EA-16F877

Mikrodenetleyici Eğitim Seti

Kullanım ve Uygulama Kitabı

Eyüp ÖZAVCI

ÖNSÖZ

Günümüzde farkında olmadan kullandığımız çevremizdeki araçların pek çoğunda akıllı yongalar dediğimiz mikroişlemciler ve/veya mikrodenetleyiciler kullanılmaktadır. İlk bakışta sayabileceğimiz cihazlar: otomobillerin kontrol panelleri, fax makineleri, bilgisayarlar, cep telefonları, yazıcılar, saatler, müzik setleri, fotoğraf makinaları ve daha pek çok cihaz. Aslında çevremizde bu akıllı yongaların kullanılmadığı cihazlar bulmak çok daha zordur. Mikrodenetleyiciler tek yonga üzerinde pek çok işlevi, birimi (RAM, flash bellek, EPROM, ADC, DAC, Seri iletişim birimleri: UART, I2C, zamanlayıcı, USB, Ethernet ...) barındıran işlem birimleridir. Bu akıllı yongaların tasarımlarda kullanılabilmesi ve bunların programlarının geliştirilebilmesi için ihtiyaç duyulan en önemli şey bu yongalara ait Geliştirme Setleridir. Bu setler kullanıcıya ihtiyaç duyduğu bütün fonksiyonları ve işlevleri sağlayan donanımlar ve bunlarla ilgili dökümantasyon ile birlikte çeşitli yazılımların bulunduğu bir tasarım ortamıdır. Bu tür setlerin çoğunlukla yurt dışından gelmesi nedeniyle Türkçe desteği bulunmamakta ve yerli özgün ürünlerin ihtiyacı hissedilmektedir. Şirketimiz OGEM bu ihtiyacı karşılayacak ürün yelpazesini sunmayı ve bu ürünleri çeşitli türkçe dokümanlar ve projelerle desteklemeyi hedeflemektedir. İlk ürünümüz **EA-80C552** Mikrodenetleyici Eğitim Seti, **MCS-51** ailesine ait **8051** mikrodenetleyicisi tabanlı bir eğitim setidir ve bu konudaki çalışmaları hedef almaktadır. Diğer ürünümüz **EA-16F877** Mikrodenetleyici Eğitim Seti ile ürün yelpazesine Microchip Firmasının üretimi olan PIC (**P**eripheral **I**nterface **C**ontroller) mikrodenetleyiciler ailesini de katmış olduk. Şu anda okumakta olduğunuz Kullanım ve Deney Kitabı ürünümüz PIC ailesinden **PIC16F877** mikrodenetleyisini kullanmakta olan **EA-16F877** Mikrodenetleyici Eğitim Setine aittir. Tamamen kendi tasarımımız olan bu ürün ile PIC mikrodenetleyiciler ailesine yönelik eğitim seti eksiliğini büyük ölçüde gidereceğini ve Memleketimizin teknik eğitim ve öğretimine katkıda bulunacağına inanıyoruz. Görevimiz kaliteli ürünlerle eğitim dünyasına katkıda bulunmaktır.

Saygılarımızla,
Eyüp ÖZAVCI
OGEM Otomasyon Gereçleri ve Elektromekanik San. Tic. Ltd. Şti.
ODTÜ-KOSGEB Teknoloji Geliştirme Merkezi
No:403 ODTÜ, ANKARA

Eylül 2003

EA-16F877 Mikrodenetleyici Eğitim Seti

KULLANIM ve DENEY KİTABI

İÇİNDEKİLER **Sayfa**

ÖNSÖZ i

BÖLÜM 1
MİKRODENETLEYİCİLER VE PIC

EA-16F877 Microdenetleyici Eğitim Seti.....
1.1. ..... 1.1
1.2. EA-16F877 Temel Set..... 1.1
1.3. Deney Kartları ve İşlevleri..... 1.3
1.4. PIC16f877 Programlama Yazılımı..... 1.4

BÖLÜM 2
16F877 MİKRODENETLEYİCİSİNİN ÖZELLİKLERİ

2.1. Genel Tanımlar..... 2.1
2.2. PIC Mikrodenetleyici Özellikleri..... 2.2
2.3. PIC Mikrokontrolcülerinin Donanımsal İncelenmesi..... 2.3
2.3.1 PIC Mikrodenetleyicilerin İç Yapısı..... 2.3
2.3.2 Genel Tanımlama..... 2.5
2.3.3 Gelişme Desteği..... 2.7
2.3.4 Elektrikle Silinebilen Mikrodenetleyiciler..... 2.7
2.4 Mimari Olarak İncelenmesi..... 2.7
2.5 Komut Akımı / Bilgi İletimi..... 2.7
2.6 Bellek Organizasyonu..... 2.8
2.7 Veri Bellek Organizasyonu..... 2.8
2.8 Genel Amaçlı Kayıt Dosyası (GPR)..... 2.10
2.8.1 Özel Fonksiyon Kayıtçıları..... 2.10
2.9 PIC Mikrodenetleyicilerinde Yazılım (Komutlar)..... 2.12

BÖLÜM 3

MICROCHIP - MPLAB KULLANIMI

3.1.	MPLAB'ın Kurulması ve Başlatılması.....	3.1
3.2.	MPLAB'ın Kullanılması.....	3.1
3.3.	Assembly Programının Yazılımı.....	3.2
3.4.	MPSIM-PIC Simulatörü.....	3.7

BÖLÜM 4
DENEYLER

DENEY 1.	Mikrodenetleyici sisteminden veri çıkışı.....	4.1
DENEY 2.	Ledli yürüyen ışık uygulaması.....	4.6
DENEY 3.	İki yönlü yürüyen ışık uygulaması.....	4.10
DENEY 4.	Mikrodenetleyici sistemine veri girişi ve veri çıkışı.....	4.14
DENEY 5.	Dijital Saat Uygulaması.....	4.25
DENEY 6.	4X4'lük Tuş Takımından Girilen Harflerin Görüntülenmesi.....	4.39
DENEY 7.	4-İşlem Hesap Makinası.....	4.53
DENEY 8.	Optik Asansör Simülasyonu.....	4.69
DENEY 9.	Adımlı Motor Arabirimi ve Kontrolu Uygulaması.....	4.77
DENEY 10.	Analog-Dijital (A/D) Konvertör Uygulaması.....	4.88

EKLER:

EK A.	RS-232 Kablo Yapısı ve Devre Şemaları.....	A.1
	Komut Listesi.....	A.7

BÖLÜM 1.

1.1 EA-16F877 Microdenetleyici Eğitim Seti :

EA-16F877 Mikrodenetleyici Eğitim Setimiz donanım ve yazılım altyapısı, cihazın kullanımını ve uygulamalarını anlatan kitabı ile tüm ihtiyaçları karşılamaktadır. Sahip olduğu özellikleri ile setimiz öğrenci veya kişisel olarak teknik uygulama yapmak isteyenler için çok faydalı bir eğitim aracıdır. Gömülü yonga kullanılarak tasarlanan elektronik uygulamalarında kişinin yaratıcı ve üretici özelliklerine katkıda bulunur.

Eğitim setimiz aşağıdaki birimlerden oluşmaktadır:

- EA-16F877 Temel Set (Ana Kart ve Güç ünitesinden oluşan)
- Display ve Led Deney Modülü
- Asansör Deney Modülü
- Adımlı Motor Deney Modülü
- Sıcaklık Kontrol Deney Modülü
- RS-232 Arabirim Kablosu
- Deney Kitabı
- Yardımcı Programlar CD'si. Çeşitli yardımcı Programlar bulundurmaktadır.

1.2 EA-16F877 Temel Set:

EA-16F877 Temel Set, üzerindeki özel tasarım programlayıcısı ile PIC16F877 ve 877A mikrodenetleyicisini programlamaktadır. Flash memory teknolojisi ile üretilen bu mikrodenetleyicilerin belleğine yüklenen program bellğin kalıcı olmasından dolayı uygulanan enerji kesilse bile silinmez. Flash bellekler bu özellikleri ile EEPROM belleklere benzemektedir.

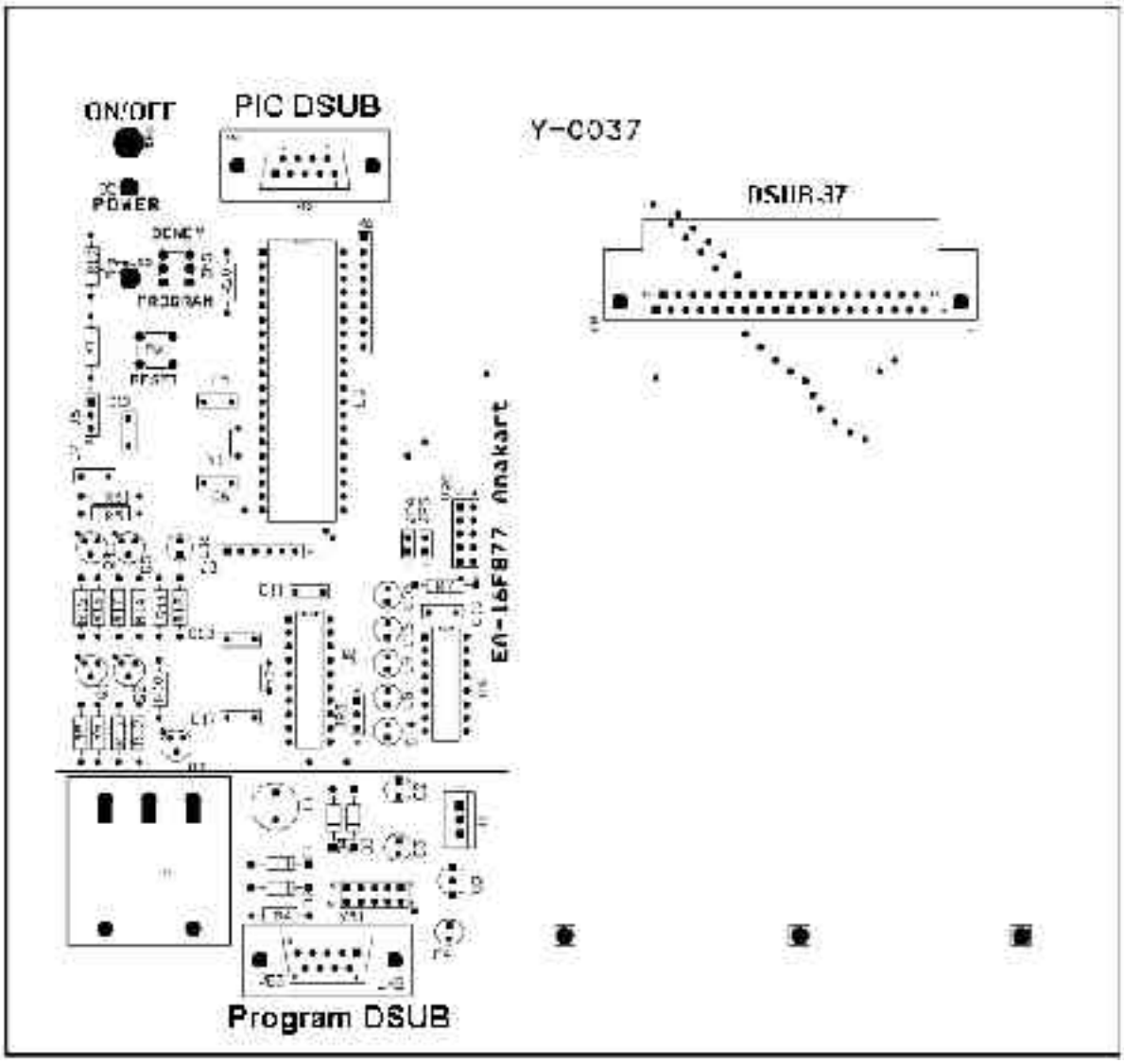
Flash belleği olan PIC16F877'nin ve benzer ürünlerin en önemli özelliği belleğin en az 1000 defa silinip programlanabilmeleridir. Biz bu özelliğini kullanarak ilgili deney programlarımızı üzerine programlayıp deneyeceğiz ve hatalarımızı bulup düzelttikten sonra belleği silip tekrar programlayacağız. Bu işlem bizim deneylerimizi düzgün çalıştırana kadar devam edecektir. Bu özellik sayesinde aynı yongayı bütün deneylerimizde kullanabileceğiz. Yonganın sahip olduğu bu bellek sayesinde setimiz aynı zamanda oldukça yüksek performanslı ucuz bir geliştirme setidir.

Setimiz RS232 arabirimine sahip her türlü PC ile çalışabilme özelliğine sahiptir. PC ile haberleşmeyi RS-232C seri port üzerinden yapmaktadır.

Anakart üzerinde +5V ve +12V DC gerilimler üreten bir besleme katı vardır. Bu kat programlayıcının ve deney kartlarının ihtiyaç duyduğu besleme gerilimlerini sağlar. Ayrıca ilave olarak PC ile haberleşmesini ve aktarılan programı PIC16F877'nin belleğine programlama işlemini gerçekleştiren AT89C2051

mikrodenetleyicisinden ve MAX232 entegresinden oluşan bir programlama katı bulunmaktadır. Kart üzerinde PIC16F877 mikrodenetleyicisi ile PC ile haberleşme için programlama katına ait bir Dişi DSUB-9 ile PIC16F877 mikrodenetleyicisinin haberleşmesinde kullanılan dişi DSUB-9 konnektörleri ve Deney kartlarının takıldığı dişi DSUB-37 konnektörü bulunmaktadır.

EA-16F877 Temel Eğitim Setinin Üstündeki Led ve Tuşların Görevleri:



Şekil 1.1. EA-16F877 Yerleşim planı

RESET butonu: PIC16F877'nin 4 nolu bacağı ile toprak arasına bağlanan bir push butondur. Butona basıp bıraktığımızda ucu toprak seviyesine çekilip tekrar VCC seviyesine getirilerek programın ORG 00H adresinden (RESET vektöründen) tekrar başlamasını sağlar.

PROGRAM/DENEY Anahtarı: Programlayıcıya yaptırmak istediğimiz işi seçmemiz için kullanılır. PIC16F877'ye program yüklerken program konumuna, deney yaparken de deney konumuna alırız.

PROGRAM Ledi: Yandığında setin program konumunda olduğunu ve PIC16F877'nin programlanmakta olduğunu gösterir. Kırmızı renkli leddir.

POWER Ledi: Temel sete enerji geldiğini gösterir. Cihazın kullanıma hazır olduğunu gösterir. Yeşil renkli leddir.

ON/OFF Toggle Anahtarı: Eğitim setine gelen gücü açıp kapamak için kullanılır.

Program D-sub 9'lu: Mikrodenetleyicinin programlanması esnasında RS-232C arabirim kablosunun takıldığı konnektörüdür.

PIC D-sub 9'lu: Mikrodenetleyicinin PC ile haberleşmesi (Deneyler esnasında) için RS-232C arabirim kablosunun takıldığı konnektörüdür.

D-sub 37'li: Deney kartlarının takıldığı konnektörüdür.

1.3 DENEY KARTLARI ve İŞLEVLERİ:

Tuş Takımı Ve Görüntüleme Kartı

- Uygulamanın özelliğine göre tanımlanabilen çok amaçlı 4x4'lük tuş takımı.
- 4 adet yedi parçalı LED gösterge.
- Basit Giriş/Çıkış uygulamaları için 8 det 3mm led.
- Kartın üst tarafındaki jumper modülün birden fazla uygulamalarda kullanılması içindir. Led uygulamalarında led, tuş ve yedi parçalı gösterge uygulamalarında ise tuş konumuna alınır.

Asansör Deney Modülü

- Bu kart bir mekanik asansörün benzetimi yapılarak asansörün işleyişini göstermek amacıyla kullanılmıştır.
- Kartın sol tarafında kat butonları bulunmaktadır.
- Kartın sağ tarafında ise yukarı-aşağı ledleri ve kat arası ledleri bulunmaktadır.
- Asansör yukarı çıktığında yukarı ledleri, aşağı indiğinde aşağı ledleri ve kat arasında olduğunda da kat arası ledleri yanar.

Adımlı Motor Deney Modülü

- Bir adımlı motorun dönme yönü ve dönüş hızının kontrolü içindir.
- Kart üzerindeki artır butonlarıyla gireceğimiz sayı değeri adımlı motorun hızını ayarlar.
- Başla/ Dur butonuyla adımlı motoru çalıştırırız.
- Yön değiştir butonuyla da motorun dönüş yönünü değiştiririz.

Sıcaklık Kontrol Deney Modülü

- Bir ısıtıcının kontrolü için tasarlanmıştır.
- Girdiğimiz 0-99 °C arası bir sıcaklık değeri ile ortamın o anki sıcaklık değeri karşılaştırılır.
- Karşılaştırma sonucu ortamın ısınması gerekiyorsa ısıtıcı çalışır.
- Eğer ortamın sıcaklığı istenen değerde ise ısıtıcı çalışmaz.
- Kart üzerindeki led ısıtıcının yerine kullanılmıştır.
- Butonlardan biri artırma işlemini yapar. Diğeri de ölçülen ortam sıcaklığının değerini gösterir. Butona basıldığında o anki ortam sıcaklığı görüntülenir, bırakıldığında ise yine ayarlanan sıcaklık değeri görüntülenir.

1.4 PIC16F877 PROGRAMLAMA YAZILIMI

Bu bölümde tanımlanan bir programın yazılıp derlenerek makine diline çevrilmesi ve bu hazırlanan kodun EA-16F877 Temel Seti Programlayıcısı kullanılarak PIC'e aktarılması aşamalarında kullanılan işlemler üzerinde durulacaktır.

Programın Yazılması:

Mikrodenetleyicinin çalışmasını ve işletilmesini sağlayan bilgi programın yazılımıdır. Başarılı bir uygulama için yazılım hatasız (bug) olmalıdır. Yazılım C, Pascal veya Assembly gibi çeşitli dillerde veya ikilik (binary) kod olarak yazılabilir. Biz burada yazılımı PIC16F877'nin komutlarını kullanarak, assembly dilinde yazılım geliştirme yapmaktayız.

PIC'ler RISC mimarisi kullanılarak üretilmişlerdir. Bunun amacı PIC'i programlamak için kullanılacak komutların basit ve sayıca azaltılmış olmasıdır. PIC16F877 mikrodenetleyicisi toplam 35 komut kullanılarak programlanmaktadır.

Programın Derlenmesi:

Assembler, bir text editöründe assembly dili kurallarına göre yazılmış olan komutları PIC'in anlayabileceği hexadecimal kodlara çeviren programdır. Microchip firmasının hazırladığı MPASM bu işi yapan assembler programdır. Assembler'e çoğu zaman compiler da denir ancak assembler bir dönüştürücü programdır.

Tüm programlama dillerinde olduğu gibi assembly dili programlarını yazmadan önce akış şemaları çizmek iyi bir alışkanlıktır. Uzun ve karmaşık mantık işlemlerinin yoğun olduğu programları yazarken direkt olarak komutları yazmaya başlamak kısa bir süre sonra içinden çıkılmaz hale gelebilir. Bu nedenle programları yazmadan önce akış şemaları çizerek, komutların hangi sıraya göre yazılacağını görmek için görsel bir düşünme ortamı oluşturulur. Akış şeması çizildikten sonra işlemleri gerçekleştirecek olan assembly komutları yazılır. PIC programı hazırlanmış olur. Hazırladığımız bu programı bir edit, text veya MPLAB programında yazıp .asm uzantılı olarak kaydederiz. Sonra MPASM'de programı derleriz. Programda hatalar varsa onları temizleyip programı tekrar derleriz. (MPLAB programının kullanılması daha sonraki bölümlerde ayrıntılı olarak açıklanmıştır.)

Programın PIC'e Aktarılması:

Hatalardan arındırdığımız programı derleyip makine kodlarını oluştururuz. Makine kodlarını içeren dosya .hex uzantılı dosyadır. Bu dosyayı PIC16F877'ye aktarmak için PC üzerinde çalışan PicProgramlayıcısı yazılımı kullanılır.

Programlamaya başlamadan önce EA-16F877 Temel Eğitim Setinin güç kablosu prize takılır. RS-232C arabirim kablosu PC'nin COM1 portuna bağlanır. ON/OFF anahtarı ON konumuna alınır. Program/Deney anahtarı Program konumuna alınır. Bu durumda Setin programlayıcı birimi PIC'i programlamak için hazırdır. PC'den PicProgramlayıcısı yazılımı aşağıdaki gibi kullanılarak programın makine kodları(.hex dosyası) PIC'e aktarılır.

PicProgramlayıcısı:

Bu program yazdığımız programların makine kodlarını PIC'e aktarır. Kullanımı için aşağıdaki basamaklar izlenir.

- Konfigürasyon bitlerini ayarlarız. Ancak configürasyon bitlerinin tanımları Program içinde verilmişse, ki bunu programın taşınabilirliği açısından tercih ederiz, bu ayarlar dosyanın okunması esnasında otomatik yapılır. Yapılmamış ise ve yapılması gerekiyorsa PIC için hangi osilatörü kullanıyorsak onu seçeriz.

- Dosya ikonunun altından Aç seçeneğini tıklayarak yüklemek istediğimiz dosyayı açarız. (yalnızca .hex uzantılı dosyalar içindir.)

- Üstteki dosya bölümünde yüklemek istediğimiz programın ismi ve yeri görünmektedir. Yükleyeceğimiz dosyayı belirtir.

- Programla ikonuna tıklayarak programı PIC'e aktarırız. Eğer PIC belleğinde bilgi varsa önce PIC'in belleğini temizleriz. Bu işlemi de Sil(Yonga) ikonuna tıklayarak yaparız. Daha sonra PIC'i programlarız.

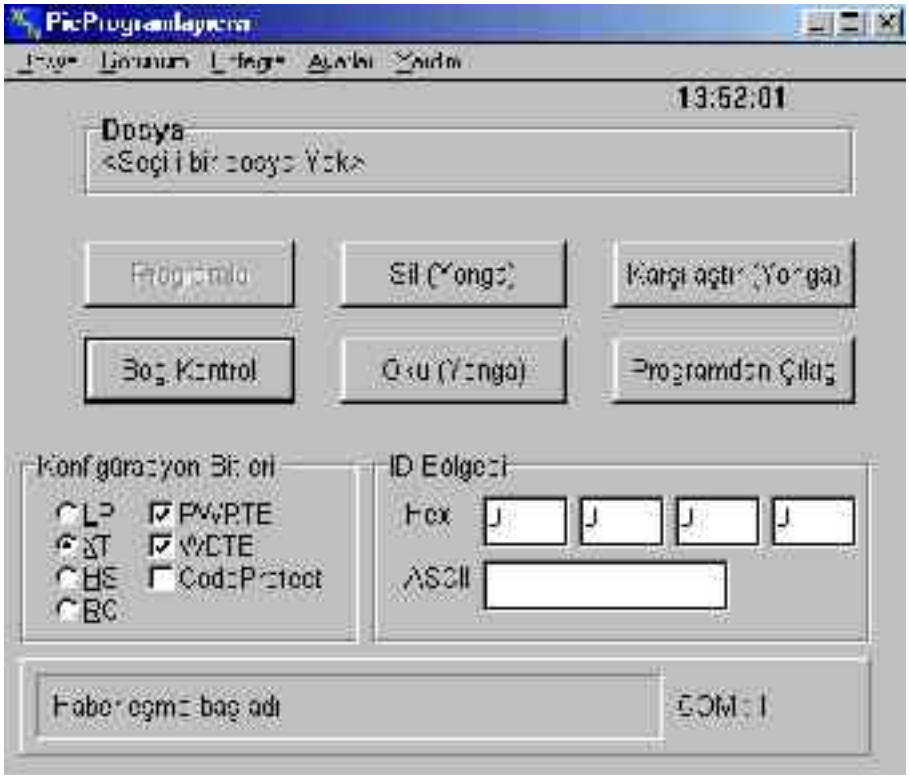
- Programlama işlemi 0000 adresinden başlar ve 2040'a geldiğinde biter. Bu durumda programlama işlemi de tamamlanmıştır. "Programlama işlemi bitmiştir..." mesajını verir.

- Görünüm düğmesinin altında Pic Belleği seçeneğini seçerek PIC16F877'nin programlanması için tampon belleğe aldığımız program içeriğini okuyabiliriz. Oku (Yonga) ikonuna tıklayarak PIC belleğinin Bilgisayarın tampon belleğine okunması, aktarılmasıda işlemini yaparız.

- EA-16F877 Temel Seti hangi seri porta bağlanacağını seçmemiz için Seri Port Seçim Formu vardır. Setimizi hangi porta bağladıysak o seri portu seçmemiz gerekir.

- Karşılaştır(Yonga) ikonu PIC belleği ile Bilgisayardaki tampon belleğe yüklediğimiz programın makine kodlarını karşılaştırır.

Program PIC'e yüklendikten sonra EA-16F877 Temel Set üzerindeki DSUB-37 konnektörüne deneye ait uygun kart takılır. Program/Deney anahtarı Deney konumuna alınır ve deneyin işlem basamakları takip edilir.



$\overline{MCLR}$ $\overline{MCLR}$

BÖLÜM 2.

16F877 MİKRODENETLEYİCİSİNİN ÖZELLİKLERİ

2.1. GENEL TANIMLAR :

- **G/Ç (Giriş / Çıkış) :** Mikrodenetleyicinin dış dünya ile ilişkisini sağlayan, giriş veya çıkış olarak ayarlanabilen bir bağlantı pinidir. G/Ç noktası mikrodenetleyicinin dış dünyayla iletişim kurmasına, dış dünyayı kontrol etmesine veya dış dünyadan bilgi almasına izin verir.

- **Yazılım :** Mikrodenetleyicinin çalışmasını ve işlevin yerine getirilmesini sağlayan komutlar kümesidir. Başarılı bir uygulama için yazılım hatasız (bug) olmalıdır. Yazılım C, Pascal veya Assembler gibi çeşitli dillerde veya ikilik (binary) olarak yazılabilir.

- **Donanım :** Mikrodenetleyici; bellek, arabirim bileşenleri, güç kaynakları, sinyal düzenleyici devreler ve bunları çalıştırmak ve arabirim görevini üstlenmek için bu cihazlara bağlanan tüm bileşenlerdir.

- **Simülatör :** PC üzerinde çalışan ve mikrodenetleyicinin içindeki işlemleri simüle eden MPSIM gibi bir yazılım paketidir. Hangi olayların ne zaman meydana geldiği biliniyorsa bir simülatör kullanmak tasarımları test etmek için kolay bir yol olacaktır. Öte yandan simülatör, programları tümüyle veya adım adım izleyerek hatalardan arındırma fırsatı sunar. Şu anda en gelişmiş simülatör programı Microchip firmasının geliştirdiği MPLAB programıdır.

- **ICE :** PIC MASTER olarak da adlandırılır (In-Circuit Emulator / İç devre takipçisi). PC ve mikrodenetleyicinin yer alacağı soket arasına bağlanmış yararlı bir gereçtir. Bu gereç yazılım, PC de çalışırken devre kartı üzerinde bir mikrodenetleyici gibi davranır. ICE, bir programa girilmesini, mikro içinde neler olduğunu ve dış dünyayla nasıl iletişim kurulduğunun izlenilmesini mümkün kılar.

- **Programlayıcı :** Yazılımın mikrodenetleyici üzerindeki kalıcı belleğe kaydedilmesini ve böylece ICE' nin yardımı olmadan çalışmasını sağlayan bir birimdir. Çoğunlukla seri port'a (örneğin PICSTART, PROMASTER, EA-16F877...) bağlanan bu birimler çok çeşitli biçim, ebat ve fiyatlara sahiptir.

- **Kaynak Dosyası :** Hem asembler'in hem de tasarımcının anlayabileceği dilde yazılmış bir programdır. Kaynak dosya mikrodenetleyicinin kalıcı belleğine aktarılmadan önce assemble edilmiş, ikili sayı sisteminde makine diline çevrilmiş olmalıdır.

- **Assembler :** Kaynak dosyayı bir nesne dosyaya dönüştüren yazılım paketidir. Hata araştırma bu paketin yerleşik bir özelliğidir. Bu özellik assemble edilme sürecinde hatalar çıktıkça programı hatalardan arındırırken kullanılır. MPASM, tüm PIC ailesini elinde tutan Microchip'in son assembler'idir.

- **Nesne dosyası (object file) :** Assembler tarafından üretilen bu dosya; programcı, simülatör veya ICE'nin anlayabilecekleri dosyadır. Dosya uzantısı assemble edicinin emirlerine bağlı olarak , .OBJ veya .HEX olur.

2.2 PIC MİKRODENETLEYİCİ ÖZELLİKLERİ

- **Güvenirlik:** PIC komutları RISC mimarisinden dolayı bellekte çok az yer kaplarlar. PIC ailesinde bu komutlar 12 (16C52), 14 (16F84) veya 16(17F, yada 18F serisi) bit'ten oluşan bir word'lük sembollerdir. Harvard mimarisi kullanılmayan mikrodenetleyicilerde (CISC mimarisinde) komut sabit bir uzunlukta (8, 16, 32) olup sadece işlevi tanımlamaktadır. İşlem için gerekli veriler komutu takip eden program belleği alanındadır. Gerekli bu verilerin okunarak işlemin tamamlanması gerektiğinden her komut işlem süresinde farklılıklar olmaktadır ve bu aynı zamanda işlemi yavaşlatmaktadır.

- **Hız :** PIC oldukça hızlı bir mikrodenetleyicidir. Her bir komut döngüsü 1sn'dir. Örneğin 5 milyon komutluk bir programın 20Mhz'lik bir kristalle işletilmesi yalnız 1sn sürer. Bu süre 386SX33 hızının yaklaşık 2 katıdır. Ayrıca RISC mimarisi işlemcisi olmasının hıza etkisi oldukça büyüktür.

- **Komut seti :** PIC'in 16C5X ailesinde bir yazılım yapmak için 33 komuta ihtiyaç duyarken 16CXX araçları için bu sayı 35'tir. PIC tarafından kullanılan komutların hepsi

kaydedici (register) temellidir. Komutlar 16C5X ailesinde 12 bit, 16CXX ailesindeyse 14 bit uzunluğundadır. PIC'te CALL, GOTO, bit test eden BTFSS ve INCFSZ gibi komutlar dışında diğer komutlar 1 saykıl çeker. Belirtilen komutlar ise adres değişimi durumunda 2 saykıl, bir sonraki adresten devam ediyorsa 1 saykıl çeker.

- **Statik İşlem :** PIC tamamıyla statik bir işlemcidir. Yani saat durdurulduğunda da tüm yazmaç içeriği korunur. Pratikte bunu tam olarak gerçekleştirebilmek mümkün değildir.

PIC mikrodenetleyicisi programı işletilmediği zaman uyuma (sleep) moduna geçirilerek mikrodenetleyicinin çok düşük akım çekmesi sağlanır. PIC uyuma moduna geçirildiğinde, saat durur ve PIC uyuma işleminden önce hangi durumda olduğunu çeşitli bayraklarla ifade eder (elde bayrağı, 0 (zero) bayrağı ... vb.). PIC uyuma modunda 1μA'den küçük değerlerde akım çeker (Standby akımı).

- **Sürme özelliği (Sürücü kapasitesi) :** PIC yüksek bir çıktı kapasitesine sahiptir. Tek bacaktan 25mA akım çekebilmekte ve entegre toplamı olarak 150mA akım akıtma kapasitesine sahiptir. Entegrenin 4MHz osilatör frekansında çektiği akım çalışırken 2mA, stand-by durumunda ise 2μA kadardır.

- **Seçenekler :** PIC ailesinde her türlü ihtiyaçların karşılanacağı çeşitli hız, sıcaklık, kılıf, I/O hatları, zamanlama (Timer) fonksiyonları, seri iletişim portları, A/D ve bellek kapasite seçenekleri bulunur.

- **Çok yönlülük :** PIC çok yönlü bir mikrodenetleyicidir ve ürünün içinde, yer darlığı durumunda birkaç mantık kapısının yerini değiştirmek için düşük maliyetli bir çözüm bulunur.

- **Güvenlik :** PIC endüstride en üstünler arasında yer alan bir kod koruma özelliğine sahiptir. Koruma bit'inin programlanmasından itibaren, program belleğinin içeriği, program kodunun yeniden yapılandırılmasına olanak verecek şekilde okunmaz.

- **Geliştirme :** PIC program geliştirme amacıyla programlanabilip tekrar silinebilme özelliğine sahiptir (EPROM, EEPROM). Aynı zamanda seri üretim amacıyla bir kere programlanabilir (OTP) özelliğine sahip PIC'lerde vardır.

- **Liste dosyası :** Assembler tarafından oluşturulan ve kaynak dosyadaki tüm komutları hexadecimal sistemdeki değerleri ve tasarımcının yazmış olduğu yorumlarıyla birlikte içeren bir dosyadır. Bir programı hatalardan arındırırken araştırılacak en yararlı dosya budur. Çünkü bu dosyayı izleyerek yazılımlarda neler olup bittiğini anlama şansı kaynak dosyasından daha fazladır. Dosya uzantısı .LST dir.

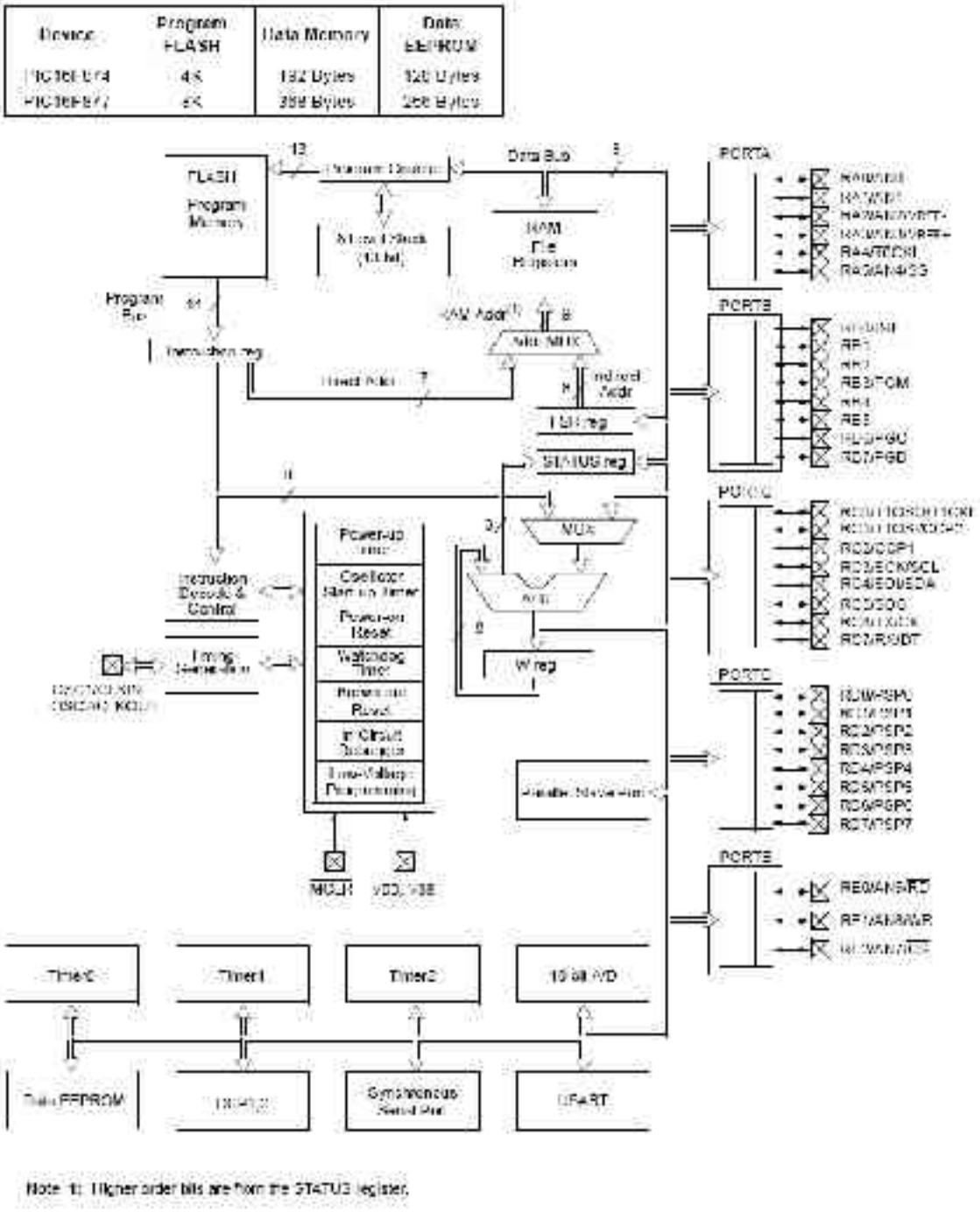
- **Diğer dosyalar :** Hata dosyası (Error file : uzantısı .ERR) hataların bir listesini içerir ancak bunların kaynağı hakkında hiç bir bilgi vermez. Uzantısı .COD olan dosyalar emülatör tarafından kullanılırlar.

- **Hatalar (Bug) :** Tasarımcının farkında olmadan yaptığı hatalardır. Bu hatalar; basit yazılım hatalarından, yazılım dilinin yanlış kullanımına kadar uzanır. Hataların çoğu derleyici tarafından bulunur ve bir .LST dosyasında görüntülenir. Kalan hataları bulmak ve düzeltmekte programcıya düşer.

2.3 PIC MİKROKONTROLCÜLERİNİN DONANIMSAL İNCELENMESİ

2.3.1 PIC MİKRODENETLEYİCİLERİN İÇ YAPISI

CPU bölgesinin kalbi ALU'dur (Aritmetic Logic Unit-Aritmetik mantık birimi). ALU, W (Working-Çalışan) adında bir kaydedici içerir. PIC, diğer mikroişlemcilerden, aritmetik ve mantık işlemleri için bir tek kaydediciye sahip oluşuyla farklılaşır. W kaydedicisi 8 bit genişliğindedir ve CPU'daki herhangi bir veriyi transfer etmek üzere kullanılır.



Şekil 2.1 : PIC16F877 blok diyagramı

CPU alanında ayrıca iki kategoriye ayırabileceğimiz Veri kaydedici dosyaları (Data Regıster Files) bulunur. Bu veri kaydedici dosyaları; I/O ve kontrol şeklinde çalışanlar ve tamamen RAM gibi çalışanlardır.

PIC'ler de Harvard Mimarisi kullanılır. Harvard mimarisi, mikrodenetleyicilerde veri akış miktarını hızlandırmak ve yazılım güvenliğini arttırmak amacıyla kullanılır. Ayrı bus'ların kullanımıyla veri ve program belleğinde hızlı erişim sağlanır.

PIC mikrodenetleyicilerini donanımsal olarak incelerken PIC16F877 üzerinde durarak bu PIC'i temel alıp donanım incelenecektir. Bellek ve bazı küçük farklılıklar dışında burada anlatılanlar bütün PIC'ler için geçerlidir.

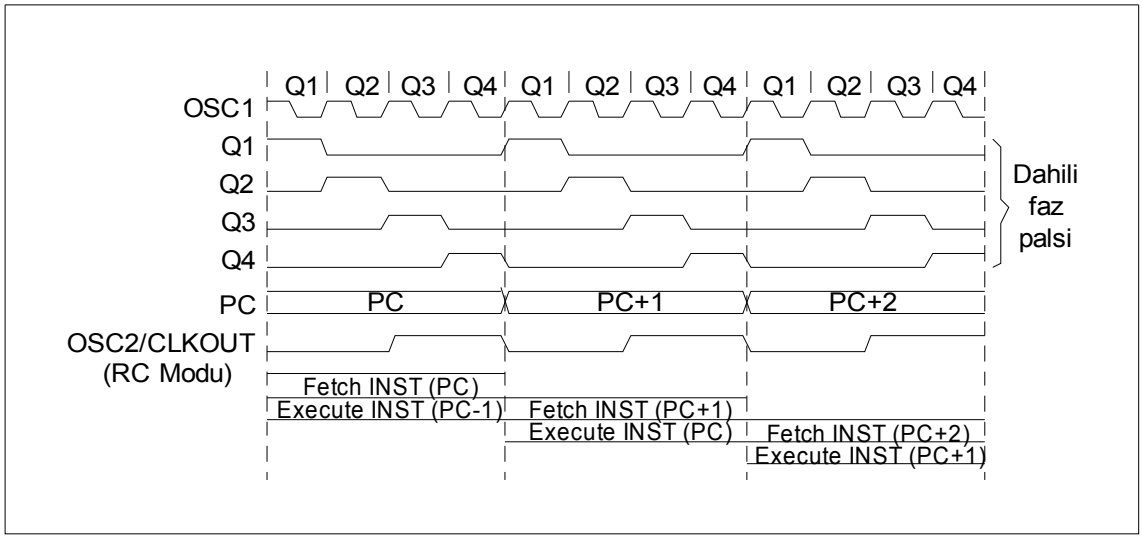
2.3.2 GENEL TANIMLAMA

PIC16F877 düşük maliyetli, yüksek performanslı, CMOS, full-statik, 8 bit mikrodenetleyicidir. Tüm PIC mikrodenetleyicileri RISC mimarisini kullanmaktadır. Harvard Mimarisinin ayrı komut ve veri taşıyıcısıyla ayrı 8 bitlik geniş veri taşıyıcılı, 14 bitlik geniş komut kelimesine imkan vermektedir

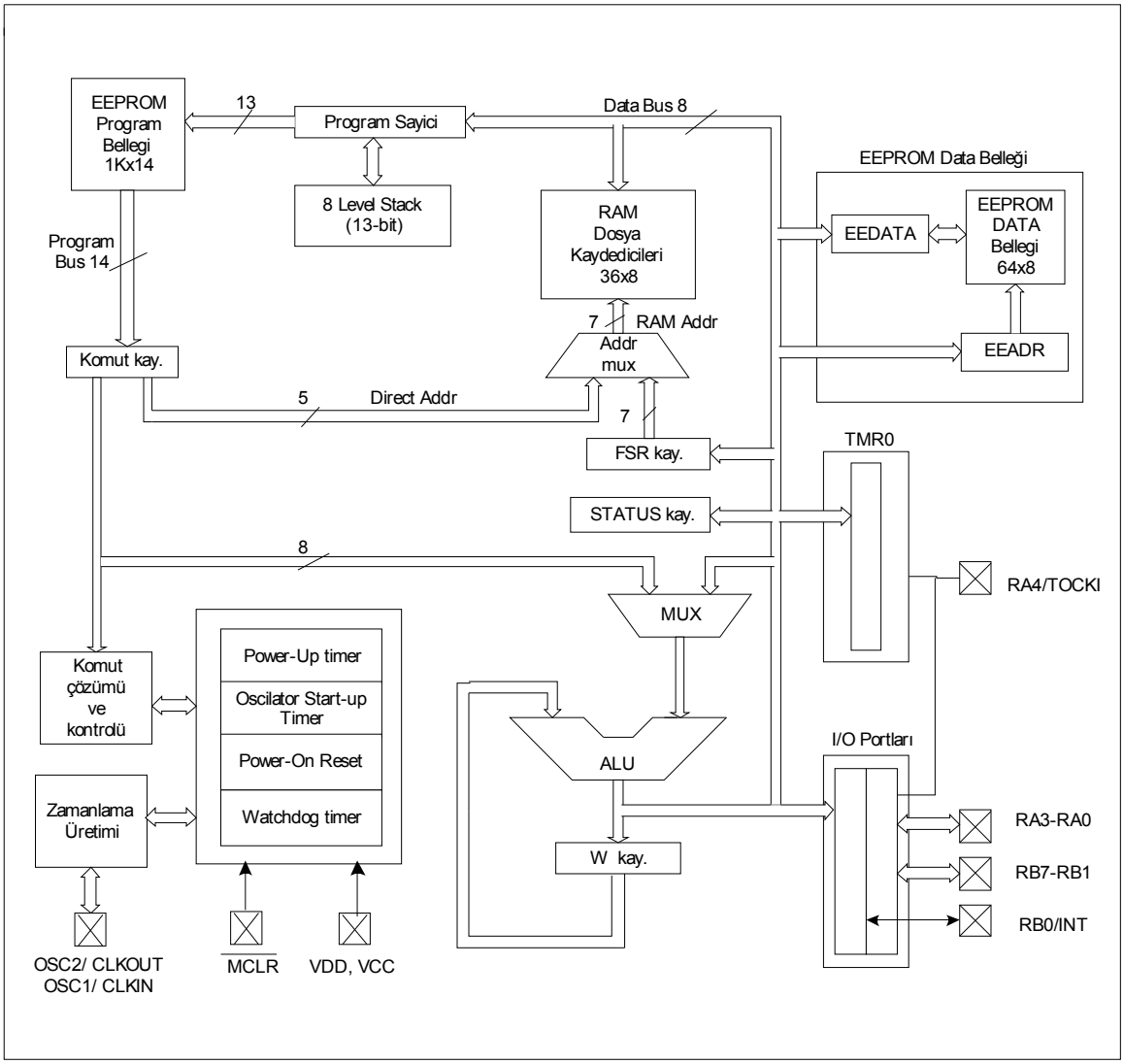
PIC16F877 microchip'i 368 bitlik veri belleğine, 256 bayt EEPROM belleğine ve A,B,C,D,E portlarına sahiptir. Bunun yanı sıra üzerinde timer ve sayaç ile Seri haberleşme, PWM, Paralel haberleşme ve 10-Bitlik ADC birimleri mevcuttur.

PIC ailesi dış elemanları azaltacak kendine has özelliklere sahiptir ve böylece maliyet minimuma inmekte, sistemin güvenirliği artmakta, enerji sarfiyatı azalmaktadır. Bunun yanı sıra tüm PIC ler de 4 adet osilatör seçeneği mevcuttur. Bunlarda tek pin li RC osilatör, düşük maliyet çözümünü sağlamakta (4 MHZ); LP osilatör (Kristal veya seramik rezonatör), enerji sarfiyatını minimize etmekte (asgari akım)(40 KHZ); XT kristal veya seramik rezonatör osilatörü standart hızlı ve HS kristal veya seramik rezonatörlü osilatör çok yüksek hıza sahiptir (20 MHZ).

PIC mikrodenetleyicileri sleep modla kullanılmaktadır. Bu mod ile PIC işlem yapılmadığı durumlarda uyuma moduna geçerek çok düşük akım çeker (5μA). Kullanıcı bir kaç iç ve dış kesmelerle PIC'i uyuma modundan çıkarabilmektedir. Yüksek güvenilirlikli Watchdog Timer kendi bünyesindeki chip üstü RC osilatörü ile yazılımı kilitlemeye karşı



korumaktadır.



	PIC16F873	PIC16F874	PIC16F876	PIC16F877
Maksimum İşlemci Frekansı (MHz)	DC-20	DC-20	DC-20	DC-20
Reset (ve Gecikmeler)	POR, BOR (PWRT,OST)	POR, BOR (PWRT,OST)	POR, BOR (PWRT,OST)	POR, BOR (PWRT,OST)
Flash Program Belleği (14-Bit)	4K	4K	8K	8K
Data Belleği (Bayt)	192	192	368	368
Data EEPROM (Bayt)	128	128	256	256
Timer Modül	3	3	3	3
Interrupt Kaynakları	13	14	13	14
I/O Portları	A,B,C	A,B,C,D,E	A,B,C	A,B,C,D,E
Yakala/Kıyasla/PWM modülleri	2	2	2	2
Seri Haberleşme kanalları	MSSP,USART	MSSP,USART	MSSP,USART	MSSP,USART
Paralel Haberleşme	-	PSP	-	PSP
10-bit Analog/Sayısal Modülü	5 giriş kanalı	8 giriş kanalı	5 giriş kanalı	8 giriş kanalı

Komut Seti	35 komut	35 komut	35 komut	35 komut

Tablo 2.1 : PIC16F87X ailesi özellikleri

PIC16F877 FLASH program belleği, aynı aygıt paketinin orijinali ve üretimi için kullanılmasına olanak vermektedir. Yeniden programlanabilirliği mikroyu uygulamanın sonundan kaldırmadan kodu güncelleştirmeye izin vermektedir.

Bu aygıtın kolayca erişilemediği, fakat prototipinin kod güncelleştirmesi gerekli olduğu durumlarda, bir çok uygulamanın geliştirilmesinde yararlıdır. Bunun yanı sıra bu kodun güncelleştirilmesi diğer ayrı uygulamalarda da yararlıdır.

Şekil 2.2 : PIC 16F84'ün basitleştirilmiş iç yapısı

PIC'ler özellikle de PIC16F84 yüksek hızlı otomobillerden, motor kontrolü uygulamaları, düşük enerji sarfiyatlı uzaktan çalışan sensörler, elektronik kilitler, güvenlik aygıtları ve akıllı kartlara kadar bir çok uygulamada kullanılırlar. EEPROM teknolojisi; Transmitter kodları, motor hızları, alıcı frekansları, güvenlik kodları vb. programların uygulamasını son derece hızlı ve uygun hale getirmektedir. Küçük boyutlarıyla bu mikrodenetleyiciler alan sınırlaması bulunan uygulamalarda kusursuzdur. Düşük maliyet, düşük enerji sarfiyatı, yüksek performans, kullanım kolaylığı ve I/O esnekliği özellikle de PIC 16F84 mikrodenetleyicisinin daha önce kullanımı hiç düşünülmeyen alanlarda kullanılmasını sağlamaktadır. Bunlar; timer fonksiyonları, seri komünikasyon, PWM fonksiyonları ve birlikte işlemci uygulamaları...

2.3.4 GELİŞME DESTEĞİ

PIC16CXX sınıfı tam özellikli mikroişlemci, yazılım simülatörü, devre içi emülatör, düşük maliyetli program geliştirme ve tam özellikli programlayıcı ile desteklenmiştir. PIC16F84, PIC16C5X mikrodenetleyicilerinin geliştirilmiş halidir. PIC16C5X için yapılan devrelerde kolaylıkla PIC16F84 kullanılabilir

2.3.5 ELEKTRİKLE SİLİNEBİLEN MİKRODENETLEYİCİLER

Bu mikrodenetleyiciler, silinip yeniden yazılabilme özelliğine sahiptir ve düşük maliyetli plastik ambalajlar halinde bulunmaktadır. Aynı zamanda bu tip mikrodenetleyiciler üretimi kadar prototipinin geliştirilmesi ve pilot programlar için kullanılma olanağı sağlamaktadır. Bunun daha ötesindeki avantajlarından biri, bunların devre içi veya Microchip's PICSTART® plus, PROMATE II veya EA-16F877 programlayıcıları tarafından silinebilmesi ve yeniden programlanabilmesidir.

2.4 MİMARİ OLARAK İNCELENMESİ

PIC16CXX sınıfı RISC mikroçiplerinde bulunan birçok mimari özelliklere sahiptir. Başlangıç olarak PIC16CXX Harvard mimarisini kullanmaktadır. Bu mimari ayrı belleklerden erişilen program ve verilere sahiptir. Programların ve veri belleklerinin ayrılması komutların 8 bitlik geniş veri kelimesinden farklı boyutlandırılmasına olanak vermektedir. PIC16CXX

mikrodenetleyicileri tekli kelimeye imkan veren 14 bit taşıyıcı üzerinden 14 bit komutu tek bir süreçte uygulamaktadır.

PIC16CXX mikrodenetleyicileri, kayıt dosyalarına ve veri belleğine doğrudan veya dolaylı olarak yönlenebilmektedir. Program sayacı dahil bütün özel fonksiyon kayıtları veri belleğine yerleştirilmiştir. Adres modunu kullanarak herhangi bir kaydın üstüne herhangi bir işlemin üstlenmesini mümkün kılan Ortogonal (simetrik) komutlarda kurulmuştur. Simetrik özelliği ve "özel optimal durumların" eksikliği PIC16CXX ile programlamayı daha da etkin kılmaktadır. İlaveten enformasyon eğrisi önemli ölçüde azaltılmıştır.

PIC16CXX mikroları 8 bitlik ALU'ya ve W(working) kaydedicisine sahiptir. W kaydedicisindeki veri ile herhangi bir dosya kaydedicisi arasında aritmetik ve boolean fonksiyonları uygulanmaktadır.

ALU 8 bit enindedir ve toplama, çıkarma, değiştirme ve çeşitli lojik işlemleri içerir. İki bilgili komutlarda (Subwf sayac, Sublw b'00011100' vb.) bir bilgi tipik olarak W kaydedicisidir, diğer bilgi ise dosya kaydedicisi (sayac) veya hazır sabit (b'00011100') değerdir. Tekli komutlarda bilgi ya W kaydedicisinde ya da dosya kaydedicisindedir.

Yürütülen komutlara dayanarak ALU, STATUS kaydedicisindeki Caryy(C), Digit Caryy(DC) ve Zero(Z) bitlerini etkileyebilmektedir. C ve DC bitleri, çıkarmalarda, nispeten çıkarma işleminde ödünç alan ve sayısal ödünç alan bit olarak işlemektedir.

2.5 KOMUT AKIMI / BİLGİ İLETİMİ

'Komut süreci' dört Q sürecinden oluşmaktadır. (Q1, Q2, Q3 ve Q4). Komut devri ve yürütülmesi şöyle iletilmektedir. Devir bir komut sürecini üstlenirken decode ve yürütme diğer komut sürecini üstlenmektedir. Bununla birlikte bilgi iletim nedeniyle, her bir komut etkin olarak bir süreçte yürütülür. Eğer komut program sayacının değişmesine neden olmuşsa (örn. GOTO komutu) o zaman komutun tamamlanması için iki süreç gereklidir.

Şekil 2.3: Saat palsi ve komut yürütme akımı

Devir süreci her Q1 de değeri bir artan program sayacı (PC) ile başlar. Yürütme sürecinde işleyen komut Q1 sürecindeki 'Komut kaydı' na gönderilir. Daha sonra bu komut Q2, Q3 ve Q4 süreçleri boyunca decode edilir ve yürütülür. Veri belleği Q2 boyunca okunur (Bilgi okunması) ve Q4 boyunca yazılır (Yazım hedefi).

2.6 BELLEK ORGANİZASYONU

PIC16F84'de iki bellek bloğu mevcuttur. Bunlar program belleği ve veri belleğidir. Her bir bellek kendi taşıyıcısına sahiptir; böylece her bir bloğa erişim aynı osilatör süreci boyunca meydana gelebilmektedir.

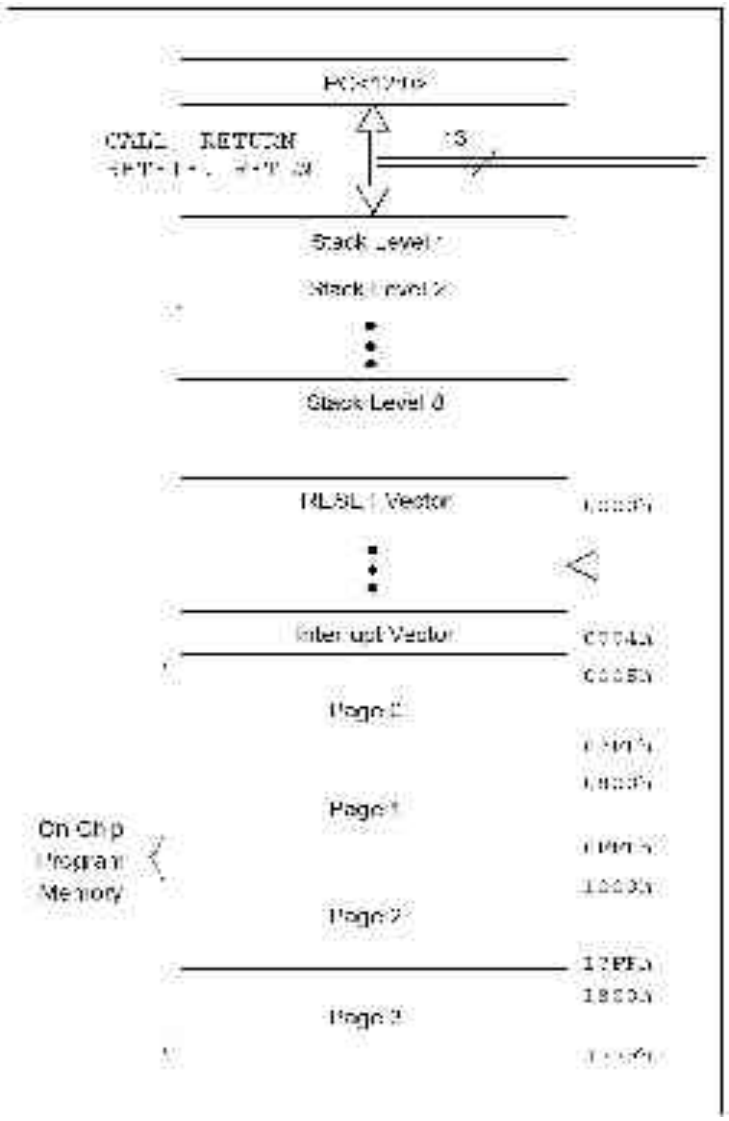
Bunun ötesinde, veri belleği genel amaçlı RAM ve özel fonksiyon kayıtları (SFR_S) olmak üzere ikiye bölünür. SFR`ler özel modülleri kontrol etmek için kullanılmaktadır.

Veri belleği EEPROM veri belleğini de içermektedir. Bu bellek, direkt veri belleğine planlanmamış, fakat indirekt olarak planlanmıştır ve indirekt adres göstergeleri okuma/yazma için EEPROM belleğinin adresini belirlemektedir. EEPROM belleği 64 bayt ve 10h-3Fh adres enine sahiptir.

2.7 VERİ BELLEK ORGANİZASYONU

Veri belleği ikiye ayrılır. Birincisi özel fonksiyon kayıt alanı (SFR), diğeri ise genel amaçlı kayıt (GPR) alanıdır. SFR'ler aygıtın işlemini kontrol eder.

Veri belleğinin bölümleri kümelenmiştir. Bu kümeler BANK adını alırlar. Bu hem SFR alanı hem de GPR alanı içinde geçerlidir. GPR alanı genel amaçlı RAM`ın 16 baytından daha fazlasına olanak sağlanabilmesi için kümelenmiştir. SFR'nin kümelenmiş alanı özel fonksiyonları kontrol eden kayıtlara aittir. Küme seçimi için kontrol bitleri gerektirmektedir. Bu kontrol bitleri STATUS kaydedicisinde yer almaktadır. Şekil2.4 veri bellek haritası organizasyonunu göstermektedir.



Şekil 2.4: Program hafızası ve küme(Yığın)

Veri belleğin tümüne ya direkt her kayıt dosyasının mutlak adreslerini kullanarak, yada dolaylı yoldan dosya seçim kaydedicisi (FSR) üzerinden erişilebilir. Dolaylı adresleme, veri belleğinin kümelenmiş alanına erişmek için RP1, RP0'ı kullanmaktadır. RP0 bitinin (STATUS <5> yani 5. Bit RP0 bitidir.) silinmesiyle BANK 0 seçilir. RP0'ın kurulmasıyla BANK 1'e geçilir. Her bir BANK 7Fh (128 bayt) kadar uzanır. Her bir kümenin ilk on iki yerleşimi özel fonksiyon kaydı için rezerve edilmiştir. Kalan bellek alanı ise statik RAM olarak kullanılmaktadır.

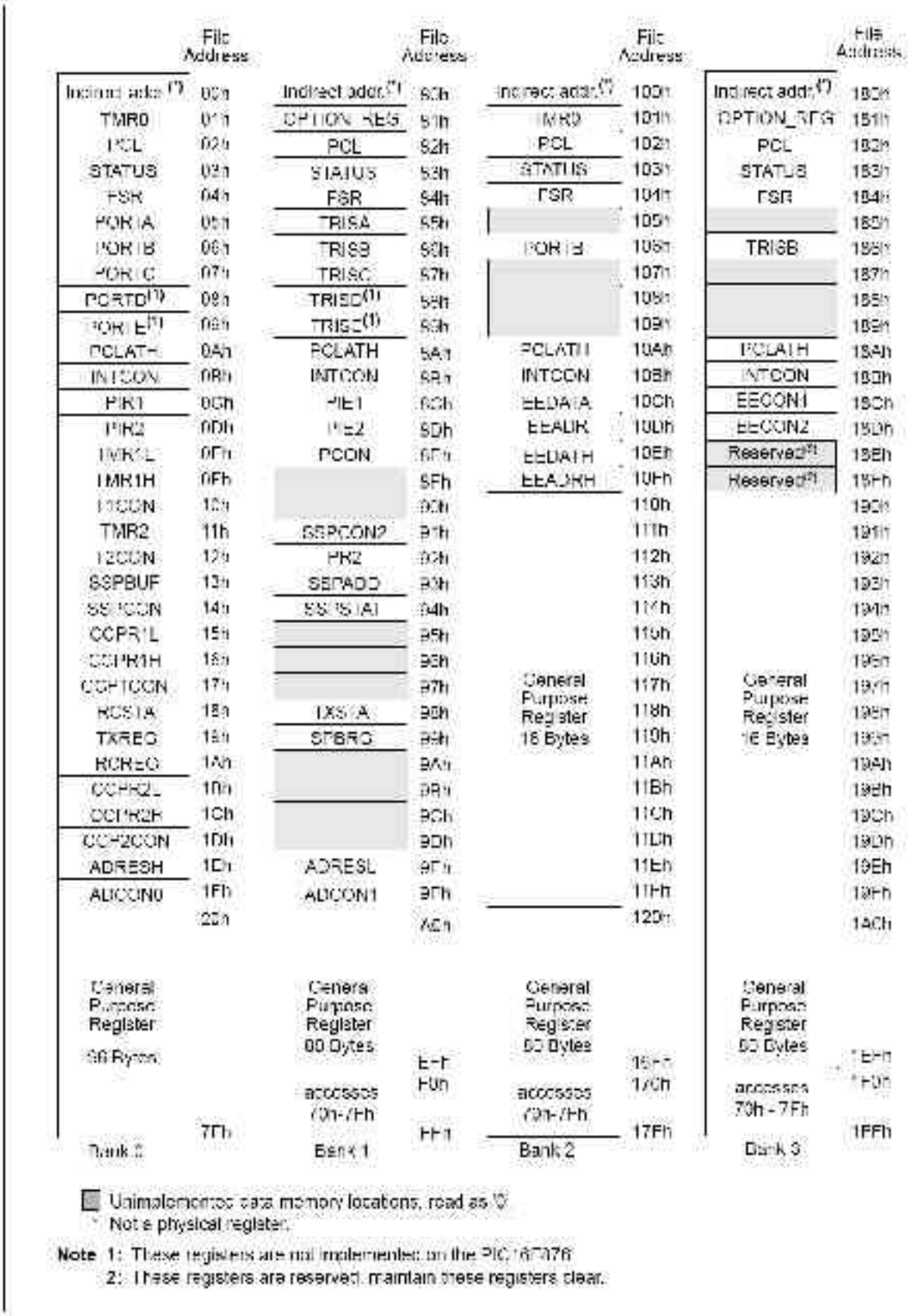
2.8 GENEL AMAÇLI KAYIT DOSYASI (GPR)

Bütün aygıtlar belirli bir miktarda genel amaçlı kaydedici (GPR) alanına sahiptir. Her bir GPR 8 bit enindedir ve dolaylı yada doğrudan FSR üzerinden erişilmektedir.

BANK 1` deki GPR adresleri BANK 0`daki adreslere tanımlanır. Örnek olarak, 0Ch veya 8Ch adresleme yerleşimi aynı GPR ` ye erişecektir

2.8.1 ÖZEL FONKSİYON KAYITÇILARI

Özel fonksiyon (Şek 2.5) kayıtçıları, aygıtın işleyişini kontrol etmek için CPU ve özel fonksiyonlar tarafından kullanılmaktadır. Bu kayıtçılar statik RAM`lerdir



Şekil 2.5 : Kayıtçı Dosyasının Haritası

2.9 PIC MİKRODENETLEYİCİLERİNDE YAZILIM

Komutlar:

Her bir komut, komutu tanımlayan ve işlevi ile ilgili gerekli bilgileri de (operand) içeren ve opcode olarak adlandırılan 14 bit sabit uzunluktan oluşan kelimelerdir.
Komutları üç gruba ayırabiliriz. Bunlar:

1. Bit'e Yönelik Komutlar: "f" dosya kaydedici atayıcısını ve "d" hedef atayıcıyı temsil etmektedir. Hedef kaydedici atayıcısı işlem sonucunun nereye yerleştirileceğini belirtir. Eğer "d" 0 ise sonuç "W" kaydedicisine yerleştirilir. "d" 1 ise sonuç komutta belirtilen "f" kaydedicisine yerleştirilir. Örnek: COMF 0x21, F.
2. Byte'a Yönelik Komutlar: "b" bellek alanındaki 8-bit ile adreslenen bit'i işaret eder içindeki işlem tarafından etkilenen biti seçer. "f" ise bitin yerleştirildiği kaydediciyi seçer. Örnek: BCF 0x12, 1
3. Bilgi(Literal)/Kontrol Komutları: "k" 8-bit'lik bir sabit veriyi veya 11-bit'ten oluşan adresi ifade eden etiket değerlerini temsil eder.

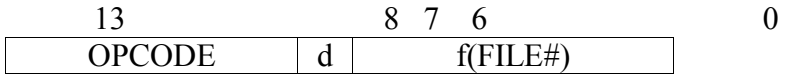
Çoğu komutun işlenmesi 1 saat çevrimi kadar sürer (osilatör frekansı/4). Fakat test işlemleri içeren komutlar komut şartı sağlandıysa 2, şart sağlanmadıysa 1 saat çevrimi kadar sürer. Komutları tanımlarken kullanılan harflerin anlamları:

ALAN:	**TANIM:**
f	Kaydedici dosya adresi (00'dan 7F'e kadar)
W	Çalışma kaydedicisi (akümülatör)
b	Bit tanımlayıcı (0-7)
k	Bilgi (sabit veri veya etiket)
x	Don't care alanları (=0 veya 1) Assembler x=0 kodunu üretecektir.
d	Atayıcı seçimi d=0 ise W'ye sakla d=1 ise f'e sakla (Kurulum yapılmadıysa d=1 olarak varsayılır.)
Label	Etiket ismi
TOS	Yığının en üst düzeyi
PC	Program sayıcı
PCLATH	Program sayıcı kilidi
GIE	Global aktif kesme biti
WDT	Watchdog timer
$\overline{TO}$	Zaman aralığı biti
$\overline{PD}$	Alçak güç biti
dest	Hedef yer (ya W kaydedicisi ya da belirtilen f kaydedicisi)
[ ]	Seçenekler
( )	İçerikler
→	...'e atanan
< >	Kayıt bit alanı
ε	...'ın kurulmasında

italics	Kullanıcı tarafından tanımlanan terim

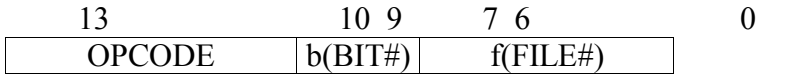
Komutların sahip olabileceği genel biçimler:

Byte'a yönelik komutlar



d=0 W hedefi için
d=1 f hedefi için
f= 7-bit kaydedici adresi

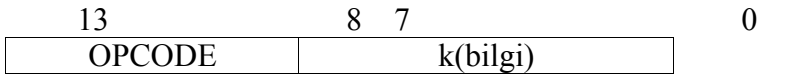
Bit'e yönelik komutlar



b=3-bit bit adresi
f= 7-bit kaydedici adresi

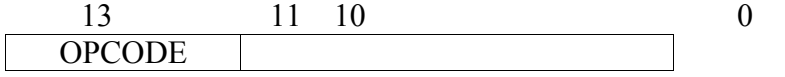
Bilgi ve Kontrol komutları

Genel



k= 8-bit sabit sayı/karakter

Yalnızca CALL ve GOTO komutları



k= 11-bit sabit sayı(adres/etiket)

Byte'a Yönelik Komutlar:

Komut:	**Status Yazmacı:**	**Anlamı:**
ADDWF f,d	C, DC, Z	W kaydedicisinin içeriğini f kaydedicisiyle toplar. Sonuç W veya f kaydedicisine yazılır.
		Örnek: ADDWF sayı,f
ANDWF f,d	Z	W kaydedicisi ile f kaydedici içeriğine AND işlemini uygular. Sonuç W veya f kaydedicisine yazılır.
		Örnek: ANDWF sayı,f
CLRF f	Z	f kaydedicisinin içeriğini siler.
		Örnek: CLRF sayı
CLRW -	Z	W kaydedicisinin içeriğini siler
		Örnek: CLRW
COMF f,d	Z	f kaydedicisinin içindeki sayı terslenir. Yani tüm 1'ler 0, 0'lar 1 olur. Sonuç f veya W kaydedicisine yazılır.
		Örnek: COMF sayı,f
DECF f,d	Z	f kaydedicisinin içindeki sayıyı 1 azaltır. Kaydedicinin içeriği h'00' ise, 1 eksilttiğinde h'FF' olur. Sonuç f veya W kaydedicisine yazılır.
		Örnek: DECF sayı,f
DECFSZ f,d	-	f kaydedicisinin içeriğini 1 azaltır. Kaydedicinin içeriği 0'sa bir sonraki komuta atlar. Sonuç W veya f kaydedicisine yazılır.
		Örnek: DECFSZ sayı,f
INCF f,d	Z	F kaydedicisinin içeriğini 1 artırır. Kaydedicinin içeriği h'FF' ise, 1 artırıldığında h'00' olur. Sonuç f veya W kaydedicisine yazılır.
		Örnek: INCF sayı,f
INCFSZ f,d	-	f kaydedicisinin içeriğini 1 artırır. Kaydedicinin içeriği 0'sa bir sonraki komuta atlar. Sonuç W veya f kaydedicisine yazılır.
		Örnek: INCFSZ sayı,f

IORWF f,d	Z	W kaydedicisi ile f kaydedici içeriğine OR işlemini uygular. Sonuç W veya f kaydedicisine yazılır. Örnek: IORWF sayı,f
MOVF f,d	Z	F kaydedicisinin içeriğini W veya f kaydedicisine yükler. Örnek: MOVF sayı,f
MOVWF f	-	W kaydedicisinin içeriğini f kaydedicisine yükler. Örnek: MOVWF sayı,f
NOP -	-	Bir komut saykılı boyunca işlem yapmaz. Zaman geciktirme işlemlerinde kullanılır. Örnek: NOP
RLF f,d	C	F kaydedicisi içindeki sayıyı bir bit sola kaydırır. Kaydediciden taşarak Carry bayrağına yazılan bit, LSB'ye yazılır. Sonuç W veya f kaydedicisine yazılır. Örnek: RLF sayı,f
RRF f,d	C	F kaydedicisi içindeki sayıyı bir bit sağa kaydırır. Kaydediciden taşarak Carry bayrağına yazılan bit, MSB'ye yazılır. Sonuç W veya f kaydedicisine yazılır. Örnek: RRF sayı,f
SUBWF f,d	C, DC, Z	f kaydedicisinden W kaydedicisinin içeriğini çıkarır. Sonuç W veya f kaydedicisine yazılır. Örnek: SUBWF sayı,f
SWAPF f,d	-	F kaydedicindeki ilk dört bit ile son dört biti yer değiştirir. Sonuç W veya f kaydedicisine yazılır. Örnek: SWAPF sayı,f
XORWF f,d	Z	W kaydedicisi ile f kaydedici içeriğine XOR işlemini uygular. Sonuç W veya f kaydedicisine yazılır. Örnek: XORWF sayı,f

Bit'e yönelik Komutlar:

Komut:	**Status Yazmacı:**	**Anlamı:**

BCF f,b	-	F kaydedicisinin içerisindeki sayının b bitini sıfırlar. Örnek: BCF sayı,5
BSF f,b	-	F kaydedicisinin içerisindeki sayının b bitini 1 yapar. Örnek: BSF sayı,3
BTFSC f,b	-	F kaydedicisinin b.ninci bitini test eder. Eğer bu bit 0 ise program akışı bir sonraki komuta geçer. Örnek: BTFSC sayı,2
BTFSS f,b	-	F kaydedicisinin b.ninci bitini test eder. Eğer bu bit 1 ise program akışı bir sonraki komuta geçer. Örnek: BTFSS sayı,7

Bilgi (Literal) ve Kontrol Komutları:

Komut:	**Status Yazmacı:**	**Anlamı:**
ADDLW k	C,DC,Z	W kaydedicisinin içeriğini k sabitiyle toplar. Sonuç W kaydedicisine yazılır. Örnek: ADDLW b'00001101'
ANDLW k	Z	W kaydedicisi ile k sabitine AND işlemini uygular. Sonuç W kaydedicisine yazılır. Örnek: ANDLW b'00111101'
CALL k	-	Program akışı k etiketinin bulunduğu yerdeki alt programa dallanır. Örnek: CALL bekle
CLRWDT -	$\overline{TO}, \overline{PD}$	Watchdog timerı sıfırlar. Ayrıca watchdog timerın prescaler değerini de sıfırlar. Status bitlerinden TO ve PD yi 1 yapar. Örnek: CLRWDT
GOTO k	-	Program k etiketi/ adresine dallanır. Örnek: GOTO bekle
IORWF k	Z	W kaydedicisi ile k sabiti OR işlemini uygular. Sonuç W kaydedicisine yazılır. Örnek: IORWF b'00100110'
MOVLW k	-	K sabitini W kaydedicisine yükler. Örnek: MOVLW h'07'
RETFIE -	-	Program akışını interrupt alt programından ana programa döndürür. Örnek: RETFIE
RETLW k	-	Program akışını alt programdan ana programa k değerini w kaydedicisine yükleyerek döndürür. Örnek: RETLW bir

RETURN -	-	w kaydedicisinin içeiğini değiştirmeden alt programdan ana programa döner.
		Örnek: RETURN
SLEEP -	$\overline{TO}, \overline{PD}$	Pic uyuma veya bekleme durumuna gelmiştir. Kaydedici içerikleri korunur. Güç harcaması azalır.
		Örnek: SLEEP
SUBLW k	C,DC,Z	K sabitinde W kaydedicisinin içeriğini çıkarır. Sonuç W kaydedicisine yazılır.
		Örnek: SUBLW b'11001011'
XORLW k	Z	W kaydedicisi ile k sabitine XOR işlemini uygular. Sonuç W kaydedicisine yazılır
		Örnek: XORLW b'01101110'

µ BÖLÜM 3.

MICROCHIP - MPLAB KULLANIMI

MPASM derleyicisi, Microchip firmasının ürettiği PICmicro mikrodenetleyicilerinin assembly kodlarını yazmak için kullanılan DOS veya Windows tabanlı bir metin editörüdür. MPLAB ile programcı aynı ortamda assembly kodlarını yazar, hatalarını ayıklar (debug) ve simüle edip programın çalışıp çalışmadığını kontrol eder.

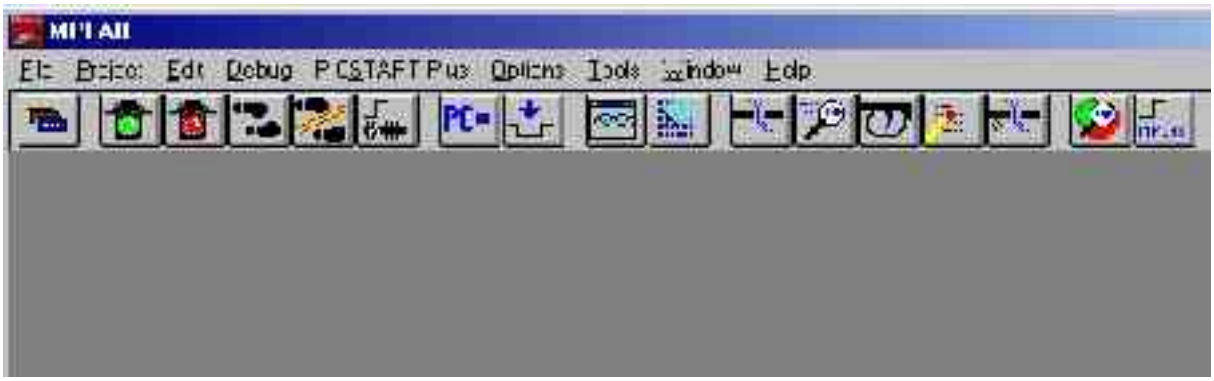
MPLAB Arizona Microchip tarafından geliştirilmiş entegre tasarımı programıdır.

3.1. MPLAB'ın KURULMASI ve BAŞLATILMASI

MPLAB'ı herhangi bir windows programının kurulumu gibi kurunuz. Bilgisayardan Başlat-Programlar-MPLAB ikonuna tıklayarak programı başlatınız.

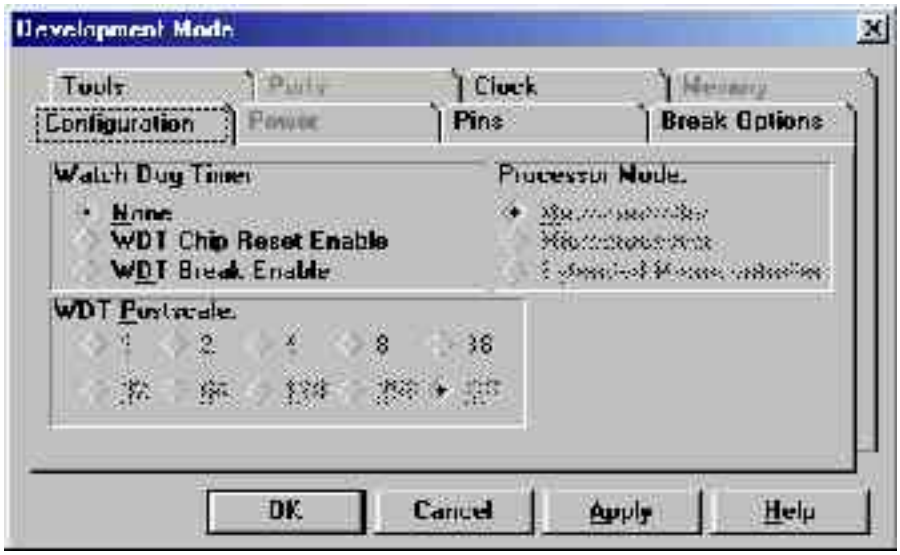
3.2. MPLAB'ın KULLANILMASI

MPLAB açıldığı zaman karşınıza araç çubukları olan bir windows penceresi gelir.

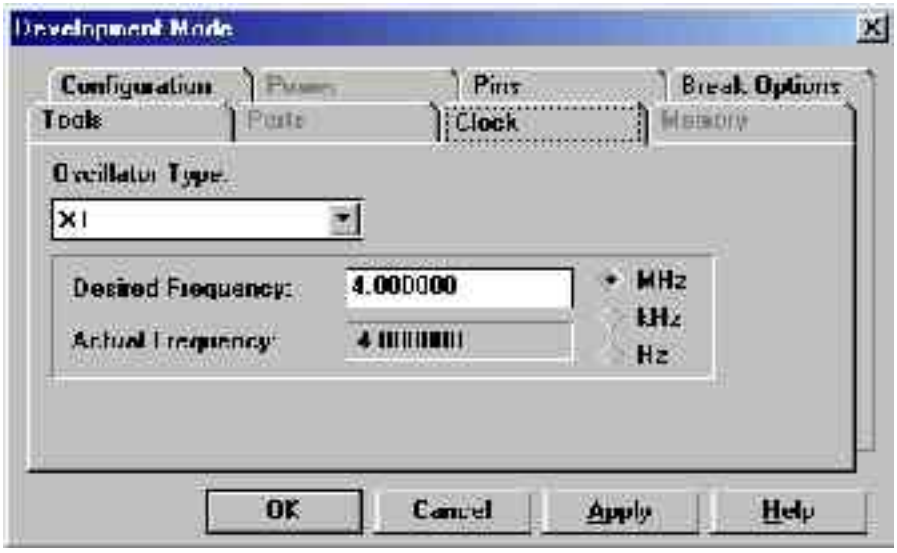


Burada ilk yapacağınız işlem kullanacağınız pıc'i seçmektir. Bunun içinde araç çubuğundan "Options" mesünü açınız. Buradan "Development Mode" seçeneğini seçiniz.

Eğer “MPLAB-SIM Simulator” işaretlenmemişse bu seçeneği işaretleyiniz. Kullanmak istediğiniz pıc çeşidini “Processor” menüsünden seçiniz.



Programınızda Watch Dog Timer kullanmıyorsanız “None” seçeneğini seçin. Kullanıyorsanız diğer seçenekleri seçerek alt kısımda bulunan “WDT Postscale” kısmından kullanacağınız preskalar oranını işaretleyiniz.

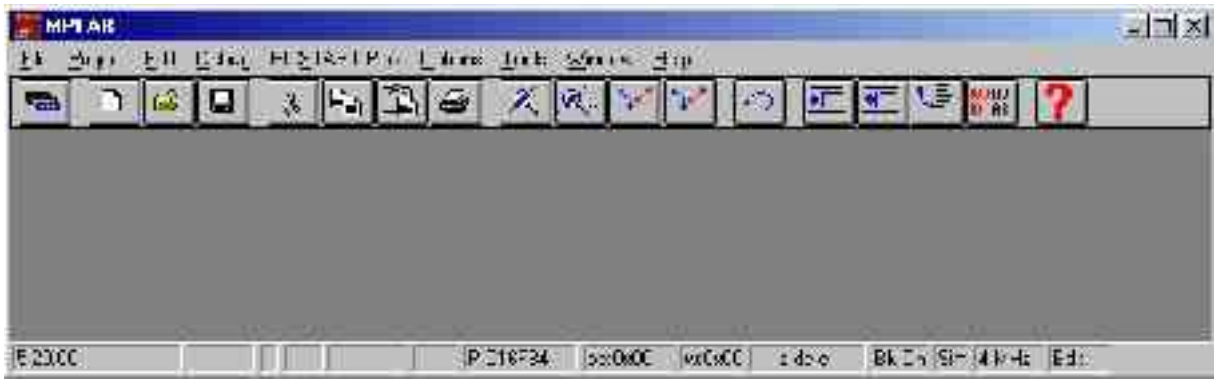


"Clock" menüsüne gelerek kullanmak istediğiniz osilatör tipini ve frekansını seçiniz. "OK" tuşuna basınız. Bir süre sonra yazılım seçtiğiniz PIC mikrodenetleyicisine ait gerekli değerleri yükler. Böylece MPLAB'ın hazırlık aşamaları bitti.

3.3. ASSEMBLY PROGRAMININ YAZILIMI

İlk önce MPLAB'ın araç çubuklarını inceleyelim. Araç çubuğunun en solundaki tuşa basarak araç çubuklarını değiştirebilirsiniz. MPLAB'ın "Edit", "Project", "User" ve "Debug" olmak üzere dört farklı araç çubuğu vardır.

- "Edit" araç çubuğu



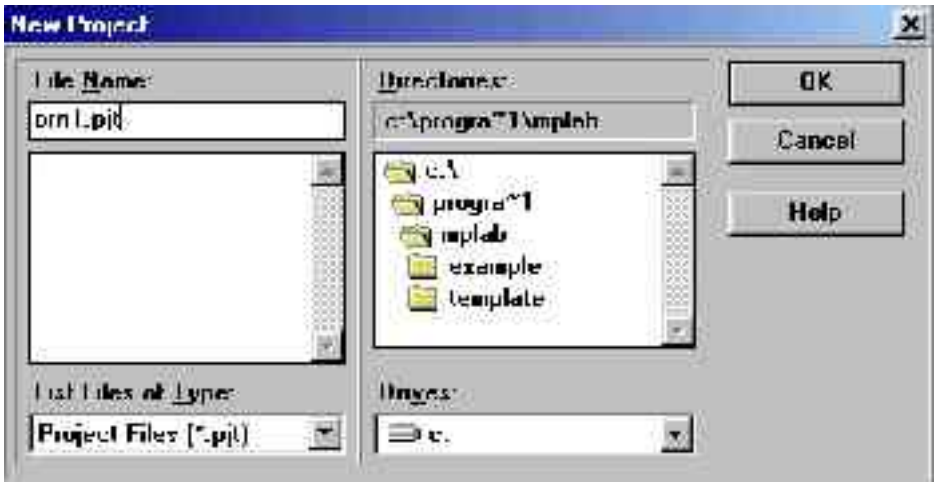
- "Project" araç çubuğu

♦ “User” araç çubuğu

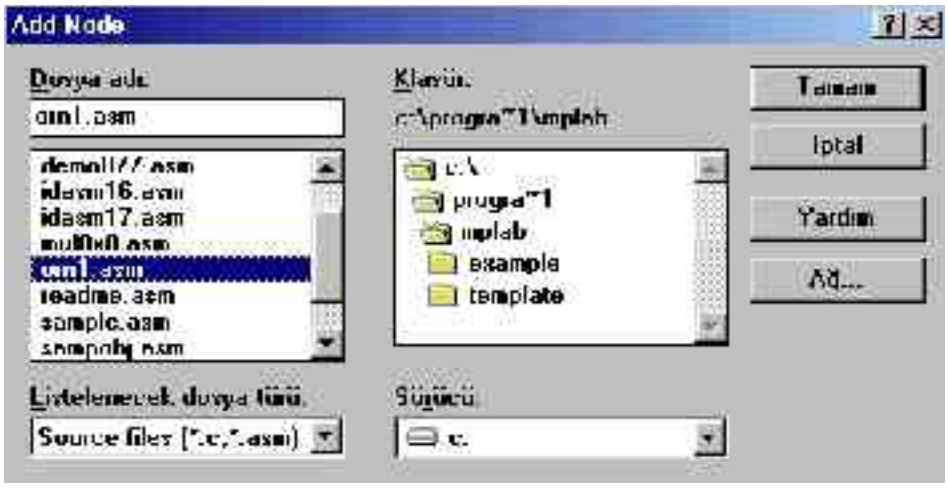
♦ “Debug” araç çubuğu

Araç çubuğunun en solundaki tuşa basarak “Edit” araç çubuğunu ekrana getirin. “File” menüsünden “New Source” seçeneğini seçin. Ekrana boş bir pencere gelecektir. Assembly programınızı buraya yazıp (orn1.asm gibi) kaydedin.

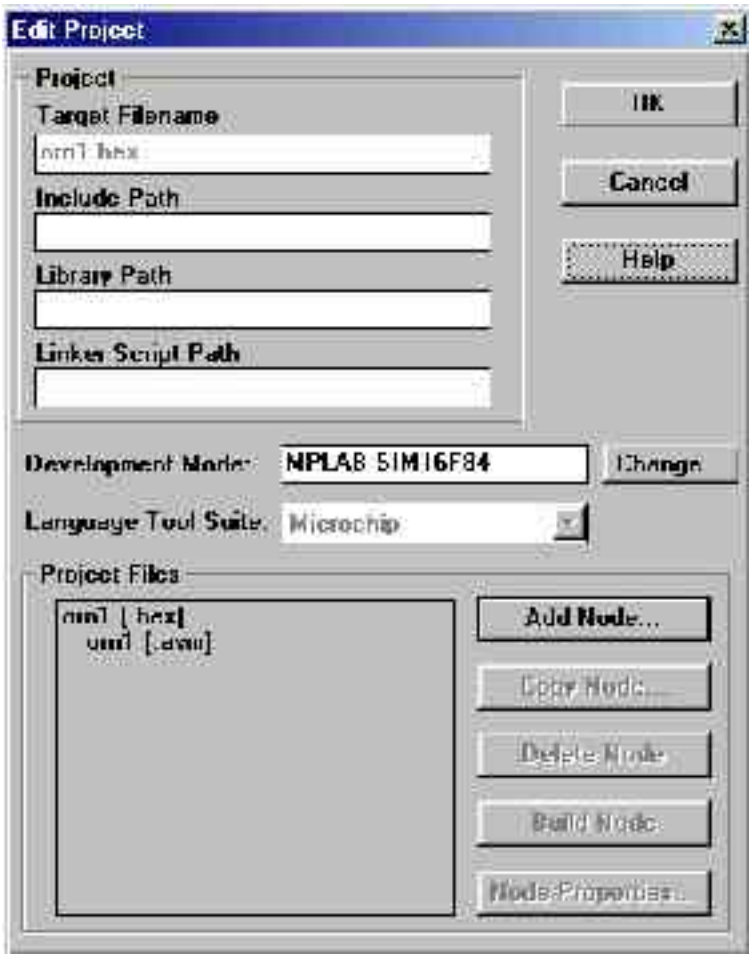
Daha sonra yazdığınız programın project dosyasını oluşturunuz. “Project” menüsünden “New Project” seçeneğini seçiniz. Karşınıza “New Project” penceresi gelir.



“File Name” yazan kutuya programınızın ismini yazınız ve “OK” tuşuna basanız (orn1.pjt gibi).

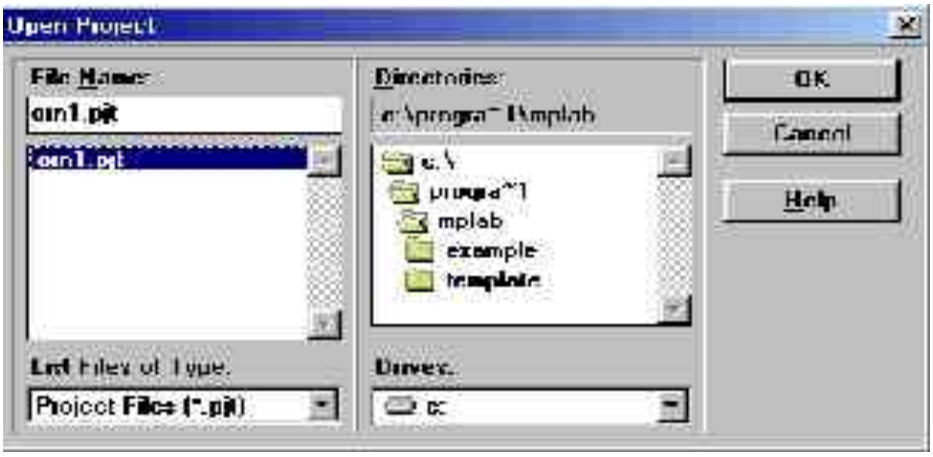


Burada yazdığınız programa ait .asm dosyasını seçip "Tamam" tuşuna basınız.



Hedef dosyanız otomatikman proje dosyası olan "orn1.pjt"nin ismini alarak "orn1.hex" olmuştur. İsterseniz .hex dosyasını başka bir isimle de kaydedebilirsiniz.

Proje oluşturmanın ve kullanmanın en önemli yanı, tasarladığımız uygulamada sonradan yapacağımız değişikliklerde, her seferinde sistem konfigürasyon ayarlarını yapmaya gerek yoktur.



Herhangi bir anda "Project" menüsünde "Open Project" seçeneğinden "orn1.pjt" dosyasını seçtiğimizde, projede hangi konfigürasyonu ve hangi dosyaları (.asm,.hex v.b gibi) tanımlamışsak, bunlar bir daha tanımlamaya gerek kalmaksızın MPLAB tarafından otomatikman devreye girecektir. Bu da birden fazla projeyle uğraşıldığında büyük kolaylıklar sağlayacaktır.

Araç çubuğunun en solundaki butona basarak "Project" araç çubuğunu ekrana getiriniz. Kaynak kod dosyasını derleyebilirsiniz. Bunun içinde "Project" menüsünden en soldaki butondan başlayarak 8. butona basınız (mause butonun üstüne geldiği zaman ekranın sol alt köşesinde "Build Full Project" diye butonun ismi görülür). Aynı işlemi "Project" menüsü altındaki "Build All" seçeneğini seçerekte aynı işlemi yapabilirsiniz. MPLAB kaynak kod dosyanızı derlemeye başlayacaktır.

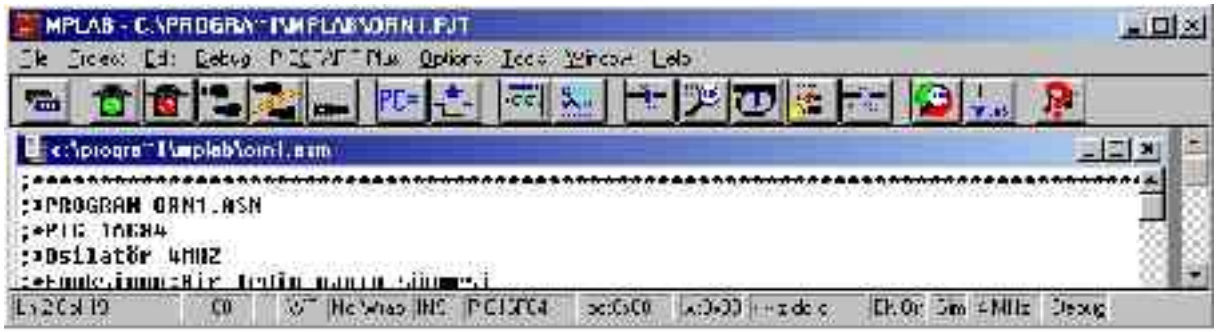
Eğer kaynak kod dosyanızda hata yoksa MPLAB derleme işlemi başarıyla tamamlayacaktır. Aşağıdaki gibi "Built Results" penceresi ekrana gelecektir. "Build completed succesfully." Derleme işlemi başarıyla tamamlandı mesajı gelir.

Hata varsa hata listesini ekrana getirecektir. Hatanın olduğu satıra gelerek kontrol ediniz. Bu hataları düzelterek derleme işlemini tekrarlayınız. Bütün hatalar düzeltildikten sonra MPLAB yazdığınız programa ait .hex dosyasını (orn1.hex gibi) oluşturacaktır.

3.4. MPSIM-PIC SİMULATÖRÜ

Burada programınızı bilgisayar ortamında çalıştırabilirsiniz. Portlardaki bilgilere bakabilirsiniz.

Araç çubuğunun en solundaki butona basarak "Debug" araç çubuğunu seçiniz.



"Debug" araç çubuğundaki bazı butonların görevleri şunlardır;

"**Swap Toolbar" butonu:** Araç çubuğunun en solunda yer alan bu butona basarak "Edit", "User", "Debug" ve "Project" araç çubuklarını ekrana getirebilirsiniz.

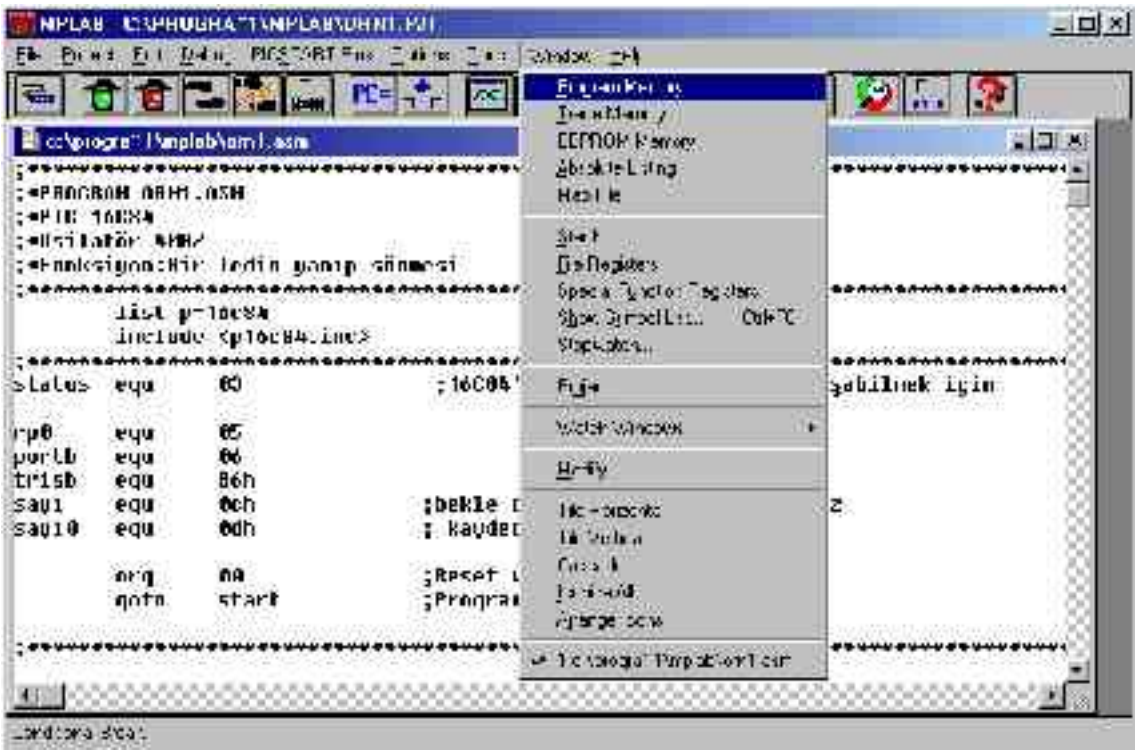
"Run" butonu: Yeşil trafik ışığı şeklindeki butona bastığınızda PIC simülasyona başlar.

"Halt Processor" butonu: Kırmızı trafik ışığı şeklindeki butona bastığınızdaysa simulasyon durur. Yeniden "Run" butonuna bastığınızda ise simulasyon kaldığı yerden devam eder.

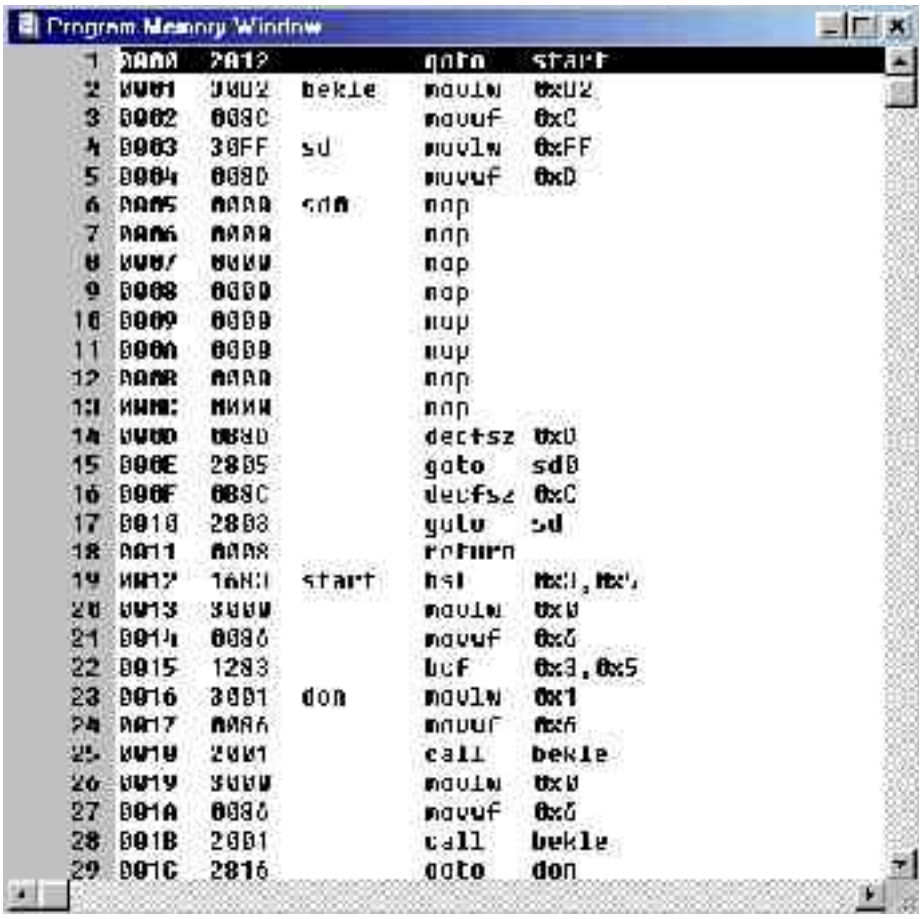
"Step" butonu: Simulasyonun, butona her basılış da bir komut ilerleyecek şekilde basamak basamak yapılmasını sağlar.

"Reset" butonu: Simulasyonu tümüyle durdurup, Pıc'i reset eder.

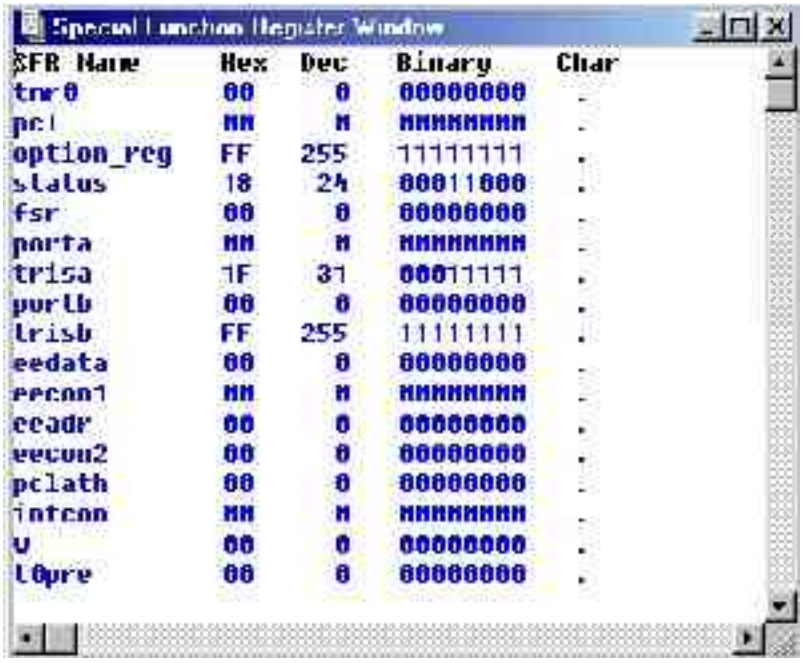
“Window” menüsünü seçiniz.



“Window” menüsünden sırasıyla “Program Memory”,



“Special Function Register” ve



“Stopwatch” pencerelerini açınız.

"Step" butonuna basarak "orn1.asm" kaynak kodunun adım adım simulasyonunu görebilirsiniz. Butona her basılışta sonraki komut devreye girer ve hangi komut yerine getiriliyorsa "Program Memory" ve "orn1.asm" dosyasında o komutun yer aldığı satır seçilidir. Buna paralel olarak "Special Function Register" penceresinde programdaki SFR değerlerinin adım adım değişimini de izleyebilirsiniz. "Stopwatch" penceresinde saat çevrim sayısını ve gerçek zamanda geçmesi gereken süre izlenir.

Eğer programa dışardan sinyal uygulamak gerekirse; bu durumda dışardan uygulanan sinyalleri simule etmede kullanılan "Stimulus" özelliği kullanılır.

"Debug" menüsünden "Simulator Stimulus" seçeneğinden "Asynchronous Stimulus" seçiniz.

"Asynchronous Stimulus Dialog" penceresi ekrana gelir.

Mausu "Stim1" tuşuna getirip sağ tuşa basarsanız aşağıdaki gibi bir pencere açılır.

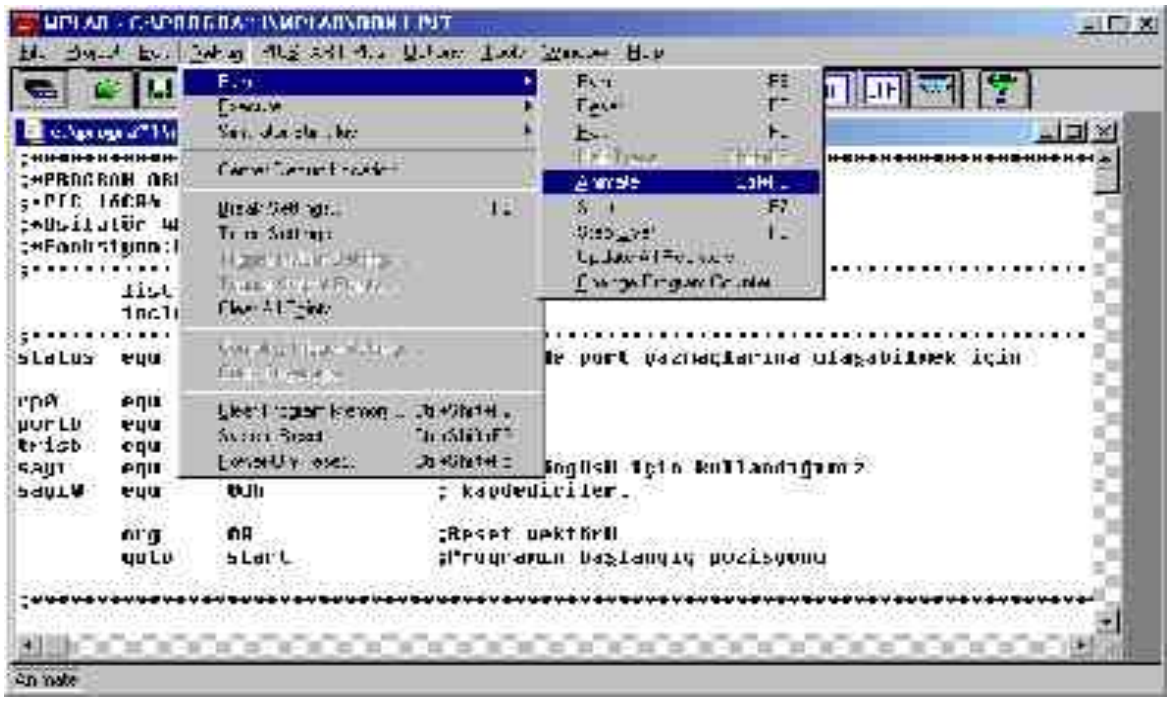
Örnek olarak "Stim1"e RA1 portunu ve porta yapılacak uyarıyı ise bir durum değiştiren sinyal yani "Toggle" olarak tanımlayalım. Durum değiştiren sinyal dışında RA1 portuna uygulanacak sinyali "0", "1" ve darbe olaraktan tanımlayabilirsiniz.

Pinlere isimlerini vermek için ise yukardaki şekilde de görüldüğü gibi “Assign Pin” seçeneği seçilir ve ekrana aşağıdaki gibi bir pencere gelir.

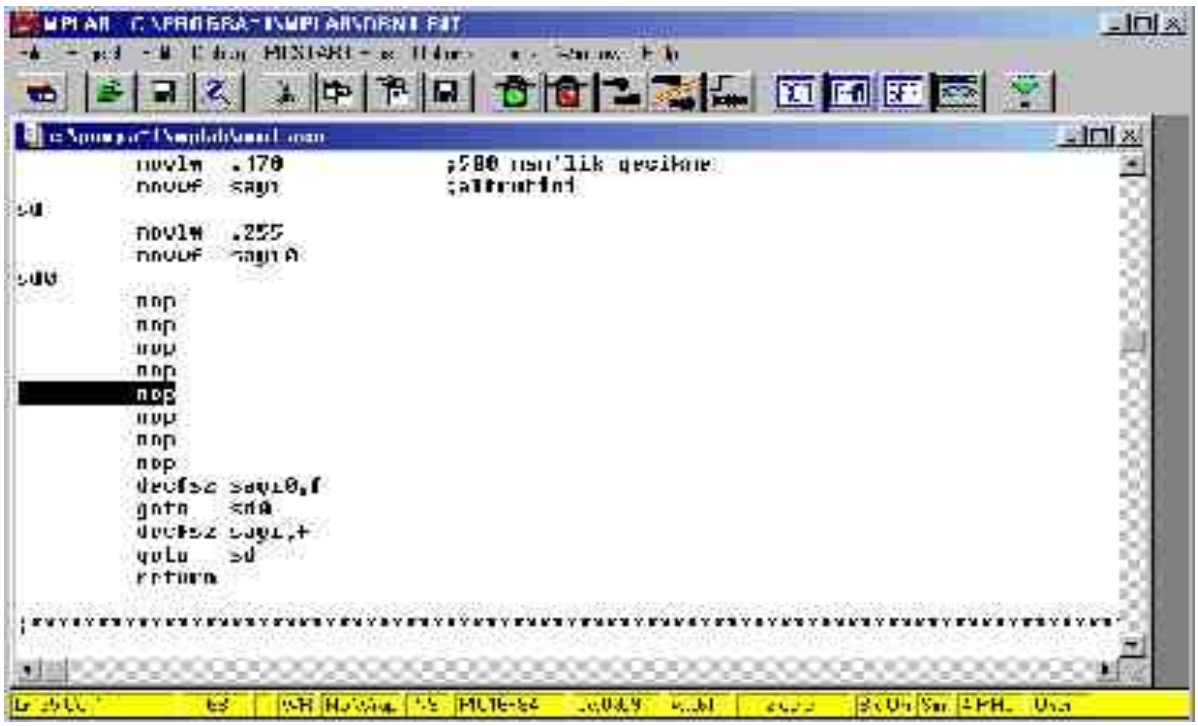
Buradan istediğimiz butonu seçeriz. Böylece devrenizdeki butonları da simulasyon ortamına getirmiş olursunuz.

Devrenizde kullandığınız bütün butonları buraya getirebilirsiniz. Butona basıldığında 1 bilgisini programa göndermiş olursunuz.

Bütün işlemleri bitirdikten sonra MPLAB’da simulasyonu başlatabilirsiniz. Bunun içinde “Debug” menüsünden “Run”ı seçip buradan da “Animate” komutunu seçiniz. Program iki komut satırı arasında hızlı bir şekilde devinir.



Programın işleyişini aşağıdaki gibi görebilirsiniz.



Aynı zamanda Portlardaki bilgiyi görebilmek ve programın akışını takip edebilmek için “Special Function Register Window” ile “Program Memory Window” pencerelerini seçin. Window menüsünden “Tile Vertical” seçeneğini seçerek aynı ortamda ikisini de görüntüleyebilirsiniz.

Bu açıklamaların hepsi MPLAB ortamında simulasyona bir giriş özelliğindedir. Daha ayrıntılı bilgi için MPLAB’ın “Help” menüsüne bakınız.

BÖLÜM 4.

DENEYLER

DENEY NO : 1
DENEYİN ADI : MİKRODENETLEYİCİ SİSTEMİNDEN VERİ ÇIKIŞI

1) DENEYİN AMACI :

- Mikrodenetleyiciden ikilik bilgi çıkışının nasıl gerçekleştirileceğini öğrenmek.
- Mikrodenetleyicinin portlarının çıkış olarak kurulmasını öğrenmek.
- Mikrodenetleyiciden alınan ikilik bilgilerin ledde nasıl görüntüleneceğini göstermek.
- Programda gecikme süresini hesaplamak ve öğrenmek.

2) GEREKLİ MALZEME:

2.1. EA-16F877 uygulama seti
2.2. Display ve Led Deney Modülü

3) GİRİŞ:

Bu deneyde mikrodenetleyici sisteminden veri çıkışının nasıl yapılacağını göstereceğiz. Bu veri çıkışının ledi nasıl yakıp söndürdüğünü inceleyeceğiz. Gecikme süresinin nasıl hesaplanacağını ve lede olan etkisi üzerinde duracağız.

4) YÖNTEM:

4.1. Bilgisayarınızda LED adlı bir directory açınız.
4.2. EK-1 de verilen programı MPLAB editöründe yazınız. Programa ait akış diyagramı şekil1.1'de gösterilmiştir. Programı daha iyi analiz edebilmek için akış diyagramı önemlidir. Yazdığınız programı LED/led1.asm olarak kaydediniz.
4.3. LED/led1.asm kaynak kodunu LED/led1.pjt adı ile proje dosyası haline getiriniz.
4.4. LED/led1.pjt yi assemble ediniz(derleyiniz). Assemble işlemi başarılı mı?
Başarısız ise; yazdığınız LED/led1.asm yi EK-1'deki (Bir ledin yanıp sönmesi) programı ile karşılaştırarak hatalarınızı düzeltiniz ve başarılı bir assemble yapılana dek 4.3'e kadar olan işlem basamaklarını tekrar ediniz.
4.5. Assemble başarılı ise LED/led1.hex dosyası elde edilir.
4.6. MPLAB'ı kapatınız. Bilgisayarınızı kapatarak programlayıcıyı bilgisayarınıza bağlayınız. Programlayıcının program/deney anahtarını kullanarak program konumuna alınız.
4.7. Bilgisayarınızı açarak PRG -PIC yazılımını çalıştırınız.
4.8. Oluşturulan LED/led1.hex dosyasını seçerek programlayıcıya programı yükleyiniz. "Programlama işlemi bitmiştir..." mesajını alana kadar bekleyiniz.
4.9. Program/Deney anahtarını deney konumuna alınız.
4.10. Deney kartını programlayıcıya takınız.
4.11. Deney kartının sağ üst köşesindeki LED/TUŞ jumperını LED konumuna alınız.
4.12. Led yandı mı?
4.13. Bir süre sonra led söndü mü?
4.14. Cevabınız "Evet" ise deneyi başarı ile tamamladınız. Tebrikler...

4.15. **Gecikme süresinin hesaplanması:**
Öncelikle tek döngü ile yapılan gecikme altyordamını inceleyelim.

Bekle

```
        movlw   .255        ;1 msn'lik gecikme
        movwf   sayac       ;altyordamı

Tekrar
        decfsz  sayac,f
        goto    Tekrar
        return
```

Yukarıdaki programda sayac kaydedicisine değer yüklenir. Sonra sayac kaydedicisinin içeriği sıfır olana kadar sayac birer birer azaltılır. Buradaki azaltma işlemi bizim döngümüzü oluşturmaktadır. Sayacın içeriği sıfırlanınca return komutu ile ana programa dönülür.

Bu gecikme altyordamında gecikme süresi hesabı komutların programda kaç saykıl tuttuğuna bakılarak hesaplanır (PIC16F84 komut tablosuna bakınız). Burada dikkat edilmesi gereken husus şartlı dalma komutlarının şart sağlandığında 2 saykıl; şart sağlanmadığında ise 1 saykıl çekeceğidir.

$$1 + 1 + (1+2)*sayac + \underline{1} + 2 = 2 + 3.255 + 3 = 770\ \mu s$$

(şart sağlandığı için gelen 1 saykıl)

Çoğu zaman tek döngü ile elde ettiğimiz gecikme altyordamları kısa bir gecikme süresi sağlar. Uzun bir gecikme elde etmek için ise içiçe döngüler kullanırız. Şimdi de içiçe döngü kullanarak elde edilen gecikme altyordamının nasıl hesaplanacağını görelim.

```
bekle
        movlw   .245        ;500 msn'lik gecikme
        movwf   sayac2      ;alyordamı
bekle1
        movlw   .255
        movwf   sayac1
bekle2
        nop
        nop
        nop
        nop
        nop
        decfsz  sayac1,f
        goto    bekle2
        decfsz          sayac2,f
        goto    bekle1
        return
```

Yukarıdaki gecikme altyordamı içiçe iki döngü kullanılarak elde edilmektedir. Buradaki döngü sayısı çoğaltılarak gecikme süresi artırılabilir. NOP kullanmak mikrodenetleyicinin işlem yapmadan 1 komut saykılı beklemesini sağlar. Bu da bize gecikme süresini artırma kolaylığı verir.

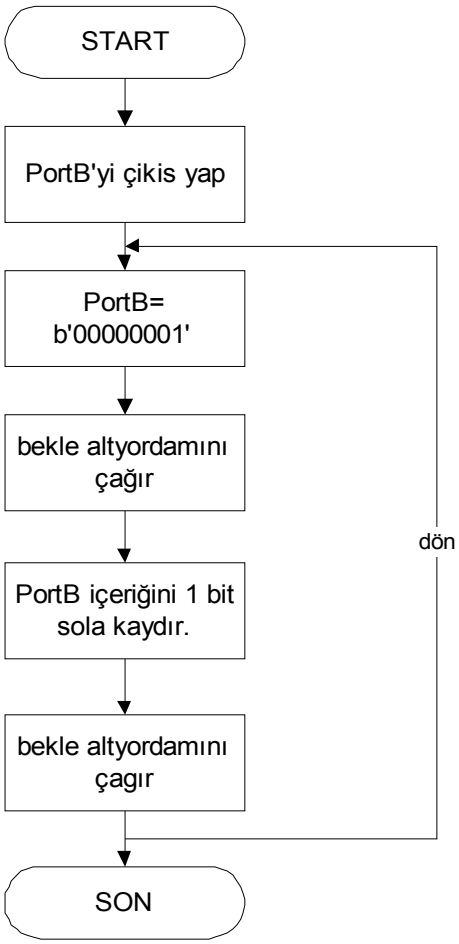
Döngü1'de (bekle2) sayac1'in içeriği sıfır olana kadar birer birer azaltılmaktadır. Sayac1 sıfıra eşitlenince döngü2 (bekle1) devreye girmektedir. Sayac2'nin içeriği sıfır olana kadar döngü2 çalışacaktır. Sayac2 sıfıra eşitlendiğinde ise döngü2 çalışmasını bitirir ve ana programa dönülür.

1 + 1 + [1 + 1 + (1 + 1 + 1 + 1+ 1 + 1 + 2).sayac1 + 1 + 1 + 2].sayac2 + 1 + 2 ≅ 500mS

Programda gecikme süresini azaltınız ve gecikme süresini hesaplayınız. Programı kaydedip 4.3 den 4.11 e kadar olan işlem basamaklarını tekrarlayınız. Gecikme süresi az olduğu zaman ne gözlediniz? Led daima yanıyor mu? Neden? (Aslında leddeki yanıp sönme devam etmektedir. Ama insan gözü saniyede 25'ten daha az ışık değişimlerini farkedebildiği için 25 'ten daha hızlı ışık değişimleri ledin daima yanması şeklinde algılar. Bu nedenle Film makineleri gözün bu özelliğini kullanarak saniyede 25 resim karesi peş peşe gösterilerek resimlerin hareketli görülmesi sağlanır).

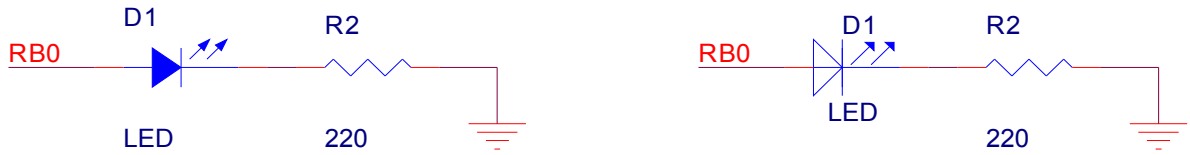
4.16. Şimdi de gecikme süresini artıralım. Programı kaydedip 4.3'den 4.11'e kadar olan işlem basamaklarını tekrarlayınız. Gecikme süresini artırdığınızda ne gözlediniz? Ledin yanıp sönmesini rahat bir şekilde gözlemlediniz mi?

4.17. Gecikme süresini hesaplayarak ledin yanma süresini ayarlayabiliriz.



Şekil 1.1. Programın akış diyagramı

4.18. **Port/B'deki Ledlerin Sürülmesi:**



Şekil 1.2:a) Ledin yanması Mantık1 bilgisi **b)**Ledin sönmesi Mantık0 bilgisi

4x4 Tuş Takımı-Display ve Led Modülündeki ledler katodu 220Ω'luk dirençle toprağa, anodu Port/B'ye bağlanmıştır. Port/B'nin RB0 pininden gönderilen Mantık1 (+5Volt) bilgisi ledi yakar, Mantık0 (0Volt) bilgisi ledi söndürür. Bu bilgiler sırayla gönderilerek ledin yanıp sönmesi sağlanır.

TARTIŞMA:

Led1.asm ana programındaki bekle altyordamı sayac1 ve sayac2 kaydedicisini kullanan bir gecikme altyordamıdır. İç döngü sayma işlemini bitirdikten sonra dış döngü sayma işlemine geçmektedir. Dolayısıyla döngülerin kaç defa işleyeceği döngülere atılan değerlerle belirlenmektedir. Doğal olarak, yüklenen sayı büyükse, gecikme süresi de uzun olur. Ancak sayaç kaydedicilerine en fazla .255 değeri yüklenebilir. Neden?

.

DENEY NO : 2
DENEYİN ADI : LEDLİ YÜRÜYEN IŞIK UYGULAMASI

1) DENEYİN AMACI :

- Mikrodenetleyicinin portlarının çıkış olarak kullanılmasını öğrenmek.
- Mikrodenetleyiciden alınan ikilik bilgilerin ledde kaydırılarak görüntülenmesini öğrenmek.

2) GEREKLİ MALZEME:

2.1. EA-16F877 uygulama seti
2.2. Display ve Led Deney Modülü

3) GİRİŞ:

Bu deneyde mikrodenetleyicinin portundan alınan ikilik bilgilerle ledlerin sürülmesini inceleyeceğiz.

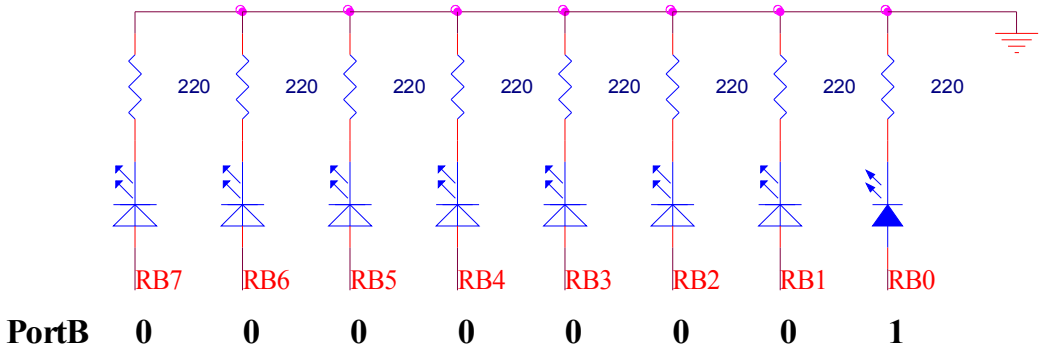
4) YÖNTEM:

4.1. EK-2 de verilen programı MPLAB editöründe yazınız. Programa ait akış diyagramı Şekil2'de gösterilmiştir, inceleyiniz. Yazdığınız programı LED/led2.asm olarak kaydediniz.
4.2. LED/led2.asm kaynak kodunu LED/led2.pjt adı ile proje dosyası haline getiriniz
4.3. LED/led2.pjt yi assemble ediniz(derleyiniz). Assemble işlemi başarılı mı?

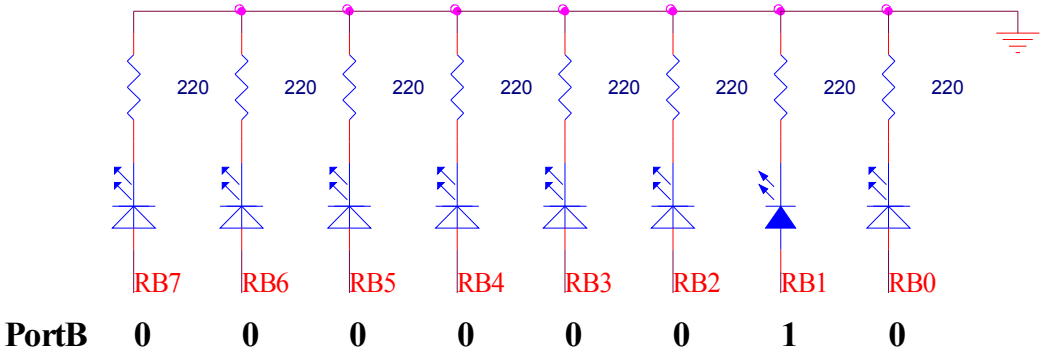
Başarısız ise; yazdığınız LED/led2.asm yi EK-2'deki . (Ledli yürüyen ışık uygulaması) programı ile karşılaştırarak hatalarınızı düzeltiniz ve başarılı bir assemble yapılana dek 4.3'e kadar olan işlem basamaklarını tekrar ediniz.
4.4. Assemble başarılı ise LED/led2.hex dosyası elde edilir.
4.5. MPLAB'ı kapatınız. Bilgisayarınızı kapatarak programlayıcıyı bilgisayarınıza bağlayınız. Programlayıcının program/deney anahtarını kullanarak program konumuna alınız.
4.6. Bilgisayarınızı açarak PRG -PIC yazılımını çalıştırınız.
4.7. Oluşturulan LED/led2.hex dosyasını seçerek programlayıcıya programı yükleyiniz. "Programlama işlemi bitmiştir..." mesajını alana kadar bekleyiniz.
4.8. Program/Deney anahtarını deney konumuna alınız.
4.9. Deney kartını programlayıcıya takınız.
4.10. Deney kartının sağ üst köşesindeki LED/TUŞ jumperını LED konumuna alınız.
4.11. Sağdan sola sürekli kayan kırmızı ışık görüyorsunuz değil mi? Işığın bir pozisyondan diğerine kaymadan önce bekleme süresi ne kadar? Bu süreyi isteğiniz şekilde uzatmak veya kısaltmak için programdaki gerekli değişiklikleri yapınız ve programınızı deneyiniz.

4.12. **Port/B deki bilginin kaydırılması:**

Ana program, aküye kaydedilen b'00000001'bilgisini Port/B'ye gönderir. Port/B'nin RB0 ucuna bağlı olan led yanar, diğer ledler sönüktür. Bekle altyordamını çağırır. Daha sonra bu bilgiyi bir pozisyon sola kaydırır. Port/B'de b'00000010' bilgisi elde edilir. Artık Port/B'nin RB1 ucuna bağlı olan led yanacaktır. Program bu şekilde işlevini sürdürür. Bilginin her kaydırılışında farklı bir led yanacağından kayan bir ışık dizgesi elde edilir.



Şekil 2.1: b'00000001' bilgisinin ledlerde görüntülenmesi



Şekil 2.2: b'00000010' bilgisinin ledlerde görüntülenmesi

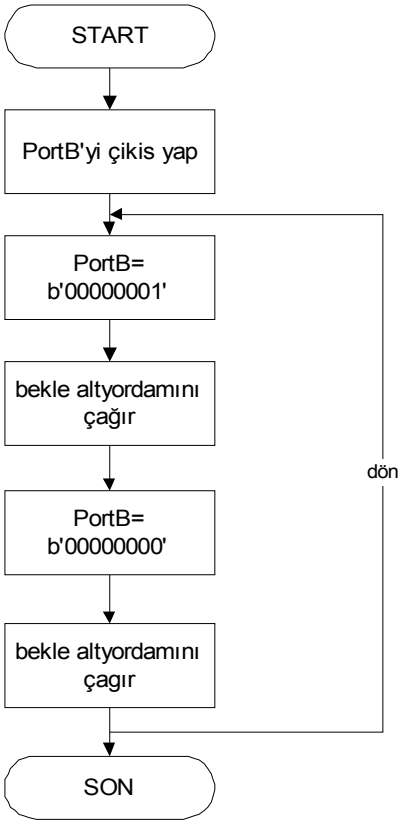
TARTIŞMA:

Bu deneydeki program, deney1 ile aynı modülü kullanarak kayan ışık dizgeleri uygulamasına bir örnektir. Programın başlangıcındaki port seçimi, bir önceki programla aynı olup Port/B çıkış olarak seçilmektedir.

Işığın her pozisyonda bekleme süresi, daha önce açıkladığımız gecikme altyordamı ile sağlanır. Ancak burada elde edilebilecek en uzun gecikme süresi 521 mSn dir. Daha uzun gecikme süresi istenirse programda ne gibi değişiklikler yapılabilir?

Birincisi bekle alt yordamındaki NOP'ları artırabiliriz. Bu ise programcılık açısından istenmeyen bir durumdur. Ya da içice döngü sayısı uzatılabilir.

Gecikme zamanını artırdığınızda ne gözlediniz? Yürüyen ışığın hareketini rahat bir şekilde görüyor musunuz?



Şekil 2.3. Programın akış diyagramı

DENEY NO : 3
DENEYİN ADI : İKİ YÖNLÜ YÜRÜYEN IŞIK UYGULAMASI

1. DENEYİN AMACI :

- Mikrodenetleyicinin portlarının çıkış olarak kullanılmasını öğrenmek.
- Mikrodenetleyiciden alınan ikilik bilgilerin ledde iki yönlü kaydırılarak görüntülenmesini sağlamak.

2. GEREKLİ MALZEME:

2.1. EA-16F877 uygulama seti
2.2. Display ve Led Deney Modülü

3. GİRİŞ:

Bu deneyde mikrodenetleyiciden alınan ikilik bilgilerin ledlerde çift yönlü kaydırılarak görüntülenmesini inceleyeceğiz.

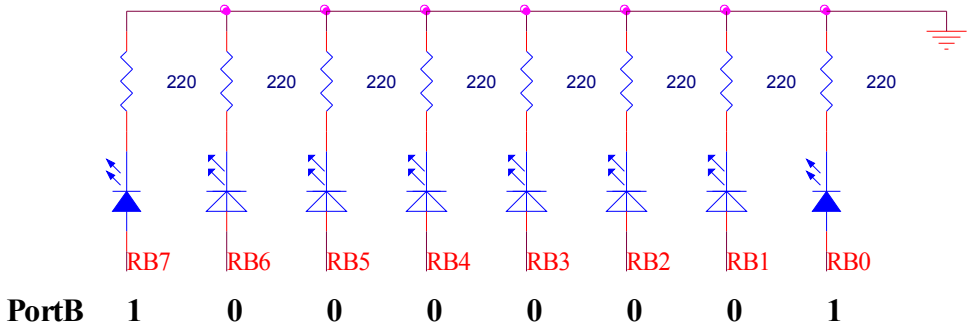
4. YÖNTEM:

4.1. EK-3 de verilen programı MPLAB editöründe yazınız. Yazdığınız programı LED/led3.asm olarak kaydediniz.
4.2. LED/led3.asm kaynak kodunu LED/led3.pjt adı ile proje dosyası haline getiriniz.
4.3. LED/led3.pjt yi assemble ediniz(derleyiniz). Assemble işlemi başarılı mı?
Başarısız ise; yazdığınız LED/led3.asm yi EK-3'deki (İki yönlü yürüyen ışık uygulaması) program ile karşılaştırarak hatalarınızı düzeltiniz ve başarılı bir assemble yapılana dek 4.3'e kadar olan işlem basamaklarını tekrar ediniz.

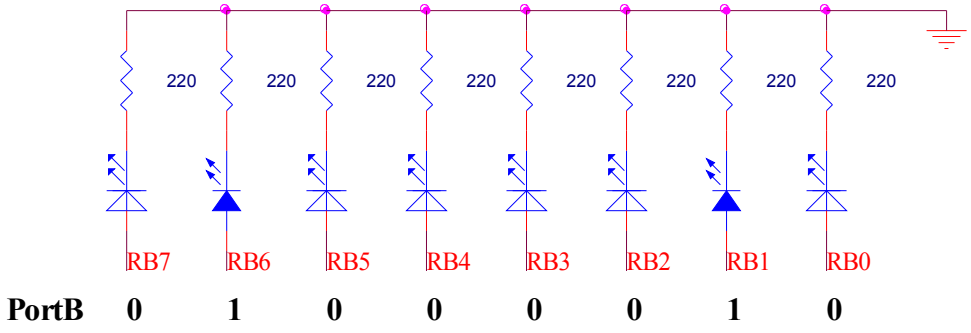
4.4. Assemble başarılı ise LED/led3.hex dosyası elde edilir.
4.5. MPLAB'ı kapatınız. Bilgisayarınızı kapatarak programlayıcıyı bilgisayarınıza bağlayınız. Programlayıcının program/deney anahtarını kullanarak program konumuna alınız.
4.6. Bilgisayarınızı açarak PRG -PIC yazılımını çalıştırınız.
4.7. Oluşturulan LED/led3.hex dosyasını seçerek programlayıcıya programı yükleyiniz. "Programlama işlemi bitmiştir..." mesajını alana kadar bekleyiniz.
4.8. Program/Deney anahtarını deney konumuna alınız.
4.9. Deney kartını programlayıcıya takınız.
4.10. Deney kartının sağ üst köşesindeki LED/TUŞ jumperını LED konumuna alınız.
4.11. İki yönlü kayan kırmızı ışık dizgeleri görüyorsunuz değil mi? Işığın bir pozisyondan diğerine kaymadan önce bekleme süresi ne kadar? Bu süreyi isteğinize göre uzatmak veya kısaltmak için programdaki gerekli değişiklikleri yapınız ve programınızı deneyiniz.

4.12. **Port/B deki bilginin iki yönlü kaydırılması:**

Ana programda, ilk olarak sola kaydedicisinde ikilik '00000001' bilgisi, sağa kaydedicisinde de ikilik '10000000' bilgisi vardır. Böylece Port/B'nin RB0 ve RB7 uçlarına bağlanan ledler yanar. Sonra sağa kaydedicisinin içeriğini sağa doğru, sola kaydedicisinin içeriğini de sola doğru kaydırırız. Bu işlem sonucu Port/B'de b'01000010' bilgisi elde edilir. Port uçlarında lojik1(+5 Volt) bilgisi bulunan ledler yanar. Program bu şekilde işlevini sürdürür. Bilginin her kaydırılışında farklı ledler yanacağından iki yönlü kayan ışık dizgeleri elde edilir.



Şekil 3.1: b'10000001' bilgisinin ledlerde görüntülenmesi



Şekil 3.2: b'01000010' bilgisinin ledlerde görüntülenmesi

TARTIŞMA:

Bu deneydeki program, deney1 ve deney2'deki modülü kullanarak iki yönlü kayan ışık dizgeleri uygulamasına bir örnektir. Programın başlangıcındaki port seçimi, bir önceki programla aynı olup Port/B çıkış olarak seçilmektedir.

Bu deneyleri örnek olarak düşünürsek, yine aynı donanımı kullanarak çok değişik ışık dizgeleri düşünebilir ve bunların programlarını yazabilirsiniz. Deneyin!

Örnek: iki yönlü yürüyen ışık uygulaması,ledli 255'e kadar sayaç,üç displayli 255'e kadar sayan sayaç,dört bit dört bit yer değiştirme uygulaması gibi...

DENEY NO : 4

DENEYİN ADI : MİKRODENETLEYİCİ SİSTEMİNE VERİ GİRİŞİ ve ÇIKIŞI

- **DENEYİN AMACI:**

3.1. Dış dünyadan mikrodenetleyiciye veri girişinin nasıl yapılacağını öğrenmek
3.2. Mikrodenetleyiciden veri çıkışının nasıl yapılacağını öğrenmek.
3.3. Alınan verinin görüntüleme altyordamları kullanarak displayde görüntülenmesini sağlamak.
3.4. Tuş takımında tuşların nasıl kullanılacağını öğrenmek.
3.5. Tablo kullanımını öğrenmek

2) GEREKLİ MALZEME:

2.1. EA-16F877 uygulama seti
2.2. Display ve Led Deney Modülü

3) GİRİŞ:

Bu deneyde mikrodenetleyici sistemine veri girişinin nasıl yapılacağını göstereceğiz. Mikrodenetleyici'ye veri transferi yapmak için tuşları kullanacağız.

4) YÖNTEM:

4.1. Bilgisayarınızda TUS adlı bir directory açınız.
4.2. EK-4 de verilen programı MPLAB editöründe yazınız. Programa ait akış diyagramı şekil4.3'de gösterilmiştir. Yazdığınız programı TUS/tus1.asm olarak kaydediniz.
4.3. TUS/tus1.asm kaynak kodunu TUS/tus1.pjt adı ile proje dosyası haline getiriniz.
4.4. TUS/tus1.pjt yi assemble ediniz(derleyiniz). Assemble işlemi başarılı mı?

Başarısız ise; yazdığınız TUS/tus1.asm'yi EK-4'deki program ile karşılaştırarak hatalarınızı düzeltiniz ve başarılı bir assemble yapılana dek 4.4'e kadar olan işlem basamaklarını tekrar ediniz.

4.5. Assemble başarılı ise TUS/tus1.hex dosyası elde edilir.
4.6. MPLAB'ı kapatınız. Bilgisayarınızı kapatarak programlayıcıyı bilgisayarınıza bağlayınız. Programlayıcının program/deney anahtarını kullanarak program konumuna alınız.
4.7. Bilgisayarınızı açarak PRG -PIC yazılımını çalıştırınız.
4.8. Oluşturulan TUS/tus1.hex dosyasını seçerek programlayıcıya programı yükleyiniz. "Programlama işlemi bitmiştir..." mesajını alana kadar bekleyiniz.
4.9. Program/Deney anahtarını deney konumuna alınız.
4.10. Deney kartını programlayıcıya takınız.
4.11. Deney kartının sağ üst köşesindeki LED/TUŞ jumperını TUŞ konumuna alınız.
4.12. Butona bastığınızda artırma işlemi yapıyor mu? Yapıyorsa program çalışmaktadır. Yapmıyorsa programı kontrol edip tekrar yükleyiniz.

4.13. **Portların giriş çıkış olarak kurulması:**

Bu deneyde mikrodenetleyiciden hem veri girişi hem de veri çıkışı yapılmaktadır. Burada veri girişini tuş takımından seçtiğimiz bir tuş sağlamaktadır. Veri çıkışı ise 7 parçalı göstergede görüntü olarak elde edilmektedir.

Program içinde bir port aynı anda hem giriş hem de çıkış olarak kullanılırken aşağıdaki işlemler yapılır;

1. Kullanılacak olan port ilk olarak (program ilk çalıştığında) çıkış mı, giriş mi olacağı belirlenir.

2. *STATUS* yazmacının *RP0* biti 1 yapılarak *bank1*'e geçilir burada istenen portlar giriş yada çıkış olarak seçilir.
3. Daha sonra *RP0* biti 0 yapılarak *bank0*'a geçilir. Artık portlarımız istenen konumdadır.

Örnek: Aşağıdaki programda Port/A ve Port/B çıkış olarak seçilmiştir. (Bu program parçası görüntüleme altyordamında portlar için gerekli olan kurulumdur.)

Bsf	Status,	RP0
Movlw	00h	
Movwf	trisb	
Movwf	trisa	
Bcf	Status,	RP0

- Aynı program içinde daha önce ayarlanmış olan portun konumunu değiştirmek istediğimizde yine *bank1'*e geçilerek portlar istenildiği gibi giriş veya çıkış olarak ayarlanır. Sonra yine *bank0* 'a geçilir. Böylece portların kurulması tamamlanmış olur.

Bu deneyde ilk olarak Port/B; yedi parçalı göstergede görüntülemeyi sağlamak için çıkış olarak seçildi. Sonra tuş bilgisini okuyabilmek için giriş olarak seçildi.

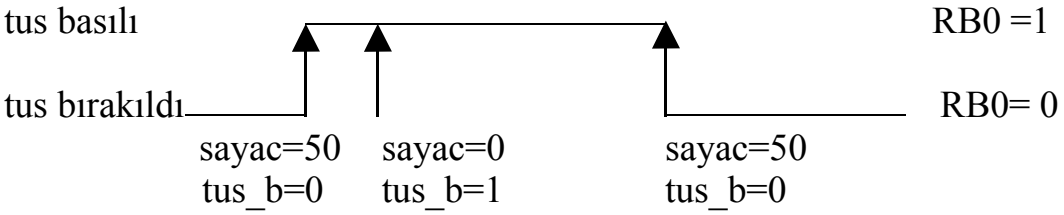
bsf	status,	RP0
movlw	b'00001111'	
movwf	trisb	
bcf	status,	RP0
movlw	b'00010000'	
movwf	portb	

Bu örnekte Port/B'nin *RB0-RB3* bitleri giriş; *RB4-RB7* bitleri ise çıkış konumundadır. (Bu tuşun algılanması için gerekli olan kurulumdur.)

4.14. **Tuş kontrolünün gerçekleştirilmesi :**

Tuş kontrolünde dikkat edilmesi gereken en önemli husus tuşun basılı kalma süresidir. Yani tuşa bastığımız anda mı mikrodenetleyici bunu bir bilgi olarak algılayacak, yoksa tuş bırakıldıktan sonra mı algılayacak?

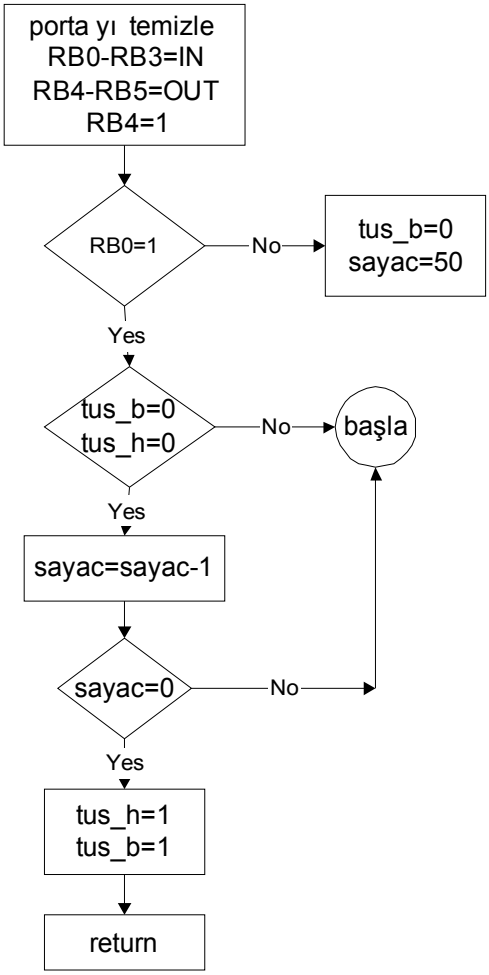
Tuş basıldığı zaman bunu bir bilgi olarak algıladığımızı kabul edersek bu sefer yeni bir sorun ortaya çıkar. Bizim tuşa basıp-çekme hızımız en az 1-2 sn kadardır. Fakat mikrodenetleyicinin çalışma hızı ise 1usn'dir. Mikrodenetleyici bu süre içinde butona defalarca basılmış gibi algılar. Tuşa her basıldığında mikrodenetleyiciye bir işlem yaptırmak için ;



tus_h=0 tus_h=1 tus_h=X

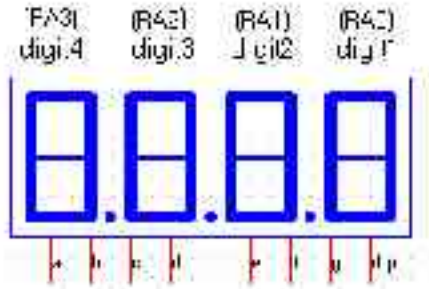
- Önce tuşun basıldığı algılanır,
- Sayaç azaltılır,
- Sayaç 0 olduğunda *tus_h* ve *tus_b* bayrağı birlenir,
- Tuş bırakıldığında *tus_b* bayrağı sıfırlanır ve sayaç değeri yeniden yüklenir.

Böylece tuşa basılıp çekilme süresince bir kere *tus_h* bayrağı birlenir. Program içerisinde *tus_h* bayrağı kontrol edilerek işlem yapılır. Eğer tus_h=1 ise artırma işlemi yapılır. Ve *tus_h* bayrağı temizlenir.



Şekil 4.1 :Kontrol altyordamının akış diagramı

4.15.**Görüntüleme işlemi:**



Şekil 4.2: Yedi parçalı gösterge

Tuş takımı modülünde dört tane 7 parçalı gösterge kullanılmaktadır. Bu displaylerin segmentleri (a , b , c , d , e , f , g , dp) kendi aralarında birleştirilmiştir. Bu segmentlere bilgi PortB 'den gönderilir.

Port/B	Segmentler	Port/A	Digitler
RB0	a	RA0	Digit1
RB1	b	RA1	Digit2
RB2	c	RA2	Digit3
RB3	d	RA3	Digit4
RB4	e		
RB5	f		
RB6	g		
RB7	dp		

Her bir displayin katodları Port/A tarafından kontrol edilmektedir. Haneler sırayla sürülmektedir. Bu işlem saniyede 200 defa gerçekleştiği için tüm displayler aynı anda yanıyormuş gibi algılanır.

Movf	Digit1,	W
Call	Kod	
Movwf	Portb	
Movlw	b'00000001'	
Movwf	Porta	
Call	Bekle	

Görüntüleme alt yordamında digit1 bilgisinin yedi parçalı gösterge kod karşılığı bulunduktan sonra bu bilgi Port/B ye gönderilir. Port/A'nın *RA0* biti birlenir ve birler hanesi sürülür. Bu displayde digit1 bilgisi görüntülenir. Bu işlem sırayla yapılarak tüm displayler aktif hale gelir.

4.16. **Tablo kullanımı**

Çevrim tabloları bir kodu başka bir koda çevirmek için kullanılır. Programda digitler artırılarak işlem yapıldı. Her artış değeri görüntülendi . İşlemleri binary olarak yaptık. Yedi parça gösterimde ise bu değerlerin karşılıkları daha farklıdır. Displayde her bir segment harf ile ifadelendirilmiştir. Bu harfler sırası ile a,b,c,d,e,f,g,h dır. Burada istenen rakamın yakılabilmesi için gerekli segmentlere

lojik 1 bilgisi gönderilir. Örneğin “1” rakamının gösterilebilmesi için *b* ve *c* segmentleri 1 yapılmalıdır.

Displayda 1 rakamını görüntüleyebilmek için

h g f e d c b a
0 0 0 0 0 1 1 0 ikili bilgi kullanılır.

b’00000001’ = yedi parçalı göstergede (00000110) tır.

Bu her ikili rakam için farklı bir kod üretilmesi gerektiğini ifade etmektedir.Bunun için çevrim tabloları kullanılır.

8

Kod			
	addwf	PCL,1	
	retlw	b'00111111'	;0
	retlw	b'00000110'	;1
	retlw	b'01011011'	;2
	retlw	b'01001111'	;3
	retlw	b'01100110'	;4
	retlw	b'01101101'	;5
	retlw	b'01111101'	;6
	retlw	b'00000111'	;7
	retlw	b'01111111'	;8
	retlw	b'01101111'	;9

Yukarıdaki programda yedi parçalı gösterge gösterimi için kullandığımız , 1 den 9 a kadar olan rakamların yedi parçalı kod karşılık tablosu verilmiştir. Burada ;

PCL: PIC16F877’ün 13 bitlik program sayıcısı vardır. Pic programlarında program sayıcı PC sembolü ile gösterilir. Program sayıcının alt 8 bitine PCL , üst 5 bitine PCH adı verilir.

Programlar genellikle 256 Kbayt’ı (2= 256 bayt) geçmeyeceğinden, program belleğinin ilk 8 bitini (PCL) kullanmak yeterlidir. Eğer 256 baytı aşan programlar yazmak gerekirse PCLATH kaydedicisinin 0. , 1. ve 2. bitini de kullanmak gerekir.

Tablo okunması yapılırken PCL’ye offset değeri ilave edilerek istenen tablo satırına gidilir.

Retlw: Bir alt programdan ana programa dönüş için kullanılan RETURN komutuna çok benzer. RETLW de ana programa dönüş için kullanılır. Tek bir fark; ana programa dönüş esnasında W kaydedicisine bir sabit sayı yüklenir. Örneğin, ana programa dönüldüğünde W kaydedicisinin içinde h’3F’ sayısının olması istenirse, komut aşağıdaki gibi yazılır.

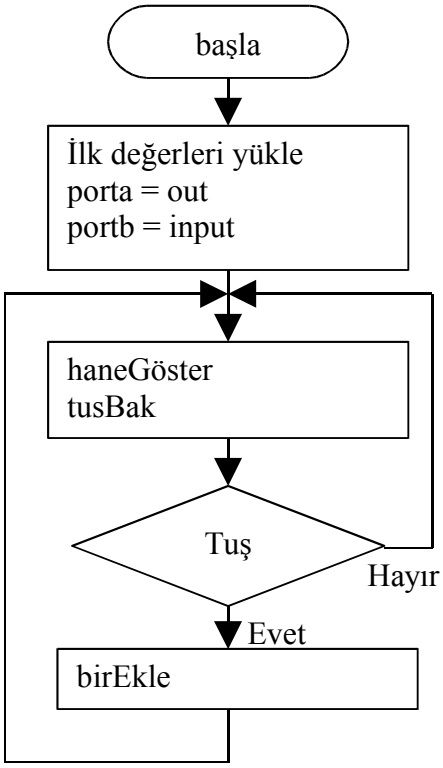
```
RETLW       h'3F'   ;W kaydedicisine h'3F' yükle ve ana programa dön.
```

TARTIŞMA:

TUS 1 ana programında veri girişi olarak kullanılan tuşun 1 yada 0 olma durumu ; Port/B'nin RB4 bitinden gönderilen 1 bilgisinin RB0 bitinden alınmasıyla karar verilir. Bu programda farklı bir tuş kullanmak istersek programda ne gibi değişiklikler yapılmalıdır?

Eğer biz bu programda, Port/B'nin RB5 bitine 1 bilgisini gönderip Port/B'nin RB0 bitinden okumak istersek veri girişi olarak hangi tuşu kullanmış oluruz?

Programda sayıcı 999 a kadar sayabilmektedir. Programda gerekli değişiklikler yapıldığında sayıcı en fazla kaça kadar sayabilir?



Şekil 4.3: Programın akış diyagramı

DENEY NO : 5
DENEYİN ADI : DİJİTAL SAAT UYGULAMASI

- DENEYİN AMACI:

    - Mikrodenetleyici ile yedi parçalı göstergenin sürülmesini öğrenmek.
    - Mikrodenetleyicide kesmenin ve kesme alt programının nasıl kullanılacağını öğrenmek .
    - Mikrodenetleyicinin dış dünya ile iletişimini sağlayan saat, dakika, ileri ve geri tuşlarının kullanılmasını öğrenmek.

- **GEREKLİ MALZEME:**

2.1. EA-16F877 mikrodenetleyici uygulama seti
2.2. Display ve Led Deney Modülü

- **GİRİŞ:**

Bu deneyde 4X4'lük tuş takımının ilk dört tuşunun (saat , dakika ve her ikisi içinde artırma ve azaltma tuşlarının) kullanılması incelenecektir. Zamanlamanın önemli olduğu durumlarda kesme altyordamlarının kullanılması gerekmektedir. Kesme altyordamı ve TMR0 zaman aşımı kesmesi incelenecektir.

- **YÖNTEM:**

4.1. Bilgisayarınızda TUS adlı bir directory açınız.
4.2. EK-5 de verilen programı MPLAB editöründe yazınız. Yazdığınız programı TUS/saat.asm olarak kaydediniz.
4.3. TUS/saat.asm kaynak kodunu TUS/saat.pjt adı ile proje dosyası haline getiriniz.
4.4. TUS/saat.pjt yi assemble ediniz(derleyiniz). Assemble işlemi başarılı mı?
Başarısız ise; yazdığınız TUS/saat.asm yi EK-5'deki program ile karşılaştırarak hatalarınızı düzeltiniz ve başarılı bir assemble yapılana dek 4.4'e kadar olan işlem basamaklarını tekrar ediniz.
4.5. Assemble başarılı ise TUS/saat.hex dosyası elde edilir.
4.6. MPLAB'ı kapatınız. Bilgisayarınızı kapatarak programlayıcıyı bilgisayarınıza bağlayınız. Programlayıcının program/deney anahtarını kullanarak program konumuna alınız.
4.7. Bilgisayarınızı açarak PRG-PIC yazılımını çalıştırınız.
4.8. Oluşturulan TUS/saat.hex dosyasını seçerek programlayıcıya programı yükleyiniz. "Programlama işlemi bitmiştir..." mesajını alana kadar bekleyiniz.
4.9. Program/Deney anahtarını deney konumuna alınız.
4.10. Deney kartını programlayıcıya takınız.
4.11. Deney kartının sağ üst köşesindeki LED/TUŞ jumperını TUŞ konumuna alınız.
4.12. Displaylerde ilk olarak ne gördünüz? İlk açılışta 0000 bilgisini gördüyseniz program çalışmaktadır. Görmüyorsanız programı tekrar kontrol ederek editöre yazınız.

4.13. **Kontrol tuşlarını kullanarak saati şöyle ayarlarız.**

Saat	Yukarı	Aşağı	Dakika

Şekil 5.1: 1X4'lük tuş takımında tuşların görevi

İlk açılışta karşımıza 00 00 bilgisi gelmektedir. Saat tuşunu basılı tutup yukarı ya da aşağı tuşuna basarak saati ayarlarız. Daha sonra dakika tuşunu basılı tutup aşağı veya yukarı tuşunu kullanarak dakikayı ayarlarız. Ayar yapabilmek için aynı anda 2 tuşa birden (saat veya dakika tuşuna sürekli, aşağı veya yukarı tuşuna istenilen değere gelene kadar) basmak gerekir. Bunun amacı saat çalışırken yanlışlıkla bir tuşa basıldığında saatin değerinin değişmemesidir. Yani bir tür koruma işlemi yapılmaktadır.

AYARLAR :	**KONTROL TUŞLARI**			
	Saat	Yukarı	Aşağı	Dakika
Saat ileri				
Saat geri				
Dakika ileri		**X**		**X**
Dakika geri			**X**	**X**

Şekil 5.2: Kontrol tuşlarının konumları

4.14. **KESME (INTERRUPT):**

Mikroişlemcilerde ve mikrodenetleyicilerde kesme kavramı programın herhangi bir anda kesilip; daha önceden belirlenmiş işlemlerin yapılıp daha sonra programa devam edilmesi olarak açıklanabilir.

Bunu sınıfta bir öğretmenin ders anlatmasına benzetecek olursak öğretmen konuları belli bir program dahilinde sıralayıp bu sıraya göre sınıfta anlatır. Eğer anlatma esnasında bir öğrenci parmak kaldırırsa öğretmen anlatmayı bırakıp öğrencinin sorusunu dinler. Değerlendirip cevaplandırdıktan sonra tekrar konuyu kaldığı yerden anlatmaya devam edecektir. Burada öğretmenin işlediği konuyu ana programa benzetirsek öğrenci de parmak kaldırarak dersi (programı) kesmiştir.

PIC mikrodenetleyicileri de program çalışırken herhangi bir anda gelen yazılım ve donanım kesmelerine cevap verebilir. PIC16F877'de 14 kesme vardır. Bunlardan 4 tanesi;

1. TMR0 sayıcısında FFH den 00H a geçerken oluşan zaman aşımı kesmesi (bu dijital saat deneyinde ayrıntılı olarak anlatılacaktır)
2. Port/B 4-7 bitlerinin durumlarındaki değişiklik
3. Port/B 0.biti harici kesmesi
4. EEPROM belleğe yazma işleminin tamamlamasında meydana gelen taşma

INTCON REGISTER:

INTCON (Interrupt control) kayıtçısı her bir kesme kaynağı için bir bayrak tutar. RAM bellekte 0BH adresinde bulunmaktadır.

Bit7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
GIE	**PEIE**	**T0IE**	**INTE**	**RBIE**	**T0IF**	**INTF**	**RBIF**

Bit 7; GIE: tüm kesme işlemlerini iptal etme bayrağı
0= tüm kesmeler geçersiz(disable)
1=aktif yapılmış olan tüm kesmeler(enable)
Bit6; PEIE: Çevrebirimlerinin kesme etkinlik biti
0=tüm çevresel kesmeler geçersiz (disable)
1=tüm engellenmemiş kesmeler geçerli (enable)
Bit5; T0IE: TMR0 sayıcı kesmesini aktif yapma bayrağı
0=geçersiz (disable)
1=geçerli (enable)
Bit4; INTE: Harici kesme aktif yapma bayrağı
0=geçersiz (disable)
1=geçerli (enable)
Bit3; RBIE: PORTB(4,5,6,7 bitleri) değişiklik kesmesini aktif yapma bayrağı
0=geçersiz (disable)
1=geçerli (enable)
Bit2; T0IF: TMR0 sayıcısı zaman aşımı bayrağı
0=Zaman aşımı yok
1=Zaman aşımı var
Bit1; INTF: Harici kesme bayrağı
0=Harici kesme yok
1=Harici kesme var
Bit0; RBIF: PORT/B değişiklik bayrağı
0=Değişiklik yok
1=Değişiklik var

TMR0 SAYICISI:

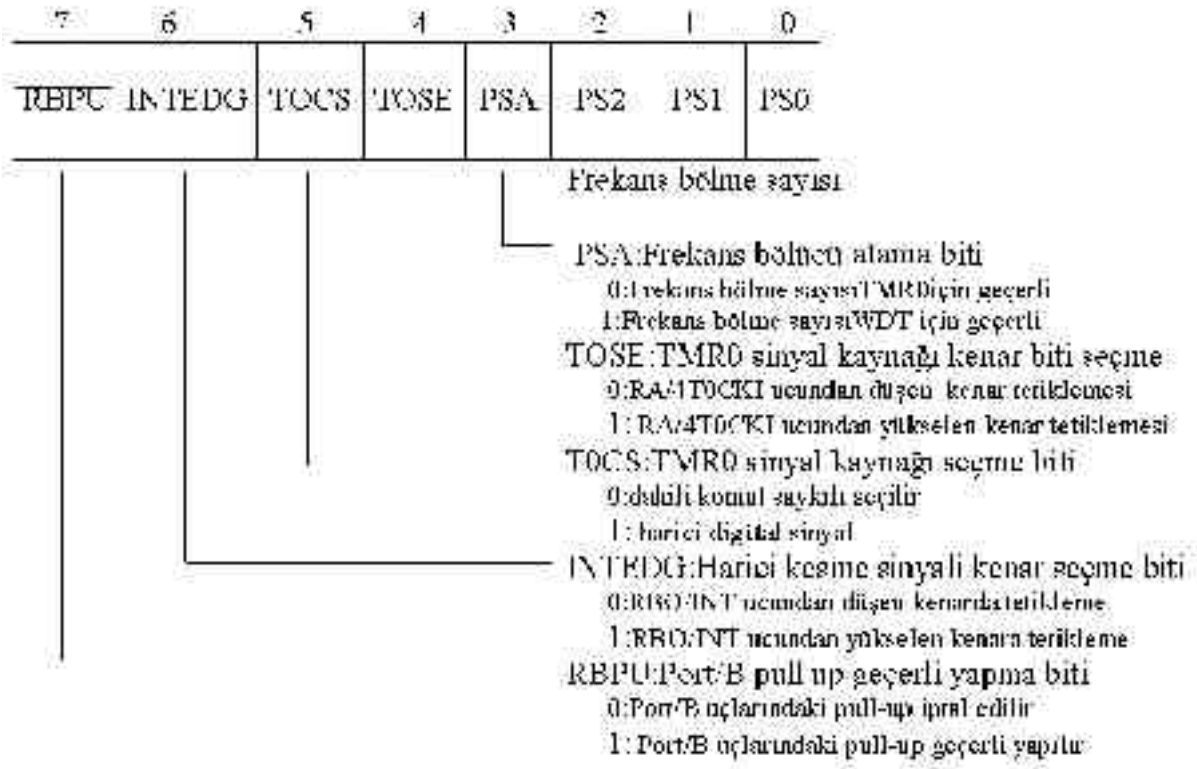
01h adresinde bulunan TMR0 kaydedicisi aynı zamanda 8 bitlik bir sayıcıdır. Bu sayıcının en önemli özelliği FFH tan 00h a geçerken meydana gelen taşmanın INTCON kayıtçısının TOIF bayrağını etkilemesidir. Bu bayrak kontrol edilerek TMR0 kesmesi oluşturulur.

OPTION REGISTER:

Option register/Seçenek kayıtçısı aşağıdaki özelliklere sahiptir;

- TMR0 ve WDT'da frekans bölme sayısı için gerekli bit durumlarının ayarlanması;
- TMR0 ve harici kesme sinyali kaynağı kenar seçme biti;

- Port/B pull up yapma bitini içeren 8 bitlik bir kayıtçıdır. Bu kayıtçı bank1 de 81H adresinde bulunmaktadır.



Şekil 5.3: OPTION register bitleri

FREKANS BÖLME SAYISININ BELİRLENMESİ:

Frekans bölme sayısı	TMR0 Oranı	WDT oranı
000	½	1/1
001	¼	½
010	1/8	¼
011	1/16	1/8
100	1/32	1/16
101	1/64	1/32
110	1/128	1/64
111	1/256	1/128

Şekil 5.4 : TMR0 ve WDT frekans bölme sayısının ayarlanması

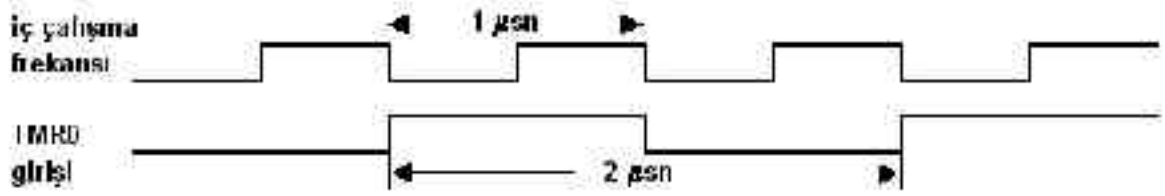
TMR0 ve WDT sinyalini bölmek için option kaydedicisinin ilk 3 biti (PS0-PS2) kullanılır. Bu kayıtçı ile TMR0 sayıcı sinyalini 8 ayrı frekans değerine bölebiliriz. Deney setinde 4 MHZ lik osilatör kullanıldı. Bu nedenle içerideki çalışma hızı 4/4 =1 Mhz dir.

T=1/F

T=1/1MHZ = 1 µsn

Yani her bir komutun işlemesi için 1 µsn süre gerekmektedir. TMR0 bir üst sayıya 1 µsn de bir geçecektir. Option kayıtçısında prescaler/önsayaç oranını ayarlayarak TMR0 veya WDT nin artma hızının kaç saat darbesinde bir geçeceğini belirleriz. TMR0 prescaler oranına göre bir üst sayıya geçtiği için:

Prescaler oranı oranı ½ ise TMR0 2 komut saykılında bir üst sayıya geçecek. Bir komutu saykılı 1 µsn olduğuna göre 2 komut saykılı 2 µsn sürecektir.



TMR0 2 µsn de bir üst sayıya geçecek ve FF h tan 00h a geçerken bir kesme göndereceği için (256 defa sayması gerekir)

TMR0 gecikmesi = Prescaler oranı x TMR0 değeri
= 2 µsn x 256
= 512 µsn lik gecikme elde edilir.

Prescaler oranını ve TMR0 sayma değerini ayarlayarak istediğimiz sürelerde gecikmeler elde edebiliriz. Saat deneyinde 4 msn'lik gecikme elde edebilmek için (TMR0 125 yüklendi) ve prescaler oranında 1/32 olarak aldık.

TMR0 gecikmesi = Prescaler oranı x TMR0 değeri
= 32 µsn x 125
= 4 msn

4.15.**KESME ALT YORDAMI :**

Int RTCC :Burada ilk olarak yapılan Option kayıtçısında prescaler oranını istenilen şekilde seçmek ve ayarlamaktır. TMR0 kayıtçısının sayma değeri belirlenir.TMR0 sayma değeri aşağıdaki şekilde yüklenir.

```
movlw   D'255'-D'124'   ; TMR0=125
movwf   TMR0
```

TMR0 kayıtçısına yüklenmesi gereken değer sayması gereken değerin 1 eksiği 255' ten çıkarılarak elde edilir. Daha öncede anlatıldığı gibi kesme olarak kullanılabilmesi için TMR0 içeriğinin FFh' dan 00h a geçmesi gerekir. Saat programında TMR0' a 131 değeri yüklenerek, kesme gelene kadar 125 defa sayması sağlanır. TMR0 gecikmesi hesaplanırken Prescaler oranı göz önüne alınmalıdır; çünkü TMR0' ın artışı prescaler oranına bağlı olarak artar. Saat programında TMR0 32 saykılda 1 artmaktadır.

Prescaler oranı ve TMR0 değerleri belirlendikten sonra ise tutulan timer_f bayrağı da bu altyordamda sıfırlanır. Daha sonra INTCON kayıtçısında 7. bit (GIE) 1 yapılarak kesmeler aktif hale getirilir. 5.bit (TOIE) 1 yapılarak ise TMR0 zamanlayıcı kesmesi aktif hale getirilir.

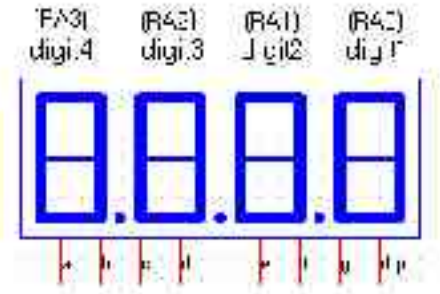
Service RTCC ***:***Programın çalışması esnasında sayan TMR0, FFh dan 00h a geçtiği zaman kesme oluşur, program Service RTCC kısmından devam eder. Burada ise ;

1. programın o andaki W ve Status kayıtçıları içeriği geçici olarak w_temp ve st_temp kayıtçılarına alınır.
2. Burada TOIF biti yani INTCON kayıtçısının 2.biti kontrol edilir. Eğer 0 ise Service Other kısmından devam edilir. 0 değil 1 ise TMR0 zaman aşımı kesmesi var demektir.
3. Timer_f bayrağı önce sıfırlanıp, sonra complementi alınarak set edilir. TMR0 değeri yeniden yüklenir.
4. INTCON kayıtçısının içeriği yeniden kesme geldiğinde kontrol edilebilir olması için temizlenir. TOIF biti yani 5.bit 1 yapılarak TMR0 zamanlayıcı kesmesi tekrar aktif hale getirilir.

Service Other ***:***Burada Service RTCC kısmında geçici kayıtçılara alınan W ve Status içerikleri yeniden yüklenir; ve retfie komutu ile *kesme alt yordamından* çıkılır. Ana program kaldığı yerden devam eder.

4.16. **TARAMA ALTYORDAMI:**

Dijital saat deneyinde yedi parçalı göstergelerin sürülmesi tarama alt yordamıyla yapılmaktadır. Adigit kaydedicisinde 1 bilgisi tutulur. Bu kaydedici ile Port/A'daki yedi parçalı göstergelerin katodları sırayla sürülür. İlk yedi parçalı göstergeyi sürüp digit1 kaydedicisinin içindeki bilgiyi Port/B'ye göndeririz. Böylece birinci yedi parçalı göstergenin yanması sağlanmıştır. Daha sonra adigit kaydedicisindeki 1 bilgisini kaydırarak tarama altyordamından çıkılır. Burada dört adet yedi parçalı gösterge kullanıldığı için adigit'teki kaydırma dört defa yapılmaktadır. 4'ten sonra adigit kaydecisine tekrar 1 bilgisi atılıp çıkılır.



Şekil 5.5: yedi parçalı gösterge

Adigit (Port/A)	Yanan yedi parçalı gösterge (Port/B)
00000001	Digit1
00000010	Digit2
00000100	Digit3
00001000	Digit4

Şekil 5.6: adigit'in kaydırılması sonunda yanan yedi parçalı gösterge

Tarama altyordamının her çağırılışında bir adet yedi parçalı gösterge sürülür. Program 4mS' de (her kesme geldiğinde) bir kere *tarama altyordamını* çağırır. Böylece saniyede 250 defa tarama yapıldığı için yedi parçalı göstergelerin hepsi yanıyormuş gibi algılanır.

TARTIŞMA

Deneyde ilk satırda saat, dakika, aşağı ve yukarı tuşlarının kullanılması RB4'ten 1 bilgisinin verilip, RB0_RB3 bitlerinde okunmasıyla gerçekleştirilmektedir. Burada tuşların yeri istenilen şekilde konumlandırılabilir. Tuşlar en alt satıra yerleştirilebilir. Tuşların yerini değiştirmek için programda ne gibi değişiklikler yapılmalıdır?

Option kayıtçısının hangi bitlerini değiştirerek prescaler oranını 16 yapabilirsiniz?Eğer prescaler oranını değiştirdiğinizde, zamanlama gecikmesini sabit tutmak istiyorsanız TMR0' da 2 kat artırmanız gerekmektedir.

DENEY NO : 6
DENEYİN ADI : 4X4'LÜK TUŞ TAKIMINDAN GİRİLEN HARFLERİN GÖRÜNTÜLENMESİ

- **DENEYİN AMACI:**

1.1. Alınan verinin farklı görüntüleme altyordamı kullanarak displayde görüntülenmesini sağlamak.
1.2. 4x4'lük tuş takımında tuşların nasıl kullanılacağını öğrenmek.

2) GEREKLİ MALZEME:

2.1. EA-16F877 mikrodenetleyici uygulama seti
2.2. Display ve Led deney modülü

3) GİRİŞ:

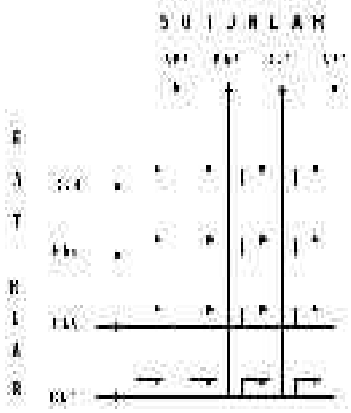
Bu deneyde 4x4'lük tuş takımından girilen harflerin displayde görüntülenmesi ve dört harfli anlamlı kelimeler üretilmesi incelenecektir.

4) YÖNTEM:

4.1. EK-6 da verilen programı MPLAB editöründe yazınız. Yazdığınız programı TUS/tus_th.asm olarak kaydediniz.
4.2. TUS/tus_th.asm kaynak kodunu TUS/tus_th.pjt adı ile proje dosyası haline getiriniz.
4.3. TUS/tus_th.pjt yi assemble ediniz(derleyiniz). Assemble işlemi başarılı mı? Başarısız ise; yazdığınız TUS/tus_th.asm yi EK-6'daki program ile karşılaştırarak hatalarınızı düzeltiniz ve başarılı bir assemble yapılana dek 4.3'e kadar olan işlem basamaklarını tekrar ediniz.
4.4. Assemble başarılı ise TUS/tus_th.hex dosyası elde edilir.
4.5. MPLAB'ı kapatınız. Bilgisayarınızı kapatarak programlayıcıyı bilgisayarınıza bağlayınız. Programlayıcının program/deney anahtarını kullanarak program konumuna alınız.
4.6. Bilgisayarınızı açarak PRG-PIC yazılımını çalıştırınız. Oluşturulan TUS/tus_th.hex dosyasını seçerek programlayıcıya programı yükleyiniz. "Programlama işlemi bitmiştir..." mesajını alana kadar bekleyiniz.
4.7. Program/Deney anahtarını deney konumuna alınız.
4.8. Deney kartını programlayıcıya takınız.
4.9. Deney kartının sağ üst köşesindeki LED/TUŞ jumperını TUŞ konumuna alınız.
4.10. Displaylerde ilk olarak ne gördünüz? İlk açılışta BETİ yazısını gördüyseniz program çalışmaktadır. Görmüyorsanız programı tekrar kontrol ederek editöre yazınız.

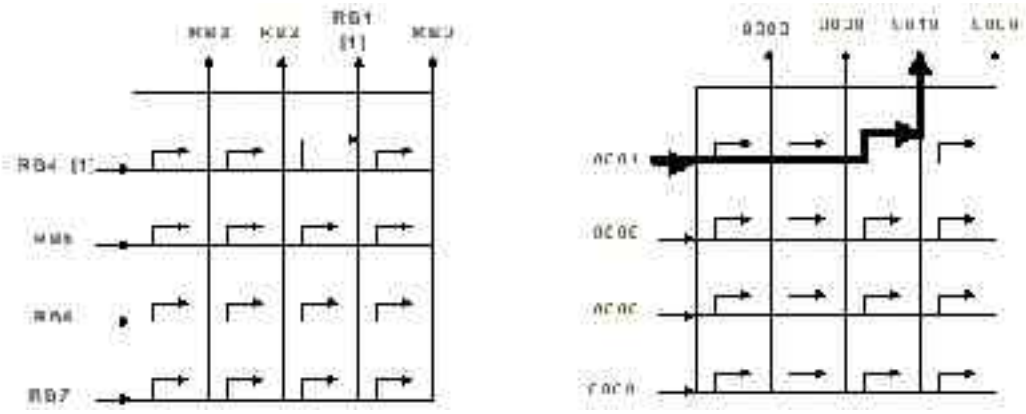
4.11.Tuş Takımının Kullanımı:

4x4'lük tuş takımını 4 satır ve 4sütundan oluşan bir matris gibi düşünebiliriz. Bu matrisin sütunlarını Port/B'nin ilk 4 biti (RB0-RB3), satırlarını ise Port/B'nin son 4 biti (RB4-RB7) kontrol etmektedir.



Şekil 6.1 : Tuş takımında satır ve sütunları oluşturan portun I/O durumları

Program yazılırken dikkat edilmesi gereken en önemli husus Port/B'nin RB0-RB3 bitlerinin giriş, RB4-RB7 bitlerinin de çıkış seçilmesidir. Çünkü; Port/B'nin RB4, RB5, RB6 ve RB7 pinlerine bağlanan diyotlar bu bitlerin yalnızca çıkış olarak kullanılmasına izin vermektedir. Port/B'nin RB0, RB1, RB2 ve RB3 pinleri ise hem giriş hem de çıkış olarak kullanılabilmektedir.



(a) **(b)**

Şekil 6.2 : (a) RB4 ten gelen "1" bilgisinin RB0-RB3 portlarından okunması
(b) 1 bilgisinin RB1 'de okunması tuşun basılı olması

Tuş takımından bilgi okunurken; satırlara (RB4-RB7) lojik 1 bilgisi sırayla gönderilir. Sütunlardan (RB0-RB3) kontrol edilir. Basılı olan tuşun bulunduğu sütundan 1 bilgisi alınır.

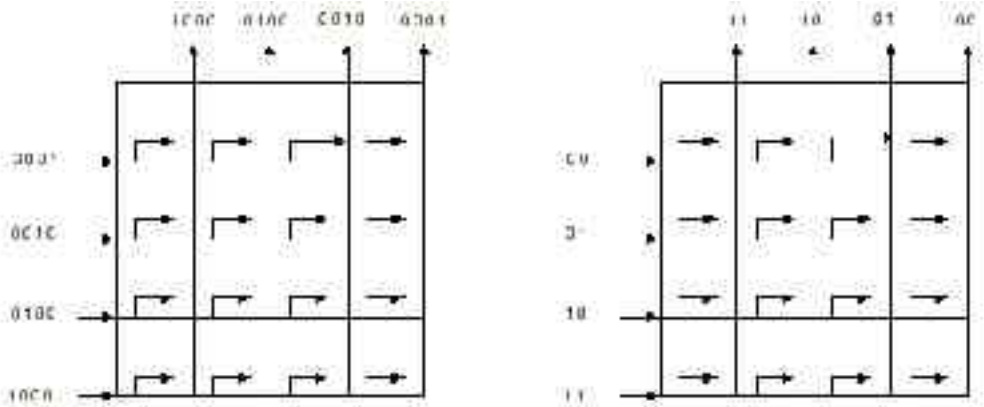
Örneğin; RB4' ten lojik 1 bilgisi gönderelim, bu satırdan herhangi bir tuşa basılmış ise bu tuşun bağlı olduğu Port/B bitine (RB0-RB3) 1 bilgisi iletilmiş olur. Eğer şekil 6.2'de görülen tuşa basılmış ise RB1'den 1 bilgisi okunacaktır. Port/B uçlarında b'00010010' bilgisi vardır. Programda bu işlemler tus_oku altyordamı ile yapılır.

4.12. **Tuş bilgisinin elde edilmesi :**

Herhangi bir tuşa basıldığı anda Port/B'de 8 bitlik bir bilgi oluşur. Bu 8 bitlik bilgi 256 (2^8) ayrı durumu ifade eder. Deney modülünde ise (4x4) 16 tane tuş bulunmaktadır. Yani 256 durumdan 16 tanesi tuş bilgisini ifade etmektedir. Port/B'deki 8 bitlik tuş bilgisini 4 bit ile tanımlamak için kod2bin dönüşüm tablosu kullanılır.

SÜTUN			SATIR		
PORTB	4 bitlik ifade	2 bitlik ifade	PORTB	4 bitlik ifade	2 bitlik ifade
RB0	0001	00	RB4	0001	00
RB1	0010	01	RB5	0010	01
RB2	0100	10	RB6	0100	10
RB3	1000	11	RB7	1000	11

Şekil 6.3 : 4 bitlik satır veya sütun bilgilerinin 2 bitlik ifadesi



(a) (b)

Şekil 6.4 : tus_bilgi kayededicisinin içeriği

a-)kod2bin tablosona gitmeden önce b-)sonra

tusbilgi = [00010010] tusbilgi = [0000/0001]

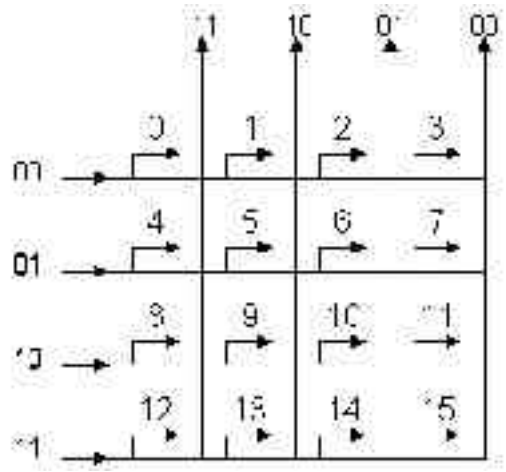
Kullandığımız her tuşa 0-15 arasında bir kod vermek için tuş bilgisini kod2bina tablosuna göndeririz. 00. satır 00. sutuna ait tuşa basıldığında tus_bilgi kaydedicisi 3 değerini alır (Şekil 6.5).

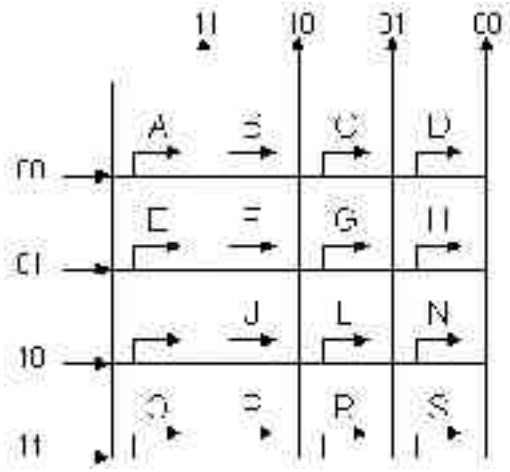
Programda üç tane çevrim tablosu kullanılmıştır. Bunlar:

1) **kod2bin tablosu:** Port/B'deki 8 bitlik bilgiyi 4 bitlik bilgiye çevirir.

2) **kod2bina tablosu:** Tuşlara 0 ile 15 arasında değer verir.

3) **kod tablosu:** Displayde görüntülenecek harflerin karşılığını verir.





Şekil 6.5 : kod2bina tablosu

Şekil 6.6 : kod tablosu

4.14.Çevir alt yordamı:

Tus_th programında her harfin karşılığı binary bir sayıdır. Bunu şekil 6.5 ve şekil 6.6 üzerinde görebiliriz. Örneğin; "A" harfinin karşılığı "0" yani b'00000000' ifadesidir. Bu binary ifadeler 8 bitlik sayıH ve sayıL kaydedicilerin de saklanmaktadır. SayıH kaydedicisini H ve L, sayıL'yi de H ve L olmak üzere 4 bitlik 4 kaydedici gibi düşünmekteyiz. Bu 4 kaydedici içeriği çevir alt yordamı içersinde kod tablosundan yedi parçalı gösterge kodları alınarak digitlere atanmaktadır. Yedi parçalı gösterge kodu alınan; sayıH/H digit4' e, sayıH/L digit3'e, sayıL/H digit2'ye ve sayıL/L digit1 kaydedilir.

Hexdecimal bilgi	H	L	H	L
Seven segment kod	digit4	digit3	digit2	digit1
Binary ifade	sayıH		sayıL	

İçerisin de binary bilgi bulunan sayıH ve sayıL kaydedicileri hane_kontrol alt yordamında H ve L olarak hexdecimal ifadeye dönüştürülmüştür. Sayıların H ve L kısmında 0-F arası hexdecimal bilgi bulunmaktadır.digit kaydedicileri içersinde ise istenilen harfi ilgili displayde yakacak yedi parçalı gösterge kodu bulunmaktadır. Aşağıda çevir alt yordamının bir parçasını incelersek;

dısplay4		
	movf	sayıH,w
	movwf	digit
	movlw	b'11110000'
	andwf	digit,f
	swapf	digit,w
	call	kod
	movwf	digit4

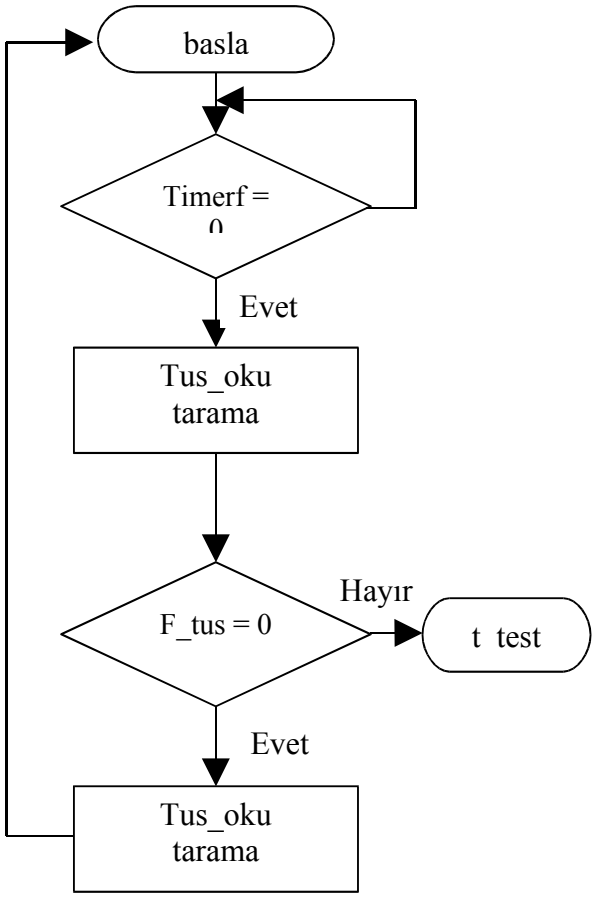
İlk olarak sayıH digit kaydedicisinde b'11110000' ile maskelenerek H kısmı alınır, kod tablosundan H kısmının karşılığı bulunur ve bu değer digit4'e eşitlenir. Bütün digitler bu şekilde kodları bulunarak Port/B'ye çıkış bilgisi olarak sıra ile gönderilir. Bu işlem ise *tarama alt yordamında* gerçekleşir.

TARTIŞMA :

Bu deneyde tuşları kullanmayı öğrendiniz. Dolayısıyla tuş takımındaki harfleri kullanarak en fazla 4 harfli anlamlı kelimeler türetilebilirsiniz.

Örneğin ; baba , para , pop , yalı , halı , sıra , nasa , darı , sene , fay , can , fare , sos , v.b.Bu kelimeleri çoğaltabiliriz. En fazla kaç tane kelime yazılabilir ? Düşünün !

Bu harfler dışında alfabedeki başka hangi harfleri tanımlayıp görüntüleyebilirsiniz. Örneğin U harfini displayde görüntülemek için programda ne gibi değişiklikler yapılmalı? programınızı düzenleyerek daha farklı harflerin görüntülenmesini sağlayabilirsiniz. Böylece kelime sayısını artırabilirsiniz. Deneyin !



Şekil 6.7: 4x4 tuş takımın akış diyagramı

DENEY NO : 7
DENEYİN ADI : 4- İŞLEM HESAP MAKİNESİ

1. DENEYİN AMACI:

1.1. Temel matematiksel işlemlerin yapılması.
1.2. 16 bitlik sayının 8 bit ile ifade edilmesi.

2) GEREKLİ MALZEME:

2.1. EA-16F877 mikrodenetleyici uygulama seti
2.2. Disolay ve Led deney Modülü

3) GİRİŞ:

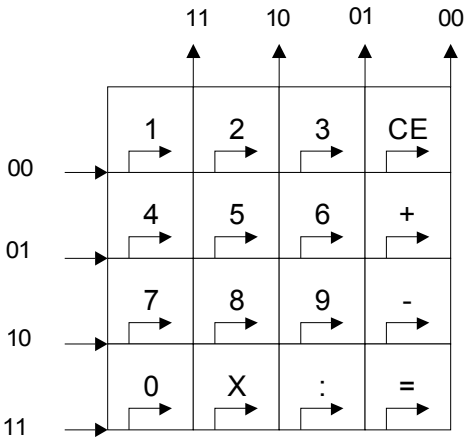
Bu deneyde PIC16F877 ile basit matematiksel işlemlerin gerçekleştirilmesi ve 16 bitlik toplama ve çıkarma işleminin nasıl yapıldığı detaylı olarak incelenecektir. Daha önceki deneylerdeki bilgilerinizi hatırlayarak 4X4' lük tus takımının hesap makinesi olarak kullanılması öğrenilecektir. Deneyde işlemler pozitif sayılar için yapılacaktır..

4) YÖNTEM:

4.1. EK-7 de verilen programı MPLAB editöründe yazınız. Yazdığınız programı TUS/saat.asm olarak kaydediniz.
4.2. TUS/hesap_m.asm kaynak kodunu TUS/hesap_m.pjt adı ile proje dosyası haline getiriniz.
4.3. TUS/hesap_m.pjt yi assemble ediniz(derleyiniz). Assemble işlemi başarılı mı? Başarısız ise; yazdığınız TUS/hesap_m.asm yi EK-7'deki program ile karşılaştırarak hatalarınızı düzeltiniz ve başarılı bir assemble yapılana dek 4.3'e kadar olan işlem basamaklarını tekrar ediniz.
4.4. Assemble başarılı ise TUS/hesap_m.hex dosyası elde edilir.
4.5. MPLAB'ı kapatınız. Bilgisayarınızı kapatarak programlayıcıyı bilgisayarınıza bağlayınız. Programlayıcının program/deney anahtarını kullanarak program konumuna alınız.
4.6. Bilgisayarınızı açarak PRG-PIC yazılımını çalıştırınız. Oluşturulan TUS/hesap_m.hex dosyasını seçerek programlayıcıya programı yükleyiniz. "Programlama işlemi bitmiştir..." mesajını alana kadar bekleyiniz.
4.7. Program/Deney anahtarını deney konumuna alınız.
4.8. Deney kartını programlayıcıya takınız.
4.9. Displayler de ilk olarak ne gördünüz? İlk açılışta displayde 0'ı gördüyseniz program çalışmaktadır. Görmüyorsanız programı tekrar kontrol ederek editöre yazınız.
4.10. Toplama ve çıkarma işlemi PIC mikrodenetleyicisinde ikilik 8 bit ve 16 bit olarak yapılır. Burada ondalık olarak 4 haneli pozitif sayılarla işlem yapmaktayız. Negatif sayılar ise 2' nin tersi(complement) alınarak bulunur.

Ondalık bir rakamı ifade etmek için 4 bit kullanırız. Dört basamaklı (2001) bir sayıyı ifade etmek için ise 16 bit kullanılır. Fakat PIC16F877'ün kaydedicileri 8 bittir. Yani bir kaydedici ile en fazla ondalık olarak iki basamaklı bir sayıyı ifade edebiliriz. Dört basamaklı herhangi bir sayı için ise 2 tane kaydedici kullanılır. (***1 nibble= 4 bit 1 bayt= 8 bit 2 bayt= 16 bit***)

4.11. **Fonksiyon Alt Yordamı :**



Şekil 7.1: Hesap makinesinde kullanılan tuşlar

Fonksiyon alt yordamında; fonksiyon tuşlarından birine basıldığında yapılacak olan matematiksel işlem belirlenir. Yukarıda şekil 7.1'de 4X4'lük tuşların hesap makinesi deneyine göre düzenlemesi görülmektedir.

Tuş takımı CE silme, matematiksel işlemler(toplama, çıkarma, çarpma, bölme ve eşitlik) ve 0-9 arası sayıları içermektedir.

Programda yapılacak olan işlem belirlenirken; basılı olan tusun karşılığı olan kod fonksiyon alt yordamında matematiksel işlemlerin kodları ile karşılaştırılır.

İŞLEM	
CE (SİLME)	***B'00001100'***
TOPLAMA	***B'00001101'***
	B'00001110'
	B'00001111'

Tablo 7.1. İşlemlerin kodları

Örneğin "+" tusuna basıldı ise fonksiyon altyordamında bu tusun kod karşılığı işlem kodu olan ***B'00001101'*** bilgisine eşit olacağından toplama işlemi gerçekleştirilir.

Bu deneyde sadece matematiksel işlem olarak CE silme, eşitlik, toplama ve çıkarma işlemleri yapılabilmektedir. Çarpma ve bölme işlem kodları tanımlı olmadığından yapılamamaktadır.

4.12. **CE silme işlemi:**

Tuş takımında "CE" tuşuna basıldığında displaydeki bilgi sıfırlanmakta fakat yapılan işlemin sonucu değişmemektedir. Sıfırlama işlemi ise *sayıH* ve *sayıL* kaydedicilerinin içeriğinin sıfırlanması ile sağlanmaktadır. Silme işlemi aşağıdaki gibi yapılır.

CE

```
movlw   00H
movwf   sayıH
movwf   sayıL
return
```

4.13. **Toplama işlemi :**

Matematiksel işlemler için gerekli olan komutlar;

KOMUT	AÇIKLAMA
Addlw	Sabit sayıyı W kaydedicisi ile toplar.
Addwf	W kaydedicisi ile F kaydedicisini toplar.
Sublw	Sabit sayıdan W kaydedicisini çıkarır.
Subwf	W kaydedicisinden F kaydedicisini çıkarır.
Rrf	F kaydedicisi içerisindeki bitleri elde(carry) ile birlikte 1 bit sağa kaydırır.
Rlf	F kaydedicisi içerisindeki bitleri elde(carry) ile birlikte 1 bit sola kaydırır

Tablo 7.2. İşlemler için kullanılan komutlar

Girilen sayılar toplandıktan sonra , işlem sonucu reg_Hh, reg_Hl, reg_Lh ve reg_Ll kaydedicilerinde saklanır. Sonucun görüntülenebilmesi için bu kaydediciler sayıH ve sayıL kaydedicilerine eşitlenir. Aşağıda görüldüğü gibi;

reg_Hh (4 bit)	**reg_Hl (4 bit)**	**reg_Lh (4 bit)**	**reg_Ll (4 bit)**
***SayıH* (8 bit)**		***sayıL*(8 bit)**	

Tabloda kaydedicilerin eşitliği görülmektedir. Sonuç 8 bitlik iki kaydedicide saklanır. Bütün işlemlerin sonucu bu şekilde görüntülenir.

4.14. **Çıkarma işlemi :**

Çıkarma işleminde işlemin yapılabilmesi için önce büyük sayı daha sonra küçük sayı girilmelidir. Çünkü deneyde negatif işlemler yapılmamaktadır. Eğer 2. Sayı büyük değerde ise sonuç ilk girilen sayı olacak, çıkarma işlemi gerçekleşmeyecektir.

Programda yapılan işlemin hangisi olduğu bir bayrak tarafından belirlenir. Bu bayrak programda fonk_f kaydedicisi ile tanımlıdır. "=" tuşuna basıldığında bu bayrağa bakılır, eğer fonk_f = 1 ise yapılan işlem toplama işlemidir, fonk_f = 0 ise işlem çıkarma işlemidir.

TARTIŞMA:

Sizde kendinize göre hesap makinesi için bir tuş takımı düzenlemesi yapıp programda gerekli değişiklikleri yaparak deneyin! Çarpma ve bölme işlemleri için altyordamlar yazarak programa ekleyip programı yeniden deneyin.

DENEY NO : 8
DENEYİN ADI : OPTİK ASANSÖR SİMÜLASYONU

1) DENEYİN AMACI:

- INCLUDE dosyalarının kullanımını öğrenmek.
- Optik asansör simülasyonunu öğrenmek.
- Bit tanımlama işlemini öğrenmek.

2) GEREKLİ MALZEME:

a. EA-16F877 mikrodenetleyici uygulama seti
b. Asansör Deney Modülü

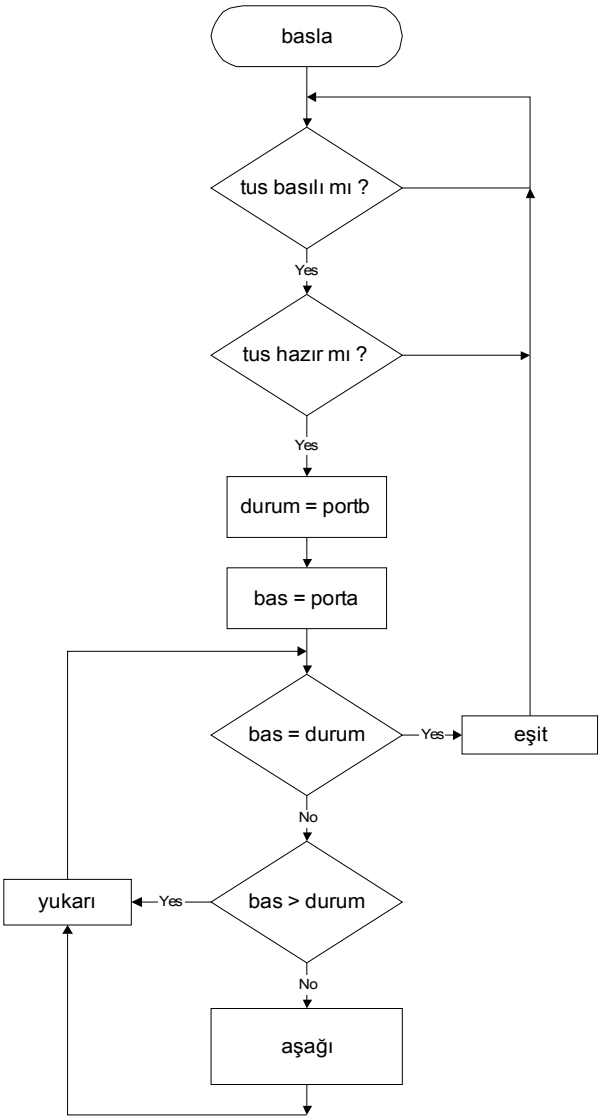
3) GİRİŞ:

Bu deneyde mekanik asansörün optik asansör olarak simüle edilmesi incelenecektir.

4) YÖNTEM:

1. Bilgisayarınızda ASA adlı bir directory açınız.
2. EK-8 de verilen programı MPLAB editöründe yazınız. Programa ait akış diyagramı şekil 8.1'de gösterilmiştir. Yazdığınız programı ASA/asansor.asm olarak kaydediniz.
3. ASA/asansor.asm kaynak kodunu ASA/asansor.pjt adı ile proje dosyası haline getiriniz.
4. ASA/asansor.pjt yi assemble ediniz(derleyiniz). Assemble işlemi başarılı mı? Başarısız ise; yazdığınız ASA/asansor.asm yi EK-8'deki program ile karşılaştırarak hatalarınızı düzeltiniz ve başarılı bir assemble yapılana dek 4.4'e kadar olan işlem basamaklarını tekrar ediniz.
5. Assemble başarılı ise ASA/asansor.hex dosyası elde edilir.
6. MPLAB'ı kapatınız. Bilgisayarınızı kapatarak programlayıcıyı bilgisayarınıza bağlayınız. Programlayıcının program/deney anahtarını kullanarak program konumuna alınız.
7. Bilgisayarınızı açarak PRG-PIC yazılımını çalıştırınız.
8. Oluşturulan ASA/asansor.hex dosyasını seçerek programlayıcıya programı yükleyiniz. "Programlama işlemi bitmiştir..." mesajını alana kadar bekleyiniz.
9. Program/Deney anahtarını deney konumuna alınız.
10. Deney kartını programlayıcıya takınız.
11. Displaylerde ilk olarak ne gördünüz? İlk açılışta alttaki yedi parçalı göstergede 0'ı gördüyseniz program çalışmaktadır. Görmüyorsanız programı tekrar kontrol ederek editöre yazınız.
12. Bu deneyde kat butonlarından birine bastığımızda o kata ait yedi parçalı gösterge ve asansörün yönüne göre de aşağı ya da yukarı ledleri yanacaktadır. Asansörün katlar arasındaki geçişini ise kırmızı ledler göstermektedir.

Port/B'nin RB0, RB2, RB4 ve RB6 pinleri katları gösteren yedi parçalı göstergelere RB1, RB3 ve RB5 pinleri kat arası ledlerine bağlanmıştır. Port/A'nın RA0, RA1, RA2 ve RA3 pinleri kat butonlarına bağlıdır.



Şekil 8.1: Asansor programının akış diyagramı

4.13. **INCLUDE Dosyaları:**

Assembly komutları ile bir program yazarken kullanacağımız kaydedicileri tanımlamamız gerekir. Bu işlem de programın giriş kısmında yapılır. PIC mikrodenetleyicilerinin RAM belleğinde özel kaydedicilerin adresleri sabittir. Dolayısıyla programı yazarken bu adresleri tekrar tanımlamak gereksizdir. Bu yüzden INCLUDE dosyaları oluşturulmuştur. MPASM ile derlenecek her PIC mikrodenetleyicisi (PIC16F84, PIC16F877 vb.) için ayrı bir INCLUDE dosyası vardır. Bu dosyalar MPLAB programı içerisindedir.

include <p16F877.inc> komutunu kullanarak MPLAB'ın içindeki .inc uzantılı dosyayı kullanmış oluruz. Bu komutu kullanılmazsa bütün kaydediciler EQU komutu kullanılarak aşağıdaki gibi tanımlanır.

RP0	EQU	H'0005'
PCL	EQU	H'0002'
STATUS	EQU	H'0003'
PORTA	EQU	H'0005'
PORTB	EQU	H'0006'
TRISA	EQU	H'0085'
TRISB	EQU	H'0086'

4.14. **Bit Tanımlama:**

 # define deyimi program içinde tanımlanan kaydedicilerin bitlerine isim atamak için kullanılır. Bu programcıya bit işlem komutlarını (bcf, bsf , btfsc , btfss gibi) kullanırken büyük kolaylık sağlar.

Programda Port/A'nın RA0, RA1, RA2 ve RA3 bitleri kat butonları için kullanıldı. Bu bitlere uygun isimler verilerek bu butonların kontrolü yapılır. Bit tanımlama işlemi aşağıdaki gibi yapılır.

#	define	Bite verilen ad	Kullanılan kaydedici	Tanımlanan bit
#	define	katb0	porta,	0

Böylece Port/A'nın RA0 bitini "katb0" olarak adlandırılır. Artık program içerisinde *Porta,0* yerine *katb0* kullanılır. Aşağıdaki gibi.

basla			
	btfss	katb0	;katb0 basılımı
	goto	ttest	;evet ttest git
	btfss	katb1	; hayır katb1 basılımı
	goto	ttest	;evet ttest git
	btfss	katb2	; hayır katb2 basılımı
	goto	ttest	;evet ttest git
	btfss	katb3	; hayır katb3 basılımı

TARTIŞMA:

Kat sayısını artıramayız. Neden? Katta kalma sürelerini artırarak programı yeniden düzenleyin. Artık sizde kendi programlarınız için include dosyalarını oluşturabilirsiniz. Böylece programınızın .asm dosyasını daha kısaltmış olursunuz. Fakat .hex uzantılı dosyanızın boyutu değişmemektedir.

DENEY NO : 9

DENEYİN ADI : ADIMLI MOTOR ARABİRİMİ
ve
KONTROLÜ UYGULAMASI

2. DENEYİN AMACI:

- Adımlı motorun kontrolünü öğrenmek.
- Bir adımlı-motorun maksimum dönme hızını göstermek

2) GEREKLİ MALZEME:

- EA-16F877 mikrodenetleyici uygulama seti
- Adımlı Motor Deney Modülü

3) GİRİŞ:

Adımlı motorların temel özelliklerini kullanarak, PIC16F877 mikrodenetleyicisi ile bir adımlı motorun dönme yönünün ve dönüş hızının nasıl kontrol edilebileceği incelenecektir.

4) YÖNTEM:

4.2.Bilgisayarınızda MOTOR adlı bir directory açınız.
4.3.EK-9 da verilen programı MPLAB editöründe yazınız. Yazdığınız programı MOTOR/amotor.asm olarak kaydediniz.
4.4.MOTOR/amotor.asm kaynak kodunu MOTOR/amotor.pjt adı ile proje dosyası haline getiriniz.
4.5.MOTOR/amotor.pjt yi assemble ediniz(derleyiniz). Assemble işlemi başarılı mı? Başarısız ise; yazdığınız MOTOR/amotor.asm yi EK-9'daki program ile karşılaştırarak hatalarınızı düzeltiniz ve başarılı bir assemble yapılana dek 4.4'e kadar olan işlem basamaklarını tekrar ediniz.
4.6.Assemble başarılı ise MOTOR/amotor.hex dosyası elde edilir.
4.7.MPLAB'ı kapatınız. Bilgisayarınızı kapatarak programlayıcıyı bilgisayarınıza bağlayınız. Programlayıcının program/deney anahtarını kullanarak program konumuna alınız.
4.8.Bilgisayarınızı açarak PRG -PIC yazılımını çalıştırınız.
4.9.Oluşturulan MOTOR/amotor.hex dosyasını seçerek programlayıcıya programı yükleyiniz. "Programlama işlemi bitmiştir..." mesajını alana kadar bekleyiniz.
4.10.Program/Deney anahtarını deney konumuna alınız.
4.11.Deney kartını programlayıcıya takınız.
4.12.Displayler de ilk olarak ne gördünüz? İlk açılışta yedi parçalı göstergede 0'ı gördüyseniz program çalışmaktadır. Görmüyorsanız programı tekrar kontrol ederek editöre yazınız.

4.12. **Adımlı Motorun Çalışması:**

Senkron motorların tek çalışma modları vardır ki bu da "sürekli dönme"dir. Adımlı-motorlar ise küçük ve hassas bir açı (veya adım) kadar döndürülebilir ve orada durdurulabilir. Her adımı attırabilmek için motor sargılarına sayısal akım darbesi uygulamak gerekir. Böyle (n) darbe peş peşe uygulandığında, motor (n) adım döner ve durur. Adım açısı motor tasarımına bağlı olup genelde 1.8 ile 30 derece arasında değişmektedir. Bu deneyde önerilen

motorların adım açısı 7.5 derecedir ki bu da bize motorun 48 adımda bir tur tamamlayacağını gösterir. Akım darbeleri motor sargılarına kontrol transistörleri ile uygulanır. Darbelerin uygulama sırası, motorun dönme yönünü, uygulama sıklığı ise motorun dönme hızını tayin eder.

Standart adım açıları ;
1.8^o - 200 adımda bir tur
3.75^o - 96 adımda bir tur
7.5^o - 48 adımda bir tur
15^o - 24 adımda bir tur.

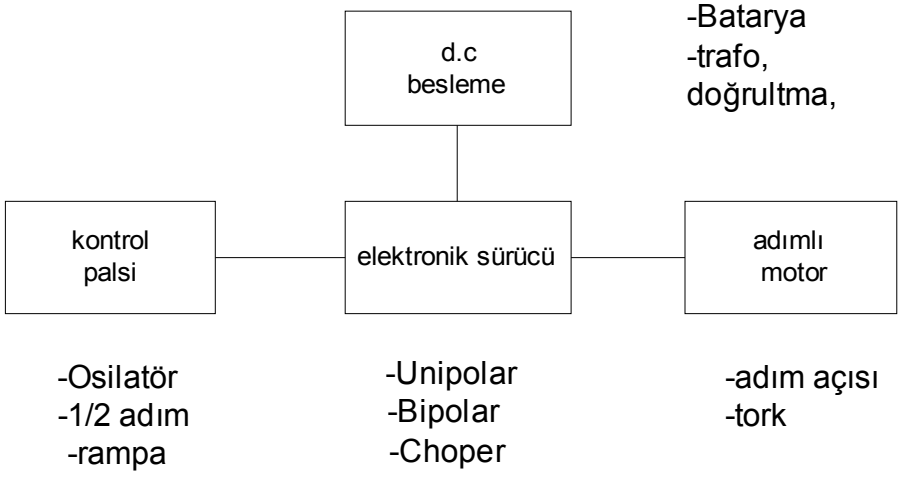
Adım sayısı ile adım açısının çarpımı motor milinin hareket açısını verir. Örneğin 7.5^o'lik 6 adım motora 45^o'lik hareket verir.

Adımlı motorlar genellikle dijital kontrol işlemlerinde, elektronik cihazların kesin ve hızlı hareket etmeleri gereken durumlarda kullanılır. Bunlar;

- Manyetik teyp sürücüleri,
- Tele teyp ve şerit yazıcılar,
- Kamera iris kontrolü,
- Plotterlerin ayarlanmasında, artan grafik kayıtçıları ve değişken hızlı grafik sürücüleri,
- Medikal aletler; kan örnekleyici, akciğer analizörleri, diyaliz pompaları,
- Sıvı yakıt kontrolü, valf kontrolü,
- Taksimetreler, kart okuyucular, üretim bandı pals sayıcıları, kantarlarda ve etiketleme sistemlerinde,
- Dijital analog çeviricilerde ve uzaktan kontrollü cihazlarda....

Bu örneklerin hepsinde ortak nokta harekettin kontrolüdür. Hareket ve/veya pozisyon kontrolünün gerekli olduğu her yerde step motor kullanılabilir. Bu genelde daha avantajlıdır.

Uygulamada kullanılacak adımlı motor sürücü devre ve motor karakteristiklerine göre seçilir. Aşağıdaki şekil adımlı motor sisteminin dört önemli öğesini göstermektedir.



Çalışma voltajı	12 volt
Tipik faz akımı	175 mA
Adım açısı	7.5 derece
Max. hız	40 adım/san.
Tork	20 mNm
Ağırlık	170 gr.

Tablo 9.1: 9904 112 31004 seri kodlu adımlı motor özellikleri

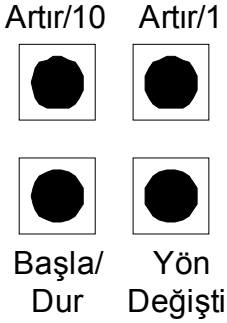
Adımlı motorların dönen kısmı (rotor) sabit mıknatıstan yapılmaktadır. Duran kısmında (stator) ise belirli aralıklarla yerleştirilmiş elektromıknatıslar bulunmaktadır. Elektromıknatısın içerisinden geçen akımın yönüne göre N-S kutuplarının yönü de değişir. Bir adımlı motorun döndürülmesi için belli bir sırayla bu elektromıknatısların enerjilenmesini sağlayan gerilimler motor uçlarından uygulanır. Böylece sabit mıknatıs, duran kısmın enerjilenen kutupları tarafından yönlendirilir.

	Sargı 1	Sargı 2	Sargı 3	Sargı 4	
	1	1	0	0	
↓	0	1	1	0	↑
Saat	0	1	0	1	Saat
yönü	1	0	0	1	Yönünün Tersi

Tablo 9.2: 9904 112 31004 seri kodlu adımlı-motorun adım dizesi

Genel olarak adımlı motorlarda dört kablo (sargı) çıkışı vardır (bipolar adımlı motor). PIC mikrodenetleyicisinin çıkış uçlarını adımlı motorun çıkış kablolarına direkt bağlamak doğru değildir. Çünkü PIC16F877'ün verebileceği akım 20-25 mA arasındadır. Adımlı motorun çekeceği akım ise daha fazladır. Bundan dolayı PIC16F877 ile adımlı motor sürmek istenilirse aşağıdaki gibi transistörlü sürücü devre kullanılmalıdır.

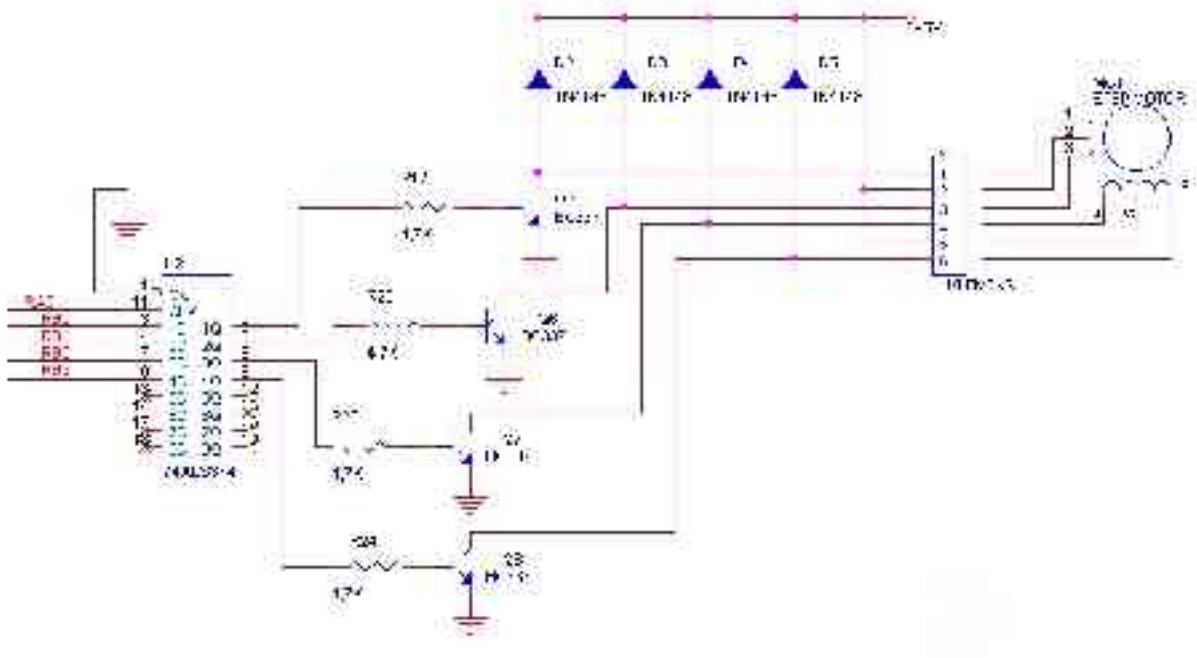
4.13. **Tuşların Görevleri:**



Artır/10: Yedi parçalı göstergenin onlar hanesini 0 ile 9 arasında artırır. Aynı zamanda gecikme süresininde onlar basamağı artırılır.

Artır/1: Yedi parçalı göstergenin birler hanesini 0 ile 9 arasında artırır. Aynı zamanda gecikme süresininde birler basamağı artırılır.

Başla/Dur: Adımlı motorun dönmesini başlatır. Eğer adımlı motor dönerken tekrar basılırsa durdurur.
Yön Değiştir: Tuşa her basıldığında adımlı motorun dönme yönünü değiştirir.



Şekil 9.1 :Adımlı motor sürücü devresi

TARTIŞMA:

Bu şekilde gerçekleştirilebilecek bir tasarım ile neler yapılabilir. Örneğin burada kullanılan butonlar yerine foto-transistör kullanılabilir. Buton basılı mı yerine ışık var mı/yok mu sorgulaması yapılır. Ona göre; adımlı motor çalışır yada durur. Bu şekilde ışık kontrollü bir adımlı motor uygulaması gerçekleştirilir. Daha farklı uygulamaları siz de deneyebilirsiniz.

DENEY NO : 10
DENEYİN ADI : ANALOG-DİJİTAL (A/D) KONVERTÖR UYGULAMASI

1 DENEYİN AMACI:

1.1 A/D dönüştürücü uygulaması.
1.2 OP-AMP ın buffer olarak kullanılması.
1.3 LM 336 ile referans kaymasının sağlanması.
1.4 LM 335 Sıcaklık sensorünün kullanılması.
1.5 Mikrodenetliyici ile sıcaklık kontrol sisteminin gerçekleştirilmesi.

2 GEREKLİ MALZEME:

- EA-16F877 mikrodenetleyici uygulama seti,
- Sıcaklık kontrol deney modülü,

3 GİRİŞ:

Bu deneyde; dışardan girilebilecek 0-99 °C arası bir sıcaklık değeri ile ortamın o anki

sıcaklık değeri karşılaştırılır. Karşılaştırma sonucu ortamın sıcaklığı daha düşük ise ısıtıcı çalışır, eğer ortamın sıcaklığı istenen değerde veya daha yüksek ise ısıtıcı çalışmaz.

4 YÖNTEM:

1.1. Bilgisayarınızda ADC adlı bir directory açınız.
1.2. EK-10 da verilen programı MPLAB editöründe yazınız. Yazdığınız programı ADC/adc.asm olarak kaydediniz.
1.3. ADC/adc.asm kaynak kodunu ADC/adc.asm adı ile proje dosyası haline getiriniz.
1.4. ADC/adc.pjt yi assemble ediniz(derleyiniz). Assemble işlemi başarılı mı? Başarısız ise; yazdığınız ADC/adc.asm yi EK-9'daki program ile karşılaştırarak hatalarınızı düzeltiniz ve başarılı bir assemble yapılana dek 4.4'e kadar olan işlem basamaklarını tekrar ediniz.
1.5. Assemble başarılı ise ADC/adc.hex dosyası elde edilir.
1.6. MPLAB'ı kapatınız. Bilgisayarınızı kapatarak programlayıcıyı bilgisayarınıza bağlayınız. Programlayıcının program/deney anahtarını kullanarak program konumuna alınız.
1.7. Bilgisayarınızı açarak PRG -PIC yazılımını çalıştırınız.
1.8. Oluşturulan ADC/adc.hex dosyasını seçerek programlayıcıya programı yükleyiniz. "Programlama işlemi bitmiştir..." mesajını alana kadar bekleyiniz.
1.9. Program/Deney anahtarını deney konumuna alınız.
1.10. Deney kartını programlayıcıya takınız.
1.11. Displayler de ilk olarak ne gördünüz? İlk açılışta yedi parçalı göstergede 0'ı gördüyseniz program çalışmaktadır. Görmüyorsanız programı tekrar kontrol ederek editöre yazınız.

Başla tuşuna basarak o anki ortamın sıcaklığını görebilirsiniz. Artır butonuna basarak istediğiniz bir sıcaklık değeri ayarlayınız(örneğin 29º C). Sıcaklık değerini ayarladıktan sonra ısıtıcı ledini gözleyiniz. Ayarlanan sıcaklık değeri ortamın sıcaklığından daha büyükse ısıtıcı ledi yanar. Daha küçükse bu led söner.

Isıtıcı ledi yanıyorsa LM 335 sıcaklık sensörünü ısıtınız (örneğin parmağınızla) ve ısıtıcı ledini gözleyiniz. Belirli bir süre sonra ısıtıcı ledi sönecektir. Eğer vücut ısısı ısıtıcı ledinin sönmesi için yeterli değil ise bir ısı kaynağı örneğin havya kullanarak bir tornavidayı çok az ısıtıp LM 335 e dokundurarak sönmesini temin ediniz. Isıtıcı ledinin sönmesi ortamın sıcaklığının; sizin artır butonu ile ayarladığınız değere ulaşmış olduğu anlamına gelir. Isıtıcı ledi söndükten sonra LM 335 den ısıttığınız tornavidayı ya da elinizi çekin ve bir süre sonra ısıtıcı ledinin tekrar yandığını göreceksiniz. Isıtıcı ledinin tekrar yanması ortamın sıcaklığının ayarlanan değer altına düştüğünü göstermektedir.

4.12. **LM335 Sıcaklık Sensörü :**

Sıcaklığı gerilime dönüştüren devre elemanlarından biri de LM335'dir. National firmasının (TO-46 ve TO-92 transistör paketinde) ürettiği bir entegredir. LM335 kolay kalibre edilebilen bir sıcaklık sensörüdür. Bu eleman sıcaklığa karşı duyarlı bir zener diyot gibi görev yapmaktadır. Ortam sıcaklığının 1 değişmesi durumunda elemanın çıkışındaki gerilim 10 mV değişmektedir. 0=273 olduğundan LM335 çıkış gerilimi ifadesi:

Vz = V0+VX .T şeklinde yazılabilir. Burada:
V0 = 2,73 V,
VX = 10 mV/ ve
T = olarak ortam sıcaklığıdır.
T = 20 iken Vz =2,93 V bulunur.

Bu demektir ki LM335 çıkışında 20de 2.93 V görülür. LM335 1 Ω dan daha düşük bir dinamik empedansı vardır. 400µA den 5 mA kadar geniş bir akım değişmesine karşı performansı değişmez. LM335 in en önemli özelliği lineer bir çıkışının olmasıdır. Düşük empedans ve lineer çıkışı sayesinde kullanımı çok kolaydır. -40°C ile 100°C arasında 1°C den daha düşük bir hata ile çalışır.

4.13. **LM336 2.5V Referans Kaynağı :**

Ölçme ve kontrol devreleri için gerekli Vref gerilimleri dış etkenlere bağımlılığı çok az olan referans gerilim kaynakları ile sağlanmalıdır. Aksi halde ölçme ve kontrol işlemlerinde hatalar olur. LM336 referans gerilim kaynağı ile bu kararlı gerilimler sağlanabilir.

4.14. **OP-AMP ın buffer olarak kullanılması :**

Genel olarak op-amp, çok yüksek kazançlı bir DC yükselteçtir. Çeşitli özellikleri, devreye dışardan bağlanan devre elemanları ve bunların sağladığı geri besleme ile, kontrol altına alınabilir.

Op-amp devresi tek başına düşünüldüğünde, 5 önemli özelliğe sahiptir. Bunlar;

- Kazancı çok fazladır (örneğin 200.000).
- Giriş empedansı çok yüksektir (5 M Ω).
- Çıkış empedansı sıfıra yakındır.
- Bant genişliği fazladır (1MHz gibi)
- Girişe 0 volt uygulandığında çıkışta yaklaşık 0 volt elde edilir.

Op-amp'a dışardan çeşitli şekillerde geri besleme devreleri eklenerek değişik çalışması sağlanabilir. Op-amp bu deneyde buffer (tampon) olarak kullanılmaktadır. Gerilim izleyici de denebilir, yani gerilim kazancını 1 ve giriş-çıkış işaretlerinin aynı fazda olduğu bir yükselteçtir.

4.15. ADC0804 Analog-Dijital Çevirici :

ADC0804, 8 bitlik bir CMOS A/D çeviricidir. Tüm mikroişlemcilerle kolay bir arabirim oluşturduğu gibi tek başına da çalışma özelliğine de sahiptir. Yaygın olarak kullanılan mikroişlemcilerle çalışmak üzere tasarlanmıştır. İçerisinde eşdeğer devre olarak 256 dirençten oluşan bir direnç grubu vardır. Analog anahtarlar, ardışıl yaklaşımlı lojik tarafından devreye sokulup çıkarılarak, farklı analog gerilimlerinde 256 dirençli grubu devreye sokup çıkarırlar. İlk olarak MSB biti test edilir. Daha sonra 64 clock periyodunda 8 bitlik binary kod (11111111 tam skala) çıkış tutuculara transfer edilir ve INTR (Interrupt) çıkışı lojik-0'a düşer. Entegre tek başına serbest modda çalıştırılacaksa INTR ucu WR ucuna bağlanmalı ve CS ucu da lojik 0'a bağlanmalıdır.

ADC0804'ün teknik özellikleri:

- Bütün mikroişlemcilerle kolay kullanım veya tek başına kolay bağlantı
- Diferansiyel voltaj girişleri
- Lojik giriş ve çıkışların MOS ve TTL voltaj seviyesi uygunluğu
- Chip üzerinde saat üreteci
- 0-5 V giriş aralığında 5 volt besleme kaynağı ile çalışabilme
- sıfır ayarı gerektirmemesi
- 0.3'' standart 20 pin DIP paket
- 8 bit çözünürlük
- ±1 LSB maksimum hata
- 100 uS dönüştürme zamanı

4.16. Sıcaklık Kontrol Devresi:

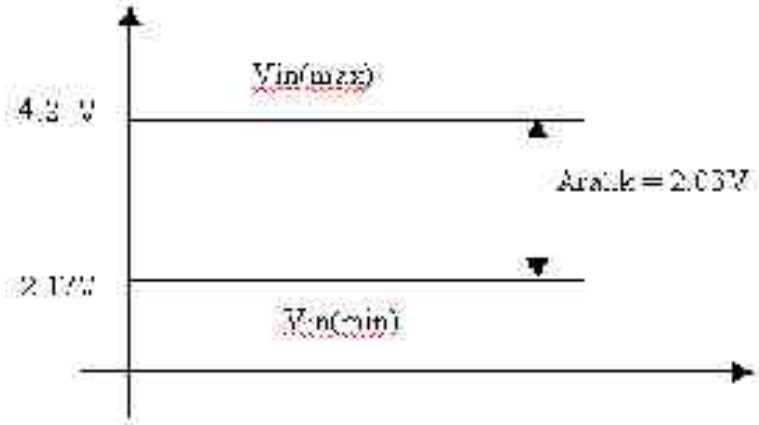
Bu deneyde sıcaklık algılayıcı olarak LM335 sıcaklık sensörü kullanılmıştır. Bu sensörün üzerine 25ºC (298ºK) de 2.98V düşmekte ve her bir ºC için +10mV artmaktadır . ADC0804'ün VREF/2 ve VIN(-) girişleri 0ºC ve 100ºC aralığında ölçülecek şekilde ayarlanmıştır. Bunun için sıcaklık sensörü 0ºC'deki su-buz karışımı içine sokulur. Teorik olarak bu sırada sensör uçlarında 2.73V olmalıdır. ADC çıkışında (0000 0000) görülecek şekilde (1K değerindeki çok turlu trimpot) TA(min) ayarlanır. TA (max) ayarı için ise , sensör kaynayan suyun içersine (100ºC) sokulur. Burada da sensörün uçlarında 3.73V olmalıdır. ADC çıkışı, 100 değerini gösterecek şekilde ayarlanır. Böylece sensör uçlarındaki her 10mV'luk artış için çıkışın 1 artması ve üzerine 2.73 düşmesi durumunda da çıkışın 0'ı göstermesi ayarlanmış olur. Bu ayarı kısaca şöyle hesaplayabiliriz:

Sensör uçlarındaki minimum gerilim (0ºC) 2.73V ise Vin (+) girişi gerilim bölücüden dolayı

$$\text{Vin(min)} = 2.73 . \frac{390K}{390K + 100K} = 2.17\text{V}$$

Sensör uçlarındaki max gerilim (teorik olarak ADC çıkışı 1111 1111=255) 5.28V ise

$$\text{Vin(+) max} = 5.28 . \frac{390K}{390K + 100K} = 4.2\text{V olarak bulunur.}$$



Bulunan giriş aralığı 2.03V olduğuna göre VREF/2 girişi = $\frac{G.A.}{2}$ = 1.015V olmalıdır. Bu ayar Vin(+) girişi 4.2V olduğundan çıkışın FF $_{HEX}$ =1111 1111olmasını sağlar.

Eğer 0ºC 'ta çıkıştan 0000 0000_2 değerini almak istiyorsak Vin(-) girişini de 2.17V a ayarlamalıyız.

Devre de bulunan LM336 2.5V referans diyodudur ve sıcaklıkla değişmeyen 2.5V gerilim sağlar. LM358 op-amp'ı da yük ile değişmeyen çıkış gerilimi elde etmemizi sağlar.

ADC0804'ün RD ve WR pinleri birbiriyle irtibatlandırılarak PIC16F877'nin Port/A girişine bağlanmıştır . Bunun amacı , bu girişler lojik 1 olduğunda çıkışa bilgi gitmeyecek , 0 olduğunda ise çıkışta sıcaklık bilgisi bulunacak ve PIC16F877 B portu uçlarından sıcaklık bilgisini okuyacaktır. PIC16F877 birim zamanın 10 da birinde ADC nin kontrol ucunu açtığı zaman sıcaklık bilgisi B portu vasıtası ile okunacak, 10 da 9 unda ise displayi sürerek ayarlanan sıcaklığı ve ortam sıcaklığını yazdıracaktır.

TARTIŞMA :

Sizce bu tür bir sistem nerelerde kullanılabilir?Artır butonu ile ayarlanan sıcaklık değeri deneyde en fazla kaç olur? Açıklayınız.

OGEM®

Otomasyon Gereçleri ve Elektromekanik San. Tic. ltd.şti.
ODTÜ-KOSGEB Teknoloji Geliştirme Merkezi No:403
06531 ODTÜ Kampüsü ANKARA
Tel:0-312-210 13 00 / 403 Fax:0-312-210 13 09
http//:www.ogemsan.com.tr
e-mail:ogem@ogemsan.com.tr