ONDOKUZ MAYIS ÜNİVERSİTESİ

MÜHENDİSLİK FAKÜLTESİ

ELEKTRİK ELEKTRONİK MÜHENDİSLİĞİ

MİKROİŞLEMCİLER LABORATUVARI

PİC UYGULAMA NOTLARI

BÖLÜM 1.

1.1 16F84 Mickrodenetleyicisi:

PIC16F84 18 pinli bir mikrodenetleyicidir. Bu pinlerden 13 tanesi I/O (giriş/çıkış) portudur. Bunların dışında Vdd, Vss, MCLR ve osilatör girişleri bulunur.

Sahip olduğu flash bellek sayesinde clock girişlerine uygulanan sinyal kesilse de registerleri içindeki veri saklandığından clock sinyali yeniden verildiği zaman program kaldığı yerden çalışmaya devam eder. I/O portlarından girilen digital sinyalleri yüklenmiş olan programa göre işleyerek digital çıkışlar verir.

Temel Özellikleri

* Çalışma gerilimi 2 V - 5.5 V 'tur.
* 4 MHz - 20 MHz arasındaki hızlarda çalışabilir.
* PIC16F84 1 Kbyte 'lık bir program belleğine sahiptir. Bellek hücrelerinden her birinde 14 bitlik veri saklayabilir.
* Program belleği elektriksel olarak yazılıp silinebilir (flash), programın çalıştığı sırada ise sadece okunabilir.
* PIC16F84 'ün iki banktan meydana gelen 68x8 byte 'lık bir RAM belleği vardır.
* PIC CPU sunun çalışmasını kontrol eden RAM bellekteki file register 'lardır.
* File register 'ların haricindeki bellek alanı ise normal RAM bellek olarak kullanılır.
* PIC16F84 64 byte 'lık bir EEPROM veri belleğine sahiptir.
* PIC16F84 mikrodenetleyicisinin 13 tane I/O protundan 5 tanesi A portu (RA0 - RA4),
8 tanesi de B portudur (RB0 - RB7).
*Portların giriş ve çıkış yönlendirmeleri PIC içerisinde bulunan özel bir yönlendirme registeri olan TRIS registeri ile yapılır.

1.2 EA-16F84 Temel Set:

EA-16F84 Temel Set, üzerindeki özel tasarım programlayıcısı ile PIC16F84 mikrodenetleyicisini programlamaktadır. Flash hafıza teknolojisi ile üretilen bu mikrodenetleyicilerin belleğine yüklenen program belleğin kalıcı olmasından dolayı uygulanan enerji kesilse bile silinmez. Flash bellekler bu özellikleri ile EEPROM belleklere benzemektedir.

Flash belleği olan PIC16F84'ün 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.

EA-16F84 Temel Eğitim Setinin Üstündeki Led ve Tuşların Görevleri:

RESET butonu: PIC16F84'nin 4 no'lu 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. PIC16F84'e program yüklerken program konumuna, deney yaparken de deney konumuna alırız.

PROGRAM Ledi: Yandığında setin program konumunda olduğunu ve PIC16F84'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 PIC16F84 PROGRAMLAMA YAZILIMI

Bu bölümde tanımlanan bir programın yazılıp derlenerek makine diline çevrilmesi ve bu hazırlanan kodun EA-16F84 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 PIC16F84'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. PIC16F84 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.

Programın PIC'e Aktarılması:

Hatalardan arındırdığımız programı derleyip makine kodlarını oluştururuz. Makine kodlarını içeren dosya .hex uzantılı dosyadır. Bu dosyayı PIC16F84'ye aktarmak için PC üzerinde çalışan PicProgramlayıcısı yazılımı kullanılır.

Programlamaya başlamadan önce EA-16F84 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 PIC16F84'nin programlanması için tampon belleğe aldığımız program içeriğini okuyabiliriz. Oku (Yonga) ikonuna tıklayarak PIC belleğinin Bilgisayarın tampon belleğine okunması, aktarılmasıda işlemini yaparız.

- EA-16F84 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-16F84 Temel Set üzerindeki DSUB-37 konnektörüne deneye ait uygun kart takılır. Program/Deney anahtarı Deney konumuna alınır ve deneyin işlem basamakları takip edilir.

BÖLÜM 2

16F84 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-16F84...) 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.

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 PIC16F84 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

PIC16F84 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 olanak vermektedir

PIC16F84 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 birkaç iç ve dış kesmelerle PIC'i uyuma modundan çıkarabilmektedir. Yüksek güvenilirlikli Watchdog Timer kendi bünyesindeki chip üstü RC osilatörü ile yazılımı kilitlemeye karşı korumaktadır.

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.

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 (SFRS) 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.

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

2.7.1 ÖZEL FONKSİYON KAYITÇILARI

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

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.

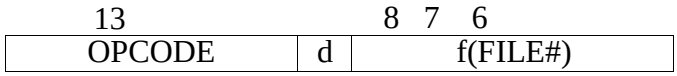
Çoğu komutun işlenmesi 1 saat çevrimi kadar sürer (osilatör frekansı/4). Fakat test işlemleri içeren komutlar komut şartı sağlandıysa 2, şart sağlanmadıysa 1 saat çevrimi kadar sürer. Komutları tanımlarken kullanılan harflerin anlamları:

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
italics	Kullanıcı tarafından tanımlanan terim
→	...'e atanan
< >	Kayıt bit alanı
ε	...'ın kurulmasında

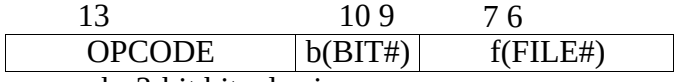
Komutların sahip olabileceği genel biçimler:

Byte'a yönelik komutlar



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

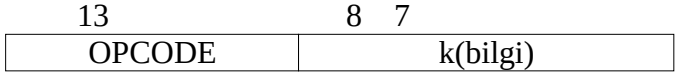
Bit'e yönelik komutlar



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

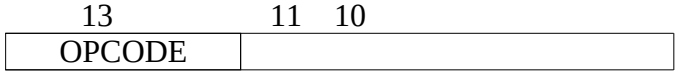
Bilgi ve Kontrol komutları

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'

BÖLÜM 3.

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-16F84 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.

Program kodu

```
;*Fonksiyon:Bir ledin yanıp sönmesi
;*********************************************************
status  equ     03      ;16C84'de port yazmaçlarına ulaşabilmek için kullanılmaktadır
rp0     equ     05
portb   equ     06
trisb   equ     86h

porta   equ     05
trisa   equ     85h

sayı    equ     0ch             ;bekle döngüsü için kullandığımız
sayı0   equ     0dh             ; kaydediciler.

        org     00              ;Reset vektörü
        goto    start           ;Programın başlangıç pozisyonu

;*************************************************************************************************
;gecikme altyordamı

bekle
        movlw       .178            ;500 msn'lik gecikme
        movwf       sayı            ;altyardamı
sd
        movlw       .255
        movwf       sayı0
sd0
```

```
        nop
        nop
        nop
        nop
        nop
        nop
        nop
        nop
        decfsz sayı0,f
        goto    sd0
        decfsz sayı,f
        goto    sd
        return

;*********************************************************
;Ledi yakıp söndüren ve aralarda gecikme altyordamına dallanan program

start
        bsf     status,rp0              ;1. yazmaç bankasına geç
        movlw           00h
        movwf           trisb                   ;B portunu  çıkış yap
        movwf  trisa
        bcf     status,rp0              ;0. yazmaç sayfasına dön.
        bsf    porta,4

don
        movlw           b'00000001'             ;bilgiyi  aküye al
        movwf           portb                   ;aküdeki bilgiyi portb ye taşı
        call    bekle                   ;bekle döngüsünü çağır.

        movlw           b'00000000'             ;bilgiyi aküye al
        movwf           portb                   ;aküdeki bilgiyi portb ye taşı
        call    bekle                   ;bekle döngüsünü çağır.
        goto    don                     ;don etiketine git.

        End
```

3.1. 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 sayaç kaydedicisine değer yüklenir. Sonra sayaç kaydedicisinin içeriği sıfır olana kadar sayaç 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.

$$1 + 1 + (1+2)*\text{sayaç} + \underline{1} + 2 = 2 + 3.\,255 + 3 = 770\ \mu s$$

(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 iç içe döngüler kullanırız. Şimdi de iç içe döngü kullanarak elde edilen gecikme altyordamının nasıl hesaplanacağını görelim.

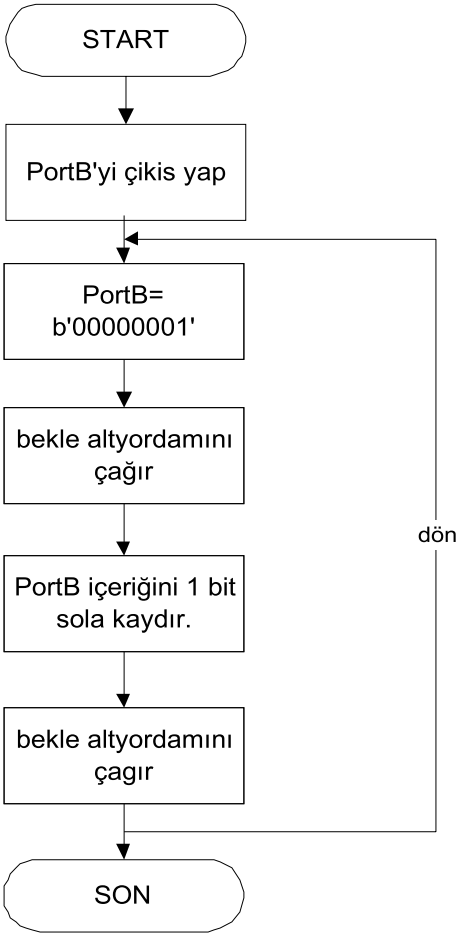
bekle bekle1 bekle2

```
movlw       .245        ;500 msn'lik gecikme
movwf       sayac2      ;alyordamı

movlw       .255
movwf       sayac1

nop nop nop nop nop
decfsz      sayac1,f
goto        bekle2
decfsz          sayac2,f
goto        bekle1 return
```

Yukarıdaki gecikme altyordamı iç içe 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.

$$1 + 1 + [\,1 + 1 + (\,1 + 1 + 1 + 1 + 1 + 1 + 2\,).\text{sayac1} + \underline{1} + 1 + 2].\text{sayac2} + \underline{1} + 2 \approx 500\text{mS}$$



Şekil 1.1. Programın akış diyagramı

3.2. 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.

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:

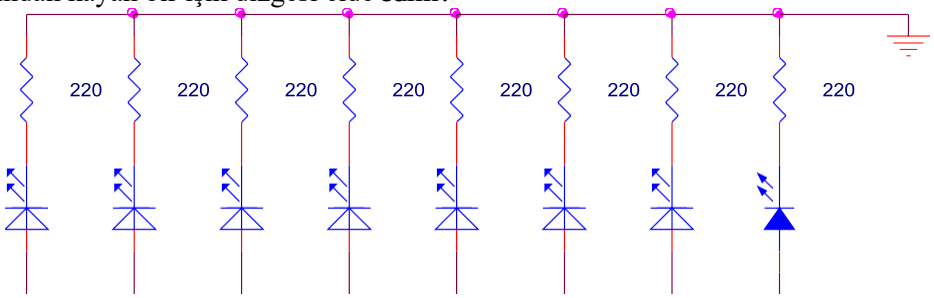
2.1. EA-16F84 uygulama seti
2.2. Display ve Led Deney Modülü

3) GİRİŞ:

Bu deneyde mikrodenetleyicinin portundan alınan ikilik bilgilerle ledlerin sürülmesini inceleyeceğiz

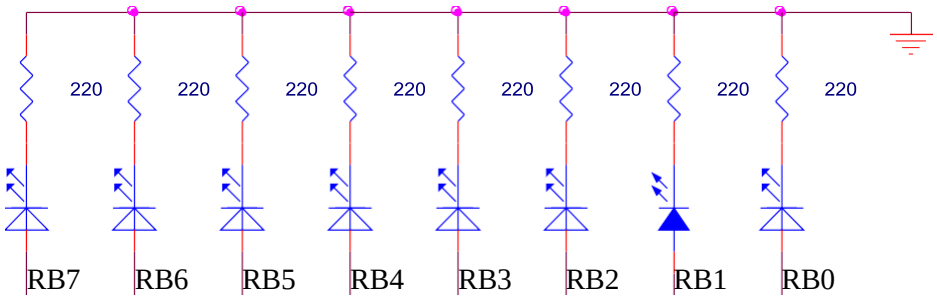
4.1. Port/B deki bilginin kaydırılması:

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.



	RB7	RB6	RB5	RB4	RB3	RB2	RB1	RB0
PortB	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**1**

Şekil 2.1: b'00000001' bilgisinin ledlerde görüntülenmesi



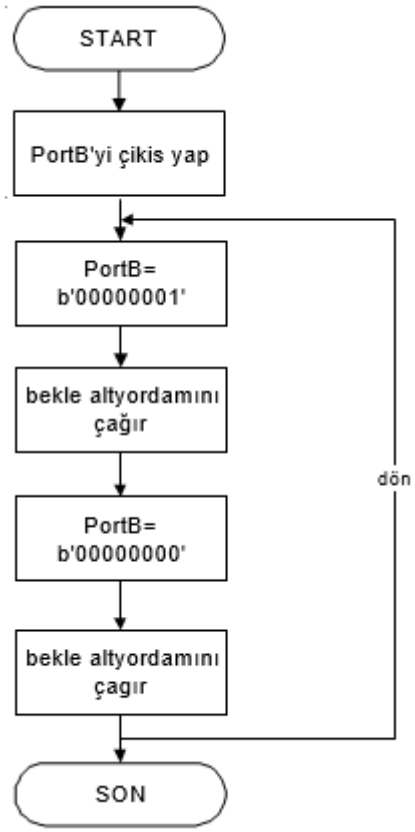
	RB7	RB6	RB5	RB4	RB3	RB2	RB1	RB0
PortB	**0**	**0**	**0**	**0**	**0**	**0**	**1**	**0**

Şekil 2.2: b'00000010' bilgisinin ledlerde görüntülenmesi

Program kodu

```
;Fonksiyon:Ledli yürüyen ışık uygulaması


status  equ     03      ;16C84'de port yazmaçlarına ulaşabilmek için kullanılmaktadır.

rp0     equ     05
```

```
portb   equ     06
trisb   equ     86h

porta   equ     05
trisa   equ     85h

sayı    equ     0ch             ;bekle döngüsü için kullandığımız
sayı0   equ     0dh             ; kaydediciler.

        org     00              ;Reset vektörü
        goto    start           ;Programın başlangıç pozisyonu

;*****************************************************************************************


bekle
        movlw .178      ;500 msn'lik gecikme
        movwf sayı      ;altyordamı
sd
        movlw .255
        movwf sayı0
sd0
        nop
        nop
        nop
        nop
        nop
        nop
        nop
        nop
        decfsz sayı0,f
        goto sd0
        decfsz sayı,f
        goto sd
        return

;*************************************************************************

start
        bsf status,rp0          ;1. yazmaç bankasına geç
        movlw 00h
        movwf trisb             ;B portunu  çıkış yap
        movwf trisa
        bcf status,rp0          ;0. yazmaç sayfasına dön.
        bsf porta,0

geri
        movlw b'00000001'       ;bilgiyi al aküye taşı
        movwf portb             ;aküdeki bilgiyi portb ye taşı
        call bekle              ;bekle döngüsünü çağır.

sol
        rlf portb,f             ;portb kaydedicisindeki bilgiyi bir bit sola kaydır.
        call bekle              ;bekle döngüsünü çağır.
        goto sol                ;sol etiketine git.

        end
```



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-16F84 uygulama seti
2.2. Display ve Led Deney Modülü

3. GİRİŞ:

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

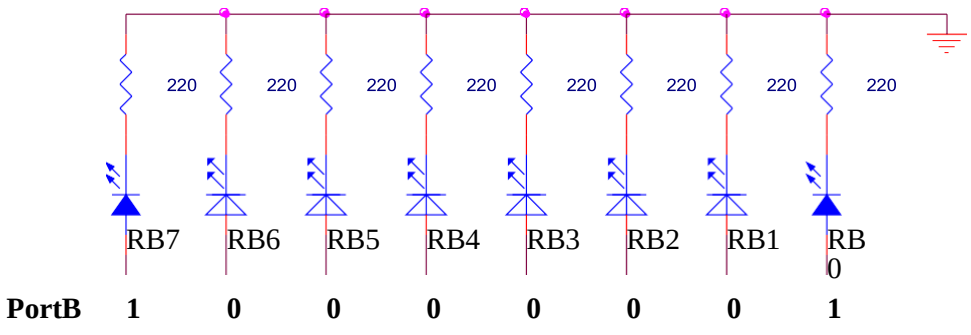
```
;Fonksiyon: İki yönlü yürüyen ışık uygulaması

status  equ     03

rp0     equ     05

portb   equ     06
trisb   equ     86h
porta   equ     05
trisa   equ     85h
```

```
sayı    equ     0ch             ;bekle döngüsü için kullandığımız
sayı0   equ     0dh             ; kaydediciler.
sola    equ     0eh
saga    equ     0fh

        org     00              ;Reset vektörü
        goto    start           ;Programın başlangıç pozisyonu

;*********************************************************
bekle
        movlw .178      ;500 msn'lik gecikme
        movwf sayı      ;altrutini
sd
        movlw .255
        movwf sayı0
sd0
        nop
        nop
        nop
        nop
        nop
        nop
        nop
        nop
        decfsz sayı0,f
        goto sd0
        decfsz sayı,f
        goto sd
        return

;*********************************************************

start
        bsf status,rp0
        movlw 00h
        movwf trisb
        movlw  00h
        movwf  trisa
        bcf status,rp0
        movlw b'00000001'
        movwf sola
        movlw b'10000000'
        movwf saga
```

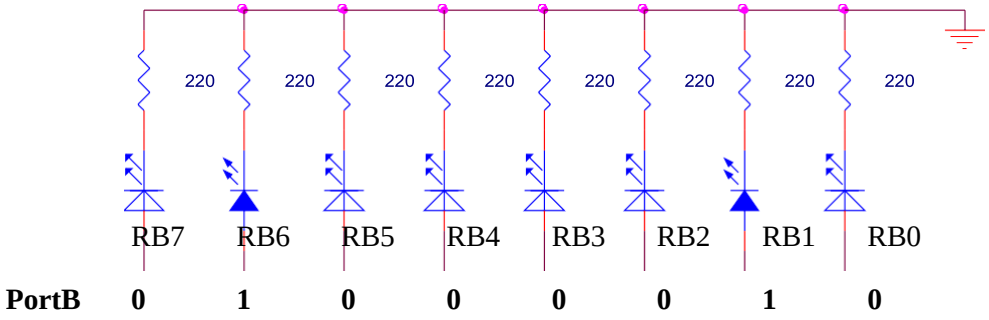
```
        bsf     porta,0

don
        rlf sola,f
        btfss status,C
        goto a1
        movlw b'00000001'
        movwf sola

a1
        rrf saga,f
        btfss status,C
        goto a2
        movlw b'10000000'
        movwf saga

a2
        movf sola,w
        addwf saga,w
        movwf portb
        call bekle
        goto don

        end
```

3.3. 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

DENEY NO : 4
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-16F84 mikrodenetleyici uygulama seti
b. Asansör Deney Modülü

3) GİRİŞ:

Bu deneyde mekanik asansörün optik asansör olarak simüle edilmesi incelenecektir.

3.4. 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'

3.5. 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

Asansör.ASM Uygulaması

```
;*******************************************************

porta           equ     5
portb           equ     6
trisb           equ     86h
trisa           equ     85h
status          equ     3
rp0             equ     5

MSB             EQU     H'0023'
LSB             EQU     H'0024'
BEKLEM          EQU     H'0025'
BEKLER          EQU     H'0026'
BEKLES          EQU     H'0027'
sayac           EQU     H'0028'
durum           EQU     H'0029'
bas             EQU     H'002A'
sayac1          EQU     H'002B'
tus_f           EQU     H'002C'
;************Bit Tanımlamaları**************************************
;*********Katları Gösteren Displayler******
#define         kat0    portb,0
#define         kat1    portb,2
#define         kat2    portb,4
#define         kat3    portb,6
;*********Katlararası Ledleri**************
#define         kat0_1  portb,1
#define         kat1_2  portb,3
#define         kat2_3  portb,5
;*********Butonlar************************
```

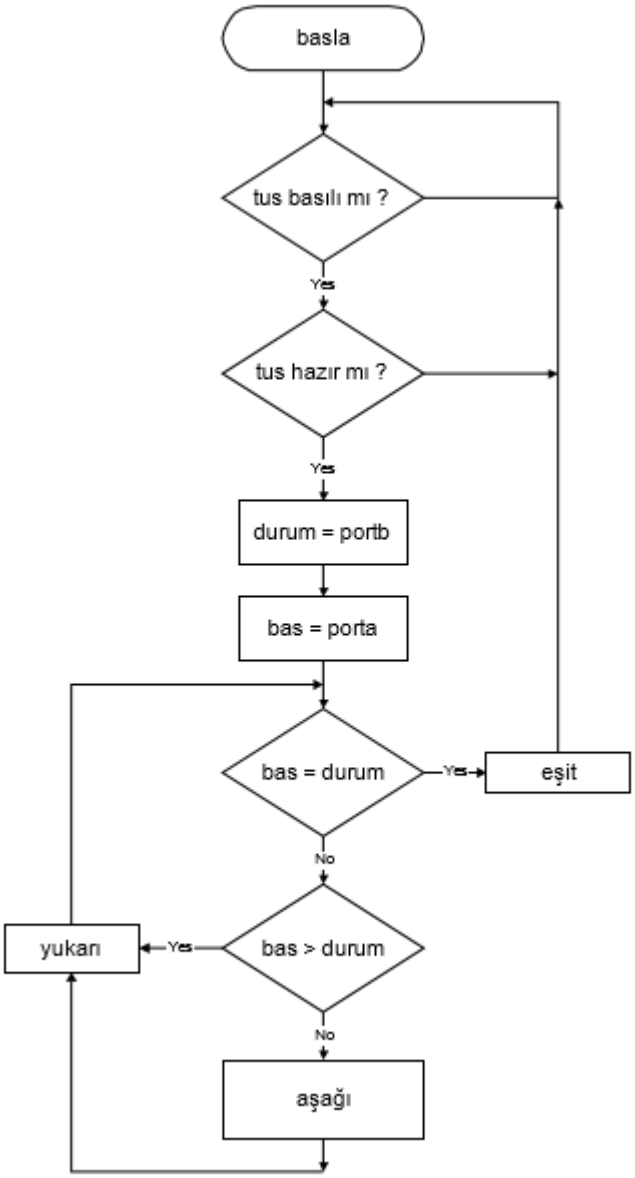
```
#define         katb0   porta,0
#define         katb1   porta,1
#define         katb2   porta,2
#define         katb3   porta,3
;********Aşağı/Yukarı Ledleri**************
#define         yukarı  porta,4
#define         asagı   portb,7
;********Kontrol Bayrakları*****************
#define         t_basılı tus_f,6
#define         t_hazır         tus_f,7

;-----------------------------------------------------------------
        org     00H
        goto    Start
        org     05H

;-----------------------------------------------------------------
bekle
        movlw   .25
        movwf   MSB
D110
        movlw   .250
        movwf   LSB
D220
        nop
        decfsz  LSB,F
        goto    D220
        decfsz  MSB,F
        goto    D110

        return
;-----------------------------------------------------------------
Start
        bsf     status,RP0      ;SELECT REGISTER BANK 1
        movlw   b'00001111'     ;SET PORTA,0123 INPUTS
        movwf   trisa           ;SET PORTA,4 OUTPUT
        movlw   b'00000000'
        movwf   trisb           ;SET PORTB TO ALL OUTPUTS
        bcf     status,RP0

        movlw   .254
        movwf   portb
```

```
        movlw  b'00011111'
        movwf  porta
        clrf   tus_f
;-----------------------------------------------------------------
basla

        btfss  katb0
        goto   ttest
        btfss  katb1
        goto   ttest
        btfss  katb2
        goto   ttest
        btfss  katb3

        goto   ttest
      bcf   t_basılı
      movlw  .8
      movwf  sayac
        call   bekle
        goto  basla


;-------------------------------------------------
ttest
        btfss  t_hazır
        btfsc  t_basılı
        goto   basla
        decfsz sayac
        goto   basla
        bsf    t_hazır
        bsf    t_basılı


cal
        bcf    t_hazır
        movf   portb,w
        andlw  b'01111111'
        movwf  durum
        movf   porta,W
        movwf  bas
        comf   bas,F
        movf   bas,w
```

```
        andlw   b'00001111'
        call    tablo
        movwf   bas
kontrol

        movlw   .40             ;1sn lik gecikme icin sayacı kur
        movwf   sayac
        movlw   .80             ;2sn lik gecikme icin sayacı kur
        movwf   sayac1

        movf    durum,W
        andlw   b'01111111'
        subwf   bas,W
        btfss   status,C
        goto    yukari
        btfsc   status,Z
        goto    esit
asagi
        movf    durum,W
        iorlw   b'10000000'
        movwf   durum
        bsf     status,C
        rrf     durum,f
        movf    durum,w
        andlw   b'01111111'
        movwf   portb
a1      call    bekle
        decfsz  sayac1
        goto    a1
        bsf     status,C
        rrf     durum,f
        movf    durum,w
        andlw   b'01111111'
        movwf   portb
a2      call    bekle
        decfsz  sayac
        goto    a2
        goto    kontrol
yukari
        bsf     status,C
        rlf     durum,f
        movf    porta,w
```

```
        andlw   b'00001111'
        movwf   porta
        movf    durum,w
        movwf   portb
a3      call    bekle
        decfsz  sayac1
        goto    a3
        bsf     status,C
        rlf     durum,f
        movf    durum,w
        movwf   portb
a4      call    bekle
        decfsz  sayac
        goto    a4
        goto    kontrol
esit
        movf    porta,w
        iorlw   b'00010000'
        movwf   porta
        movf    portb,w
        iorlw   b'10000000'
        movwf   portb
        call    bekle
        goto    basla
tablo
        addwf   PCL,1
        retlw   b'01111110'     ;0000
        retlw   b'01111110'     ;0001
        retlw   b'01111011'     ;0010
        retlw   .0              ;0011
        retlw   b'01101111'     ;0100
        retlw   .0              ;0101
        retlw   .0              ;0110
        retlw   .0              ;0111
        retlw   b'00111111'     ;1000
        retlw   .0              ;1001
        retlw   .0              ;1010
        retlw   .0              ;1011
        retlw   .0              ;1100
        retlw   .0              ;1101
        retlw   .0              ;1110
        retlw   b'01111110'     ;1111
```

end



Şekil 8.1: Asansor programının akış diyagramı

Kaynaklar

1. Nursel Ak, Pic Programlama, Alfa yayıncılık
2. 16F877 Mikrodenetleyici Eğitim Seti Kullanım ve Uygulama Kitabı