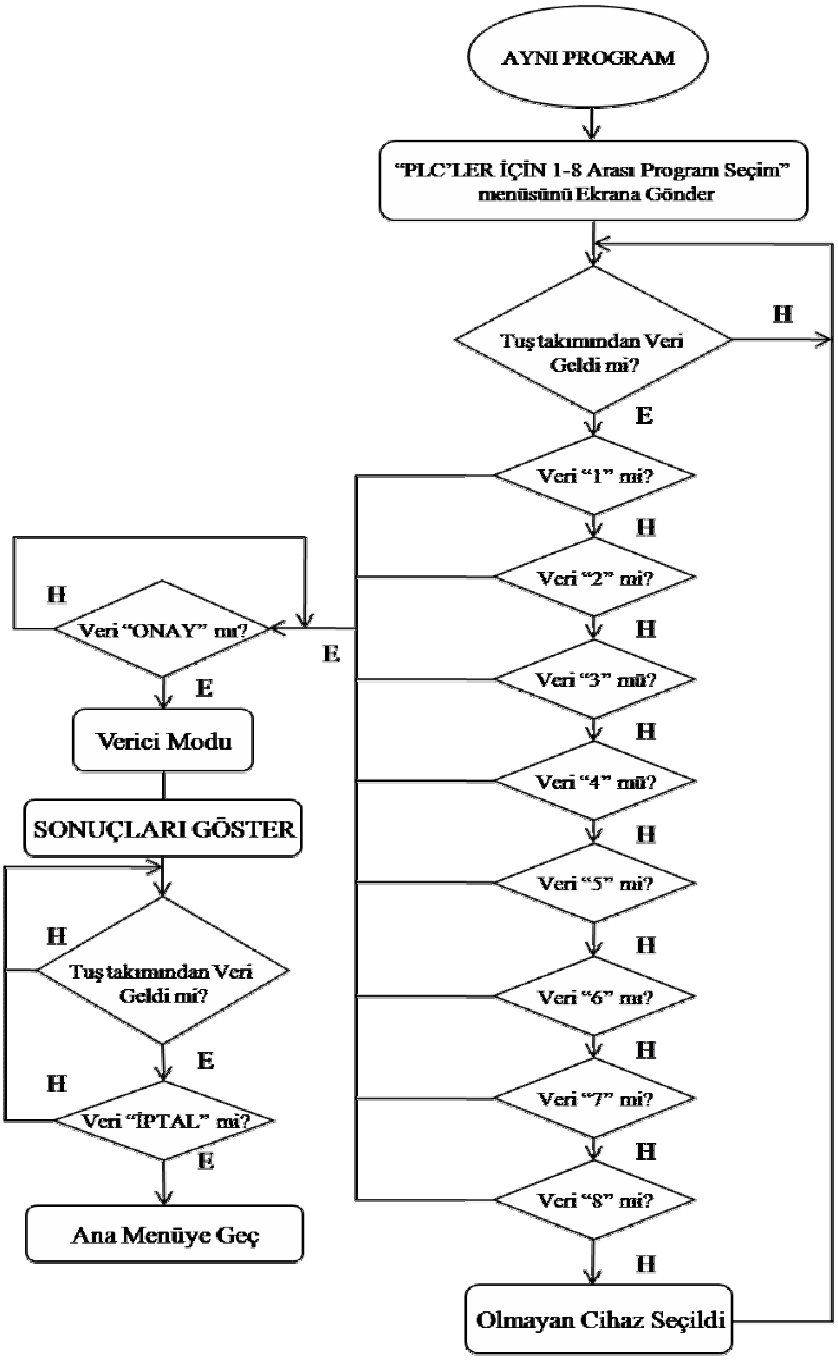
Şekil 5.2 Manüel kontrol alt menüsüne ait akış diyagramı

Şekil 5.2'de görülen diyagram ile manüel kontrol çalışmasında tuş takımından 1 ve 2 değeri girilerek, program yükleme işlemi menülerine geçilir.

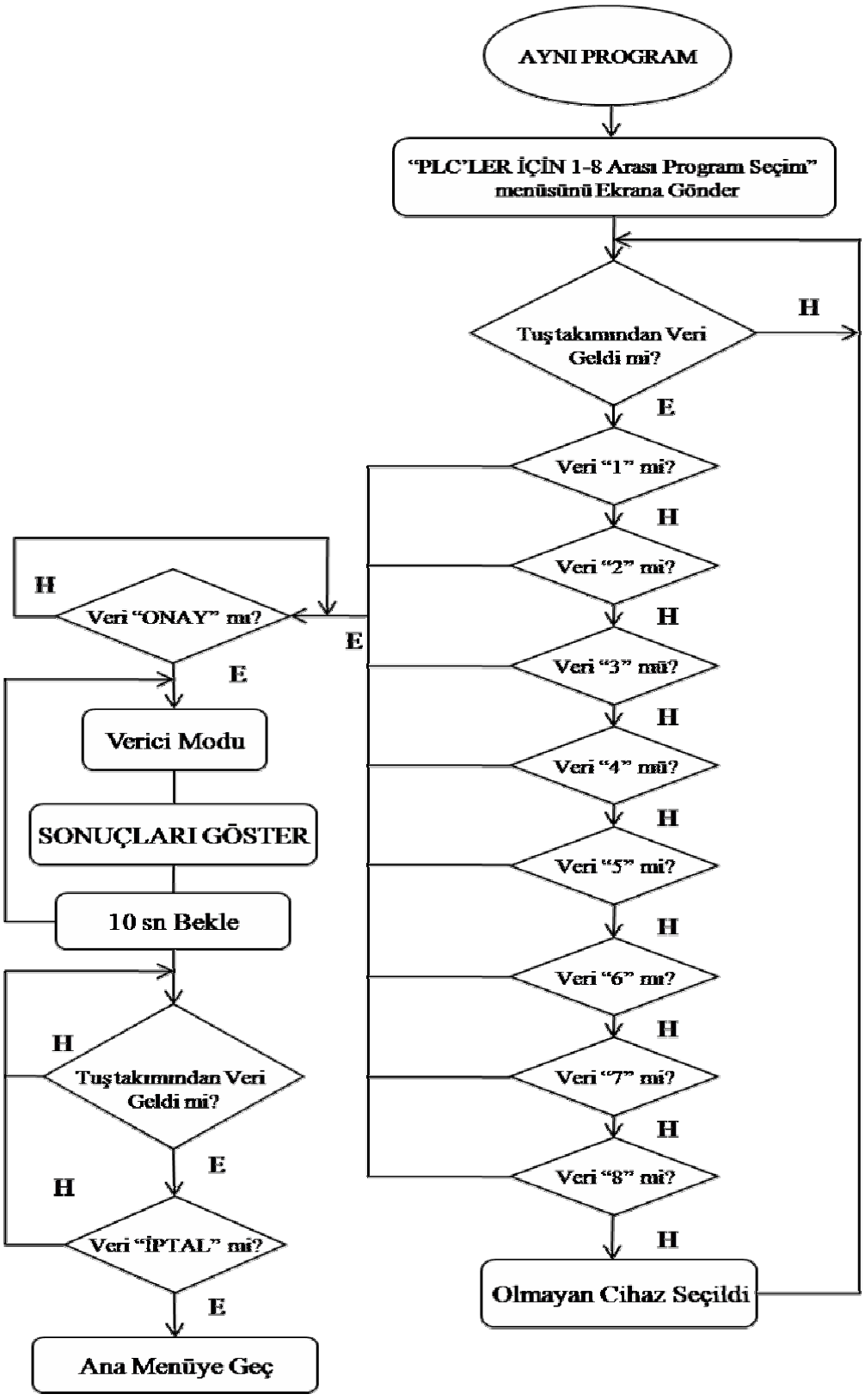


Şekil 5.3 Otomatik kontrol alt menüsüne ait akış diyagramı

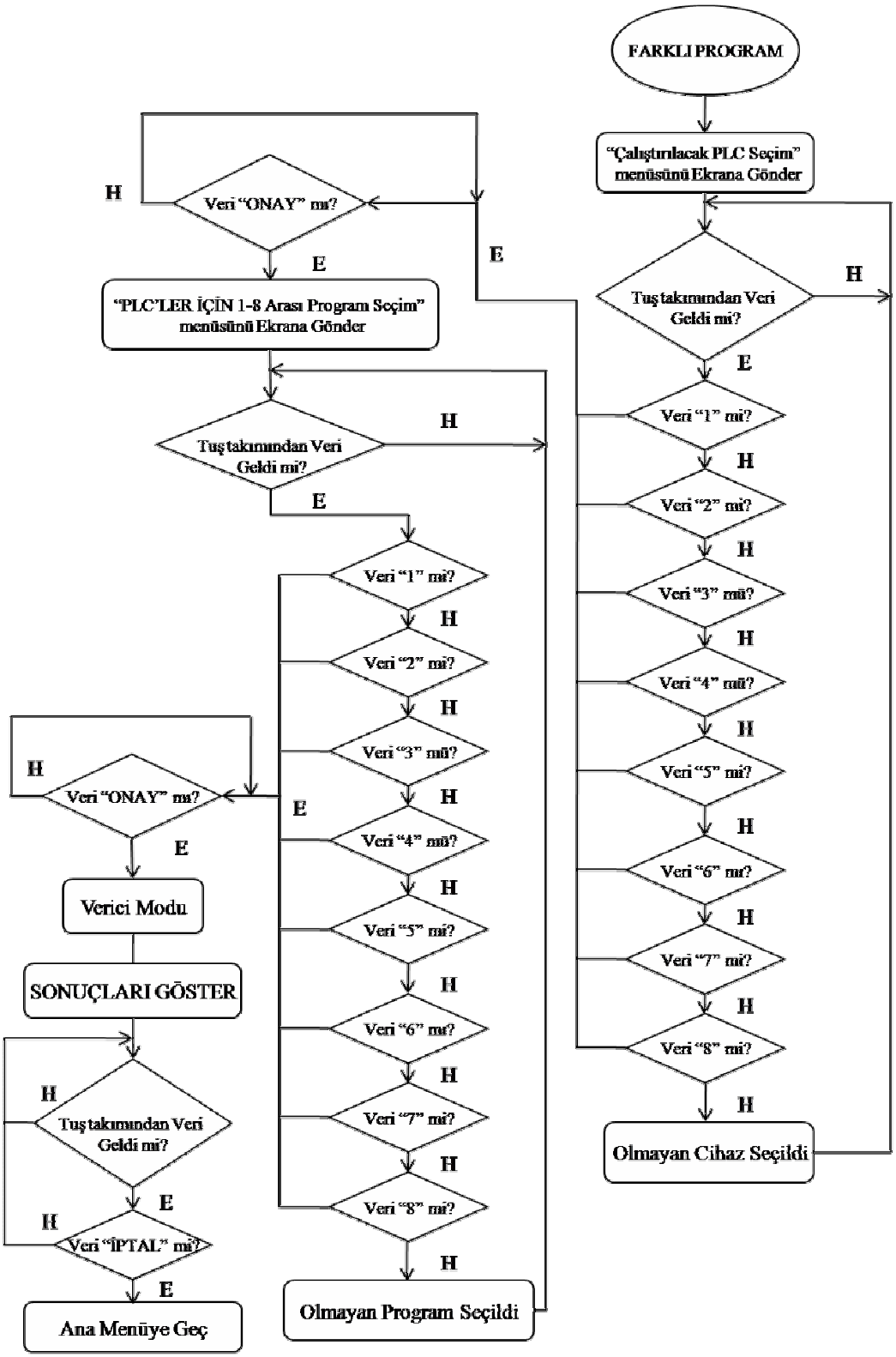
Şekil 5.3'te otomatik kontrol çalışmasında iken tuş takımından 1 ve 2 değeri girilerek program yükleme işlemi menülerine giriş sağlanır.



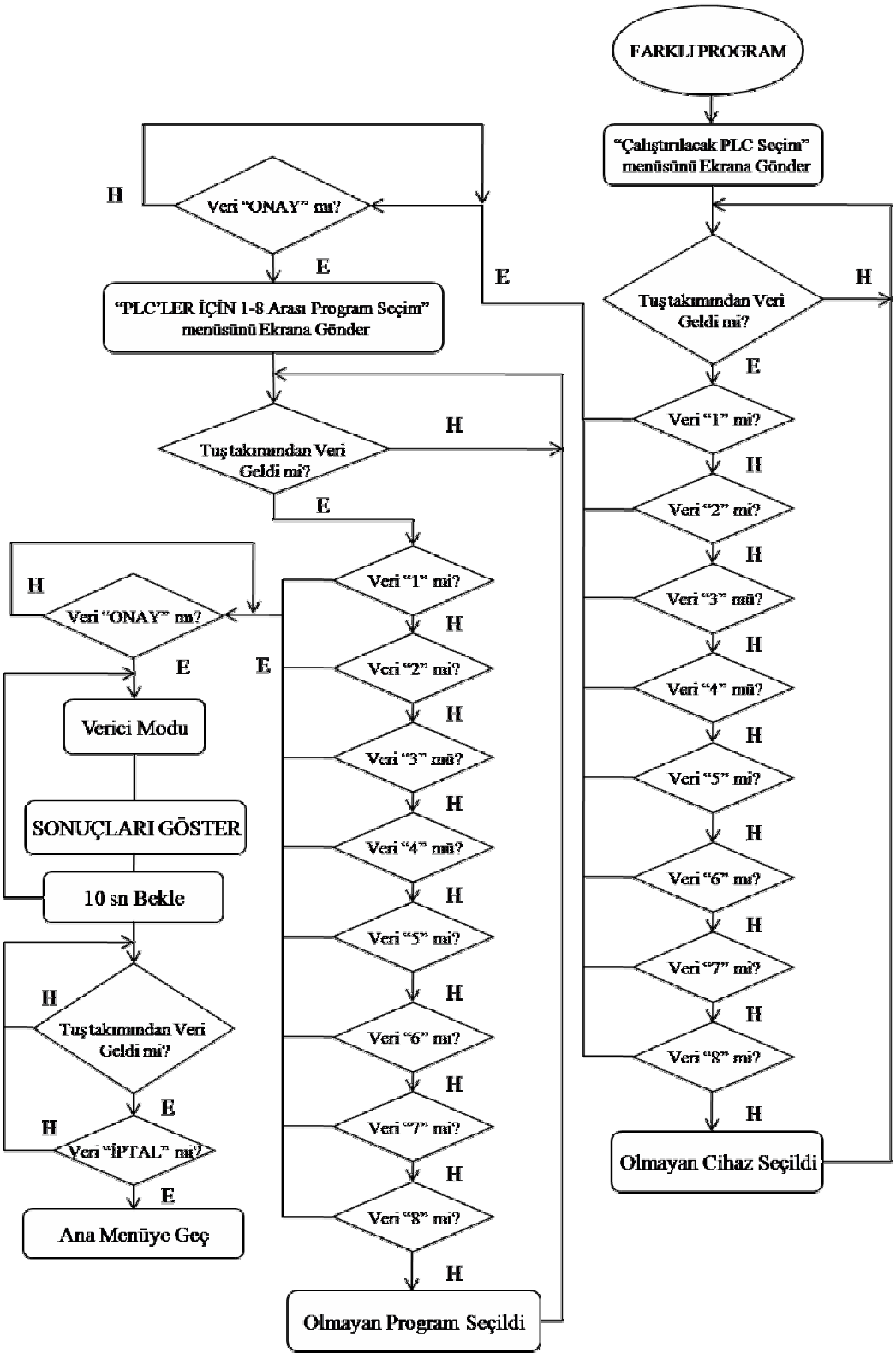
Şekil 5.4 Manüel kontrol-aynı program alt menüsüne ait akış diyagramı



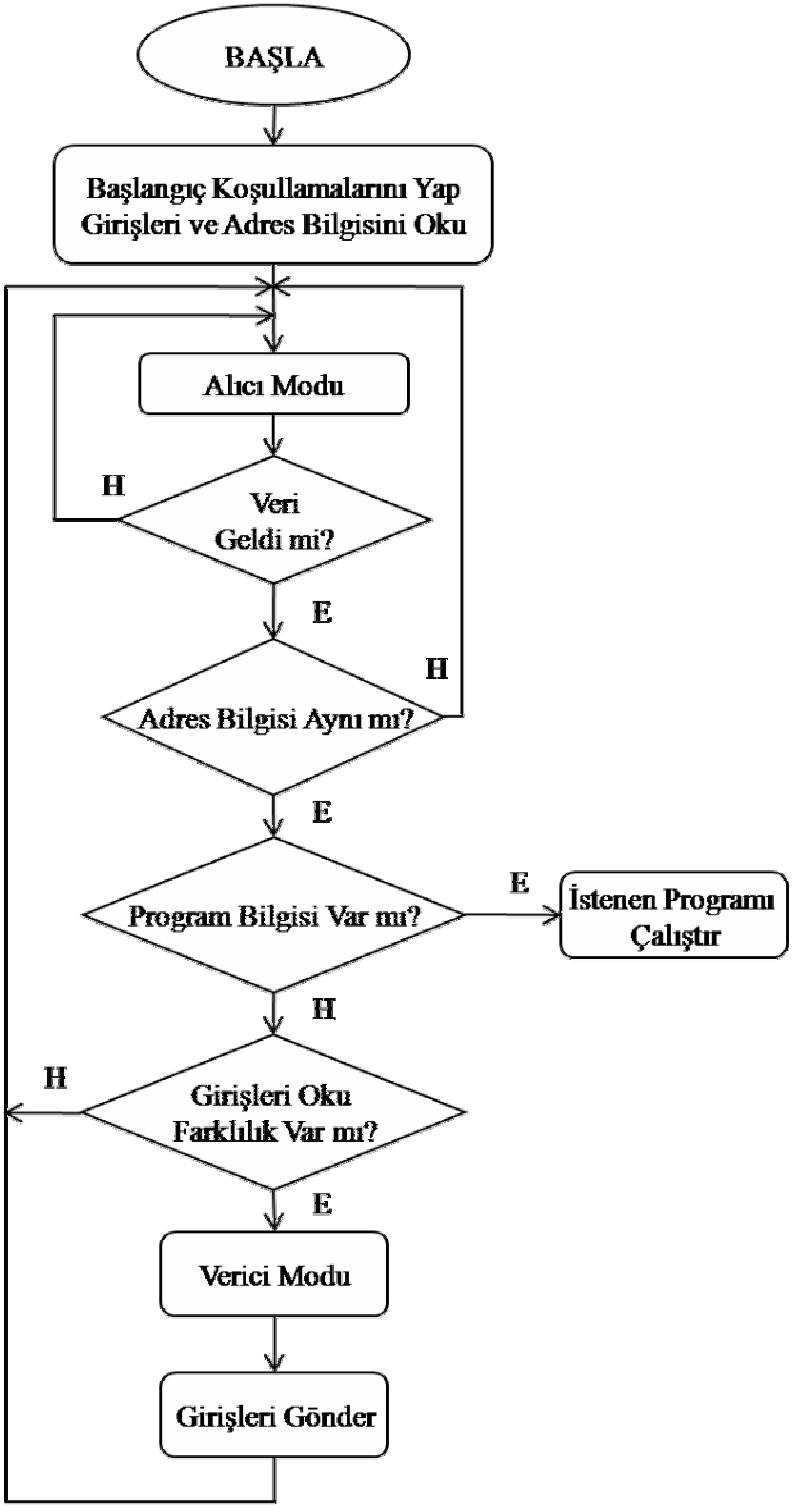
Şekil 5.5 Otomatik kontrol-aynı program alt menüsüne ait akış diyagramı



Şekil 5.6 Manüel kontrol-farklı program alt menüsüne ait akış diyagramı



Şekil 5.7 Otomatik kontrol-farklı program alt menüsüne ait akış diyagramı



Şekil 5.8 PLC kartına ait akış diyagramı

Şekil 5.4'teki diyagram da tüm PLC kartlarına seçilen program gönderilir. PLC kartlarından gelen giriş değerleri binary olarak ekranda gösterilir.

Şekil 5.5'te tüm PLC kartlarına seçilen program gönderilir. PLC kartlarından gelen giriş değerlerinde farklılık olanlara yeni program gönderilir.

Şekil 5.6'da seçilen PLC kartlarına seçilen programlar gönderilir. PLC kartlarından gelen giriş değerleri binary olarak ekranda gösterilir.

Şekil 5.7'de seçilen PLC kartlarına seçilen programlar gönderilir. PLC kartlarından gelen giriş değerlerinde farklılık olanlara yeni program gönderilir.

Şekil 5.8'de PLC kartına ait programın akış diyagramı verilmiştir. Veri alındıktan sonra ilgili program çalıştırılır. Kart sürekli olarak alıcı modunda, vericisi kapalı alıcısı açık olarak bekler. Eğer gelen veri, giriş değerlerini soruyorsa tüm PLC'ler verici moduna geçer, alıcısını kapatıp vericisini açar ve girişlerindeki değerleri sıralı olarak sekiz adet PLC 10 sn içinde gönderir. Çizelge 5.1'de kumanda biriminden gönderilen protokol bilgisinin miktarı ve şekli verilmiştir.

Çizelge 5.1 İletişim veri paketi

PLC Adresi	Yüklenecek Program	Program İcra Komutu	Giriş Bilgisi İste Komutu	Hata Toplamı
----------	0x68 + Prg.Bilgisi	0x28	0x58	Crc = 0
1 bayt	1 bayt + 1 bayt	1 bayt	1 bayt	1 bayt

BÖLÜM VI

SONUÇ VE ÖNERİLER

PIC 18F452 kullanarak gerçekleştirilen RF kontrollü PLC uygulaması olan bu çalışma ile 18 serisi PIC mikrodenetleyicisi kullanımı, özellikleri ve standart C dili ile programlanması öğrenilmiştir.

Sistemde RF alıcı ve verici olarak ABACOM firmasının modülleri kullanılmıştır. Öncelikle alıcı ve verici devreleri tasarlanmıştır. Anten olarak 433 MHz frekans bandında çalışan antenler kullanılmıştır. Sistemin mesafe testlerinde 1km ve üzeri mesafede çalışmasına olanak sağlayan ve antenden yayılan gücün maksimum seviyeye ulaşması için anten SWR değerini uygun değere çeken kutular tasarlanmış ve gerçekleştirilmiştir.

Kontrol kartları 24V-5V girişler ve 10A 250VAC röle çıkışları ile PLC sistemleri örnek alınarak gerçekleştirilmiştir. Röle çıkışlarına bağlanacak motor, lamba, sensör gibi elemanların kontrolü mümkündür. Dolayısı ile gerçekleştirilen bu sistem genel amaçlı bir sistemdir. Kontrol kartlarında bulunan devre üzeri seri programlama ile mikrodenetleyicileri sökmeden programlama olanağı sağlanmıştır. Her kontrol kartının ayrı ayrı programlaması yapılabilmektedir. Kumanda kutusu üzerinde yer alan LCD ve tuş takımı ile kullanıcının erişim ara yüzü sağlanmıştır.

Sanayimizde ve pratikte yeni bir uygulama olan bir çalışma ile RF kontrollü otomasyon sistemlerine öncülük edilmiştir.

KAYNAKLAR

[1] Suyabatmaz, B., Kablosuz Veri İletimi İçin Alıcı-Verici Geliştirme Kartı Tasarımı ve Örnek bir Uygulama, Yüksek Lisans Tezi, G.Y.T.Ü. Fen Bilimleri Enstitüsü, Gebze, 2006.
[2] Gümüşkaya, H., Mikroişlemciler ve 8051 Ailesi, İstanbul, 1999.
[3] Doğan, İ., PIC C ile PIC Programlama, İnfogate, İstanbul, 2004.
[4] www.microchip.com.
[5] MPLAB C18 C COMPİLER Getting Started, Microchip, 2005.
[6] MPLAB C18 C COMPİLER Getting Libraries, Microchip, 2005.
[7] Arslan, K., A'dan Z'ye C Kılavuzu, Pusula Yayıncılık, İstanbul, 1997.
[8] www.abacom.com.
[9] www.national.com.
[10] MPLAB C18 COMPİLER. [CD-ROM]. Microchip, 2005.

EK-A

SİSTEM YAZILIMI

```
**************************GENEL BAŞLIK DOSYALARI*****************************
*********************************Config.h*************************************
#ifndef __CONFIG_H
#define __CONFIG_H
#pragma config LVP = OFF
#pragma config WDT = OFF
#pragma config OSC = XT
#endif
*****************************degiskenler.h*************************************
#ifndef __DEGISKENLER_H
#define __DEGISKENLER_H
#include "hkb.h"
extern unsigned char        skor[24];
extern char                 mesaj[20];
extern uint8                genelZamanlayici;
extern uint8            gecenSure;
extern uint16           iletisimZamanAsimiSuresi;
extern uint8            gonderilen;
#endif // __DEGISKENLER_H
***************************durum.h*****************************************
#ifndef __DURUM_H
#define __DURUM_H
void anaDongu( void );
#endif // __DURUM_H
***************************Eeprom.h****************************************
#ifndef __EEPROM_H
#define __EEPROM_H
#include <global.h>
uint8 eeRead( uint8 addr );
void  eeWrite( uint8 addr, uint8 data );
#endif
***************************global.h****************************************
#ifndef __GLOBAL_H
#define __GLOBAL_H
//#define __TEST
#ifdef __TEST
```

```
#include <stdio.h>
#else
#include <p18f452.h>
#include <stdlib.h>
#include <timers.h>
#include <delays.h>
#endif
#define GECIKME_2mS()       Delay1KTCYx(3)
#define GECIKME_5mS()       Delay1KTCYx(8)
#define GECIKME_30mS()      Delay1KTCYx(37)
#define GECIKME_50mS()      Delay1KTCYx(61)
#define GECIKME_100mS()     Delay1KTCYx(122)
#define GECIKME_150mS()     Delay1KTCYx(183)
#define GECIKME_200mS()     Delay1KTCYx(244)
#define GECIKME_250mS()     Delay10KTCYx(24)
#define GECIKME_300mS()     Delay10KTCYx(36)
#define GECIKME_400mS()     Delay10KTCYx(49)
#define GECIKME_450mS()     Delay10KTCYx(55)
#define GECIKME_500mS()     Delay10KTCYx(61)
#define GECIKME_1S()        Delay10KTCYx(122)
#define GECIKME_2S()        Delay10KTCYx(244)
#define GECIKME_5S()      GECIKME_2S();GECIKME_2S();GECIKME_1S()
#define ZAMANLAYICI_ACIK()       (T0CONbits.TMR0ON)
#define ZAMANLAYICIYI_BASLAT()
WriteTimer0(HKB_TMR0_PERIYODU);T0CONbits.TMR0ON=1
#define ZAMANLAYICIYI_DURDUR()    T0CONbits.TMR0ON=0
#define GONDERME_ONCESI_BEKLE()   GECIKME_500mS()
#define GONDERME_SONRASI_BEKLE()  GECIKME_100mS()
#define DINLEME_ONCESI_BEKLE()    GECIKME_250mS()
#define GONDERME_ONCESI_BEKLEME_SURESI          10   // 500mS
#define GONDERME_SONRASI_BEKLEME_SURESI          2    // 100mS
#define TASIYICI_ZAMAN_ASIMI_SURESI             15   // 750mS
#define TASIYICI_DINLEME_ZAMAN_ASIMI_SURESI      5    // 250mS
#define GELEN_VERI_ZAMAN_ASIMI_SURESI           20   // 1S
#define GONDERME_PERIYODU_SURESI                20   // 1S
#define DINLEME_ONCESI_BEKLEME_SURESI            5    // 250 mS
#define VURULUNCA_DUSSUN                              1


typedef unsigned char byte;
```

```
typedef unsigned int  word;
typedef unsigned long dword;
typedef unsigned char uint8;
typedef unsigned int  uint16;
typedef unsigned long uint32;
typedef char int8;
typedef int  int16;
typedef long int32;
typedef void(*DurumFonk)( void );
#ifndef NULL
#define NULL 0
#endif
#endif  // __GLOBAL_H
**************************kbd.h*****************************************************
#ifndef __KBD_H
#define __KBD_H
#define getch getc
unsigned char tusBasili( void );
char klavyeKarakterDegeriniOku( void );
char getc( void );
void KlavyeyiAc( void );
#endif
**************************lcd.h*****************************************************
#ifndef __LCD_H
#define __LCD_H
#define RS_PIN     PORTEbits.RE2
#define RW_PIN     PORTEbits.RE1
#define E_PIN      PORTEbits.RE0
#define DATA_PORT              PORTD
#define TRIS_DATA_PORT     TRISD
#define DON                          0b00001111  /* Display on      */
#define DOFF                         0b00001011  /* Display off     */
#define CURSOR_ON                    0b00001111  /* Cursor on       */
#define CURSOR_OFF                   0b00001101  /* Cursor off      */
#define BLINK_ON                     0b00001111  /* Cursor Blink    */
#define BLINK_OFF                    0b00001110  /* Cursor No Blink */
#define FOUR_BIT                     0b00101111  /* 4-bit Interface  */
#define EIGHT_BIT                    0b00111111  /* 8-bit Interface */
#define LINE_5X7                     0b00110011  /* 5x7 characters, single line   */
#define LINE_5X10                    0b00110111  /* 5x10 characters */
```

```
#define LINES_5X7                       0b00111011  /* 5x7 characters, multiple line */
unsigned char lcdMesgul( void );
void lcdKomutGonder( unsigned char cmd );
void lcdVeriGonder( unsigned char data );
void lcdAc( unsigned char lcdTipi );
void lcdTemizle( void );
void putcLCD( const char c );
void putrsLCD( const rom char* buffer );
void putsLCD( const char* buffer );
void lcdKursoruAyarla( unsigned char satir, unsigned char sutun );
#endif // __LCD_H
**********KUMANDA BİRİMİNE AİT BAŞLIK DOSYALARI**********
**********degiskenler.h**********
#ifndef __DEGISKENLER_H
#define __DEGISKENLER_H
#include "hkb.h"
extern unsigned char            skor[ HEDEF_SAYISI ];
extern char                     mesaj[20];
extern uint8                    genelZamanlayici;
extern uint8                    gecenSure;
extern uint16                   iletisimZamanAsimiSuresi;
extern uint8                    gonderilen;
extern uint8                    gonderilenveri;
#endif // __DEGISKENLER_H
**********hkb.h**********
#ifndef __HKB_H
#define __HKB_H
#include <kbd.h>
#include <lcd.h>
#include <phs.h>
#define HKB_TMR0_PERIYODU       0xF100    // Fosc = 4.915 Mhz -> 50mS
#define GIRIS_TUSU                      0x3E
#define IPTAL_TUSU                      0x3F
#define YUKARI_TUSU                     0x3C
#define ASAGI_TUSU                      0x3D
#define ILERI_TUSU                      0x3B
#define GERI_TUSU                       0x3A
#define CD_PINI                 PORTBbits.RB5
#define VERICIYI_AC_PINI        PORTBbits.RB6
#define ALICIYI_AC_PINI         PORTBbits.RB7
```

```
#define ENGELLE_PINI                    PORTCbits.RC0
#define aliciyiAc()                     ALICIYI_AC_PINI=0
#define aliciyiKapa()                   ALICIYI_AC_PINI=1
#define aliciCikisiniEtkinlestir()      ENGELLE_PINI=1
#define aliciCikisiniDevreDisiBirak()   ENGELLE_PINI=0
#define vericiyiAc()                    VERICIYI_AC_PINI=0
#define vericiyiKapa()                  VERICIYI_AC_PINI=1
#define HEDEF_SAYISI      8
#define HKB_TASIYICI_DINLEME_ZAMAN_ASIMI      30
#define HKB_DINLEME_DURUMU_ZAMAN_ASIMI      30
typedef struct {
unsigned char cihaz;
unsigned char program;
}Ayarlanacaklar;
#endif
**************************protokol.h*********************************
#ifndef __PROTOKOL_H
#define __PROTOKOL_H
// protokol
#include <usart.h>
#include "hkb.h"
void gondermeyiBaslat( void );
void gondermeyiBitir( void );
void programiIcraEt( void );
void programDurdur( void );
void sonuclariIste( uint8 safha );
void programYukle( Ayarlanacaklar* ayarla );
#endif // __PROTOKOL_H
********KUMANDA BİRİMİNE AİT FONKSİYON DOSYALARI*****************************
**************************ana.c ***************************************************
/*            Program başlangıç kısmı
*/
/*#include <portb.h>                                        // PORTB library function*/
#include <usart.h>
#include <delays.h>
#include <string.h>
#include <timers.h>
#include <config.h>
#include "hkb.h"
#include <durum.h>
```

```
#include "degiskenler.h"
void ISR( void );
// Locate ISR handler code at interrupt vector
#pragma code isrcode=0x0008
void isrcode(void) { // Bu fonksiyon kesmeyi ISR fonksiyonuna yönlendirir
  _asm
  goto ISR
  _endasm
}
#pragma code
#pragma interrupt ISR
void ISR( void ) {
  if( INTCONbits.TMR0IF ) {
    if( genelZamanlayici >= 20 ) {
      gecenSure++;
      gonderilen++;
      genelZamanlayici = 0;
    }
    genelZamanlayici++;
    iletisimZamanAsimiSuresi++;
    TMR0H = 0xF1;
    TMR0L = 0x00;   // 50 ms için periyod için TMR0 yazmacına 16-Bitlik 0xF100 değeri atanır
    INTCONbits.TMR0IF = 0;
  }
}
#pragma code
void main( void ) {
  PORTB = 0x00;
  aliciyiKapa();
  vericiyiKapa();
  TRISB = 0;
  PORTC = 0x00;
  TRISC = 0x80;
  KlavyeyiAc();
  lcdAc( LINES_5X7 );
  lcdTemizle();
//          --------------------
  putrsLCD("    YILMAZ DURNA    \n");
  putrsLCD(" MUHENDIS USTEGMEN \n");
  putrsLCD(" NIGDE UNIVERSITESI\n");
```

```
  putrsLCD("       2007        ");
  Delay10KTCYx(150);
  TRISB |= 0x3F;    // sadece RB6 ve RB7 çıkış olacaktır.
  OpenUSART( USART_RX_INT_OFF &
            USART_TX_INT_OFF &
            USART_ASYNCH_MODE &
            USART_EIGHT_BIT &
            USART_CONT_RX &
            USART_BRGH_LOW,
            BAUD_DEGERI );
  OpenTimer0( TIMER_INT_ON &
             T0_16BIT &
             T0_SOURCE_INT & // Bu yapılmalı, aksi halde PIC'e dışarıdan osilatör bağlanılması gerekir.
             T0_PS_1_16 );

  genelZamanlayici = 0;
  gecenSure = 0;
  ZAMANLAYICIYI_DURDUR();
// interruptları etkinleştir
  PIR2bits.EEIF = 0;
  INTCONbits.PEIE = 1;                // Enable peripheral interrupts
  INTCONbits.GIE = 1;
  anaDongu();
}
**************************degiskenler.c*****************************************************
#include "hkb.h"
#pragma udata
uint8               genelZamanlayici;
uint8               gecenSure;
uint16              iletisimZamanAsimiSuresi;
char                mesaj[20];
uint8               skor[ ]={0,0,0,0,0,0,0,0,0};
uint8               gonderilen;
uint8               gonderilenveri;
**************************durum.c***********************************
//--------------------------------- Include dosyaları ----------------------
#include<stdlib.h>
#include<string.h>
#include<delays.h>
```

```
#include "global.h"
#include "durum.h"
#include "hkb.h"
#include "protokol.h"
#include "degiskenler.h"
#include "eeprom.h"
#include "lcd.h"

//------------------------------------------------------- Macrolar ---
#define YENI_DURUM(D) durum=(D)
#define HEDEF_SORGULAMA_SURESI      10   // 10S de bir plc bilgisi ister

//---------------------------------------- Fonksiyon Prototipleri ---
void anaMenu( void );
void Kontrol( void );
void CihazlarAyniDurumu( void );
void CihazlarFarkliDurumu( void );
void SayfaGosterDurumu( void );
void ProgramSecmeDurumu1( void );
void ProgramSecmeDurumu2( void );
void ProgramSec(DurumFonk       sonraki,
                DurumFonk       kaydir,
                DurumFonk       iptal );
char sayiGir( void );
void BilgileriGonderDurumu( void ) ;
void SonuclariGosterDurumu( void );
void iletisimiBaslat(void);
void iletisimKontrol( void );
void tasiyiciDinlemeDurumu0( void );
void tasiyiciDinlemeDurumu1( void );
void dinlemeDurumu( void );
void SonucYazBilgisi0( void );
void SonucYazBilgisi1( void );
void SonucYazBilgisi2( void );
void SonucYazBilgisi3( void );
void SonucYaz( void );
void USARTArizaGider( void );
void lcdKursoruGerilet(void);
void lcdDDRamAdresiniAyarla( unsigned char );
unsigned char lcdDDRamAdresiniHesapla( void );
```

```
/*//-------------------------------------------------------------------
  Değişkenler
*///------------------------------------------------ ram değişkenleri ---

#pragma udata

static DurumFonk    durum,
                    mesajIslemeDurum,
                    zamanAsimiDurum,
                    barGostergeDurum;

DurumFonk  programsecmedurumu[] = {ProgramSecmeDurumu1,
                            ProgramSecmeDurumu2};

uint8               c, crc, hedefNo, sayfa_satiri,
                    ayarlanancihazsayisi, prgm, kntrl;
uint16              sayi;
Ayarlanacaklar ayarlanacak[8];

//------------------------------------------------- rom değişkenleri ---

#pragma romdata

rom const char *rom secenekler[] = {"1.Flash",   "2.Karasimsek",
                                    "3.Tirtil", "4.Zigzag",
                                    "5.Sabit",  "6.Artan",
                                    "7.Azalan", "8.Durdur"};


//------------------------------------------------------ fonksiyonlar ---

#pragma code

//-------------------------------------------------------------------
void anaDongu(void){
  durum = anaMenu;

  while(1){
    if(durum != NULL)
      durum();
```

```
  }
 }

//---------------------------------------------------------------------
 void anaMenu( void ) {
  lcdTemizle();

  putrsLCD("\n1. MANUAL KONTROL\n");    // Sorgu esnasında kontrol birimi gönderim yapmaz.
  putrsLCD(  "2. OTAMATiK KONTROL\n");  // Sorgu esnasında kontrol birimi gönderim yapar.

  while( 1 ) {
   c = getch();
   if( c == '1' ) {
     kntrl = 0;           // Fonksiyonlar ortak kullanıldığından,
                          // Manual kontrol seçildiğini belirtir
     YENI_DURUM( Kontrol ); return;
   } else if( c == '2') {
     kntrl = 1;                 // Fonksiyonlar ortak kullanıldığından,
                                // Otamatik kontrol seçildiğini belirtir
     YENI_DURUM( Kontrol ); return;
   }
  }
 }
//---------------------------------------------------------------------
 void Kontrol( void ) {
  lcdTemizle();

  putrsLCD("1. AYNI PROGRAM\n");       // Tüm hedefler aynı programı çalıştırır. Ortak ID gönderilir.
  putrsLCD("2. FARKLI PROGRAM\n");   // Her bir hedef kendi ID'si ile ayrı ayrı programlanır.
  putrsLCD("3. SONUCLARI GOSTER");  // Hedeflerin PLC dğerlerini 10 sn de bir gösterir.

  ayarlanancihazsayisi = 0;
  while( 1 ) {
   c = getch();
   if( c == '1' ) {
     YENI_DURUM( CihazlarAyniDurumu ); return;
   } else if( c == '2') {
     YENI_DURUM( CihazlarFarkliDurumu ); return;
   } else if( c == '3') {
     YENI_DURUM( SonuclariGosterDurumu ); return;
```

```
   } else if( c == IPTAL_TUSU) {
     YENI_DURUM( anaMenu ); return;
   }
  }
}
//---------------------------------------------------------------------
void CihazlarAyniDurumu(void){
  prgm=0;
          // Fonksiyonlar ortak kullanıldığından,
          // CihazlarAyniDurumu'ndan gelindiğini belirtir.
  lcdTemizle();
  putrsLCD("PLC'LER ICIN\n1..8 ARASI\nBIR PROGRAM SECINIZ\n ");

  GECIKME_1S();
  sayfa_satiri = 0;                    // Sayfa göster fonksiyonu için satırlar indisi.
  ayarlanacak[ayarlanancihazsayisi].cihaz =  0;        // Hedefler için ortak ID sıfırdır.
  YENI_DURUM( SayfaGosterDurumu );

}
//---------------------------------------------------------------------
void CihazlarFarkliDurumu(void){
  prgm=1;
          // Fonksiyonlar ortak kullanıldığından,
          // CihazlarFarkliDurumu'ndan gelindiğini belirtir.
  lcdTemizle();
  putrsLCD("CALISTIRILACAK\nCIHAZI SECIN\n'ONAY' BASLATIR\ncihaz=");
  while(1){
    switch(sayiGir()){
      case 0:                          // Sayı girilmeden 'GIRIS_TUSU' na basılması durumu,
                                       // böylece 'ayarlanancihazsayisi' kadar hedef girilimiş oluyor.
        if(ayarlanancihazsayisi==0){                // Daha önce hiç hedef seçilmeden 'GIRIS_TUSU' na
basılması durumu.
          YENI_DURUM( CihazlarFarkliDurumu );
          return;
        }
        YENI_DURUM( BilgileriGonderDurumu ); // Hedefler seçilip 'GIRIS_TUSU' na basılması
durumu.
        return;
      case 1:                                // Sayı girilerek 'GIRIS_TUSU' na basılması durumu.
        if( sayi > 8 ) {                     // Geçersiz sayı durumu
```

```
            lcdTemizle( );
            //      --------------------
            putrsLCD("OLMAYAN BIR\n");
            putrsLCD("CIHAZ SECILDI\n");
            GECIKME_2S();
            YENI_DURUM( CihazlarFarkliDurumu ); return;
          }
          else                                          // Geçerli sayı durumu
            ayarlanacak[ayarlanancihazsayisi].cihaz =  sayi;
            goto prg_secim;
        case 2:                                         // 'IPTAL_TUSU' na basılması durumu.
          YENI_DURUM( anaMenu ); return;
      }
    }
prg_secim:
          // Bir hedef seçilmiştir, o hedefin program seçimi için listeyi gösterir
  lcdTemizle();
  putrsLCD("PLC ICIN\n1..8 ARASI\nBIR PROGRAM SECINIZ\n");

  GECIKME_1S();
  sayfa_satiri = 0;
  YENI_DURUM( SayfaGosterDurumu );

}

//---------------------------------------------------------------------
void SayfaGosterDurumu(void){

 lcdTemizle();

putrsLCD(secenekler[sayfa_satiri]);  putcLCD('\n');  // Seçenekler listesinin 'sayfa_satiri' ncı elemanı
 putrsLCD(secenekler[sayfa_satiri+1]);putcLCD('\n'); // Seçenekler listesinin 'sayfa_satiri+1' nci elemanı

  putrsLCD(secenekler[sayfa_satiri+2]);putcLCD('\n'); // Seçenekler listesinin 'sayfa_satiri+2' nci elemanı

  putrsLCD(secenekler[sayfa_satiri+3]);putcLCD(' prgm='); // Seçenekler listesinin 'sayfa_satiri+3' ncü
elemanı

  YENI_DURUM( programsecmedurumu[prgm] );    // Cihazlar aynı/farklı durumuna göre dallanır
}
```

```
//---------------------------------------------------------------------
void ProgramSecmeDurumu1(void){
            // Cihazlar aynı durumunda dallanılan fonksiyon
  ProgramSec(BilgileriGonderDurumu,
            SayfaGosterDurumu,
            Kontrol);
 }

//---------------------------------------------------------------------
void ProgramSecmeDurumu2(void){
            // Cihazlar farklı durumunda dallanılan fonksiyon
  ProgramSec(CihazlarFarkliDurumu,
            SayfaGosterDurumu,
            Kontrol);
 }

//---------------------------------------------------------------------
void ProgramSec(DurumFonk       sonraki,
              DurumFonk       kaydir,
              DurumFonk       iptal ) {

  while(1) {

    switch(sayiGir()){
      case 1:
          // Sayı girilerek 'GIRIS_TUSU' na basılması durumu.
        if( sayi > 8 ) {                                                    //
Geçersiz sayı durumu
            lcdTemizle( );
            //      --------------------
            putrsLCD("OLMAYAN BIR\n");
            putrsLCD("PRG. SECILDI\n");
            GECIKME_2S();
            YENI_DURUM( SayfaGosterDurumu ); return;
          }
          else{
            // Geçerli sayı durumu
            ayarlanacak[ayarlanancihazsayisi].program =  sayi;
            ayarlanancihazsayisi++;
```

```
            YENI_DURUM( sonraki ); return;
          }
        case 2:
            // 'IPTAL_TUSU' na basılması durumu.
          YENI_DURUM( iptal ); return;
        case 3:
            // 'YUKARI_TUSU' na basılması durumu.
          sayfa_satiri--;
          YENI_DURUM( kaydir ); return;
        case 4:
            // 'ASAGI_TUSU' na basılması durumu.
          sayfa_satiri++;
          YENI_DURUM( kaydir ); return;

      }
    }
}

//---------------------------------------------------------------------
char sayiGir( void ) {

  sayi = 0;
  lcdKomutGonder( DON  & CURSOR_ON  & BLINK_ON );

  while( 1 ) {
     c = getch();
     if( c == GIRIS_TUSU ) {
       lcdKomutGonder( DON & CURSOR_OFF & BLINK_OFF );
       if(sayi == 0) {return 0;}                  // Sayı girilmeden 'GIRIS_TUSU' na basıldı
       else {return 1;}                           // Sayı girilip 'GIRIS_TUSU' na basıldı
     }
     else if( c == IPTAL_TUSU ) {
       lcdKomutGonder( DON & CURSOR_OFF & BLINK_OFF );
       return 2;
     }
     else if( c == GERI_TUSU ) {
       if(sayi > 0 ){                                                                    //
Herhangi bir sayı girilmişken 'GERI_TUSU' na basıldı
       putcLCD('\b');
       sayi /= 10;
```

```
        }
      }
      else if( c >= '0' && c <='9' ) {
        putcLCD( c );
        sayi *= 10;
        sayi += c -'0';
      }
      else if( c == YUKARI_TUSU ){
        if (sayfa_satiri > 0 ){              // Ekranda en üstte ilk eleman yok, kaydırmak için dön.
        return 3;
      }}
      else if( c == ASAGI_TUSU ){
       if (sayfa_satiri < 4 ){   // Ekranda en altta sonuncu eleman yok, kaydırmak için dön.
        return 4;
     }}
   }

}

//-------------------------------------------------------------------------
void BilgileriGonderDurumu( void ) {
  int i;
  lcdTemizle();
  putrsLCD("BILGILER\n");
  putrsLCD("GONDERILIYOR\n");
  GECIKME_200mS();
  lcdKursoruAyarla( 2, 3 );
  putrsLCD("Tx");
  GECIKME_200mS();
  lcdKursoruAyarla( 2, 15 );
  putrsLCD("Rx");
  lcdKursoruAyarla( 2, 6 );

  gondermeyiBaslat();
  for( i=0; i<ayarlanancihazsayisi; ++i ) {                    // 'ayarlanancihazsayisi' kadar
    programYukle(&ayarlanacak[ayarlanancihazsayisi-1]);        // girilen bilgiler gönderilir.
  }

  for(i=0; i<8; ++i){
                                                  // Bilgilerin gittiğini belirten
```

```
  putcLCD('>');
          // 8 tane '>' ekrana yazılır.
  GECIKME_100mS();
  }

  programiIcraEt();
                              // Bu komutu alan hedefler çalışmaya başlar.
  gondermeyiBitir();

  putrsLCD("\nGONDERME TAMAMLANDI");
  GECIKME_500mS();

  YENI_DURUM( SonuclariGosterDurumu );
}

//-------------------------------------------------------------------------
void SonuclariGosterDurumu( void ) {

  lcdTemizle();
  lcdKursoruAyarla( 0,0 );  GECIKME_100mS();
  putrsLCD("P1");
  lcdKursoruAyarla( 0,5 );  GECIKME_100mS();
  putrsLCD("P2");
  lcdKursoruAyarla( 0,10 );  GECIKME_100mS();
  putrsLCD("P3");
  lcdKursoruAyarla( 0,15 );  GECIKME_100mS();
  putrsLCD("P4");

  lcdKursoruAyarla( 2,0 );  GECIKME_100mS();
  putrsLCD("P5");
  lcdKursoruAyarla( 2,5 );  GECIKME_100mS();
  putrsLCD("P6");
  lcdKursoruAyarla( 2,10 );  GECIKME_100mS();
  putrsLCD("P7");
  lcdKursoruAyarla( 2,15 );  GECIKME_100mS();
  putrsLCD("P8");

  for(hedefNo=1; hedefNo<=8; hedefNo++){          // 8 hedef için ekrana en son alınmış bilgiler yazılır.
    GECIKME_100mS();
    SonucYaz();
```

```
  }

  gondermeyiBaslat();
  sonuclariIste( 0 );
        // Bilgileri güncellemek için hedeflerden sonuçlar istenir.
  gondermeyiBitir();

  iletisimiBaslat();
        // Alıcı çıkışı dinlemeyi aktifleştirir.

  while( 1 ) {
  if( gecenSure >= HEDEF_SORGULAMA_SURESI ) {          // 10 sn de bir sonuçlar istenir.
                gondermeyiBaslat();
                sonuclariIste( 0 );
          gondermeyiBitir();
          gecenSure = 0;
    }

    if( tusBasili() ) {
     c = klavyeKarakterDegeriniOku(); /// !!!!!
     if( c == IPTAL_TUSU ) {                                                    //  İptale
basılınca dinleme kesilir, kontrol ekranına dönülür.
                mesajIslemeDurum = NULL;
                zamanAsimiDurum = NULL;
       aliciyiKapa();
                YENI_DURUM( Kontrol ); return;
     }
    }
    iletisimKontrol();
        // Alıcı dinlenir, gelen data varsa alınır, zamanlama kontrol edilir.
  }
}

//---------------------------------------------------------------------------
void iletisimiBaslat(void) {
  aliciCikisiniDevreDisiBirak();                    // Engelle pini
  zamanAsimiDurum  = NULL;// Zamanlama kontrolünü yapan fonksiyon işaretçisi
  mesajIslemeDurum = tasiyiciDinlemeDurumu0;     // Alıcıyı dinleyip data alan fonksiyon işaretçisi
  gecenSure = 0;                                // 1 sn periyodunda artan değişken
  iletisimZamanAsimiSuresi = 0;                 // 50 ms periyodunda artan değişken
```

```
  ZAMANLAYICIYI_BASLAT();              // Timer çalıştırılır.
}

//-------------------------------------------------------------------------
void iletisimKontrol( void ) {

  if( mesajIslemeDurum != NULL ) {
    mesajIslemeDurum();
  }

  if( zamanAsimiDurum != NULL ) {
    zamanAsimiDurum();
  }

}

//-------------------------------------------------------------------------
void dinlemeyiKes( void ) {
  aliciCikisiniDevreDisiBirak();
  zamanAsimiDurum  = NULL;
  mesajIslemeDurum = tasiyiciDinlemeDurumu0;
}

//-------------------------------------------------------------------------
void tasiyiciDinlemeZamanAsimiDurumu( void ) {              // 1.5 sn de dinleme kesilir
  if( iletisimZamanAsimiSuresi >= HKB_TASIYICI_DINLEME_ZAMAN_ASIMI ) {
    dinlemeyiKes();
  }
}

//-------------------------------------------------------------------------
void dinlemeZamanAsimiDurumu( void ) {                      // 1.5 sn de dinleme kesilir
  if( iletisimZamanAsimiSuresi >= HKB_DINLEME_DURUMU_ZAMAN_ASIMI ) {
    dinlemeyiKes();
  }
}

//-------------------------------------------------------------------------
void tasiyiciDinlemeDurumu0( void ) {
  if( !CD_PINI ) {
```

```
    mesajIslemeDurum =  tasiyiciDinlemeDurumu1;
    iletisimZamanAsimiSuresi = 0;
    zamanAsimiDurum =  NULL;
   }
}

//-----------------------------------------------------------------------------
void tasiyiciDinlemeDurumu1( void ) {
  if( CD_PINI ) {
    mesajIslemeDurum =  tasiyiciDinlemeDurumu0;
    iletisimZamanAsimiSuresi = 0;
  }else if( iletisimZamanAsimiSuresi >= TASIYICI_DINLEME_ZAMAN_ASIMI_SURESI ) {
    getcUSART();      // Boş okuma ile tampon bellek silinir.
    aliciCikisiniEtkinlestir();
    crc = 0;
    mesajIslemeDurum = dinlemeDurumu;
    iletisimZamanAsimiSuresi = 0;
    zamanAsimiDurum =  tasiyiciDinlemeZamanAsimiDurumu;
  }
}

//-----------------------------------------------------------------------------
void dinlemeDurumu( void ) {                // Kontrol byte'ı ( = 0x48 )
  if ((RCSTAbits.OERR)) {                   // USART overrun oldu ise arıza giderilir
      USARTArizaGider();
      return;
  }

  if( DataRdyUSART() ) {
    c = getcUSART();
    crc = c;
    c >>= 4;
    if( c == 0x04 ) {
      mesajIslemeDurum = SonucYazBilgisi0;
      iletisimZamanAsimiSuresi = 0;
      zamanAsimiDurum = dinlemeZamanAsimiDurumu;
    } else {
      dinlemeyiKes();
    }
  }
```

```
}

//-------------------------------------------------------------------------
void SonucYazBilgisi0( void ) {                 //Bilgi gönderen hedefin ID numarası
 if ((RCSTAbits.OERR)) {
    USARTArizaGider();
    return;
 }

 if( DataRdyUSART() ) {
  c = getcUSART();
  if(c == 0){                           //Herhangi bir hedefin ID si 0 olamaz
            dinlemeyiKes();//Eger 0 gelirse bu dikkate alınmaz
   return;
  }
  hedefNo = c;
  mesajIslemeDurum = SonucYazBilgisi1;
  crc += c;
 }
}

//-------------------------------------------------------------------------
void SonucYazBilgisi1( void ) {     //Hedefin plc'den okuduğu değer
 if ((RCSTAbits.OERR)) {
    USARTArizaGider();
    return;
 }
 if( DataRdyUSART() ) {
  c = getcUSART();
  skor[ hedefNo ] = c;
  mesajIslemeDurum = SonucYazBilgisi2;
  crc += c;
 }
}

//-------------------------------------------------------------------------
void SonucYazBilgisi2( void ) {                 //Hedefin şu an çalıştırdığı program
 if ((RCSTAbits.OERR)) {
    USARTArizaGider();
    return;
```

```
  }
  if( DataRdyUSART() ) {
   c = getcUSART();
   if(kntrl){                    //Otomatik Kontrol ise hedefe çalıştıracağı program kodu gönderilir.
    if ((skor[ hedefNo ] != c ) && (skor[ hedefNo ] != 0 ) && skor[ hedefNo ] <= HEDEF_SAYISI ){
                        // plc den okunan değer şu an hedefin çalıştırdığı program değilse
                         // ve plc den okunan değer sıfırdan farklı ise
                         // ve plc den okunan değer bir programa karşılık geliyorsa
      ayarlanacak[0].cihaz = hedefNo;
      ayarlanacak[0].program = skor[ hedefNo ];
      gondermeyiBaslat();
      programYukle(&ayarlanacak[0]);
      programiIcraEt();
      gondermeyiBitir();
     }
   }
   mesajIslemeDurum = SonucYazBilgisi3;
   crc += c;
  }
}

//---------------------------------------------------------------------
void SonucYazBilgisi3( void )
  if ((RCSTAbits.OERR)) {
     USARTArizaGider();
     return;
  }
  if( DataRdyUSART() ) {
   c = getcUSART();
   dinlemeyiKes();
   SonucYaz();    // Alınan bilgi ekranda kendisine karşılık gelen yerde yazılır.
  }
}

//-----------------------------------------------------------------------------
void SonucYaz( void ) {

  if( hedefNo <= 4 ) {        // Bilgisi yazılacak hedefin ekrandaki koordinatları ayarlanır.
   lcdKursoruAyarla( 1, (hedefNo-1) * 5 );
  } else {
```

```
   lcdKursoruAyarla( 3, (hedefNo - 5) * 5 );
  }

  putrsLCD("   ");            // Bir önce 3 basamaklı bir sayı gelmiş olabilir diye, 3 kutucuk temizlenir.
  lcdKursoruGerilet();                    // Kursör geri alınıp yeri ayarlanır.
  lcdKursoruGerilet();
  lcdKursoruGerilet();
  while( lcdMesgul() );
  lcdDDRamAdresiniAyarla( lcdDDRamAdresiniHesapla() );

  itoa( skor[ hedefNo ], mesaj );          // Tek byte olan sayı ekranda gözükebilecek karakter dizisine
çevrilir(123->'1' '2' '3')

  putsLCD( mesaj );
}

//---------------------------------------------------------------------
void USARTArizaGider( void ) {
   getcUSART();                    // 2 byte olan usart fifo'su okunarak silinir.
   getcUSART();
   dinlemeyiKes();
         while (1) {
               TXSTAbits.TXEN = 0;    // USART vericisi kapatılıp açılır.
         TXSTAbits.TXEN = 1;
         RCSTAbits.CREN = 0;          // USART alıcısı kapatılıp açılır.
         RCSTAbits.CREN = 1;
               RCSTAbits.SPEN = 0;    // USART seri iletişimi kapatılıp açılır.
               RCSTAbits.SPEN = 1;
               if( RCSTAbits.OERR ){  // Overrun temizleme başarısız ise yenile.
                     continue;
               } else {               // Overrun temizleme başarılı ise dön.
                     break;
               }
         }
}

**********kbd.c**********
#include <kbd.h>
#include <delays.h>
#include <global.h>
```

```
/*

  Hex klavye devresine göre çalışır

  5 Bit kullanılır.

  RB[0..3] 'ten basılan tuş değeri okunur.

  RB[4] ile herhangi bir tuşa basılıp basılmadığı
  kontrol edilir.

*/

unsigned char tusBasili( void ) {
  if( PORTBbits.RB4 ) {
    Delay10KTCYx( 2 ); // Biraz bekle ( Yaklaşık 20 mS )
    if( PORTBbits.RB4 ) {
      return 1; // Tuş basılı
    }
  }
  return 0; // Tuş basılı değil
}

// Sadece klavye portundaki değeri okur ASCII değerine dönüştürür

char klavyeKarakterDegeriniOku( void ) {
  return ( PORTB & 0xF ) + 0x30; // İlk dört biti maskele ve ASCII'ye dönüştürmek için HEX 30 ekle
}

// getc tuşa basılmasını bekler. Dönüş değeri olarak

char getc( void ) {
  while( tusBasili() );     // Tuşun bırakılmasını bekle
  while( !tusBasili() );  // Tuşun basılmasını bekle
  return ( PORTB & 0xF ) + 0x30; // İlk dört biti maskele ve ASCII'ye dönüştürmek için HEX 30 ekle
}

void KlavyeyiAc( void ) {
  TRISB |= 0x1F; // 5 Bit giriş olarak
```

```
}
**********************************lcd.c*********************************************
#include <lcd.h>
#include <delays.h>

//#include "degiskenler.h"

/*
        LCD fonksiyonları
*/

unsigned char lcdDDRamAdresiniHesapla( void );
void lcdDDRamAdresiniAyarla( unsigned char DDaddr );
void lcdKursoruIlerlet( void );

#define DelayLCDPOR()   Delay10KTCYx(2)   // 20ms Gecikme
#define DelayFor20TCY() Delay10TCYx(2)
#define DelayLCD()      Delay1KTCYx(5)          // 5 ms
#define ClockCmd()      E_PIN=1;DelayFor20TCY();E_PIN=0

unsigned char __satir, __sutun;

// lcd8BitModunaAyarla 8 Bit Moduna ayarlar
void lcd8BitModunaAyarla( void ) {
  DATA_PORT = 0x30; // Function set cmd(8-bit interface)
  ClockCmd();
  DelayLCD();
}

void lcdAc( unsigned char lcdTipi ) {

                        // E portuna ait ayarlamalar
  PORTE   = 0x0;
  ADCON1  = 0x0A;   // Bütün E portunu Dijital I/O olarak ayarla
  TRISE   = 0x0;    // Tüm Kontrol pinleri çıkış olarak ayarla
                        // PORTD Standart I/O olarak kullanılsın

  DATA_PORT = 0;
  TRIS_DATA_PORT = 0xFF;
```

```
  RW_PIN = 0;
  RS_PIN = 0;
  E_PIN = 0;
  DelayLCDPOR();  // LCD 'nin resetlenmesini bekle
  TRIS_DATA_PORT = 0;     // Veri portunu çıkış yap
  lcd8BitModunaAyarla();
  lcd8BitModunaAyarla();
  lcd8BitModunaAyarla();
  TRIS_DATA_PORT = 0xFF;     // Veri portunu giriş yap
  // Arabirim genişliğini ve satır saysını, fontu  ayarla
  while( lcdMesgul() );
  lcdKomutGonder( 0x38 );
  // LCD yi Kapa sonra tekrar Ac
  while( lcdMesgul() );
  lcdKomutGonder( DOFF & CURSOR_OFF & BLINK_OFF );
  while( lcdMesgul() );
  lcdKomutGonder( DON  & CURSOR_OFF  & BLINK_OFF );
  // Kursör sağa kayacak
  while( lcdMesgul() );
  lcdKomutGonder( SHIFT_CUR_RIGHT );
  lcdTemizle();
  // DD Ram'i 0 'a ayarla
  while( lcdMesgul() );
  lcdKursoruAyarla( 0, 0 );
  return;
}

void lcdKomutGonder( unsigned char cmd ) {
  TRIS_DATA_PORT = 0;             // Data port output
  DATA_PORT = cmd;               // Write command to data port
  RW_PIN = 0;                    // Set the control signals
  RS_PIN = 0;                   // for sending a command
  DelayFor20TCY();
  ClockCmd();
  DelayFor20TCY();
  TRIS_DATA_PORT = 0xff;          // Data port input
}
void lcdVeriGonder( unsigned char data ) {
  TRIS_DATA_PORT = 0;             // Make port output
  DATA_PORT = data;               // Write data to port
```

```
  RS_PIN = 1;                   // Set control bits
  RW_PIN = 0;
  DelayFor20TCY();
  ClockCmd();
  RS_PIN = 0;                   // Reset control bits
  TRIS_DATA_PORT = 0xff;        // Make port input
}

unsigned char lcdMesgul( void ) {
  RW_PIN = 1;                    // Set the control bits for read
  RS_PIN = 0;
  DelayFor20TCY();

  E_PIN = 1;
  DelayFor20TCY();
  if( DATA_PORT & 0x80 ) {                          // Meşgul bitine bak
    E_PIN = 0;                                       // Reset clock line
    RW_PIN = 0;                                     // Reset control line
    return 1;                                        // Return TRUE
  } else {
    E_PIN = 0;
    RW_PIN = 0;
    return 0;
  }
}

void lcdKursoruIlerlet( void ) { // LCD'ye göre değişir
  __sutun++;
  if( __sutun >= 20 ) {
    __sutun = 0;
    __satir++;
    if( __satir >= 4 ) {
      __satir = 0;
    }
    while( lcdMesgul() );
    lcdDDRamAdresiniAyarla( lcdDDRamAdresiniHesapla() );
  }
}

void lcdKursoruGerilet( void ) { // LCD'ye göre değişir
```

```
  if( __sutun > 0 ) {
    __sutun --;
  } else {
    __sutun = 19;
    if( __satir > 0 ) {
      __satir --;
    }
    while( lcdMesgul() );
    lcdDDRamAdresiniAyarla( lcdDDRamAdresiniHesapla() );
  }
}

void putcLCD( const char c ) { // LCD'ye göre değişir
  if( c == '\n' ) {
    __sutun = 0;
    __satir++;
    if( __satir >= 4 ) {
      __satir = 0;
    }
    while( lcdMesgul() );
    lcdDDRamAdresiniAyarla( lcdDDRamAdresiniHesapla() );
  } else if( c == '\b' ) {
    unsigned char addr;
    if( __sutun > 0 ) {
      __sutun--;
    } else {
      __sutun=19;
      if( __satir > 0 ) {
        __satir--;
      }
    }
    addr = lcdDDRamAdresiniHesapla();
    lcdDDRamAdresiniAyarla( addr );
    lcdVeriGonder(' ');
    lcdDDRamAdresiniAyarla( addr );
  } else {
    while( lcdMesgul() );     // Wait while LCD is busy
    lcdVeriGonder( c ); // Write character to LCD
    lcdKursoruIlerlet();
  }
```

```
}

void putrsLCD( const rom char* buffer ) {
  while(*buffer) {          // Write data to LCD up to null
    putcLCD( *buffer );
    buffer++;               // Increment buffer
  }
}

void putsLCD( const char* buffer ) {
  while(*buffer) {          // Write data to LCD up to null
    putcLCD( *buffer );
    buffer++;               // Increment buffer
  }
}

unsigned char lcdDDRamAdresiniHesapla( void ) { // LCD'ye göre değişir
  unsigned char __ddram = 0;

  if( __satir & 0x1 ) { // Tek numaralı satırlar için
    __ddram = 0x40;
  }

  if( __satir >= 2 ) { // satir 2 'den büyükse ddram için gerekli ayarlamayı yap
    __ddram += 20;
  }

  __ddram += __sutun;

  return __ddram;
}

void lcdKursoruAyarla( unsigned char satir,
                       unsigned char sutun ) {

  if( sutun < 20 ) {
    __sutun = sutun;
  } else {
    __sutun = 0;
  }
```

```
  if( satir < 4 ) {
    __satir = satir;
  } else {
    __satir = 0;
  }

  while( lcdMesgul() );
  lcdDDRamAdresiniAyarla( lcdDDRamAdresiniHesapla() );

}

void lcdTemizle( void ) {
  // LCD 'yi temizle
  while( lcdMesgul() );
  lcdKomutGonder(0x01);
  lcdKursoruAyarla( 0, 0 );
}

void lcdDDRamAdresiniAyarla( unsigned char DDaddr ) {

  TRIS_DATA_PORT = 0;                   // Make port output
  DATA_PORT = DDaddr | 0b10000000;      // Write cmd and address to port
  RW_PIN = 0;                           // Set the control bits
  RS_PIN = 0;
  DelayFor20TCY();
  ClockCmd();
  DelayFor20TCY();
  TRIS_DATA_PORT = 0xff;                // Make port input
}
```

**********protocol.c**********

```
// protokol

#include "protokol.h"

#include "hkb.h"

uint8     hataKontrolToplami;
void gonder( uint8 c ) {
  while( BusyUSART() );
```

```
  putcUSART( c );
  GECIKME_2mS();
  hataKontrolToplami += c;
}
void gondermeyiBaslat( void ) {
  hataKontrolToplami = 0;
  aliciyiKapa();
  vericiyiAc();
  GONDERME_ONCESI_BEKLE();
}
void gondermeyiBitir( void ) {
  gonder( hataKontrolToplami );
  while( BusyUSART() );
  GONDERME_SONRASI_BEKLE();
  vericiyiKapa();
  aliciyiAc();
}
void programiIcraEt( void ) {
  gonder( 0x28 );
}
void programDurdur( void ) {
  gonder( 0x38 );
}
void sonuclariIste( uint8 safha ) {
  gonder( 0x58 );
  gonder( safha );
}
void programYukle (Ayarlanacaklar* ayarla){

  gonder( 0x68 );
  gonder( ayarla->cihaz );
  gonder ( ayarla->program );

}
**********************************PLC KARTINA AİT YAZILIM************************
**********************************BAŞLIK DOSYALARI ******************************
********************cevre_birim.h*************************************
#ifndef __CEVRE_BIRIM_H
#define __CEVRE_BIRIM_H
#define VERICIYI_AC_PINI      PORTEbits.RE0
```

```
#define ALICIYI_AC_PINI         PORTEbits.RE1
#define ENGELLE_PINI            PORTEbits.RE2
#define aliciyiAc()             ALICIYI_AC_PINI=0
#define aliciyiKapa()           ALICIYI_AC_PINI=1
#define vericiyiAc()            VERICIYI_AC_PINI=0;
#define vericiyiKapa()          VERICIYI_AC_PINI=1;
#define aliciCikisiniEtkinlestir()          ENGELLE_PINI=1
#define aliciCikisiniDevreDisiBirak()       ENGELLE_PINI=0
#define IKAZ_LEDI                           PORTCbits.RC3
#define CD_PINI                                     PORTCbits.RC4
#define HEDEF_ID        ((PORTA>>1)&0x1F | PORTC<<5)
                        // MSB:C2C1C0A5A4A3A2A1:LSB
#define PLC PORTB
#define ROLELER PORTD
void cevreBirimleriAc( void );
#endif // __CEVRE_BIRIM_H
**********************************************config.h*********************************
#ifndef __CONFIG_H
#define __CONFIG_H
#pragma config LVP = OFF
#pragma config WDT = OFF
#pragma config OSC = XT

#endif
**********************************************degiskenler.h**************************
#ifndef __DEGISKENLER_H
#define __DEGISKENLER_H
#include "global.h"
extern ProgramBilgisi   programs;
extern uint8        gecenSure;
extern uint16       gonderimZamanlayicisi;
extern uint8        iletisimZamanAsimiZamanlayicisi;
extern uint8        genelZamanlayici;
extern uint8        gondermesirasi;
extern uint8        programzamanlayicisi;
extern uint8        yeniprogram;
#endif
**********************************global.h*****************************************
#ifndef __GLOBAL_H
#define __GLOBAL_H
```

```
#include <p18f452.h>
#include <stdlib.h>
#include <timers.h>
#include <delays.h>
#include <usart.h>
#define GECIKME_2mS()       Delay1KTCYx(3)
#define GECIKME_5mS()       Delay1KTCYx(8)
#define GECIKME_30mS()     Delay1KTCYx(37)
#define GECIKME_50mS()     Delay1KTCYx(61)
#define GECIKME_100mS()    Delay1KTCYx(122)
#define GECIKME_150mS()    Delay1KTCYx(183)
#define GECIKME_200mS()    Delay1KTCYx(244)
#define GECIKME_250mS()    Delay10KTCYx(24)
#define GECIKME_300mS()    Delay10KTCYx(36)
#define GECIKME_400mS()    Delay10KTCYx(49)
#define GECIKME_450mS()    Delay10KTCYx(55)
#define GECIKME_500mS()    Delay10KTCYx(61)
#define GECIKME_1S()       Delay10KTCYx(122)
#define GECIKME_2S()       Delay10KTCYx(244)
#define GECIKME_5S()       GECIKME_2S();GECIKME_2S();GECIKME_1S()
#define ZAMANLAYICI_ACIK()       (T0CONbits.TMR0ON)
#define  ZAMANLAYICIYI_BASLAT()
WriteTimer0(HKB_TMR0_PERIYODU);T0CONbits.TMR0ON=1
#define ZAMANLAYICIYI_DURDUR()    T0CONbits.TMR0ON=0
#define GONDERME_ONCESI_BEKLE()   GECIKME_500mS()
#define GONDERME_SONRASI_BEKLE()  GECIKME_100mS()
#define DINLEME_ONCESI_BEKLE()    GECIKME_250mS()

#define GONDERME_ONCESI_BEKLEME_SURESI        10   // 500mS
#define GONDERME_SONRASI_BEKLEME_SURESI       2    // 100mS
#define TASIYICI_ZAMAN_ASIMI_SURESI           15   // 750mS
#define TASIYICI_DINLEME_ZAMAN_ASIMI_SURESI   5    // 250mS
#define GELEN_VERI_ZAMAN_ASIMI_SURESI         20   // 1S
#define GONDERME_PERIYODU_SURESI              20   // 1S
#define DINLEME_ONCESI_BEKLEME_SURESI         5    // 250 mS
//#define BAUD_DEGERI           63 // Fosc = 4.9152 -> 1200 baud
#define BAUD_DEGERI           15 // Fosc = 4.9152 -> 4800 baud
#define HEDEF_TMR0_PERIYODU  0xF100 //50 ms
//#define BAUD_DEGERI         127 // Fosc = 4.9152 -> 600 baud
```

```
// Ortak Protokol tanımlamaları
typedef unsigned char byte;
typedef unsigned int  word;
typedef unsigned long dword;

typedef unsigned char uint8;
typedef unsigned int  uint16;
typedef unsigned long uint32;

typedef char int8;
typedef int  int16;
typedef long int32;

typedef void(*DurumFonk)( void );

typedef struct {
 unsigned char    hedefNo;
 unsigned char    prgm;
 unsigned char    siraNo;
}ProgramBilgisi;

#ifndef NULL
#define NULL 0
#endif
#endif  // __GLOBAL_H
**************************iletisim.h**************************************************
#ifndef __ILETISIM_H
#define __ILETISIM_H
extern void  iletisimAc( void );
extern void  iletisimKontrol( void );

#endif
*****************************program.h*************************************************
#ifndef __PROGRAM_H
#define __PROGRAM_H
extern void programiBaslat( char gorev );
extern void programAc( void );
extern void programiIslet(void);
```

```
extern DurumFonk programDurum;
#endif
*****************************PLC KARTINA AİT FONKSİYONLAR**********************
*****************************************adc.c*************************************
#include <adc.h>
#include "adc.h"
#include "global.h"
#include "cevre_birim.h"

#define AKU_SINIR_DEGERI    128
void adcCeviriminiBaslat( void );
void adcDegeriniOku( void );
#pragma udata
DurumFonk     akuDurum;
#pragma code
//_____________________________________________________________________________

void adcCeviriminiBaslat( void ) {

  ConvertADC();
  akuDurum = adcDegeriniOku;
}

//-------------------------------------------------------
void adcDegeriniOku( void ) {

  if( !BusyADC() ) {
    IKAZ_LEDI = ( ((uint8) ADRESH) < AKU_SINIR_DEGERI );
    akuDurum = adcCeviriminiBaslat;
  }
}
//-------------------------------------------------------

void akuSeviyesiniKontrolEt( void ) {
  if( akuDurum != NULL ) {
    akuDurum();
  }
}
//-------------------------------------------------------
```

```
void akuKontrolunuAc( void ) {

  OpenADC( ADC_FOSC_4 &
           ADC_LEFT_JUST &
           ADC_1ANA_0REF,
           ADC_CH0 &
           ADC_INT_OFF );

  akuDurum = adcCeviriminiBaslat;

}
//-----------------------------------------------------
**********************************ana.c*****************************************

#include "global.h"
#include <usart.h>
#include "degiskenler.h"
#include "cevre_birim.h"
#include "iletisim.h"
#include "program.h"
#include "aku.h"
#include "config.h"
void ISR( void );
#pragma code isrcode=0x0008
void isrcode(void) { // Bu fonksiyon kesmeyi ISR fonksiyonuna yönlendirir
  _asm
  goto ISR
  _endasm
}

#pragma code

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

#pragma interrupt ISR //save = PROD//, section( ".tmpdata" )

void ISR( void ) {
  if( INTCONbits.TMR0IF ) {
    genelZamanlayici++;
    if( genelZamanlayici >= 20 ) {  // 1 S
```

```
    gecenSure++;              // Bu iki zamanlayıcı 1 S periyodunda çalışır
    genelZamanlayici = 0;
   }
   gonderimZamanlayicisi++;
   iletisimZamanAsimiZamanlayicisi++;
   programzamanlayicisi++;
   TMR0H = 0xF1;
   TMR0L = 0x00;
   INTCONbits.TMR0IF = 0;
  }
}

#pragma code
//_____________________________________________________________________________
void main( void ) {

//----------------------------
  cevreBirimleriAc();   // Portları giriş çıkış olarak ayarlar.
  programAc();          // Program değişkenlerine ilk değerlerini verir.
  iletisimAc();         // İletişim değişkenlerini ve sistemini ayarlar.
  //GECIKME_2S();
 // deneme = 1 ;
  akuKontrolunuAc();    // Akü için ADC'yi ayarlar.

  genelZamanlayici = 0;
//----------------------------
  // OpenTimer komutu timer ayarlarını yapar
  // ve 0x0000 dan itibaren saymaya başlar
  OpenTimer0( TIMER_INT_ON &
            T0_16BIT &
            T0_SOURCE_INT & // Bu yapılmalı, aksi halde PIC'e dışarıdan osilatör bağlanılması gerekir.
            T0_PS_1_16 );

  TMR0H = 0xF1;
  TMR0L = 0x00;
//----------------------------
// interruptları etkinleştir
  INTCONbits.PEIE = 1;
  INTCONbits.GIE = 1;
```

```
//---------------------------
  for(;;) {
    //akuSeviyesiniKontrolEt();
    programiIslet();
    iletisimKontrol();

  }
 return;
}
//_____________________________________________________________________________
```

**********cevre_birimleri.c**********

```
#include "global.h"
#pragma code
void cevreBirimleriAc( void ) {
  // PORTA, A5-A1 ID'nin ilk 5 biti
  PORTA = 0x0;
  ADCON1 = 0x07;        // Bütün portlar Dijital G/Ç olarak ayarlanır E ve A portu
  TRISA = 0xFF;          // Tüm Pinler Giriş olarak ayarlanır
  // PORTB, PLC'leri okuyup kontrol birimine gönderir
  PORTB  = 0;
  TRISB  = 0xFF;        // PLC bilgisi/DIP switch bilgisi için kullanılır.Hepsi Giriş
  // PORTC
  PORTC  = 0x0;
  TRISC  = 0xB7;        // C2-C0 ID'nin ilk 3 biti;C3 akü düşük pini;C4 CD pini;C5 boş
  // PORTE Alıcı, Verici kontrolü için kullanılır
  PORTE  = 0x0;
  TRISE  = 0x0;         // Hepsi çıkış
  // PORTD  Röleleri süren port
  PORTD  = 0x0;
  TRISD  = 0x0;         // Hepsi Çıkış

}
```

**********degiskenler.c**********

```
#include "degiskenler.h"
ProgramBilgisi   programs;
uint8       gecenSure;
uint16      gonderimZamanlayicisi;
```

```
uint8       iletisimZamanAsimiZamanlayicisi;
uint8       genelZamanlayici;
uint8       gondermesirasi;
uint8       programzamanlayicisi;
uint8       yeniprogram;

**********iletisim.c**********

#include "iletisim.h"
#include <usart.h>
#include "global.h"
#include "cevre_birim.h"
#include "degiskenler.h"
#include "program.h"

#define readUSART()         RCREG
#define writeUSART(DATA)    TXREG=(DATA)

void tasiyiciDinlemeDurumu0( void );
void tasiyiciDinlemeDurumu1( void );
void gelenVeriDinlemeDurumu( void );
void programIcraKomutuDurumu( void );
void programYukleKomutuDurumu0( void );
void programYukleKomutuDurumu1( void );
void sonuclariBildirKomutuDurumu0( void );
void sonuclariBildirKomutuDurumu1( void );
void gonderilenDegiskenleriAyarla( void );

void gondermeDurumu0( void );
void gondermeDurumu1( void );
void gondermeDurumu2( void );
void gondermeDurumu3( void );
void gondermeDurumu4( void );

void USARTArizaGider(void);

#define KOMUT_SAYISI    7

#pragma romdata
```

```
rom DurumFonk durumlar[ KOMUT_SAYISI ] = {
  NULL,                          // 0x00 BOŞ
  NULL,                          // 0x01 BOŞ
  programIcraKomutuDurumu,           // 0x02
  NULL,       // 0x03
  NULL,      // 0x04
  sonuclariBildirKomutuDurumu0,      // 0x05
  programYukleKomutuDurumu0,         // 0x06
};

#pragma udata

DurumFonk   iletisimDurum;
DurumFonk   zamanAsimiDurum;

uint8       c, tx_buf[4];
uint8       gelenVeriSayisi;
uint8       gonderilecekVeriSayisi;
uint8       hedefId;
uint8       crc;
uint8       safha;
uint16      gonderimZamani;

#pragma code
//-----------------------------------------------------------
void iletisimKontrol( void ) {

  if( iletisimDurum != NULL ) {
    iletisimDurum();
  }

  if( zamanAsimiDurum != NULL ) {
    zamanAsimiDurum();
  }

}

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

void dinlemeyiKes( void ) {
```

```
  aliciCikisiniDevreDisiBirak();
  zamanAsimiDurum = NULL;
  iletisimDurum   = tasiyiciDinlemeDurumu0;
}

//-------------------------------------------------------
/*
*     ZAMAN AŞIMI RUTİNLERİ
*/
//-------------------------------------------------------

void tasiyiciZamanAsimi( void ) {
// 750 ms içinde dinlemeyi keser
  if( iletisimZamanAsimiZamanlayicisi > TASIYICI_ZAMAN_ASIMI_SURESI ) {
    dinlemeyiKes();
  }
}

void gelenVeriZamanAsimi( void ) {
  // 1 sn içinde dinlemeyi keser
  if( iletisimZamanAsimiZamanlayicisi > GELEN_VERI_ZAMAN_ASIMI_SURESI ) {
    dinlemeyiKes();
  }
}

//-------------------------------------------------------
/*
*   DURUMLAR
*/
//-------------------------------------------------------

void tasiyiciDinlemeDurumu0( void ) { // Taşıyıcıyı dinler

  if( !CD_PINI ) {
    iletisimZamanAsimiZamanlayicisi = 0;
    iletisimDurum = tasiyiciDinlemeDurumu1;
  } else if( gonderilecekVeriSayisi > 0 ) {
    if( gonderimZamanlayicisi >= gonderimZamani ) {
      iletisimDurum = gondermeDurumu0;
    }
```

```
  }

}

void tasiyiciDinlemeDurumu1( void ) { // Taşıyıcıyı dinler

  if( CD_PINI ) {
    iletisimZamanAsimiZamanlayicisi = 0;
    zamanAsimiDurum = NULL;
    iletisimDurum = tasiyiciDinlemeDurumu0;
  } else if( iletisimZamanAsimiZamanlayicisi >= TASIYICI_DINLEME_ZAMAN_ASIMI_SURESI
/*250ms*/ ) {
    readUSART(); // Kirli verileri varsa oku. Böylece RX buffer temizlenir
    readUSART(); // Kirli verileri varsa oku. Böylece RX buffer temizlenir
    crc = 0;
    aliciCikisiniEtkinlestir();
    iletisimZamanAsimiZamanlayicisi = 0;
    zamanAsimiDurum = tasiyiciZamanAsimi;
    iletisimDurum = gelenVeriDinlemeDurumu;
  }

}

void gelenVeriDinlemeDurumu( void ) { // Taşıyıcı dinler
  if ((RCSTAbits.OERR)) {
      USARTArizaGider();
      return;
  }
  if( DataRdyUSART() ) {
    c = readUSART();
    crc += c;
    c >>= 4;
    iletisimZamanAsimiZamanlayicisi = 0;
    zamanAsimiDurum = gelenVeriZamanAsimi;
    if( c >= 2 && c < KOMUT_SAYISI ) {
      iletisimDurum = (DurumFonk) durumlar[ c ];
    } else {
      dinlemeyiKes();
    }
  }
```

```
}

//-------------------------------------------------------
/*
*          Program Icra Komutu
*/
//-------------------------------------------------------

void programIcraKomutuDurumu( void ) {
  if ((RCSTAbits.OERR)) {
     USARTArizaGider();
     return;
  }
  if( DataRdyUSART() ) {
    c = readUSART();
    if (yeniprogram == 1){
      programiBaslat( programs.prgm-1 );
      programzamanlayicisi=0;
      yeniprogram = 0;
    }
    dinlemeyiKes();
    crc += c;
  }
}

//-------------------------------------------------------
/*
*                Program
*/
//-------------------------------------------------------

void programYukleKomutuDurumu0( void ) { //
  if ((RCSTAbits.OERR)) {
     USARTArizaGider();
     return;
  }
  if( DataRdyUSART() ) {  // Bütün hedefler çalıştırılacak cihaz ID'sini alır
    c = readUSART();
    programs.hedefNo = c;
```

```
   gondermesirasi++;
   iletisimDurum = programYukleKomutuDurumu1;
   crc += c;
  }
}
void programYukleKomutuDurumu1( void ) { //
  if ((RCSTAbits.OERR)) {
     USARTArizaGider();
     return;
  }
  if( DataRdyUSART() ) {  // Yalnızca HEDEF_ID'si tutan görevi kaydeder
   c = readUSART();
   if( programs.hedefNo == HEDEF_ID || programs.hedefNo == 0 ) {
     programs.prgm = c;
     programs.siraNo = HEDEF_ID;
     yeniprogram = 1; //Yeni bir programın alındığını gösterir
   }
   iletisimDurum = gelenVeriDinlemeDurumu;
   crc += c;
  }
}

//-------------------------------------------------------------
/*
*        Sonuçları Bildir  3
*/
//-------------------------------------------------------------

void sonuclariBildirKomutuDurumu0( void ) {
  if ((RCSTAbits.OERR)) {
     USARTArizaGider();
     return;
  }
  if( DataRdyUSART() ) {
   c = readUSART();
           safha = c;
   iletisimDurum = sonuclariBildirKomutuDurumu1;
   crc += c;
  }
}
```

```
void sonuclariBildirKomutuDurumu1( void ) {
 if ((RCSTAbits.OERR)) {
    USARTArizaGider();
    return;
 }
 if( DataRdyUSART() ) {
  c = readUSART();
  dinlemeyiKes();
  if( c == crc ) {
     gonderimZamani = (uint16) programs.siraNo;
     gonderimZamani *= (uint16) GONDERME_PERIYODU_SURESI;/*1s*/
     gonderimZamanlayicisi = 0;
     gonderilenDegiskenleriAyarla();
  }
 }
}
//-------------------------------------------------------
/*
*        Gönderme Rutinleri
*/
//-------------------------------------------------------

void gonderilenDegiskenleriAyarla( void ) {
 tx_buf[4] = 0x48;
 tx_buf[3] = HEDEF_ID;
 tx_buf[2] = PLC;
 tx_buf[1] = programs.prgm;

 tx_buf[0] = 0;
 tx_buf[0] += tx_buf[4];
 tx_buf[0] += tx_buf[3];
 tx_buf[0] += tx_buf[2];
 tx_buf[0] += tx_buf[1];
 gonderilecekVeriSayisi = 5;
}
//-------------------------------------------------------
void gondermeDurumu0( void ) {
 aliciyiKapa();
 vericiyiAc();
```

```
  iletisimZamanAsimiZamanlayicisi = 0;
  iletisimDurum = gondermeDurumu1;
}

void gondermeDurumu1( void ) {
// Burdaki bekleme iptal edilebilir. Karşı taraf da bu arada 250 ms bekler, start biti ile.
  if( iletisimZamanAsimiZamanlayicisi >= GONDERME_ONCESI_BEKLEME_SURESI ) {
    iletisimDurum = gondermeDurumu2;
  }
}

void gondermeDurumu2( void ) {
  if( !BusyUSART() ) {
    gonderilecekVeriSayisi--;
//    putcUSART( tx_buf[ gonderilecekVeriSayisi ] );
    writeUSART(  tx_buf[ gonderilecekVeriSayisi ] );
    GECIKME_2mS();
    if( gonderilecekVeriSayisi == 0 ) {
      iletisimDurum = gondermeDurumu3;
    }
  }
}

void gondermeDurumu3( void ) {
  if( !BusyUSART() ) {
    iletisimZamanAsimiZamanlayicisi = 0;
    iletisimDurum = gondermeDurumu4;
  }
//  GONDERME_SONRASI_BEKLE();
}

void gondermeDurumu4( void ) {
// Gonderme sonrası 100 ms bekler ki onemsizdir kalabilir.
  if( iletisimZamanAsimiZamanlayicisi >= GONDERME_SONRASI_BEKLEME_SURESI ) {
    vericiyiKapa();
    aliciyiAc();
    iletisimDurum = tasiyiciDinlemeDurumu0;
  }
}
//-------------------------------------------------------
```

```
void iletisimAc( void )  {

  iletisimDurum = tasiyiciDinlemeDurumu0;
  zamanAsimiDurum = NULL;

  gonderilecekVeriSayisi = 0;
  gonderimZamani = 0;
  gonderimZamanlayicisi = 0;
  iletisimZamanAsimiZamanlayicisi = 0;

  aliciCikisiniDevreDisiBirak();

  OpenUSART( USART_RX_INT_OFF &
            USART_TX_INT_OFF &
            USART_ASYNCH_MODE &
            USART_EIGHT_BIT &
            USART_CONT_RX &
            USART_BRGH_LOW,
            BAUD_DEGERI );

  aliciyiAc();
  vericiyiKapa();

}


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


void USARTArizaGider( void ) {
    getcUSART();      // Boş okuma ile tampon bellek silinir.
    getcUSART();      // Boş okuma ile tampon bellek silinir.
    dinlemeyiKes();
             while (1) {

                       CloseUSART();
             OpenUSART( USART_RX_INT_OFF &
            USART_TX_INT_OFF &
            USART_ASYNCH_MODE &
            USART_EIGHT_BIT &
```

```
        USART_CONT_RX &
        USART_BRGH_LOW,
        BAUD_DEGERI );
                        TXSTAbits.TXEN = 0;
                TXSTAbits.TXEN = 1;
                RCSTAbits.CREN = 0;
                RCSTAbits.CREN = 1;
                        RCSTAbits.SPEN = 0;
                        RCSTAbits.SPEN = 1;

                if( RCSTAbits.OERR ){
                        continue;
                } else {
                        break;
                }
        }
}
```

**********program.c**********

```
#include "global.h"
#include "program.h"
#include "degiskenler.h"
#include "cevre_birim.h"

DurumFonk programDurum;
#define YENI_DURUM(D)          programDurum=(D)

void Flas (void);

void Karasimsek (void);
void Karasimsek1 (void);
void Karasimsek2 (void);
void Karasimsek3 (void);

void Tirtil (void);
void Tirtil_1 (void);
void Tirtil2 (void);
void Tirtil3 (void);
```

```
void Zigzag (void);
void Zigzag1 (void);
void Zigzag2 (void);
void Zigzag3 (void);

void Sabit (void);

void Artan (void);
void Artan1 (void);
void Artan2 (void);
void Artan3 (void);

void Azalan (void);
void Azalan1 (void);
void Azalan2 (void);

void Durdur (void);

//-------------------------------------------------------------------------
void Flas (void){
if(programzamanlayicisi>=7){
  programzamanlayicisi = 0x00;
  ROLELER ^= 0xff;
 // YENI_DURUM(Flas);
 }
}

//-------------------------------------------------------------------------
unsigned int led,j;
unsigned char i;

//-------------------------------------------------------------------------
void Karasimsek (void){
   led = 0x100;
   i=0;
   led>>=1;
   YENI_DURUM(Karasimsek1);
}
void Karasimsek1 (void){
  if(programzamanlayicisi>=1){
```

```
      programzamanlayicisi=0;
      ROLELER = led;
      ++i;
      led>>=1;
    if( i>=7 ){
      YENI_DURUM(Karasimsek2);
      i=0;
    }
  }
}
void Karasimsek2 (void){
 if(programzamanlayicisi>=1){
      programzamanlayicisi=0;
      ROLELER = led;
      ++i;
      led<<=1;
    if( i>=9 ){
      YENI_DURUM(Karasimsek3);
      i=0;
    }
  }
}
void Karasimsek3 (void){
  if(programzamanlayicisi>8){
      YENI_DURUM(Karasimsek);
  }
}

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

//-------------------------------------------------------------------------
void Tirtil (void){
  led = 0x100;
  i=0;
  led>>=1;
  YENI_DURUM(Tirtil_1);
}
void Tirtil_1 (void){
 if(programzamanlayicisi>=1){
    led|=0x80;
```

```
   programzamanlayicisi=0;
   ROLELER = led;
   i++;
   led>>=1;
     if( i>=8 ){
       YENI_DURUM(Tirtil2);
       i=0;
     }
  }
}

void Tirtil2 (void){
 if(programzamanlayicisi>=1){
   programzamanlayicisi=0;
   ROLELER = led;
   i++;
   led>>=1;
     if( i>=8 ){
       YENI_DURUM(Tirtil3);
       i=0;
     }
  }
}
void Tirtil3 (void){
  if(programzamanlayicisi>8){
     YENI_DURUM(Tirtil);
  }
}


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

void Zigzag (void){
 if(programzamanlayicisi>=8){
            programzamanlayicisi=0;
            led = 0b10000001;
            i = 1;
            ROLELER = led;
            YENI_DURUM(Zigzag1);
 }
```

```
}
void Zigzag1 (void){
 if(programzamanlayicisi>=4){
            programzamanlayicisi=0;
            led = (led + i)/2 + i ;
            i *= 2;
            ROLELER =ROLELER | led;
            if(i>=8){
                        YENI_DURUM(Zigzag2);
            }
 }
}
void Zigzag2 (void){
 led = 0b00011000;
 i = 8;
 YENI_DURUM(Zigzag3);
}
void Zigzag3 (void){
 if(programzamanlayicisi>=4){
            programzamanlayicisi=0;
            ROLELER = ROLELER - led;
            led = (led - i)*2 + i/2 ;
    i /= 2;
            if(i==0){
                        YENI_DURUM(Zigzag);
    }
 }
}

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


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

void Sabit (void){
            ROLELER = TMR0L;
            YENI_DURUM(NULL);
}
//---------------------------------------------------------------------------
```

```
/*---------------------------------------------------------------------------

void Artan (void){
 if(programzamanlayicisi>=20){
            programzamanlayicisi=0;
            led = 300;
            i = 0; j = 0;
            YENI_DURUM(Artan1);
 }
}

void Artan1 (void){
 j++;
 if( j == led){
   ROLELER = 0XFF;Nop();Nop();Nop();
            ROLELER = 0;
            YENI_DURUM(Artan2);
 }
}
void Artan2 (void){
 led--; j = 0;
 if(led == 0){
            YENI_DURUM(Artan);
            programzamanlayicisi=0;
 }
 else
            YENI_DURUM(Artan1);
}
void Artan3 (void){
 if(programzamanlayicisi>=4){
            programzamanlayicisi=0;
            ROLELER = 0;
            YENI_DURUM(Artan);
 }
}
//---------------------------------------------------------------------------
```

```
void Artan (void){
i = 0;
ROLELER = i;
YENI_DURUM(Artan1);
}
void Artan1 (void){
 if(programzamanlayicisi>=20){
            programzamanlayicisi=0;
            i++;
            ROLELER = i;
 }
}
//---------------------------------------------------------------------------


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

void Azalan (void){
i = 255;
ROLELER = i;
YENI_DURUM(Azalan1);
}
void Azalan1 (void){
 if(programzamanlayicisi>=20){
            programzamanlayicisi=0;
            i--;
            ROLELER = i;
 }
}

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


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

void Durdur (void){
  ROLELER = 0;
  YENI_DURUM(NULL);
}
```

```
//-------------------------------------------------------------------------


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

DurumFonk secenekler[] = {Flas, Karasimsek,
                          Tirtil, Zigzag,
                          Sabit, Artan,
                          Azalan, Durdur};

#pragma code

void programiBaslat( char gorev ) {
  gecenSure = 0;
  if (gorev > 7 )
   YENI_DURUM( NULL );
  else
   YENI_DURUM( secenekler[gorev] );
}

void programAc( void ) {
 programDurum = NULL;
 gecenSure = 0;
 programs.siraNo = 0;
 programs.prgm = 0;
 programs.hedefNo = 0;
 gondermesirasi = 0;
}

void programiIslet( void ) {
// deneme2 = deneme2 ^ 1;
 if( programDurum != NULL ) {
   programDurum();
 }
}
```

EK-B

PLC KARTINA AİT HEX KODLAR

:020000040000FA
:06000000D2EF04F0120033
:06000800E6EF03F0120018
:06002A000100D8090000EE
:0800300090000000100000028
:0800380000018051811107E075
:1000400005D081C0FAFF00018051F96EFADF00018E
:10005000825183110 7E005D083C0FAFF000182516D
:10006000F96EFADF120084940001826B836B900EAC
:10007000806F000E816F12000001A6510F0801E28F
:10008000F2DF12000001A651140801E2ECDF1200B9
:1000900082B808D00001A66B0001C80E806F000E68
:1000A000816F11D000018A5100080DE28E510001CC
:1000B000A45D00018F510001A55905E30001B00EB8
:1000C000806F020E816F120082A80AD00001A66B19
:1000D0000001826B836B900E806F000E816F12D0D7
:1000E000050E0001A65D0EE300018C6B8484000107
:1000F000A66B0001780E826F000E836F060E806F74
:10010000010E816F1200ABA202D032D929D09EAA73
:1001100027D0AECF84F0000184518C2784390F0B97
:10012000846F0001A66B0001840E826F000E836F46
:10013000020E845D14E3070E845D11E28451F66EB5
:1001700084F00001AA050AE10001A105E66E14EC75
:1001800003F0E5520001A96B0001AA6B6CDF0001CE
:1001900084518C271200ABA202D0EAD80FD09EAABD
:1001A0000DD0AECF84F084C0A0F00001A82B0001D8
:1001B000BE0E806F010E816F84518C271200ABA29E
:1001C00002D0D6D826D09EAA24D0AECF84F08250BA
:1001D000200DF3CF00F080407F0B1F0B00100001BB
:1001E000A05D02E0A0510EE184C0A1F08250200D7C
:1001F000F3CF00F080407F0B1F0B0010A26F0001B7
:10020000010EAA6F0001060E806F010E816F8451EE
:100210008C271200ABA202D0ABD80DD09EAA0BD077
:10022000AECF84F084C08DF00001380E806F020ED6
:10023000816F84518C271200ABA202D099D81BD0B9
:100240009EAA19D0AECF84F00EDF00018C51845DE0

:1002500012E10001A25100018E6F8F6B140E8F030B
:10026000F3CF8FF08E03F3CF8EF0F4508F27000181
:10027000A46BA56B01D812000001480E896F825053
:10028000200DF3CF00F080407F0B1F0B0010886F14
:1002900081CF87F0A1C086F0856B89518527885171
:1002A000852787518527865185270E8A6F12007D
:1002B000848284900001A66B0001C40E806F020E40
:1002C000816F12000A0E0001A65D05E30001D80E41
:1002D000806F020E816F1200ACA216D000018A0757
:1002E0008A51EA6A850FE96E000EEA22EF50AD6E80
:1002F000030EE66EB4EC04F0E55200018A5104E10D
:100300000A0E806F030E816F1200ACA207D00001AD
:10031000A66B00011E0E806F030E816F1200020E8D
:100320000001A65D07E3848084920001900E806F37
:10033000000E816F12000001900E806F000E816F21
:10034000826B836B8A6B8E6B8F6B0001A46BA56BCA
:100350000001A66B84940F0EE66EE66A2C0EE66E24
:1003600021EC04F0E552E552E552849284801200BB
:1003700079EC04F079EC04F076DE4F0EAB16AC9A13
:10038000CF0E9D160F0EE66EE66A2C0EE66E21EC81
:1003900004F0E552E552E552AC9AAC8AAB98AB88D2
:1003A000AB9EAB8EABA202D0E8D7E7D700D012004D
:1003B000070E0001A95D02E3A96B831E1200000174
:1003C000B76B010EB86FBB6BD890B833B733D80E8C
:1003D000B56F030EB66F1200010E0001A95D10E3A8
:1003E000A96BB7C083FF0001BB2BD890B833B733DC
:10043000BB6B12000001A951080805E20001BE0EC5
:10044000B56F030EB66F12000001B76B010EB86FE7
:10045000BB6BD890B833B733620EB56F040EB66F6E
:100460001200010E0001A95D13E30001B78F000126
:10047000A96BB7C083FF0001BB2BD890B833B7334B
:10048000080EBB5D05E3920EB56F040EB66FBB6B35
:100490001200010E0001A95D10E3A96BB7C083FF34
:1004A0000001BB2BD890B833B733080EBB5D05E312
:1004B000BC0EB56F040EB66FBB6B12000001A951E4
:1004C000080805E20001480EB56F040EB66F120071
:1004D000080E0001A95D0DE3A96B810E0001B76F45
:1004E000B86B010EBB6FB7C083FFF40EB56F040E7F
:1004F000B66F1200040E0001A95D27E3A96B00018D
:10050000BB51016AB725006EB8510122010E0F0BD5

```
:1005100005E0D89001320032E806FBE10001BB5152
:100520000024B76F000E0120B86FBB9FBB478350FC
:10053000B711006EB8C001F00050836E080EBB5DAD
:1005400004E34C0EB56F050EB66F1200180E0001D5
:10055000B76FB86B080EBB6F620EB56F050EB66F46
:100560001200040E0001A95D2DE3A96B8350026EF9
:10057000036A0001B751025C006EB8510358016E66
:100580000050836EBB51026E036A0250B75D006E6D
:100590000350B859016E010ED89000360136E806B6
:1005A000FBE10001BB51FE0BE8420024B76F000ED7
:1005B0000120B86FBB91BB43BB5104E1D00EB56FB6
:1005C000040EB66F1200D650836E0001B56BB66B89
:1005D00012000001BB6BBBC083FFE40EB56F050EBC
:1005E000B66F1200140E0001A95D05E3A96B0001AE
:10064000E8AE02D0003403D0E750D880005404E272
:100650000001B56BB66B11D0FE0EDBCFE9FFEA6A85
:10066000E9BEEA68D890E936EA36900EE926000E2F
:10067000EA22EECFB5F0EDCFB6F0E552E7CFD9FFE5
:1006800012000001B56BB66B0001A36B0001A26BF9
:10069000A16BA06B0001A86B12000001B551B6114F
:1006A00007E005D0B6C0FAFF0001B551F96EFADFD8
:0206B000120036
:0E06B2002A0EF66E000EF76E000EF86E0001B6
:1006C0000900F550B06F0900F550B16F03E1B06754
:1006D00001D03DD00900F550AB6F0900F550AC6F6B
:1006E0000900F550AD6F09000900F550E96E0900E9
:1006F000F550EA6E090009000900F550AE6F0900D7
:10070000F550AF6F09000900F6CFB2F0F7CFB3F0A4
:10071000F8CFB4F0ABC0F6FFACC0F7FFADC0F8FF48
:100720000001AE5302E1AF5307E00900F550EE6E51
:10073000AE07F8E2AF07F9D7B2C0F6FFB3C0F7FFD4
:10074000B4C0F8FF0001B007000EB15BBFD71200C4
:10075000EAEC04F00001600EBC6F070EBD6F1200E2
:10076000E5EC04F0000917E1C450006E016AD8906E
:10077000015004E6800E005C000E015802E3000EFA
:1007800001D0010EE8B08286E8A082960001500EEA
:10079000BC6F070EBD6F12000001BC51BD1107E018
:1007A00005D0BDC0FAFF0001BC51F96EFADF12009E
:1007B000070EE66E4E0EE66E55EC04F0E552E5527D
:0C07C0000001500EBC6F070EBD6F120050
```

:0407CC00DACFE4FF9D
:1007D000E2CFDAFFE652F2A415D00001A72B140EE7
:1007E000A75D04E30001A32B0001A76B0001A42B6C
:1007F000000EA5230001A62B0001A92BF10ED76E38
:10080000D66AF294E552E5CFDAFF1100C4EC04F0A9
:1008100041EC03F09BEC01F0D8EC03F00001A76B76
:10082000930EE66E98EC04F0E552F10ED76ED66AA0
:10083000F28CF28E4DEC03F01CEC00F0FBD700D0F4
:020840001200A4
:0E084200D9CFE6FFDACFE6FFE1CFD9FFE2CF54
:10085000DAFFAC6AAB6AFD0EDBB0AC88DBA202D07B
:10086000AC8CAB8CDBB4AC8EDBA602D0AB8801D0F9
:10087000AB8ADBA802D0AC8401D0AC949E98DBACF0
:1008800002D09D8A01D09D9A9E9ADBAE02D09D88AF
:1008900001D09D98FB0EDB50AF6EAC8AAB8EE5525B
:0A08A000E5CFDAFFE7CFD9FF120021
:0608AA00D9CFE6FFDACF12
:1008B000E6FFE1CFD9FFE2CFDAFFC26AC16AFD0EDF
:1008C000DB50C16EFD0EDB50040DF350C00BC26E49
:0E08F200D9CFE6FFDACFE6FFE1CFD9FFE2CFA4
:10090000DAFFE652F20E0001BE17ABAC03D0BE9187
:10091000ABB0BE81ABB4BE85ABB2BE87AE50DF6EAE
:10092000DF50E552E552E5CFDAFFE7CFD9FF1200FD
:10093000D9CFE6FFDACFE6FFE1CFD9FFE2CFDAFF8A
:10094000FD0EDB507F0BD56ED76AD66AF294FD0E92
:10095000DBAE02D0F28A01D0F29AD58EE552E5CF15
:08096000DAFFE7CFD9FF120016
:08096800FF0EE350066E480E7D
:1009700001D04C0EE76EE72EFED7E76AE72EFED7D2
:08098000062EF7D7000012005B
:08098800806A070EC16E92683F
:10099000816A9368826AB70E946E846A966A836AE3
:0409A000956A120042
:0C09A40015EE00F025EE00F0F86A059C4E
:0C09B00059EC03F006EC04F0FDD7120037
:0409BC000000000037
:0A09C000620100000000140296011D
:0609CA00C250040B01E025
:0409D000010E120002
:0409D400C2841200C7

```
:0809D800B003BE034804D00483
:0809E000C605D205F8051E064C
:020000040030CA
:0100010021DD
:010003000EEE
:010006008178
:00000001FF
```

EK-C

KUMANDA BİRİMİNE AİT HEX KODLAR

```
:020000040000FA
:0600000065EF09F012009B
:06000800FFEF06F01200FC
:06002A0002001A130000A1
:10003000C5000000090000001613000CE000000FB
:0400400004000000B8
:0C0044000001660E806F000E816F00014D
:100050008051811 1FCE005D081C0FAFF0001805180
:10006000F96EFADFF4D712EC05F06D0EE66E0C0EA9
:1000700049DB580EE66E0C0E45DBAADB02E18E6B07
:1000800005D0320E885DF9E1010E8E6F960E806FFD
:10009000000E816F120012EC05F0470EE66E0C0E9A
:1000A00031DB340EE66E0C0E2DDB200EE66E0C0EF0
:1000B00029DB00018C6B8CDB04E1EE0E806F000EFF
:1000C00014D0320E885D04E10C0E806F010E0DD04D
:1000D000330E885D04E1F60E806F030E06D03F0EEE
:1000E000885DE9E1660E806F000E816F12000001ED
:1000F0008D6B12EC05F0F20EE66E0B0E8FDB27DB3C
:10010000EF6A780E806F010E816F12000001010E00
:100110008D6F12EC05F0BF0EE66E0B0EF3DA8FD882
:10012000020A1AE0030A09E0010AF9E100018C5110
:100130000BE0FC0E806F020E1DD051DB09E274DB78
:10014000A30EE66E0B0E77DB0C0E806F010E12D045
:10015000FEDA8FC0EFFF05D00001660E806F000E43
:1001600009D012EC05F07A0EE66E0B0E57DB780E16
:10017000806F010E816F120012EC05F000018B51AF
:1001800044DB00018B2941DB020E00018B253DDBA6
:10019000030E00018B25BCDA200EE4DA00018D513C
:1001A000E96EEA6AD890E936EA36CE0EE926000E04
:1001B000EA22EECF80F0EDCF81F01200DDDAFC0E06
:1001C000E66E020EE2D2D8DA0C0EE66E010EDDD239
:1001D000D9CFE6FFE1CFD9FF32D8040A24E0070ADD
:1001E0001FE0010A18E0030AF7E1F9DA0AE21CDB72
:1001F0006C0EE66E0B0E1FDB780E806F010E816FAA
:100200001AD0A5DAEE528FC0EFFF8C2BFD0EDBCF9C
:1002100080F0FE0E0ED0F90EDBCF80F0FA0E09D082
```

```
:1002200000018B0702D000018B2BFB0EDBCF80F08F
:10023000FC0EDBCF81F0E552E7CFD9FF12000001C1
:100240008F6B906B0F0ED2DA04EC09F00001886F0F
:100250003E0E885D08E10C0EC9DA00018F51901145
:1002600001E1000C010C3F0E885D03E10C0EBEDACB
:10027000020C3A0E885D16E1000ED8808F55000EF4
:100280009055E2E2080E6EDA0A0E0B6E0C6A8FC011
:1002900006F090C007F03FEC09F006C08FF007C0F1
:1002A00090F0D2D7300E885D1BE38851390818E3EF
:1002B00088C0E6FF92EC04F0E55200010A0E9003BC
:1002C000F3CF90F08F03F3CF8FF0F4509027300EE0
:1002D000885D106E116AE8508F2711509023B4D7B3
:1002E0003C0E885D04E18B510008AEE2030C3D0E2C
:1002F000885DAAE1040E8B5DA7E2040CD9CFE6FF6E
:10030000E1CFD9FF020EE12612EC05F0620EE66E97
:100310000B0EF8D9540EE66E0B0EF4D9F40E70DA0B
:10032000030E69DA510EE66E0B0EECD9F40E68DAA4
:100330000F0E61DA4E0EE66E0B0EE4D9060E5BDA96
:1003400074EC07F0DE6ADD6ADECF10F0DDCF11F06D
:1003500000018C51000111AE02D0D89003D0105C86
:10036000000E115818E200018C05106E116AD89029
:1003700010361136910E1026000E112210C0E6FF25
:1003800011C0E6FF9BEC07F0E552E552DF2A010EB3
:10039000DBE3DB2AD9D7DE6ADD6ADECF10F0DDCF02
:1003A00011F0D890115004E6080E105C000E1158A0
:1003B00009E23E0ED7D97A0E23DADF2A010EEDE3E9
:1003C000DB2AEBD78CEC07F07EEC07F0390EE66EFB
:1003D0000B0E98D93D0ECBD90001F60E806F030E9F
:1003E000816F020EE15C02E2E16AE552E16EE552E4
:1003F000E7CFD9FF120012EC05F0E66AE66A9CD955
:10040000360EE66E0B0E7ED9050E1ADA330EE66E48
:100410000B0E78D90A0E14DA300EE66E0B0E72D976
:100420000F0E0EDA2D0EE66E0B0E6CD9E66A020E7A
:10043000E66E82D92A0EE66E0B0E64D9050EE9D956
:10044000270EE66E0B0E5ED90A0EE3D9240EE66E79
:100450000B0E58D90F0EDDD9210EE66E0B0E52D9B8
:100460000001010E8A6F8A51080806E37A0EC8D986
:10047000FDD800018A2BF7D78FD91ED80A0E0001AC
:10048000A25D03E389D90001A26BF6EC08F0000934
:1004900011E000EC09F00001886F3F0E885D0AE171
```

```
:1004A000826B836B846B856B818E960E806F000EE2
:1004B000816F120010D8E2D777D90001A26B00013A
:1004C000A36BA46BE66AF10EE66EBEEC08F0E55293
:1004D000E552D58E1200000182518311O7E005D04C
:1004E00083C0FAFF00018251F96EFADF00018451E6
:1004F000851107E005D085C0FAFF00018451F96E2F
:10050000FADF1200D4EF03F000011E0EA35D000E0F
:10051000A45901E3F7DF120000011E0EA35D000ED7
:10052000A45901E3EFDF120081BA07D000013C0EAD
:10053000826F050E42D9846B856B120081AA09D0A7
:100540000001280E826F050E836F0001A36BA46B60
:1005500012000001050EA35D000EA4590DE39FECEF
:1005600008F082800001896B7A0E826F050E25D912
:10057000080E846F050E856F1200ABA202D074EFD7
:1005800003F09EAA14D09FEC08F00001896FE838B0
:100590000F0B886F040E885D09E1B00E826F050EA7
:1005A0000CD9180E846F050E856F1200ABDF120098
:1005B000ABA202D074EF03F09EAA0DD004D9E8508C
:1005C00002E182EF02F088C08AF0D80E826F050E39
:1005D000836F885189271200ABA202D074EF03F019
:1005E0009EAA0AD0F0D8ACD888C0EFFFFA0E826F6E
:1005F000050E836F885189271200ABA202D074EFD9
:1006000003F09EAA28D0DFD88E511EE099D8EF5073
:10061000885D1AE095D8EF5017E092D8EF5008089F
:1006200013E38AC091F08CD8EFCF92F074EC07F00E
:1006A00082EC04F068EC04F00009FCE1DFEC04F0FB
:1006B000E66E18EC05F0E552A50EE66E000EE66E4D
:1006C00000013ED8EF50E66EE66A65EC06F0E552B2
:1006D000E552E552E552A50EE66E000EE66ED6EC4A
:1006E00004F0E552E55212009FEC08F09FEC08F090
:1006F00009DFAC9AAC8AAB98AB88AB9EAB8EABB241
:10070000F8D71200E66ECDEC04F0E552E552120087
:10071000F66EF76AD890F636F736F80EF626120E11
:10072000F7220900F5CFE6FF0800F5CFE6FFCDEC94
:1007300004F0E552E552120087D87A0E00EF04F07B
:100740008A51EA6AC50FE96E000EEA2212008C5146
:10075000E96EEA6AD890E936EA36910EE926000E8B
:10076000EA221200E66E92EC04F0E5521200E66E08
:1007700011EC09F0E5521200960EE66E000EE66EE0
:10078000780EE66E010EE66E1200E66E21DD106E4A
```

:10079000060EE15E1050120074EC07F0E66A92EC6F
:1007A00007F0E5527EEF07F082900001846B856BC5
:1007B000280E826F050E836F1200836F0001A36BFA
:1007C000A46B000112009FEC08F00001886F12007A
:1007D00004EC09F00001886F310E885D1200000101
:1007E000080ED8808F55000E90551200E66E5DEC15
:1007F00004F0E5521200E66E020EE66E24EF04F0FD
:10080000E66E55EC09F0E552120082DF0A0EB2EFF7
:1008100003F0E66E020EE66E9CEF03F073DF7A0ED5
:10082000A6DF00018B6B120012EC05F0B20EE66E33
:100830000B0E82EF03F066DFF40E99DF0001120069
:10084000E66EE66A9CEF03F0F1EC04F0E552E55247
:02085000120094
:0E085200300E836E8480EED88490050EE66E24
:1008600055EC09F0E5521200ECD8846A0A0EC16E0C
:10087000966A836A9568849284948490020EE66EE8
:1008800011EC09F0E552956AE4DFE3DFE2DF9568F9
:10089000E1D8FEE1380EE6D8DDD8FEE1080EE2D858
:1008A000D9D8FEE10C0EDED8D5D8FEE1170EDAD885
:1008B000B9D8D0D8FEE1F2D8C9D0FED8F5D0FCD84E
:1008C000836E84848492FDD8849084949568BED08D
:1008D00084828494F6D883AE03D084908492010CF1
:1008E00084908492000C0001D32B140ED35D09E395
:1008F000D36BD22B040ED25D01E3D26BABD8FEE1F9
:10090000ACD812000001D351000802E2D307120054
:10091000130ED36FD251000801E2D2079BD8FEE13B
:100920002DEF05F08ED8E652FE0EDB500A080CE1E2
:100930000001D36BD22B040ED25D01E3D26B8AD8B7
:10094000FEE13DD8E66E1CD0FE0EDB5008081BE130
:100950000001D351000802E2D30706D0130ED36F73
:10096000D251000801E2D2072AD8DF6EDFCFE6FFBE
:100970005FD8E552200EE66EA2DFE552DFCFE6FF3C
:1009800057D8E55208D066D8FEE1FE0EDBCFE6FF71
:1009900096DFE552A8DFE55259D053D867D805E075
:1009A00065D86DD8FBE3DB2AF9D750D04AD86ED88A
:1009B00005E06CD864D8FBE3DB2AF9D747D041D8EF
:1009C000E652DF6A0001D2A102D0400EDF6E020EB5
:1009D000D25D02E3140EDF26D351DF26DF50E5524D
:1009E00035D02FD8140EF36EFD0EDBCFF4FFF3508D
:1009F000F45C04E2FD0EDBCFD3F002D00001D36B38

:100A0000040EF36EFE0EDBCFF4FFF350F45C04E251
:100A1000FE0EDBCFD2F002D00001D26B1BD8FEE17C
:100A20001CD814D017D8FEE1010E1CD84EEF05F0EB
:100A300043D8800939D0020EE66E71EC09F0E55218
:100A40001200D9CFE6FFE1CFD9FF1200E552E7CF80
:100A5000D9FF12003DDF00091200B1DFE66EE8DFCA
:100A6000E5521200E66E29DFE5521200FD0EDBCFE3
:100A7000F6FFFE0EDBCFF7FF0800F5501200E66E22
:100A800051DFE552FD0EDB2AFE0E1200FD0EDBCF1C
:100A9000E9FFFE0EDBCFEAFFEF501200E66AE66ADE
:100AA000A0DFE552E5521200836E8492849409D847
:100AB0008490C1DF9568CAD7C4DF956AFE0EDB500B
:0A0AC0001200B9DF84801BEF05F07F
:060ACA0020202020202066
:100AD000203230303720202020202020202000204EBF
:100AE0004947444520554E4956455253495445536C
:100AF000490A00204D5548454E444953205553540A
:100B000045474D454E200A0020202059494C4D4173
:100B10005A204455524E41202020200A00202020F7
:100B2000005038005037005036005035005034005340027
:100B3000503300503200503100A474F4E44455266
:100B40004D452054414D414D4C414E444900527851
:100B500000547800474F4E444552494C49594F5232
:100B60000A0042494C47494C45520A005052472E10
:100B700020534543494C44490A00504C43204943C3
:100B8000494E0A312E2E382041524153490A4249DA
:100B9000522050524F4752414D20534543494E49F0
:100BA0005A0A00434948415A20534543494C444955
:100BB0000A004F4C4D4159414E204249520A0043D0
:100BC000414C4953544952494C4143414B0A4349D2
:100BD00048415A4920534543494E0A274F4E4159EF
:100BE00027204241534C415449520A636968617A53
:100BF0003D00504C43274C4552204943494E0A3151
:100C00002E2E382041524153490A42495220505217
:100C10004F4752414D20534543494E495A0A2000FF
:100C2000332E20534F4E55434C41524920474F538A
:100C300054455200322E204641524B4C49205052CE
:100C40004F4752414D0A00312E2041594E49205004
:100C5000524F4752414D0A00322E204F54414D41D0
:100C800000382E44757264757200372E417A616C9B

```
:100C9000616E00362E417274616E00352E536162B2
:100CA000697400342E5A69677A616700332E54697B
:100CB0007274696C00322E4B61726173696D736579
:0A0CC0006B00312E466C6173680072
:060CCA00D9CFE6FFDACFEE
:100CD000E6FFFA0EE3CFD9FFFB0EE3CFDAFFFD0EFE
:100CE000E35209E72D0EDE6EFC0EE31EE32AFD0E35
:100CF000D8B4E306E31EE652E652E652FD0EE124C6
:100D0000E76EFF0EE220E46EE652F70EE3CFE6FF59
:100D1000F70EE3CFE6FFD5EC06F0F90EE124E96E1D
:100D2000FF0EE220EA6EC5EC06F00AE1C2EC06F026
:100D300009E1C5EC06F008E1C2EC06F007E108D0D5
:100D4000D0EC06F0CBEC06F0D0EC06F0CBEC06F0E5
:100D5000D0EC06F0DE6AF90EE126FF0EE222FF0E6D
:100D6000E3CFDAFFE552FF0EE3CFD9FFE552FC0EE9
:100D7000E124E96EFF0EE220EA6EEECFF3FFEFCF43
:100D8000F4FF1200EF380F0B1200EF500F0B02E1CF
:100D9000EE52D8841200EF380F0B300FDE6E1200C7
:100DA000EE500F0B300FDE6E1200FC0EE3CFE9FFAA
:100DB000FD0EE3CFEAFF100EE66EEE6AEE6AEF6A12
:100DC000D890FD0EE336FE0EE336ED36ED36EF3607
:100DD000FF0EE30609E0F6EC06F0EE52F6EC06F044
:100DE000EE52F6EC06F0EDD7E56E1200030EEF249E
:0E0DF000E8B6EF6E300EEF24E8BEEF6E000C9A
:020DFE00DACF4A
:100E0000E4FFE652F2A413D0140E0001A15D06E344
:100E10000001A22B0001B92B0001A16BA12B000145
:100E2000A32B000EA423F10ED76ED66AF294E552DE
:100E3000E5CFDAFF1100816A818E818C936A826A24
:100E4000800E946E0EEC09F03B0EE66E34EC04F06E
:100E5000E55212EC05F0080EE66E0B0E2CD8F30EE0
:100E6000E66E0A0E28D8DE0EE66E0A0E24D8CA0EEA
:100E7000E66E0A0E20D8960EE66E11EC09F0E552E9
:100E80003F0E93120F0EE66EE66A2C0EE66E47ECEE
:100E900008F0E552E552E552930EE66EDAEC08F002
:100EA000E5520001A16B0001A26BD59EA198F28CC6
:100EB000F28E22EF00F0E66ECDEC04F0E552E55242
:020EC00012001E
:0E0EC2004FD8ACA2FED7FE0EDBCFE6FF28EC29
:100ED00009F0E552030EE66E55EC09F0E552FE0E00
```

:100EE000DB500001D72742D00001D76B818E819C57
:100EF0003D0EE66E11EC09F0E5521200D7C0E6FF98
:100F0000E0DFE552ACA2FED77A0EE66E55EC09F0B2
:100F1000E552818C819E1200280EA6EF07F0380E54
:100F5000E5521200FD0EDBCFE9FFFE0EDBCFEAFF0C
:100F60001200D9CFE6FFE1CFD9FF1200E552E7CF5B
:040F7000D9FF120093
:0C0F74002A0EF66E000EF76E000EF86EEE
:100F800000010900F550C06F0900F550C16F03E181
:100F9000C06701D03DD00900F550BB6F0900F55086
:100FA000BC6F0900F550BD6F09000900F550E96EEE
:100FB0000900F550EA6E090009000900F550BE6FFE
:100FC0000900F550BF6F09000900F6CFC2F0F7CF56
:100FD000C3F0F8CFC4F0BBC0F6FFBCC0F7FFBDC084
:100FE000F8FF0001BE5302E1BF5307E00900F550CE
:100FF000EE6EBE07F8E2BF07F9D7C2C0F6FFC3C066
:10100000F7FFC4C0F8FF0001C007000EC15BBFD7E7
:021010001200CC
:0E10120089EC09F00001220ED46F100ED56F8C
:10102000120084EC09F0000917E1C450106E116A37
:10103000D890115004E6800E105C000E115802E3A7
:10104000000E01D0010EE8B08282E8A08292000179
:10105000120ED46F100ED56F12000001D451D511AD
:1010600007E005D0D5C0FAFF0001D451F96EFADFD0
:101070001200070EE66E4E0EE66E7BEC08F0E552AF
:0E108000E5520001120ED46F100ED56F120053
:02108E00D9CFB8
:10109000E6FFDACFE6FFE1CFD9FFE2CFDAFFAC6AB5
:1010A000AB6AFD0EDBB0AC88DBA202D0AC8CAB8CA3
:1010B000DBB4AC8EDBA602D0AB8801D0AB8ADBA858
:1010C00002D0AC8401D0AC949E98DBAC02D09D8A57
:1010D00001D09D9A9E9ADBAE02D09D8801D09D984A
:1010E000FB0EDB50AF6EAC8AAB8EE552E5CFDAFF7C
:0610F000E7CFD9FF12005A
:0A10F600D9CFE6FFDACFE6FFE1CF25
:10110000D9FFE2CFDAFFC26AC16AFD0EDB50C16EC1
:10111000FD0EDB50040DF350C00BC26EFC0EDB5015
:10112000380BC212FC0EDBAE03D09E9C9D8CF28C61
:0E113000C280E552E5CFDAFFE7CFD9FF12000B
:02113E00D9CF07

:10114000E6FFDACFE6FFE1CFD9FFE2CFDAFFE652E2
:10115000F20E0001D617ABAC03D0D691ABB0D6815E
:10116000ABB4D685ABB2D687AE50DF6EDF50E5525A
:0C117000E552E5CFDAFFE7CFD9FF12000F
:04117C00D9CFE6FFE2
:10118000DACFE6FFE1CFD9FFE2CFDAFFE652E6524F
:10119000FC0EDBCFDEFFDBCFDDFF010EDB50D76EB9
:1011A000DF50D66EE552E552E552E5CFDAFFE7CFE4
:0411B000D9FF120051
:0C11B400D9CFE6FFDACFE6FFE1CFD9FF8C
:1011C000E2CFDAFFFD0EDB507F0BD56ED76AD66A11
:1011D000F294FD0EDBAE02D0F28A01D0F29AD58EE7
:0C11E000E552E5CFDAFFE7CFD9FF12009F
:0411EC0081A807D0FF
:1011F000020EE66E11EC09F0E55281B8010C000C0C
:101200000F0E8114300F1200F1DF0009FDE1EEDF57
:101210000009FDE00F0E8114300F12001F0E931213
:021220001200BA
:10126000ACAC04D0AC900001D6B3AC80FD0EDB502A
:0E127000AD6EE552E5CFDAFFE7CFD9FF1200F1
:02127E00036A01
:10128000026A100EE76ED89006360736023603362D
:101290000B50025C0C50035805E30B50025E0C50DF
:0A12A000035A062AE72EF0D71200C9
:0612AA00FF0EE350136E7D
:1012B000480E01D04C0EE76EE72EFED7E76AE72E08
:0A12C000FED7132EF7D7000012002E
:0612CA0015EE00F025EE18
:1012D00000F0F86A0F9CBAEC07F01BEC07F0FDD7A2
:0212E0001200FA
:0E12E200FF0EE350E84E1200000000D000D0D6
:0812F00000D0E82EFAD712002D
:0812F800C20CB50CAC0CA30CF8
:081300009B0C930C8A0C810C7C
:08130800C250040B01E0010ECC
:021310001200C9
:04131200C28412007F
:04131600BC01C6014F
:06131A00000000000000CD
:03132000000000CA

EK-D

KATALOĞ BİLGİLERİ

FAIRCHILD
SEMICONDUCTOR™

GlobalOptoisolator™

VDE | UL | CSA | SETI | SEMKO | DEMKO | NEMKO | BABT

6-Pin DIP Optoisolators Transistor Output

4N25
4N26
4N27
4N28

The 4N25, 4N26, 4N27 and 4N28 devices consist of a gallium arsenide infrared emitting diode optically coupled to a monolithic silicon phototransistor detector.

- Most Economical Optoisolator Choice for Medium Speed, Switching Applications
- Meets or Exceeds All JEDEC Registered Specifications
- ***To order devices that are tested and marked per VDE 0884 requirements, the suffix "V" must be included at end of part number. VDE 0884 is a test option.***

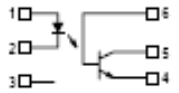
Applications

- General Purpose Switching Circuits
- Interfacing and coupling systems of different potentials and impedances
- I/O Interfacing
- Solid State Relays



STANDARD THRU HOLE

SCHEMATIC



PIN 1. LED ANODE
2. LED CATHODE
3. N.C.
4. EMITTER
5. COLLECTOR
6. BASE

MAXIMUM RATINGS (T_A = 25°C unless otherwise noted)

Rating	Symbol	Value	Unit
INPUT LED			
Reverse Voltage	V_R	3	Volts
Forward Current — Continuous	I_F	60	mA
LED Power Dissipation @ T_A = 25°C with Negligible Power in Output Detector	P_D	120	mW
Derate above 25°C		1.41	mW/°C
OUTPUT TRANSISTOR			
Collector–Emitter Voltage	V_{CEO}	30	Volts
Emitter–Collector Voltage	V_{ECO}	7	Volts
Collector–Base Voltage	V_{CBO}	70	Volts
Collector Current — Continuous	I_C	150	mA
Detector Power Dissipation @ T_A = 25°C with Negligible Power in Input LED	P_D	150	mW
Derate above 25°C		1.76	mW/°C
TOTAL DEVICE			
Isolation Surge Voltage[1] (Peak ac Voltage, 60 Hz, 1 sec Duration)	V_{ISO}	7500	Vac(pk)
Total Device Power Dissipation @ T_A = 25°C	P_D	250	mW
Derate above 25°C		2.94	mW/°C
Ambient Operating Temperature Range	T_A	−55 to +100	°C
Storage Temperature Range	T_{stg}	−55 to +150	°C
Soldering Temperature (10 sec, 1/16" from case)	T_L	260	°C

1. Isolation surge voltage is an internal device dielectric breakdown rating. For this test, Pins 1 and 2 are common, and Pins 4, 5 and 6 are common.

Şekil D.1 4N25 Optokuplörüne ait katalog bilgisi

FAIRCHILD
SEMICONDUCTOR™

ELECTRICAL CHARACTERISTICS (T_A = 25°C unless otherwise noted)[1]

Characteristic		Symbol	Min	Typ[1]	Max	Unit
INPUT LED						
Forward Voltage (I_F = 10 mA)	T_A = 25°C T_A = −55°C T_A = 100°C	V_F	— — —	1.15 1.3 1.05	1.5 — —	Volts
Reverse Leakage Current (V_R = 3 V)		I_R	—	—	100	µA
Capacitance (V = 0 V, f = 1 MHz)		C_J	—	18	—	pF
OUTPUT TRANSISTOR						
Collector–Emitter Dark Current (V_{CE} = 10 V, T_A = 25°C	4N25,26,27 4N28	I_{CEO}	— —	1 1	50 100	nA
(V_{CE} = 10 V, T_A = 100°C)	All Devices	I_{CEO}	—	1	—	µA
Collector–Base Dark Current (V_{CB} = 10 V)		I_{CBO}	—	0.2	—	nA
Collector–Emitter Breakdown Voltage (I_C = 1 mA)		$V_{(BR)CEO}$	30	45	—	Volts
Collector–Base Breakdown Voltage (I_C = 100 µA)		$V_{(BR)CBO}$	70	100	—	Volts
Emitter–Collector Breakdown Voltage (I_E = 100 µA)		$V_{(BR)ECO}$	7	7.8	—	Volts
DC Current Gain (I_C = 2 mA, V_{CE} = 5 V)		h_{FE}	—	500	—	—
Collector–Emitter Capacitance (f = 1 MHz, V_{CE} = 0)		C_{CE}	—	7	—	pF
Collector–Base Capacitance (f = 1 MHz, V_{CB} = 0)		C_{CB}	—	19	—	pF
Emitter–Base Capacitance (f = 1 MHz, V_{EB} = 0)		C_{EB}	—	9	—	pF
COUPLED						
Output Collector Current (I_F = 10 mA, V_{CE} = 10 V)	4N25,26 4N27,28	I_C (CTR)[2]	2 (20) 1 (10)	7 (70) 5 (50)	— —	mA (%)
Collector–Emitter Saturation Voltage (I_C = 2 mA, I_F = 50 mA)		$V_{CE(sat)}$	—	0.15	0.5	Volts
Turn–On Time (I_F = 10 mA, V_{CC} = 10 V, R_L = 100 Ω)[3]		t_{on}	—	2.8	—	µs
Turn–Off Time (I_F = 10 mA, V_{CC} = 10 V, R_L = 100 Ω)[3]		t_{off}	—	4.5	—	µs
Rise Time (I_F = 10 mA, V_{CC} = 10 V, R_L = 100 Ω)[3]		t_r	—	1.2	—	µs
Fall Time (I_F = 10 mA, V_{CC} = 10 V, R_L = 100 Ω)[3]		t_f	—	1.3	—	µs
Isolation Voltage (f = 60 Hz, t = 1 sec)[4]		V_{ISO}	7500	—	—	Vac(pk)
Isolation Resistance (V = 500 V)[4]		R_{ISO}	10^{11}	—	—	Ω
Isolation Capacitance (V = 0 V, f = 1 MHz)[4]		C_{ISO}	—	0.2	—	pF

1. Always design to the specified minimum/maximum electrical limits (where applicable).
2. Current Transfer Ratio (CTR) = I_C/I_F x 100%.
3. For test circuit setup and waveforms, refer to Figure 11.
4. For this test, Pins 1 and 2 are common, and Pins 4, 5 and 6 are common.

Şekil D.2 4N25 Optokuplörü elektriksel özellikleri katalog bilgisi

MICROCHIP

PIC18FXX2

28/40-pin High Performance, Enhanced FLASH Microcontrollers with 10-Bit A/D

High Performance RISC CPU:

- C compiler optimized architecture/instruction set
  - Source code compatible with the PIC16 and PIC17 instruction sets
- Linear program memory addressing to 32 Kbytes
- Linear data memory addressing to 1.5 Kbytes

Device	On-Chip Program Memory		On-Chip RAM (bytes)	Data EEPROM (bytes)
	FLASH (bytes)	# Single Word Instructions		
PIC18F242	16K	8192	768	256
PIC18F252	32K	16384	1536	256
PIC18F442	16K	8192	768	256
PIC18F452	32K	16384	1536	256

- Up to 10 MIPs operation:
  - DC - 40 MHz osc./clock input
  - 4 MHz - 10 MHz osc./clock input with PLL active
- 16-bit wide instructions, 8-bit wide data path
- Priority levels for interrupts
- 8 x 8 Single Cycle Hardware Multiplier

Peripheral Features:

- High current sink/source 25 mA/25 mA
- Three external interrupt pins
- Timer0 module: 8-bit/16-bit timer/counter with 8-bit programmable prescaler
- Timer1 module: 16-bit timer/counter
- Timer2 module: 8-bit timer/counter with 8-bit period register (time-base for PWM)
- Timer3 module: 16-bit timer/counter
- Secondary oscillator clock option - Timer1/Timer3
- Two Capture/Compare/PWM (CCP) modules. CCP pins that can be configured as:
  - Capture input: capture is 16-bit, max. resolution 6.25 ns (TCY/16)
  - Compare is 16-bit, max. resolution 100 ns (TCY)
  - PWM output: PWM resolution is 1- to 10-bit, max. PWM freq. @: 8-bit resolution = 156 kHz, 10-bit resolution = 39 kHz
- Master Synchronous Serial Port (MSSP) module, Two modes of operation:
  - 3-wire SPI™ (supports all 4 SPI modes)
  - I^2C™ Master and Slave mode

Peripheral Features (Continued):

- Addressable USART module:
  - Supports RS-485 and RS-232
- Parallel Slave Port (PSP) module

Analog Features:

- Compatible 10-bit Analog-to-Digital Converter module (A/D) with:
  - Fast sampling rate
  - Conversion available during SLEEP
  - Linearity ≤ 1 LSb
- Programmable Low Voltage Detection (PLVD)
  - Supports interrupt on-Low Voltage Detection
- Programmable Brown-out Reset (BOR)

Special Microcontroller Features:

- 100,000 erase/write cycle Enhanced FLASH program memory typical
- 1,000,000 erase/write cycle Data EEPROM memory
- FLASH/Data EEPROM Retention: > 40 years
- Self-reprogrammable under software control
- Power-on Reset (POR), Power-up Timer (PWRT) and Oscillator Start-up Timer (OST)
- Watchdog Timer (WDT) with its own On-Chip RC Oscillator for reliable operation
- Programmable code protection
- Power saving SLEEP mode
- Selectable oscillator options including:
  - 4X Phase Lock Loop (of primary oscillator)
  - Secondary Oscillator (32 kHz) clock input
- Single supply 5V In-Circuit Serial Programming™ (ICSP™) via two pins
- In-Circuit Debug (ICD) via two pins

CMOS Technology:

- Low power, high speed FLASH/EEPROM technology
- Fully static design
- Wide operating voltage range (2.0V to 5.5V)
- Industrial and Extended temperature ranges
- Low power consumption:
  - < 1.6 mA typical @ 5V, 4 MHz
  - 25 µA typical @ 3V, 32 kHz
  - < 0.2 µA typical standby current

Şekil D.3 PIC 18F452 katalog bilgisi

ABACOM Technologies

RFRQ2 Crystal controlled PLL FM Receiver Module

Features

- PLL XTAL Design
- CMOS/TTL Output
- RSSI Output
- **Standby Mode (max 100nA)**
- 5V Operation
- High stability over
- Compatible with FM RF transmitter modules such as RFTQ1, RFTQ2, FM-RTF, TXM-433 etc.
- SIL pinout for vertical or horizontal placement
- Data rates up to 9.6Kbps



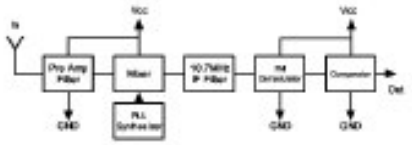
A Single in Line Package for vertical or horizontal placement. Form factor and pin-out compatible with a variety of other RF receiver modules making the RFRQ2 ideal for competitive evaluation against other RF receiver modules.

Description

These miniature RF receiver modules provide a cost effective high performance FM Radio data link, at either 315, 433.92 or 868MHz. Manufactured using laser trimmed Thick Film Ceramic Hybrid the modules exhibits extremely stable electronic characteristics over an Industrial Temperature range. The hybrid technology uses no adjustable components and ensures very reliable operation. Together with a suitable RF transmitter module such as the RFTQ1 or RFTQ2 the simple implementation of a data link at distances up to 200ft in-building and 700ft open ground may be easily achieved.

These RF receiver modules will suit one-to-one and multi-node wireless links in applications including car and building security, EPOS and inventory tracking, remote industrial process monitoring and computer networking. Because of their small size and low power requirements, these modules are ideal for use in portable, battery powered applications such as hand-held terminals.

Receiver Block Diagram



Şekil D.4 RFRQ2 alıcısı katalog bilgisi

Technical Specifications

Electrical Characteristics	Min	Typical	Max	Dimension	Notes
Supply Voltage (Vcc)	4.5	5	5.5	V	
Supply Current (Operating)		5.7	6.8	mA	
Supply Current (Standby)			100	nA	
Receive Frequency		315.00 433.92 868.35		MHz	
R.F Sensitivity 315 ,433MHZ versions 868MHz versions		 -103 -100		dBm	
3dB Bandwidth		+/-150		KHz	
Data Rate	300		9,600	Hz	
Turn on Time		8		mSecs	1
Level of Emitted Spectrum			-70	dBm	
Low Level Output Voltage			0.8	V	I = 200uA
High Level Output Voltage	Vcc-1			V	I = 200uA
Operating Temperature Range	-25		+80	$^{o}_{C}$	

Note 1: Time from applying power to stable data

Prototyping Hints:

It is essential when building any Low Power Radio System that you have a 'clean' DC power source. Typically the ripple voltage should be less than 10mV Peak to Peak. Normally a 470uF decoupling capacitor is sufficient de-coupling for an AC derived DC power source.

Never place a Transmitter or Receiver directly into plug-in type breadboard. This will severely restrict the range. Rather, use small lengths of wire from the prototyping board to the pins of the Transmitter or Receiver.

A useful antenna, for testing purposes, for both the Transmitter and Receiver on 433MHz is to use a piece of wire 16.5cm long for 433MHz soldered directly to the antenna pin.

Şekil D.5 RFRQ2 alıcısı teknik özelliklerine ait katalog bilgisi

ABACOM Technologies

RFTQ1 RFTQ2 Crystal Controlled PLL FM Transmitter Modules

- FM Radio Transmitter Modules
- Available in 315 or 433 or 868MHz
- Transmit Range Up To 700ft
- Miniature Packages
- Data Rate up to 9.6Kbps
- No Adjustable Components
- Very Stable Operating Frequency
- Operates from –20 to +85°C
- 3-12 V Operation (RFTQ2)
- 3V Operation (RFTQ1)
- SIL (RFTQ2) Package or DIL (RFTQ1) Package



A Single In Line Package incorporating a voltage regulator for 3-12V operation. (Compatible with many other RF transmitter modules available as a drop-in alternative and therefore ideal for competitive evaluation)

Applications

- Wireless Security Systems
- Car Alarms
- Remote Gate Controls
- Remote Sensing
- Data Capture
- Sensor Reporting



A Dual In Line Package operating at 3.3V. This provides the most rugged mechanical fixing to the host PCB. Power Down mode is also available.

Description

These miniature RF modules provide a cost effective high performance FM Radio data link, at either 315, 433.92 or 868MHz. Manufactured using laser trimmed Thick Film Ceramic Hybrid the modules exhibits extremely stable electronic characteristics over an Industrial Temperature range. The hybrid technology uses no adjustable components and ensures very reliable operation. This transmitter and receiver pair enables the simple implementation of a data link at distances up to 200ft In-building and 700ft open ground.

These RF modules will suit one-to-one and multi-node wireless links in applications including car and building security, EPOS and inventory tracking, remote industrial process monitoring and computer networking. Because of their small size and low power requirements, both modules are ideal for use in portable, battery powered applications such as hand-held terminals.

Şekil D.6 RFTQ2 vericisine ait katalog bilgisi

Pin Description

RTFQ1	RTFQ2	Name	Description
1	N/A	En	Enable (active high)
2	5	IN	Data Input
3	1	GND	Ground, Connect to RF earth return path
4	3	Vcc	Supply Voltage
5	4	GND	Ground, Connect to RF earth return path
6	2	EA	External Antenna

Specifications

Electrical Characteristics	MIN	TYPICAL	MAX	DIMENSION
Supply Voltage RTFQ1	2.1	3.3	4.00	V
Supply Voltage RTFQ2	2.5		12.00	V
Supply Current		7	8	mA
Standby Current (IN = EN = Low)			100	nA
Frequency		315.0 433.92 868.35		MHz
RF Output into 50Ω (Vcc=3.3V)		+5 / +5 / +1		dBm
Initial Frequency Accuracy	-35	0	+35	KHz
FM Deviation	25	30	35	KHz
Harmonic Spurious Emissions		-50		dBc
Input High Voltage RTFQ1	1.5		Vcc	V
Input High Voltage RTFQ2	1.5		5.5	V
Power up Time (En to full RF)			1	mS
Power up Time (Power on to full RF)			5	mS
Max Data Rate			9.6	KHz
Operating Temperature	-25		+80	°C

Şekil D.7 RFTQ2 vericisi bacak bağlantıları ve teknik özelliklerine ait katalog bilgisi