T.C.
SAKARYA ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

PİC MİKRODENETLEYİCİLERİ VE PİC İLE FREKANSMETRE UYGULAMASI

YÜKSEK LİSANS TEZİ

Elektrik-Elektronik Müh. İsmail Zafer TANRIVERDİ

Enstitü Anabilim Dalı : ELEKTRİK-ELEKTRONİK MÜHENDİSLİĞİ

Enstitü Bilim Dalı : ELEKTRİK

Tez Danışmanı : Prof. Dr. Ertan YANIKOĞLU

Mayıs 2003

T.C.
SAKARYA ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

PİC MİKRODENETLEYİCİLERİ VE PİC İLE FREKANSMETRE UYGULAMASI

YÜKSEK LİSANS TEZİ

Elektrik-Elektronik Müh. İsmail Zafer TANRIVERDİ

Enstitü Anabilim Dalı : ELEKTRİK-ELEKTRONİK MÜHENDİSLİĞİ

Enstitü Bilim Dalı : ELEKTRİK

Bu tez .. / .. /200 tarihinde aşağıdaki jüri tarafından Oybirliği ile kabul edilmiştir.

Prf. Dr. Ertan YANIKOĞLU	**Doç. Dr. Sadettin AKSOY**	**Doç. Dr. İ.Hakkı CEDİMOĞLU**
…….	…….	…….
Jüri Başkanı	**Üye**	**Üye**

ÖNSÖZ

Günümüzde çok yaygın olarak kullanılmakta olan mikrodenetleyiciler mikroişlemci temelli sistemlerden daha avantajlıdır. Bu gün mikrodenetleyiciler otomobillerde, kameralarda, cep telefonlarında, fax-modem cihazlarında, çamaşır makinesi, fotokopi radyo, TV gibi sistemlerde ve sayılamayacak daha bir çok alanda kullanılmaktadır.

PIC mikrodenetleyiciler ise RISC (Reduced Instruction Set Computer)mimari temelli bir mikrodenetleyicidir. RISC mimarisinde temel düşünce daha basit ve daha az komut kullanılmasıdır. Kullanım olarak esnek yapısı ve az elemana gereksinim duyması PIC ' leri popüler kılmaya başlamıştır.

Bu çalışmanın oluşumundaki ana sebep PIC mikrodenetleyicilerini donanım ve yazılım özellikleri bakımından incelemek ve PIC mikrodenetleyici (PIC16F84) ile oluşturulacak sistemlerin nasıl hazırlanacağını gözlemlemektir. PIC uygulamalarını örneklendirerek anlaşılır kılmaktır.

Bu çalışma sırasında desteğini esirgemeyen Sayın Hocam Prof. DR. Ertan YANIKOĞLU 'na bu çalışmada yer alan uygulamaların geliştirilmesinde, testlerinde fikirleriyle yardımcı olan Hasan BAYHAN'a, ve her türlü fedakarlığı gösteren aileme ve eşime teşekkürlerimi belirtmek isterim.

Konuyla ilgilenen herkese yardımcı olabilmesi dileğiyle...

İÇİNDEKİLER

ŞEKİLLER LİSTESİ

ÖZET

Anahtar Kelimeler – PIC Uygulamaları, PIC programlama, donanım yazılımlar, 16F84 Mimarisi ve Komutları

Bu çalışmada, çok fonksiyonlu lojik uygulamalarının daha hızlı ve daha ekonomik yapılması için tasarlanmış olan PIC (PERIPHERAL INTERFACE CONTROLLER) serisi mikrodenetleyicilerin temel yapıları ve gelişimi incelenmiştir. PIC serisi mikroişlemcileri kullanarak tasarlanması hedeflenen bir proje veya bir sisteme uygun PIC seçimi, bir PIC'in çalıştırılması için gerekli olan minimum donanımlar, yazılımlar ve PIC ile PC arasındaki veri iletişimini sağlayacak kompenentler ve PIC'e hazırlanan yazılımı kaydedecek PIC programlayıcılar hakkında bilgi verilmiştir. PIC uygulamaları için gerekli donanımlar, çevre birimleri, yazılımlar, yazılım aktarıcı cihazlar ve bu cihazlar için hazırlanmış derleyiciler, kullanıcı arayüz programları ele alınmıştır.

PIC uygulamalarında kullanılacak komutlar anlatılmış, uygulama örnekleri bu seri de mikrodenetleyiciler anlatılmış ve birkaç örnek uygulama devre şemaları, PIC'e yüklenecek yazılımlarla birlikte sunulmuştur.

PIC kullanımının diğer mikroişlemcilere göre kullanımının bize sağlayacağı avantajlar ve önemi ortaya konmuştur.

Frekansmetreler hakkında genel bilgi verilmiş, birkaç farklı metotla frekans metre tasarımı incenmiş ve Mikro denetleyici kullanımını örneklemek amacıyla PIC 16f84 kontrollü frekans sayıcı tasarlanmış, baskı devresi verilmiştir.

PIC MICROCOTROLLERS, APPLICATIONS AND PIC BASED FREQUENCE COUNTER

SUMMARY

KEYWORDS– PIC 16f84, programming PIC, PIC assembler, PIC hardware

This study examined the PIC (PERIPHERAL INTERFACE CONTROLLER) which designed for controling multi functional logic opretaions faster, easier and also cheaper ways. Also structures and evolution of PIC is told. I examined which microcontroller is more suitable for the new project.

Minimal system, hardware, software requarements are all investigated. I acknowladge the units which writing or erasing programme over PIC. And also the interfaces used to connect the PIC programmer to PC.

The advantages of PIC is examined. And ı underline the critical differances between PIC and other chips manifactured by other companies.

Lastly; basic knowladge is told about Frequencemeters. As an example of programming PIC, a frequence counter is designed with PIC 16f84. Circuit diagram, schema and the 16f84 hex files are given abot the frequence counter.

BÖLÜM 1. GİRİŞ

İleri seviyede lojik uygulamaları içeren elektronik otomasyon kontrol sistemleri, sistem analizi ve sistem simülasyonu yapabilen test cihazları, iletişim ağı projeleri, bilgisayar ağları ve çevre birimleri ile birlikte çalışabilen uzman sistemler, haberleşme kartları, endüstriyel imalat tezgâhları, vs gibi yazılım destekli ünitelerde işlem akışını sağlamak amacıyla mikroişlemciler kullanılmaya başlandı.

Mikroişlemci; CPU (Centrel Proccessing Unit) adı verilen bilgi işleme ünitesi, veri giriş (Input), veri çıkışı (Output) kısaca giriş-çıkış (Input-Output I/O), ve bellek (Memory) ünitelerinden oluşur. Ayrıca bu üç birimin arasında veri iletimini sağlayan veri yolları (Data Bus) ve adres yolları (Adres Bus) lardan oluşan anakartlara gereksinim vardır.

İşlemciler kendi başlarına pek işe yaramazlar ve bellek, giriş-çıkış ünitelerine ihtiyaç duyarlar. Eğer bu üç yapıyı (CPU, RAM ve I/O) bir araya getirip bir kılıf içine aldığımızda ortaya çıkan yapıya mikro denetleyici (microcontroller) denir. Bilgisayar teknolojisi gerektiren uygulamalarda kullanılmak üzere tasarlanmış olan mikro denetleyiciler, mikroişlemcilere göre basit ve ucuzdur. Artık günümüzde birçok ürünün içinde mikro denetleyiciler yer almaktadır. Mutlaka sıvı kristal bir gösterge ekranına sahiptir ve tuşlarla bu makineleri kontrol etmektedir. Bu kontrol etme olayını bize mikro denetleyiciler sağlar. Veya otomobille özellikle yeni olanlarında mutlaka bir tane mikro denetleyici vardır. Bu mikro denetleyici merkezi kilit sistemini ve gösterge panelini ve/veya klimayı kontrol eder ve bunlar hakkındaki bilgileri bize ekranda anlatır. İşte bu tür kontrol ve hesaplama olayları için mikro denetleyicileri kullanabiliriz.

Fiziksel olarak büyük olmaları, besleme gerilimlerinin yüksek olması, çok kompenentten oluşmaları dolayısıyla arıza yapma olasılıklarının fazla olması, ve en önemlisi fiyatlarının pahalı olması gibi dezavantajlar, aynı işlemleri daha hızlı ve eksiksiz yapabilen, ucuz, tek üniteden oluşan chiplerin imal edilmesini gündeme getirdi.

CPU, RAM, I/O, DATABUS gibi birimlerin tek bir chip üzerinde toplanması maliyeti düşürdü, arıza olasılığını azalttı, fiziksel olarak küçük olmaları daha çok yerlerde kullanılmasına olanak sağladı.

Günümüzde birçok mikro denetleyici üreticisi mevcuttur ve bunlar ihtiyaç duyulan işleme göre farklı mikro denetleyiciler üretmektedirler. Ama basite indiğimizde mikro denetleyici mimarisinde çok az bir değişim vardır ve bu mikro denetleyiciler yapı itibariyle hemen hemen aynıdır.

Bir projeye başlamadan önce mikro denetleyiciyi seçerken, uygulamada kullanacağımız mikro denetleyicinin ne tür özelliklerinin olması gerektiği iyice kavranmalıdır. Bu tür özellikler,

1. Kesme sayısı
2. I/O port sayısı
3. Programlanabilir dijital paralel giriş / çıkış
4. Programlanabilir analog giriş / çıkış
5. Dahili belek tipi ve kapasitesi
6. Kayan nokta hesaplaması

Bu özellikleri arttırmak mümkün, temel özellikleri şunlardır.

Bir mikro denetleyici genel olarak aşağıdaki birimlerden oluşur:

1. CPU (Merkezi işlem ünitesi - central Proccessing Unit)
2. RAM (Rast gele erişimli bellek-Random Access Memory)
3. EPROM/PROM/ROM (Silinir, yazılır sadece okunur bellek-Erasable
4. Programmable Read Only Memory)
5. I/O (Girdi/çıktı - Input/Output) - seri ve paralel
6. Timers (Zamanlayıcılar)

7. Interrupt controller (Kesmeler)

Mikro denetleyiciler özel amaçlı bilgisayarlardır ve programlandıkları şeyi en iyi şekilde yaparlar. Genel olarak özellikleri ise;

1. Mikro denetleyiciler sadece bir iş için programlanmışlardır ve sadece ve sadece bu programı işlerler ve kullandıkları program çipin içinde veya program hafızası denilen yerde saklı tutulur.
2. Mikro denetleyiciler sadece 50 mW civarında güç harcarlar.
3. Mikro denetleyicilere sadece girdi yapılmaz aynı zamanda çıktı da alınabilir.
4. LED göstergelerle, sıvı kristal göstergelerle, ikaz sesleriyle vb.
5. Mikro denetleyiciler ucuzdur. Birçok parçadan oluşan karmaşık bir devreyi kolayca küçük boyutlara ve maliyete indirmenizi sağlar.

Bu tür chipleri birçok firma üretti. (Intel, Phillips, Dallas, Siemens, Oki, Temic, Haris vb.) İçlerinde en önemli kullanım oranına ulaşan chipleri Microchip firması üretti ve PIC (PERIPHERAL INTERFACE CONTROLLER) ismini verdi. Diğer bir üretici firma olan İNTEL ise 8051 ismini verdi. Bu çalışmamda yaygın kullanımı, kaynak tarama kolaylığı, ucuz olması, derleyici, yazılım ve programlama cihazlarının kolay ve ücretsiz sağlanabilmesi gibi sebeplerden dolayı PIC mikroişlemcileri anlatılacak ve çalıştırılabilmesi için gereken minimal donanım ve yazılımlar belirlenecek ve haklarında bilgi verilecektir. Ayrıca kullanım kolaylığını, basit devre yapısını anlatmak amacı ile örnek olarak bir frekansmetre modeli PIC16f84 ile tasarlanmıştır.

BÖLÜM 2. PIC MİKRODENETLEYİCİLER

2.1. PIC Mikrodenetleyicilerine Giriş

PIC (PERIPHERAL INTERFACE CONTROLLER) adı verilen giriş-çıkış işlemcileri Microchip firması tarafından 1994 yılında üretilmişlerdir. Üretim amaçları, 16 bitlik ve 32 bitlik büyük işlemcilerin, giriş ve çıkışlarındaki yükü azaltmak ve çok hızlı ve en önemlisi ucuz bir çözüme ulaşmaktır.

Çok geniş bir ürün ailesinin ilk üyesi olan PIC16C54 bu ihtiyacın ilk meyvesidir. PIC işlemcileri RISC -benzeri işlemciler olarak anılır. PIC16C54 12 Bit komut hafıza genişliği olan 8 bitlik CMOS bir işlemcidir. 18 bacaklı dip kılıfta 13 I/O bacağına sahiptir ve 20 Mhz osilatör hızına kadar kullanılabilir. 33 adet komut içermektedir. 512 byte program epromu ve 25 byte RAM`i bulunmaktadır. Bu hafıza kapasitesi çok küçük olmasına rağmen bir risc işlemci olması birçok işin bu kapasitede uygulanmasına olanak vermektedir.

PIC serisi tüm işlemciler herhangi bir ek bellek veya giriş/çıkış elemanı gerektirmeden sadece 2 adet kondansatör, 1 adet direnç ve bir kristal ile çalıştırılabilmektedir. Tek bacaktan 40 mA akım çekilebilmekte ve entegre toplamı olarak 150 mA akım akıtma kapasitesine sahiptir. Entegrenin 4 Mhz osilator frekansında çektiği akım çalışırken 2 mA stand-by durumunda ise 20µA kadardır.
PIC 16C54 ‘un mensup olduğu işlemci ailesi 12Bit core 16C5X olarak anılır. Bu gruba temel grup adı verilir. Bu ailenin üyesi diğer işlemciler PIC16C57, PIC16C58 ve dünyanın en küçük işlemcisi olarak anılan 8 bacaklı PIC 12C508 ve PIC 12C509’dur.

İnterrupt kapasitesi ilk işlemci ailesi olan 12Bit Core 16C5X ailesinde bulunmamaktadır. Daha sonra üretilen ve Orta sınıf olarak tanınan 14Bit Core-16CXX ailesi birçok açıdan daha yetenekli bir grup işlemcidir.

Bu ailenin temel özelliği interrupt kapasitesi ve 14 bitlik komut işleme hafızasıdır. Bu özellikler Pic'i gerçek bir işlemci olmaya ve karmaşık işlemlerde kullanılmaya yatkın hale getirmiştir. PIC16CXX ailesi en geniş ürün yelpazesine sahip ailedir. 16CXX ailesinin en önemli özellikleri seri olarak devre üstünde dahi programlanmasıdır ki bu özellik PIC16C5x de çok karmaşıktı. Paralel programlanabiliyordu, interrupt kabul edebilmesi, 33 I/O,AD Converter, USART, I2C, SPI gibi endüstri standardı giriş çıkışları kabul edecek işlemcilere ürün yelpazesinde yer vermesi PIC 16CXX ailesinin en çok tanınan ve dünyada üzerinde en çok proje üretilmiş, elektroniğin gözdesi olan bireyi PIC16C84 veya yeni adıyla PIC16F84 dur.

PIC 16F84 un bu kadar popüler olması onun çok iyi bir işlemci olmasından ziyade program belleğinin Eeprom - Elektrikle silinip yazılabilen bellek olmasından kaynaklanmaktadır. Seri olarak dört adet kabloyla programlanması da diğer önemli avantajıdır. Bugüne kadar amatörce bir işlemciyle uğraşmış herkesin en büyük sıkıntısı eprom veya eeprom tabanlı işlemcileri programladıktan sonra UltraViole ışık kaynağı ile silip tekrar programlamaktır. Bu çok zahmetli ve bir amatör için donanım gerektiren yöntem olmuştur. Evde üretilmesi zor olan özel bir programlayıcı da madalyonun diğer yüzüdür.

PIC16F84 amatörler tarafından internette en bol programlayıcısı bulunan işlemcidir herhalde. Eprom silmek diye bir şey zaten söz konusu değil zira eeprom belleği programlayan programlayıcı devre 1 saniye içinde aynı belleği silebilmektedir. Bu özellik size çok hızlı ve defalarca deneyerek program geliştirme avantajını getirmektedir ki bu bir elektronikçi için bulunmaz bir nimettir. Bu denemeleri yaparken işlemciyi devrenizden sökmeniz dahi gerekmez. Bu tip programlamaya ISP -In System Programming- denmektedir.

2.1.1. PIC Mikrodenetleyicilerinin Tercih Sebepleri

a) Lojik uygulamalarının hızlı olması,
b) Fiyatının oldukça ucuz olması,
c) 8 bitlik mikro kontroller olması ve bellek ve veri için ayrı yerleşik busların kullanılması,
d) Veri ve belleğe hızlı olarak erişimin sağlanması,
e) PIC' e göre diğer mikrokontrolörlerde veri ve programı taşıyan bir tek bus bulunması, dolayısıyla PIC' in bu özelliği ile diğer mikrokontrolörlerden iki kat daha hızlı olması,
f) Herhangi bir ek bellek veya giriş/çıkış elemanı gerektirmeden sadece 2 kondansatör ve bir direnç ile çalışabilmeleri,
g) Yüksek frekanslarda çalışabilme özelliği,
h) Stand by durumunda çok düşük akım çekmesi,
i) İntterrupt kapasitesi ve 14 bit komut işleme hafızası,
j) Kod sıkıştırma özelliği ile aynı anda birçok işlem gerçekleştirebilmesi

2.1.2 PIC Mikrokontrolörlerle Çalışabilmek İçin Gereken Tanımlar

I/O (Giriş / Çıkış) Birimi: Mikro kontrolcünün dış dünya ile ilişkisini sağlayan, girdi ve çıktı şeklinde ayarlanabilen bir bağlantı pinidir. I/O çoğunlukla mikro kontrolcünün iletişim kurmasına, kontrol etmesine veya bilgi okumasına izin verir.

Yazılım: Mikro kontrolcünün çalışmasını ve işletilmesini sağlayan bilgidir. Başarılı bir uygulama için yazılım hatasız (bug) olmalıdır. Yazılım C, Pascal veya Assembler gibi çeşitli dillerde veya ikilik (binary) olarak yazılabilir.

Donanım: Mikro kontrolcü, bellek, arabirim bileşenleri, güç kaynaklan, sinyal düzenleyici devreler ve bunları çalıştırmak ve arabirim görevini üstlenmek için bu cihazlara bağlanan tüm bileşenlerdir.

Simülatör: PC üzerinde çalışan ve mikrokontrolcünün 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, programlar tümüyle veya adım adım izleyerek bug'lardan 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 Mikro kontrolcünün yer alacağı soket arasına bağlanmış yararlı bir gereçtir. Bu gereç yazılım, PC de çalışırken devre kartı üzerinde bir mikro kontrolcü gibi davranır. ICE, bir programa girilmesini, mikro içinde neler olduğunu ve dış dünya ile nasıl iletişim kurulduğunun izlenilmesini mümkün kılar.

Programcı: Yazılımın mikro kontrolcü belleğinde programlamasını ve böylece ICE' nin yardımı olmadan çalışmasını sağlayan bir birimdir. Çoğunlukla seri port 'a (örneğin PICSTART, PROMASTER) bağlanan bu birimler çok çeşitli biçim, ebat ve fiyatlara sahiptir. Çalışmamın daha sonraki bölümlerinde bir kaç programmer örneği verilecektir.

Kaynak Dosyası: Hem assemblerin hem de tasarımcının anlayabileceği dilde yazılmış bir programdır. Kaynak dosya mikrokontrolörün anlayabilmesi için önceden assemble edilmiş olmalıdır.

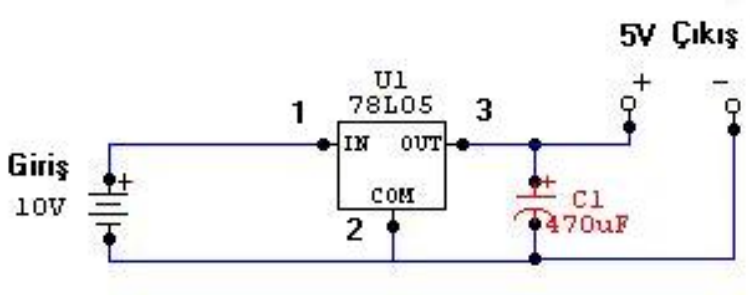
Assembler: Kaynak dosyayı bir nesne dosyaya dönüştüren yazılım paketidir. Hata araştırma bu paketin yerleşik bir özelliğidir. Bu özellik assemble edilme sürecinde hatalar çıktıkça programı bug'lardan arındırırken kullanılır. MPASM, tüm PIC ailesini elinde tutan Microchip' in son assemble edicisidir.

Nesne dosyası (object file): Assembler tarafından üretilen bu dosya; programcı, simülatör veya ICE' nin anlayabilecekleri ve böylelikle dosyanın işlevlerinin çalışmasını sağlayabilecekleri bir dosyadır. Dosya uzantısı assemble edicinin emirlerine bağlı olarak, .OBJ veya .HEX olur.

2.1.3 PIC Mikro kontrolörlerinin Çalışabilmesi İçin Gereken Donanımlar

2.1.3.1. Voltage Range (Çalışma Voltaj Aralığı)

PIC in sağlıklı çalışacağı voltaj aralığı PIC16F84 te 2-6 volt, diğerlerinde (PIC16C620-621-622-61-64-65-71-73-74) ise 3-6 volt aralığındadır. En sağlıklısı, besleme voltajını, regüle entegresi olan ve +5 volt çıkış verebilen 7805 entegresi kullanarak sağlamaktır.



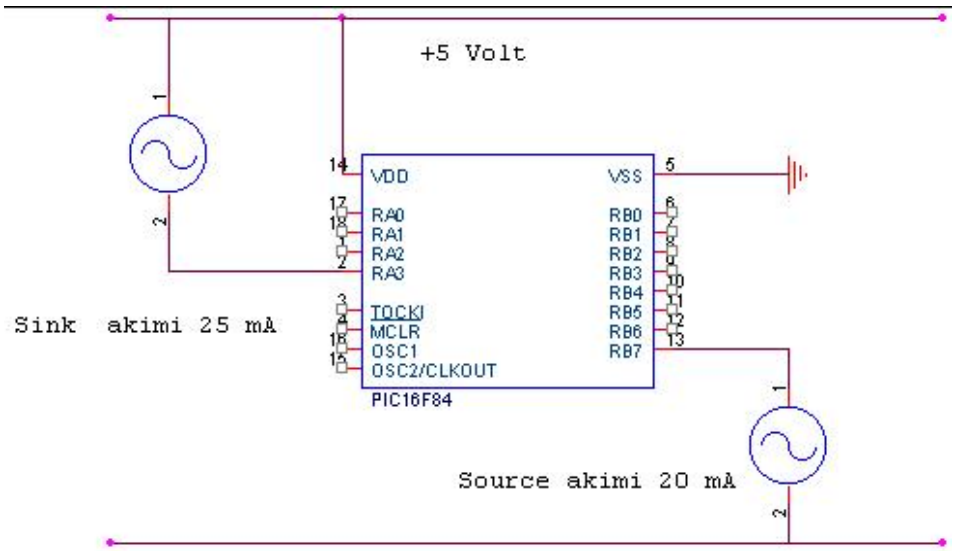
Şekil 2.1. Yapılmış Besleme Devresi

2.1.3.2. Oscillator (Osilatör)

PIC in çalışma hızını belirlememizi sağlayan, PIC in OSC1 ve OSC2 isimli bacaklarına uygun şekilde bağlanan ve XT (Kristal-Kondansatör), RC (Direnç-Kondansatör), HS (Kristal-Kondansatör yüksek hızlar için- 20 MHz.), LP (Kristal-Kondansatör min. 40 KHz.) şeklinde isimlendirilen yardımcı devrelerdir.

16F84'ün çektiği akım saat hızına bağlıdır. Saat hızı arttıkça çekilen akım artar. 4 Mhz saat hızında çekilen akım 2 mA kadardır. Eğer pic sleep modda ise bu akım 40 mikroamper'e kadar düşer.

CMOS entegrelerde giriş uçlarını +5 Volt'a bağlamak gerekir.16F84 de bir CMOS entegredir, fakat burada pic içinde I/O pinleri özel şekilde bağlanmışlardır ve program aracılığı ile giriş veya çıkış haline getirilir. Giriş pozisyonunda bu uçlar pull-up konumundadır, yani pic içinde bir şekilde +5 volta bir direnç üzerinden bağlı gibidir. Burada pull-up dirençleri 50 K kadardır.



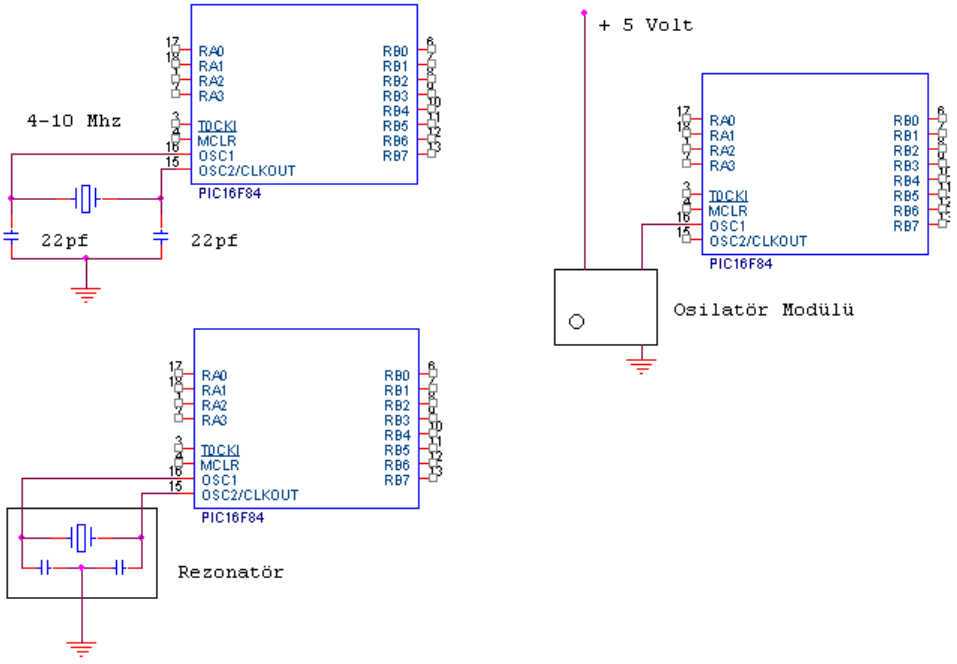
Şekil 2.2. Besleme Gerilimlerinin 16F84'e verilmesi

16F84'ün Çalıştırılmasında Kullanılan Osilatorler

1-) Xtal osilatör

2-) Seramik rezonatör

3-) Osilatör Modülü 3. şıkta bahsedilen osilatör modülleri sıcaklık değişimlerine karşı da korunmuş oldukları için çok kararlı çalışırlar.



Şekil 2.3. 16F84 için osilatör devreleri

Bir 16F84'de + gerilim ile GND uçları arasına bir adet 0.1 µf kondansatör konması, istenmeyen bazı gerilim dalgalanmalarını önler.

16 nolu bacaktan girilen osilatör sinyali, pic içinde 4'e bölünür ve frekansın 1/4'ü 15 nolu bacaktan alınabilir. Bu 4'e bölünmüş saat frekansının karşılığı olan periyota "instruction cycle" yani komut süresi denir. Bu bir komutun işlenmesi için gereken zamandır.16F84'de bu 4 Mhz de 1 mikro saniye 10 Mhz de ise 0.4 mikro saniyedir. Bu zaman, programlama esnasında çok önem arz eder, bu komut sürelerinin toplamı ile zamanlar hesaplanır.

Dışarıdan bağlanan osilatörler ile pic daha yavaş saat hızlarında da çalıştırılabilir. Bunun için kullanılan uç RA4 /TOCKI pinidir. Burası çıkış olarak RA4, giriş olarak ta TOCKI girişidir (Timer Zero Clock Input).

Kristal osilatör kullanıldığında 16F84'e 22 pf kondansatör bağlanır. Seramik rezonatör için bir özellik yoktur.

16F84'ün 13 ucunu da çıkış olarak kullanabiliriz. Bu çıkışlar A ve B diye ikiye ayrılır A çıkışları 5 adettir RA0-RA1-RA2-RA3-RA4 diye adlandırılır. B çıkışları ise 8 adettir ve bunlarda RB0-RB1-RB2-RB3-RB4-RB5-RB6-RB7 diye adlandırılırlar. Bu çıkışlar da bir tek RA4 yani TOCKI ucu diğerlerinden farklıdır ve bu uç açık kollektör özelliği gösterir, bu nedenle çıkış olarak kullanıldığında, 10 K lık bir direnç le + 5 V a asılır.

RB0 ucu giriş olarak ayarlanırsa, aynı zamanda "external interrupt" girişidir. Yine B port'un 4-5-6-7 çıkışları (RB4-RB5-RB6-RB7) internal interrupt alma özelliğindedir.

Pic power-on reset ile yani pic'e enerji verildiğinde, başlangıç adresinden başlar. Bu durumda tüm pull-uplar iptal edilmiş olur ve pic portları çıkış pozisyonundadır. Programlama ile port çıkışları giriş veya çıkış olarak yönlendirilir, bunun için "option" register içinde de disable veya enable seçenekleri vardır. Power-on reset ile disable olan pic de tüm portlar çıkıştır.

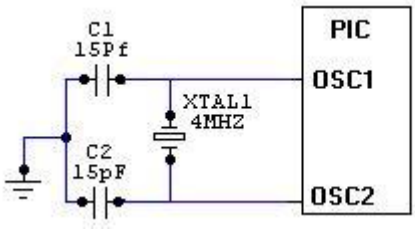
Pic'i başlangıç adresinden başlatan 3 mekanizma vardır bunlar:

1-)Power-on reset

2-) MCLR reset (Master Clear Reset) kullanıcı tarafından programı başa döndürmeye yarar.

3-) Watchdog timer (İptal edilmediğinde programı belli zaman aralıkları ile başa döndürür.

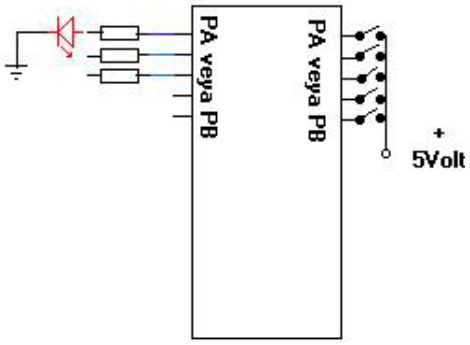
Bu portların giriş veya çıkış olarak yönlendirilmesi TRIS register ile sağlanır, veya Write register vasıtası ile aktarılan komutlar yardımı ile port lar tek tek seçilebilir.



Şekil 2.4. Xt (Kristal-Kondansatör) Osilatör Devresi

2.1.3.3. I/O (Giriş/Çıkış)

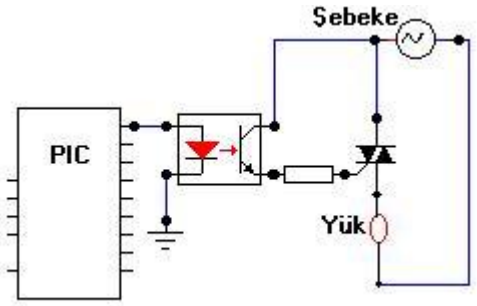
PIC16F84 te PORTA-5, PORTB-8, toplam 13 bitlik Giriş/Çıkış portu vardır. Bu bitlerden istediğinizi giriş, istediğinizi çıkış yapabilirsiniz. Deneme devrelerimizde, Giriş (I) lere butonlar üzerinden +5 Volt vererek, çıkış (O) larada genelde led bağlantısı yaparak kullanacağız.



Şekil 2.5. Portların Bağlantı Şekilleri

2.1.3.4. PIC' lerle Güç Kontrolü

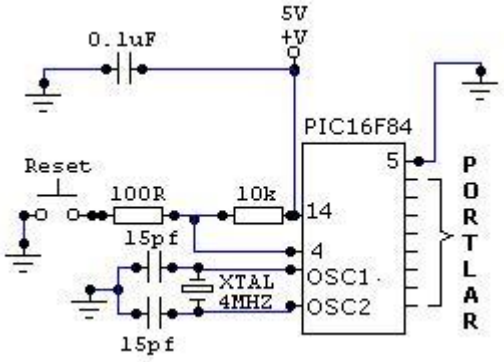
PIC' lerle büyük güçleri sürmek ilave arabirim devreleri gerektirmektedir. Bir şebeke gerilimini sürebilmek ve PIC' in sağlıklı çalışmasını sağlayabilmek için; PIC' in beslemesinin şebeke geriliminden yalıtılması gerekir. Sürülecek olan şebeke gerilimi ve PIC' çıkış portlarını bir optoizolatörle yalıtmak uygun olur.



Şekil 2.6. Yüksek Güçlerin PIC' Ten Yalıtılması

2.1.3.5. PIC' in Minimum Devre Konfigürasyonu

PIC' in minimum devre şeması aşağıdaki gibidir.



Şekil 2.7. PIC' in Minimum Devre Konfigürasyonu

2.1.4. PIC Mikro kontrolörlerinin Özellikleri

PIC komutları bellekte çok az yer kaplarlar. Dolayısıyla bu komutlar 12 veya 14 bitlik bir program bellek sözcüğüne sığarlar. Harward mimarisi teknolojisi kullanılmayan mikrokontrolörler de yazılım programının veri kısmına atlama yaparak bu verilerin komut gibi çalıştırılmasını sağlamaktadır. Bu da büyük hatalara yol açmaktadır. PIC' ler de bu durum engellenmiştir. Yani daha güvenilirdirler.

PIC oldukça hızlı bir mikrokontrolör' dür. Her bir komut döngüsü bir mikro saniyedir. Örneğin 5 milyon komutluk bir programın 20Mhz' lik bir kristalle işletilmesi yalnız l saniye sürer. Bu süre 3868X33 hızının yaklaşık 2 katıdır. Ayrıca RISC mimarisi işlemcisi olmasının hıza etkisi oldukça büyüktür.

PIC' in 16C5X ailesinde bir yazılım yapmak için 33 komuta ihtiyaç duyarken 16CXX araçları için bu sayı 35' tir. PIC tarafından kullanılan komutların hepsi yazmaç (register) temellidir. Komutlar 16C5X ailesinde 12 bit, 16CXX ailesindeyse 14 bit uzunluğundadır. PIC' te CALL, GOTO ve bit test eden BTFSS ve INCFSZ gibi komutlar dışında diğer komutlar l saykıl çeker. Belirtilen komutlar ise 2 saykıl çeker. Sonuçta maksimum 35 komut kullanılarak program yapılabilir.

PIC tamamıyla statik bir işlemcidir. Yani saat durdurulduğunda da tüm yazmaç içeriği korunur. Pratikte bunu tam olarak gerçekleştirebilmek mümkün değildir. PIC mikrosu programı işletilmediği zaman uyuma (sleep) moduna geçirilerek mikronun 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, O (zero) bayrağı, vb.) PIC uyuma modunda l mili Amper'den küçük değerlerde akım çeker. (Stand by akımı).

Sürücü kapasitesi yüksektir. Tek bacaktan 40mA akım çekebilmekte ve entegre toplamı olarak 150mA akım akıtma kapasitesine sahiptir. Entegrenin 4mHz osilatör frekansında çektiği akım çalışırken 2mA, Stand by durumunda ise 2uA kadardır.

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. Her çeşit uygulamaya cevap verecek model ve tipi vardır.

PIC çok yönlü bir mikrodur ve ürünün içinde, yer darlığı durumunda birkaç mantık kapısının yerini değiştirmek için düşük maliyetli bir çözüm bulunur.

PIC endüstride en üstünler arasında yer alan bir kod koruma özelliğine sahiptir. Koruma bitinin programlanmasından itibaren, program belleğinin içeriği, program kodunun yeniden yapılandırılmasına olanak verecek şekilde okunmaz. Yazılım korsanlığına karşı korumalıdır, dolayısıyla güvenlidir.

PIC program geliştirme amacıyla programlanabilip tekrar silinebilme özelliğine sahiptir. (EPROM, EEPROM) Aynı zamanda seri üretim amacıyla bir kere programlanabilir (OTP) özelliğine sahiptir.

Assembler tarafından yaratılan ve kaynak dosyadaki tüm komutları hexadecimal sistemdeki değerleri ve tasarımcının yazmış olduğu yorumlarıyla birlikte içeren bir liste dosyası vardır. Bir programı bug'lar dan arındırırken araştırılacak en yararlı dosya budur. Çünkü bu dosyayı izleyerek yazılımlarda neler olup bittiğini anlama şansı kaynak dosyasından daha fazladır. Dosyanın uzantısı .LST dir.

Hataların bir listesini içeren ancak bunların kaynağı hakkında hiç bir bilgi vermeyen Hata dosyası (Error file) bulunur, uzantısı .ERR dir. Uzantısı .COD olan dosyalar emülatör tarafından kullanılırlar.

Tasarımcının farkında olmadan yaptığı hatalar olabilir. 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 geliştiriciye düşer.

BÖLÜM 3. PIC MİKROKONTROLÖRLERİNİN DONANIMSAL İNCELENMESİ

3.1. PIC Mikrokontrolörlerinin İç Yapısı

CPU bölgesinin ana birimi ALU (Aritmetic Logic Unit-Aritmetik mantık birimi) denilen birimdir. ALU, W (Working-Çalışan) adında bir yazmaç içerir. PIC, diğer mikroişlemcilerden, aritmetik ve mantık işlemleri için bir tek ana yazmaca sahip oluşuyla farklılaşır. W yazmacı 8 bit genişliğindedir ve CPU'da ki herhangi bir veriyi transfer etmek üzere kullanılır. CPU alanında ayrıca iki kategoriye ayırabileceğimiz Veri Yazmaç dosyalan (Data Regıster Files) bulunur. Bu veri yazmaç dosyalarından biri, I/O ve kontrol işlemlerinde kullanılırken, diğeri RAM olarak kullanılır. PIC' ler de Harvard Mimarisi kullanılır. Harvard Mimarisi mikro kontrolcülerde veri akış miktarım hızlandırmak ve yazılım güvenliğini arttırmak amacıyla kullanılır. Ayrı bus' ların kullanımıyla veri ve program belleğinde hızlı bir şekilde erişim sağlanır.

PIC mikro kontrolörlerini donanımsal olarak incelerken PIC 16F84 üzerinde durarak bu PIC' i temel alıp donanım incelenecektir. Bellek ve bazı küçük farklılıklar dışında burada anlatılanlar bütün PIC' ler için de geçerlidir.

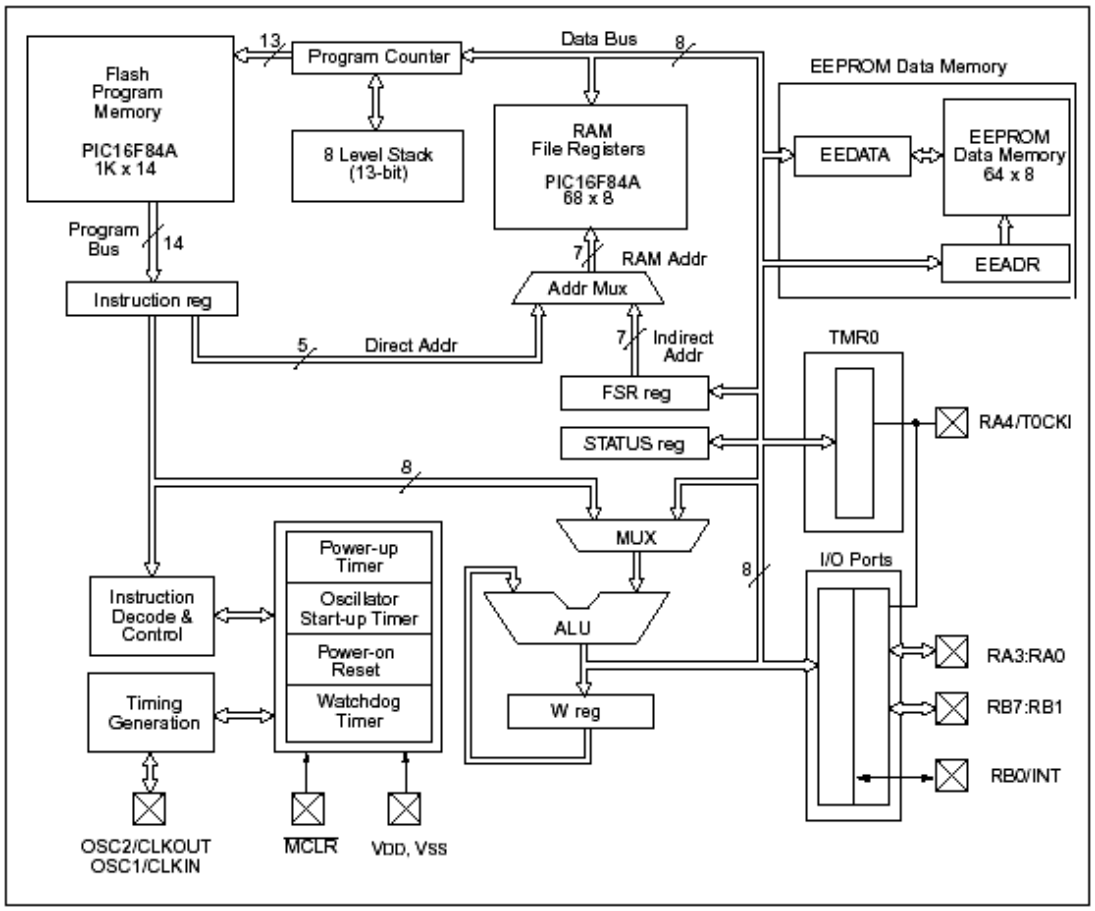
3.2. Genel Tanımlama

PIC 16F8X Serisi yüksek performanslı, CMOS, full-statik, 8 bit mikro denetleyicidir. Tüm PIC 16/17 mikro denetleyiciler RISC mimarisini kullanmaktadır. PIC16F87X mikroları birçok esas özelliklere sahiptir. 14 seviyeli, derin küme ve çoklu iç ve dış kesme kaynaklarına sahiptir. 2 aşamalı komut hattı tüm komutların tek bir saykıl' la (çevrimle) işlenmesini sağlamaktadır. Yalnızca bazı özel komutlar 2 saykıl çekerler. Bu komutlar dallanma komutlarıdır. PIC16F873/874 Microchip' i 192 baytlık RAM

belleğine, 128 bayt EEPROM belleğine ve 22/33 (PIC 16F873-22/ PIC 16F874-33) I/O pin' ine sahiptir. Bunun yanı sıra, timer ve sayaç da mevcuttur. PIC16F87X ailesi dış elemanları azaltacak spesifik 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 pinli RC osilatör, düşük maliyet çözümünü sağlamakta (4 MHZ), LP osilatör (Kristal veya seramik rezonatör), enerji sarfiyatım 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 mikrokontrolörlerinin en büyük özelliği sleep modu özelliğidir. Bu mod ile PIC işlem yapılmadığı durumlarda uyuma moduna geçerek çok düşük akım çeker. Kullanıcı bir kaç iç ve dış kesmelerle PIC' i uyuma modundan çıkara bilmektedir. Yüksek güvenilirlikli Watchdog Timer kendi bünyesindeki chip üstü RC osilatörü ile yazılımı kilitlemeye karşı korumaktadır. PIC16F87X EEPROM 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.

	PIC16F87	PIC16F874	PIC16F876	PIC16F877
Çalışma Frekansı(MHZ)	20	20	20	20
FLASH Program Belleği(14-Bit word)	4K	4K	8K	8K
Veri Hafızası (Byte)	192	192	368	368
EEPROM Veri Belleği (Byte)	128	128	256	256
Kesmeler	13	14	13	14
I/O Portlan Sayısı	Port A,B,C	Port A,B,C,D,E	Port A,B,C	Port A,B,C,D,E
10-BitADC Modülü	5 Kanal	SKanal	SKanal	8 Kanal
Komut Seti	35 Komut	35 Komut	35 Komut	35 Komut

Tablo 3.1 PIC 16F87X ailesi özellikleri



Şekil 3.1. 16F874/877 basitleştirilmiş iç yapısı

3.3. Elektrikle Silinebilen Mikro denetleyiciler

Bu mikrolar, programının silinip yeniden yazılabilme özelliğine sahiptir ve oldukça düşük maliyetli plastik ambalajlar halinde bulunmaktadır. Aynı zamanda bu tip mikroların üretimi kadar prototipinin geliştirilmesi ve pilot programlar için kullanılmasına olanak sağlamaktadır. Bunun daha ötesindeki avantajlarından biri, bunların devre içi veya Microchip's PICSTART plus veya PROMATE programlayıcıları tarafından silinebilmesi ve yeniden programlanabilmesidir.

3.4. Bellek Organizasyonu

PIC16F87X' de bellek bloğu mevcuttur. Bunlar program belleği,veri belleği ve bunları ayıran veri hattıdır. Her bir bellek kendi taşıyıcısına sahiptir; böylece her bir

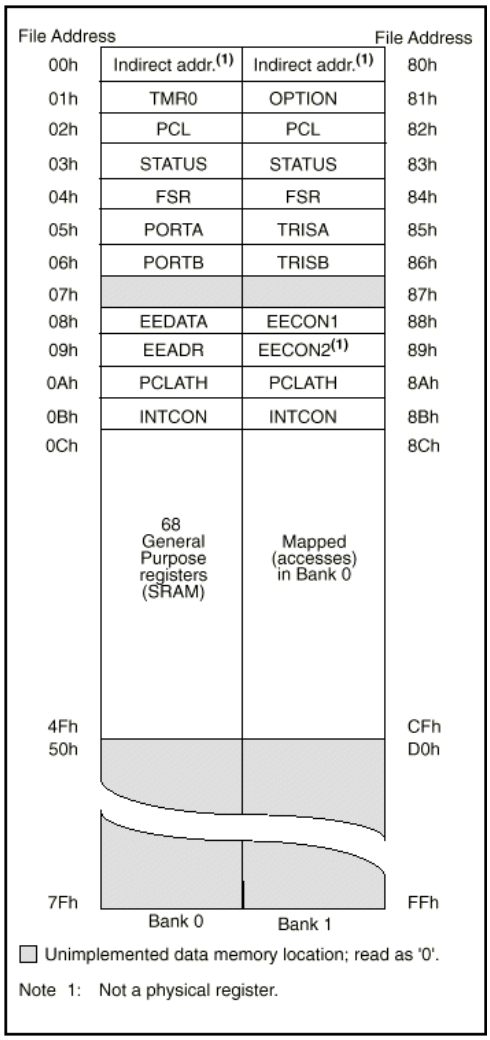
bloğa erişim aynı osilatör süreci boyunca meydana gelebilmektedir. Bunun ötesinde, veri belleği genel amaçlı RAM ve özel fonksiyon kayıtlan (SFR) olmak üzere ikiye bölünür. SFR'ler her bir bireysel özelleşmiş modülü ele alan bölümde açıklanan özel modülleri kontrol etmek için kullanılmaktadır. Veri belleği EEPROM veri belleğini de içermektedir. Bu bellek, direkt veri belleğine planlanmamış, fakat endirekt olarak planlanmıştır ve endirekt adres göstergeleri okumak/yazmak için EEPROM belleğinin adresini belirlemektedir.

Pic 16F84'de iki hafıza blok'u mevcuttur. Bunlar;
1- Program memory blok
2- Data memory blok

Program memory blok 14 bit kelime boyunda ve 1 Kbyte kapasitesindedir.13 bit ile adreslenir, program bus 14 bittir. Buraya program olarak yazıp Pic içine attığımız hex dosyası bilgileri yazılır, gerekirse silinir ve tekrar yazılır.

Data memory alanı RAM ve EEPROM olmak üzere iki tiptir. RAM alanına programın çalışması sırasında işlenen bilgiler yazılır. Data memory alanında SFR ve GPR registerler bulunur.

Pic 16F84'ün en önemli registerleri şekilde görülmektedir. 68 GPR alanında bizim tanımladığımız; sayaç, timer vs… gibi registerler yazılır.



Şekil 3.2. 16F84 Registerleri

Write Register: 16F84 içinde RAM bellek alnında, görülmeyen, direk ulaşılmayan, adresi olmayan, geçici bir depolama alanı vardır, burası "Write Register" dir. Bu registerler içine yazılan bilgiler başka registerlere aktarılır ve write register bu aktarma işleri için geçici aktarma alanı olarak kullanılır. Aritmetik ve atama işlemleri bu sayede yapılır.

Register adreslerinin bazılarını bilmek programlama işlemleri için önemlidir. Inc. dosyası kullanıldığında, register tanımları yapmaya gerek yoktur fakat temel ve en çok kullanılan registerlerin adreslerini ezbere bilmek gereklidir.

Örneğin:

```
PORTA    EQU    H'05'
PORTB    EQU     H'06'
STATUS  EQU    H'03'
TRISA     EQU  H'85'
TRISB     EQU  H'86'
```

Gibi registerler en temel registerlerdir, programlamada ilerledikçe diğer register adresleri de otomatikman bellekte kalır veya .inc dosyası kullanımı alışkanlığı ile tamamen bellekten silinir.

Aşağıda gerekli olacak register ve onların bit kısaltmaları verilmiştir. Programlama esnasında, register bitleri, kısaltma olarak da yazılabilir. Inc. dosyalarında bunlar tanımlanmıştır.

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on Power-on Reset	Value on all other resets (Note3)
Bank 0											
00h	INDF	Uses contents of FSR to address data memory (not a physical register)								---- ----	---- ----
01h	TMR0	8-bit real-time clock/counter								xxxx xxxx	uuuu uuuu
02h	PCL	Low order 8 bits of the Program Counter (PC)								0000 0000	0000 0000
03h	STATUS (2)	IRP	RP1	RP0	$\overline{TO}$	$\overline{PD}$	Z	DC	C	0001 1xxx	000q quuu
04h	FSR	Indirect data memory address pointer 0								xxxx xxxx	uuuu uuuu
05h	PORTA	—	—	—	RA4/T0CKI	RA3	RA2	RA1	RA0	---x xxxx	---u uuuu
06h	PORTB	RB7	RB6	RB5	RB4	RB3	RB2	RB1	RB0/INT	xxxx xxxx	uuuu uuuu
07h		Unimplemented location, read as '0'								---- ----	---- ----
08h	EEDATA	EEPROM data register								xxxx xxxx	uuuu uuuu
09h	EEADR	EEPROM address register								xxxx xxxx	uuuu uuuu
0Ah	PCLATH	—	—	—	Write buffer for upper 5 bits of the PC (1)					---0 0000	---0 0000
0Bh	INTCON	GIE	EEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF	0000 000x	0000 000u
Bank 1											
80h	INDF	Uses contents of FSR to address data memory (not a physical register)								---- ----	---- ----
81h	OPTION	$\overline{RBPU}$	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0	1111 1111	1111 1111
82h	PCL	Low order 8 bits of Program Counter (PC)								0000 0000	0000 0000
83h	STATUS (2)	IRP	RP1	RP0	$\overline{TO}$	$\overline{PD}$	Z	DC	C	0001 1xxx	000q quuu
84h	FSR	Indirect data memory address pointer 0								xxxx xxxx	uuuu uuuu
85h	TRISA	—	—	—	PORTA data direction register					---1 1111	---1 1111
86h	TRISB	PORTB data direction register								1111 1111	1111 1111
87h		Unimplemented location, read as '0'								---- ----	---- ----
88h	EECON1	—	—	—	EEIF	WRERR	WREN	WR	RD	---0 x000	---0 q000
89h	EECON2	EEPROM control register 2 (not a physical register)								---- ----	---- ----
0Ah	PCLATH	—	—	—	Write buffer for upper 5 bits of the PC (1)					---0 0000	---0 0000
0Bh	INTCON	GIE	EEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF	0000 000x	0000 000u

Legend: x = unknown, u = unchanged, - = unimplemented read as '0', q = value depends on condition.
Note 1: The upper byte of the program counter is not directly accessible. PCLATH is a slave register for PC<12:8>. The contents of PCLATH can be transferred to the upper byte of the program counter, but the contents of PC<12:8> is never transferred to PCLATH.
2: The $\overline{TO}$ and $\overline{PD}$ status bits in the STATUS register are not affected by a $\overline{MCLR}$ reset.
3: Other (non power-up) resets include: external reset through $\overline{MCLR}$ and the Watchdog Timer Reset.

3.5. Taklit Yazılımlara Karşı Koruma

Şöyle durumlar olabilir ki, aygıt EEPROM veri belleğine yazmak istemeyebilir. Taklit yazılımlara karşı korunmak için, değişik mekanizmalar monte edilmiş, kurulmuştur. Yüksek güçte WREN temizlenir. Bunun yanı sıra, yüksek güç timer' i (72 msn süreli) EEPROM yazımını önler. Yazılımı başlatan ardışık ve WREN biti ikisi birlikte 'Brown-Out', güç arızası veya yazılım aksamasında tesadüfi yazılımları önlemeye devam eder.

3.6. Kod Koruma Süresince Eeprom Veri İşlemi

Microişlemci, kod korumalı durumdayken düzene sokulan verileri okuyabilir ve EEPROM verisine yazabilir. ROM aygıtlarında iki koruma biti mevcuttur. Birisi ROM program belleği diğeri ise EEPROM veri belleği içindir.

3.7. Zamanlama 0 (Tımer0) Modülü Ve TMRO Kaydı

TimerO modül, timer/sayaç aşağıdaki özelliklere sahiptir.

• 8 bitlik timer/sayaç
• Okunabilir ve yazılabilir
• 8 bitlik programlanabilir prescaler.
• İçten veya dıştan saat ayarı
• FFh' tan OOh' ye taşma üzeri kesme
• Dış saatin sınır seçimi

Timer modu, TOCS bitinin (OPTION<5>) temizlenmesiyle seçilir. Timer modunda TimerO modülü her bir komut sürecini uzatır. Eğer TMRO kaydı yazılıysa, uzama takip eden 2 süreci engeller. Kullanıcı ayarlanan değeri TMRO kaydına yazarak, bunun etrafından çalışabilir. Sayaç modu TOCS bitinin (OPTION<5>) ayarlanmasıyla seçilir.

Bu modda, TMRO, RA4/TOCK1 pininin sınırlarının her bir artışında yada düşüşünde artacaktır. Genişleyen sınır, TO kaynak sınır seçim biti tarafından, TÖSE (OPTION<4>) tarafından belirlenmektedir. TÖSE bitinin temizlenmesi artan sınırlan seçecektir.

Prescaler, TimerO modülü ile Watchdog Timer arasında paylaşmaktadır. Prescaler ataması, yazılımda PSA biti kontrolü tarafından denetlenmektedir. (OPTION<3>) PSA bitinin temizlenmesi, prescaler' ı TimerO modülüne atayacaktır. Prescaler okunabilir veya yazılabilir değildir. Prescaler TimerO modülüne atandığında prescaler değeri (1:2,1:4...;1:256) yazılım tarafından seçilebilirdir.

3.8. TMRO Kesmesi

TMRO kesmesi, TMRO kaydı FFH'dan OOh'ye akışında üretilmektedir. Bu fazla akım TOIF bitini (INTCON<2>) kurar (ayarlar). Kesme, aktif TOffi bitinin (INTCON<5>) temizlenmesi ile gizlenebilir. (1NTCON<5>) TOIF biti, TimerO modülü tarafından, bu kesmenin yeniden aktifleştirilmesinden önce yazılımdan silinmelidir. TMRO kesmesi (şek.2.4) işlemciyi SLEEP' ten çıkaramaz. çünkü, SLEEP boyunca timer kapalıdır.

3.9. TMRO' IN Dıştan Saat İle Kullanımı

Dıştan saat girişleri TMRO için kullanıldığında, bazı ön şartların gerçekleştirilmesi gerekir. Dıştan saat gereksinimi, içten faz saati senkronizasyonundan kaynaklanmaktadır. Bunun yanı sıra, TMRO kaydının senkronize edilmesinden sonra, fiili artmada gecikme mevcuttur.

3.9.1 Dıştan Saat Senkronizasyonu

Hiç bir prescaler kullanılmadığı taktirde, dıştan saat girişi prescaler çıkışındaki gibidir. RA4/TOCKI pirimin içten faz saati ile senkronize edilmesi iç faz saatlerinin Q2 ve Q4 süreçlerindeki prescaler çıkışını örneklemek yoluyla yerine getirilir.(Şek2.5) Bunun için, TOCKT nin düşük değerinin en azından 2TOSC (artı ufak RC gecikmesi) olması gerekir. Prescaler kullanıldığında, dış saat girişi asenkron sayıcı tipi prescaler' a bölünür ve böylece prescaler çıkışı simetrik olur. Dış saatin örnekleme gereksinmelerini karşılamak için sayaç (counter) dikkate alınmalıdır. Böylece prescaler değerine bölünen en azından 4 TOSC periyot uzunluğuna sahip olmalıdır.

3.9.2. TMRO Gecikme Uzatılması

Prescaler çıkışı, iç saat ile senkronize edildiği için, dış saat sınırlarının meydana gelmesindeki zamandan TMRO modülünün fiili olarak uzatılması zamanına kadar küçük bir gecikme vardır.

3.10. CPU' nun Spesifik Özellikleri

Mikro kontrolör' ü diğer işlemcilerden ayıran şey, gerçek zaman uygulamalarının gereksinmeleri ile ilgili özel devrelerdir. PIC16F87X' te sistem güvenliğini maximize eden, dış elemanları ayırarak maliyeti minimize eden, güç tasarrufu, çalışma modu ve kod koruma gibi özellikleri taşımaktadır. Bu özellikler;

• OSC seçimi
• Reset
• Güç kaynağı reseti (POR)
• Yüksek güç timen (PWRT)
• Osilatör başlangıç Timer ı (OST)
• Kesmeler
• Watchdog Timer
• Sleep
• Kod koruma
• İD yerleşimleri
• Devre içi seri programlama

PIC16F87X' te yalnızca konfigürasyon bitleri tarafından kapatılabilen Watchdog Timer mevcuttur. Güvenliği arttırmak için bu kendi RC osilatörünü de çalıştırmaktadır. Yüksek güçte gereken esas gecikmeleri sağlayan 2 Timer mevcuttur. Bunlardan birisi Osilatör Başlangıç Timer 'ıdır. Bu timer, kristal osilatör durgunlaşıncaya kadar cipi resette tutar. Diğer timer ise yalnızca nominal yüksek güçte 72 ms sabit gecikme üreten Yüksek Güç Timer' ıdır. Bu güç kaynağı stabilize olurken aygıtı resette tutar. Bu iki cip üzeri Timer ile , uygulamaların çoğu hiçbir reset devrelerini gerektirmemektedir. SLEEP modu çok düşük enerjili alçak güç modunu sunmaktadır. Kullanıcı SLEEP ten çıkmak için dış reset, Watchdog Timer zaman aralığı veya kesmeleri kullanabilir. Bazı osilatör seçenekleri, kısımları uygulamaya yerleştirmek için elde edilmektedir. RC osilatör seçeneği sistem maliyetini, LP kristal seçeneği ise güç sarfiyatını düşürmektedir. Çeşitli seçenekleri seçmek için konfigürasyon bitler seti kullanılmaktadır.

BÖLÜM 4. PIC MİKROİŞLEMCİLERİN PROGRAMLANMASI

Öncelikleri şu kısa tanımların bilinmesi gerekmektedir.
ESC : Escape
FS: File Seperatör
GS: Group Seperatör
RS: Recoud Seperatör
US: Unit Seperatör
DEL: Delete
SP: Space
FF: From Feed
CR: Carriage Retum
SO: Shift Out
SI: Shiftln

4.1. Kaynak Kod Yazımı

Program yazarken, 4 temel kural izlemeniz önemlidir:

1.Yazılımınız hakkında daima açıklamalar yapın. Yoksa daha sonraki aşamalarda kaçınılmaz olarak geri döndüğünüzde , neyi neden yaptığınızı anlayabilmek güç olur.

2.Programlarınız için evrensel bir header-başlık kullanın. Bu iş yükünü hafifletir, tutarlı bir format yaratır ve hatırlamanız gereken değişken sayısını azaltır. Daha ileriki bir bölümde evrensel bir başlık örneği bulabilirsiniz.

3.Tüm alt rutinleri tek bölgede toplayın. PIC 'le çalışırken bu bölge her bellek sayfasının üstünde (00-FFh) olmalıdır. Bunun nedeni daha sonra açıklanacaktır.

4.Yazılımın ne yapmasını istediğinizi hatırlamak için bir iş akış diyagramı veya başka bir çizimden yararlanın ve gerçek dünyayla nerede arabirim yapılandıracağınızı gösterin (I/O).

Alt rutinlerin, sıçramaların ve dallanmaların nerede bulunduğunu göstermek için etiketler kullanın. Bu bir sıçramanın konumunun onaltılık değerini hatırlamaktan daha kolay olacaktır. Etiketler ayrıca programın başında bazı ad ve değerlerin tanımlanmasında kullanılır. Bu etiketlere denklik (equate) adı verilir. Örneğin ;

PORTB EQU 6: PORTB 'nin programın herhangi bir yerinde kullanılmasını sağlar; 6 değeriyle belirtileceğini gösterir.

MOVWF PORTB: Örneğin PORTB ' yi sıkça kullandığınızı düşünürsek, sayısal bir değeri hatırlatmaktansa bu değeri PORTB ' ye baştan verip onu bu etiketle kullanmak size daha fazla kolaylık sağlayacaktır.

Programda çift denklik ve etiketlerin bulunmamasına dikkat edin. Assembler bunları ayıklayamaz ve bug 'dan arındırma gerekecek şekilde tahribata yol açar.

1. TEMP EQU 5: TEMP değerini 5 yapar

2. Doğru komut ADDLW TEMP: TEMP değeri W yazmacındaki değere eklenir.

3. Yanlış komut OTO TEMP: TEMP etiketinin bulunduğu adrese gitmek istemenize rağmen programın örneğin 4511 adresinde bulunabilecek bir TEMP etiketi yerine 5H adresine sıçramasına neden olur.

Alt rutinler, makrolar ve kod yığınları için bir kütüphane kurun. Örneğin LED.TXT'den 20 satır ve SWITCH.TXT 'den 5 satır alın.

Yazdığınız kodun bir başlangıcı ve bir sonu olmalıdır. PIC 'in her reset durumunda başlangıç daima tanımlıdır. Son sizin ise tarafınızdan tanımlanmalıdır. Bu programın sonsuz kere dönemeyeceği anlamına gelmez.

Sondan yoksun program döngüye devam edecektir ve bu durumda birinin hata aramak için uğraşması çoğunlukla sonuç vermeyecektir. Programlarınız için evrensel bir başlık örneği:

```
YAZAN
;TARİH                    REVİZYON
;DOYA ADI
;PIC 16F87X için          18/28/40 BACAKLI AYGITLAR
;REZONATÖR                4 MHz
;KOMUT  SAAT  HIZI        1.00Mhz T = 1Us
;WATCHDOG                 DEVREDE/DEVRE DIŞI
;KOD KORUMA               AÇIK/ KAPALI
;YAZILIM FONKSİYONU       (Kodun ne için yazıldığı ve ne yaptığı tanımlanır)
;LİSTE                    Buraya assembler direktifleri listesi koyulur.
; bellek denklikleri
; programın gerektirdiği gibi adlan ve yer tayinlerini tanımlayın.
ORG  OOH   ;  PIC tipine göre adres değiştirilir. ;  Reset vektörü bölümüne bakınız. GOTO  INIT  ;  Programın başlamasını istediğiniz yer burasıdır.
; alt rutinler burada başlar
INIT; Program burada başlar
;PROGRAMINIZ
END; Ve burada biter bu komut programın son satırında yer almalıdır ki assembler nerede duracağını bilsin.
```

Kaynak dosyanın en üstteki bilgi kısmı hariç bu bölümü, PIC.H olarak kaydedilebilir ve bir INCLUDE PIC.H ifadesiyle kaynak kodu içerisinden çağrılabilir. I/O denklikleri PIC.H 'de göz ardı edilebilir. Bunların kullanımları programdan programa çeşitlilik gösterir. I/O denkliklerinin atlanması kaynak dosyanın büyüklüğünü çok küçük miktarda azaltır ama .OBJ ve .LST dosyalarının büyüklükleri de azalacağından faydalı olabilir.

Pic programlamak için gereken temel şeyler şunlardır;

1-) Pic programlamak için pic Assembly dilini bilmek ve program yazarak bunu hex dosyası haline getirmek
2-) Pic programlayacak donanıma sahip olmak, bunlar bilgisayar ve gerekli yazılımlardır.

Assembler veya Complier:
MS-DOS/Edit
WİNDOWS/notepad
MPLAB/Pfe

Gibi bir text editöründe Pic Assembly dili ile yazılmış programı derlemeye yani HEX file haline getirmeye yarayan programdır.Bu iş için biz Microchip firmasının MPLAB içinde gelen MPASM programını kullanacağız.

4.1.1. Pic Assembly

Pic'in yapmasını istediğimiz işlemleri, pic'in anlaması için, kısa komutlardan oluşturulmuş bir dildir. Bu dil ile yazılan program ASM dosyası olarak kaydedilir daha sonra MPASM ile HEX dosyası haline getirilip, pic içine yazılır.

4.1.1.1. Pic Assembly Genel Kuralları

Genel kurallara geçmeden önce son olarak şu konuya da dikkatinizi çekelim. Bir program yazıldıktan sonra complier'in bunu tanıması için ASM olarak kaydedilen bu dosya, complier'in yani mpasmwin'in bulunduğu direktörde olmalıdır. Eğer program yazımında inc dosyası kullanılmış ise o da burada olmalıdır. Inc dosyası program içinde yazılı font ile klasör altında bulunmalıdır, aksi halde compiler tarafından tanınmaz.

PROGRAM FİLES / MPLAB/

PicProg

mplab

mpasmwin

deneme.asm

deneme.hex

deneme. Err

deneme.lst

16f84.inc

gibi bir düzen ortaya çıkar.

Noktalı Virgül

Bir program yazılırken derlenmeyen satırların başına konur, bizi veya programı inceleyeni bilgilendirecek bir bilgiyi; işaretinden sonra yazarız, yine bu işaretten sonra çeşitli şekiller ve süslemeler yapabiliriz.

```
;===============================================
; Bir deneme programıdır
;===============================================
; ----------------ooooooooooooooooo-------------------------
```

program içine bir satır yazıldıktan sonra satır yanına açıklayıcı bilgi (;) işaretinden sonra yazılır.

```
SAYAC   EQU    h'0D'          ; h'0D ' adresinde sayac diye bir register tanımla
BSF     STATUS,5 ; bank 1 ' e geç
```

Burada da complier noktalı virgül den sonraki açıklamaları görmez, bu sebeple buraya istediğimiz açıklamaları yazabiliriz.

Bir pic'i programlamak için yapmamız gerekenler sırası ile şöyledir:

1-) Bilgisayarımıza gerekli Software'i yüklemek

2-) Pic Assembly dilinin genel kurallarını öğrenmek

3-) 16F84'ün 35 komut setini öğrenmek

4-) Program yazıp ASM olarak kaydetmek

5-) Programı hatasız yazıp Compile edebilmek

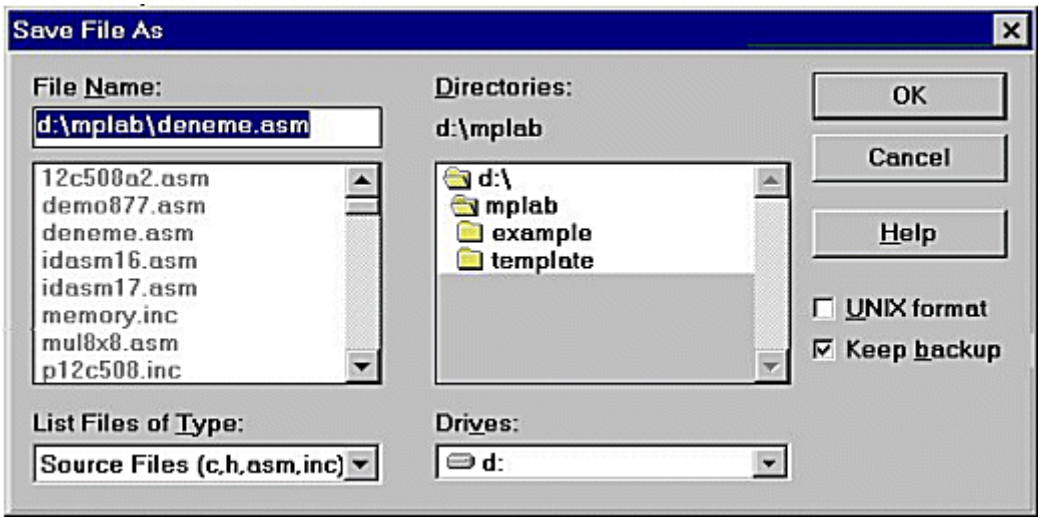
6-) Compile edilip HEX haline gelmiş dosyayı EASYPİC ile pic'e aktarmak.

4.1.2. Software

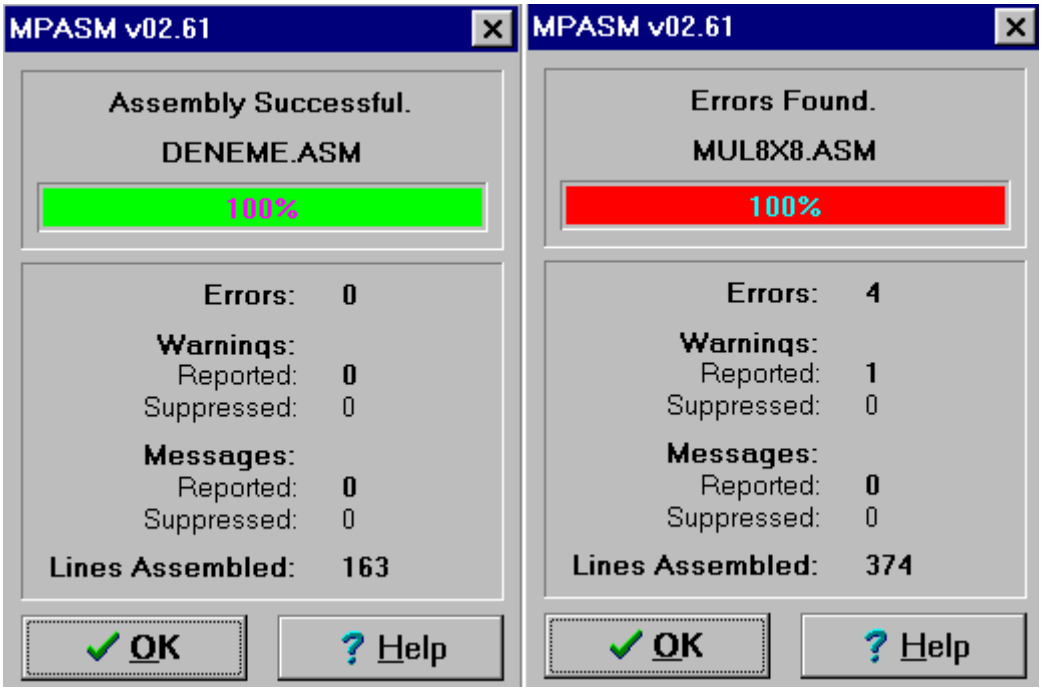
Microchip firmasının sitesinden ücretsiz olarak indirilen program MPLAB v5.20, file manager içinde MP520ful 10.659 Kb Application dosyası şeklinde görülür. Bu dosyanın yeni versiyonunun adı ve boyu değişik olabilir. Burada zip file olarak verilen dosya budur ve internetten indirebilirsiniz.

Program kendisini C /Program Files/MPLAB /altında install eder. Burada oluşan Mpasmwin ve Mplab application dosyalarını kısayol olarak masa üstüne taşıyınız.

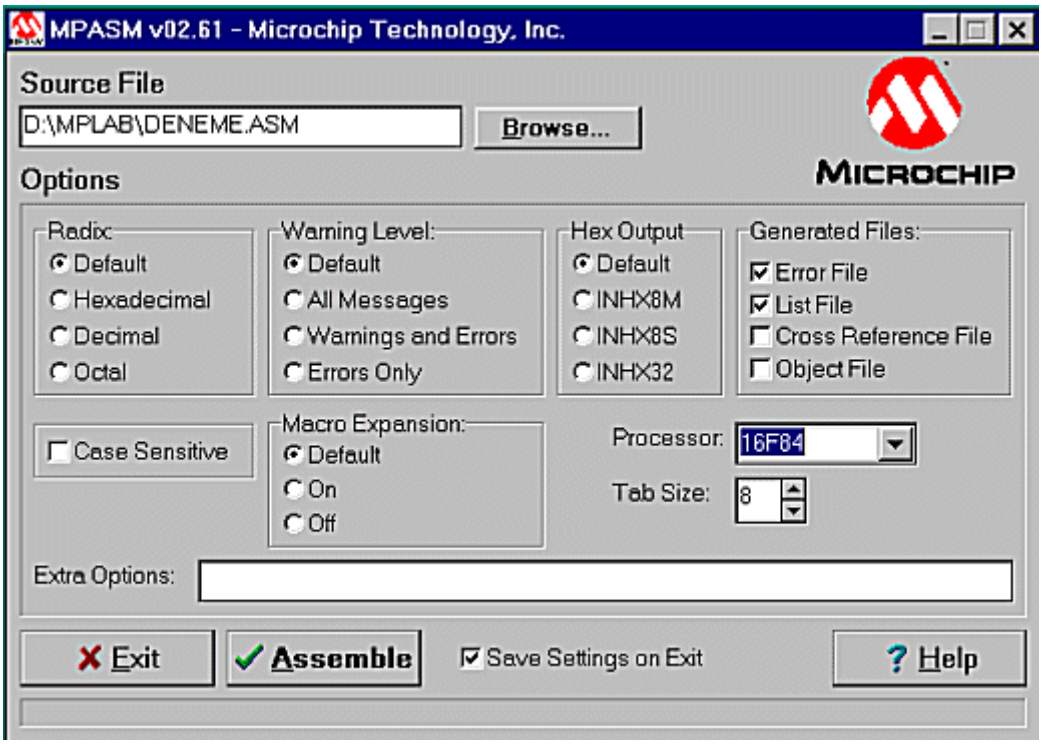
Masaüstünden MPLAB'ı tıklayarak File /new altından yeni bir sayfa açınız. Bu sayfaya artık pic assembly komutlarını kullanarak programınızı yazabilirsiniz. Bu programı sonu ASM olacak şekilde kaydetmeniz gerektiğini tekrar söyleyelim.



Programdan çıkmadan Project/build ile dosyayı HEX haline getirebilirsiniz ve ASM dosyası ile aynı yerde, aynı isimde HEX dosyanız açılmış olur. Bu işlem sırasında yazdığınız program doğru ise, başarılı olduğuna ilişkin, hatalı ise hatalı olduğuna dair bir mesaj verilir. MPLAB içinde yazıp ASM olarak kaydettiğiniz dosyayı, masa üstünden MPASM'ı açarak ve dosyanızı "source file name" penceresinden bularak compile edebilirsiniz. Bu sırada başarılı ise yeşil, başarısız ise kırmızı uyarı işareti verilir.

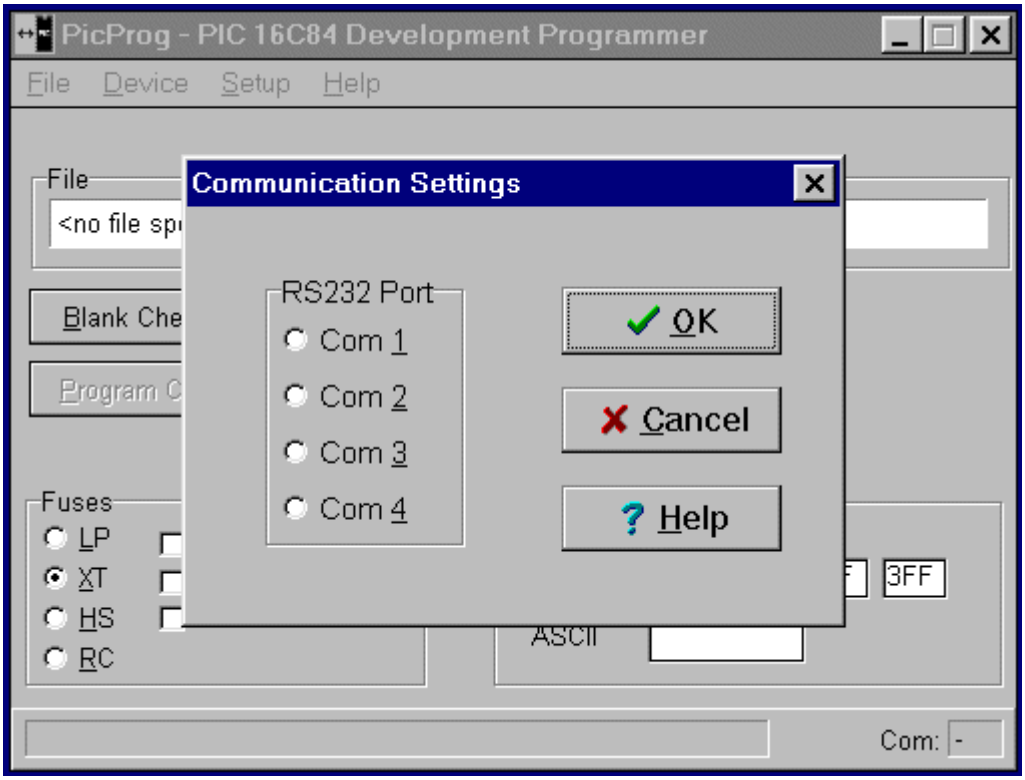


Başarısız ise, oluşturulmuş bulunan err dosyasını editörde açarak,hatanızın nerede olduğunu görebilirsiniz.Bu iş için şekilde görülen dosyalar içinde error file kısmının işaretli olması gerekir.

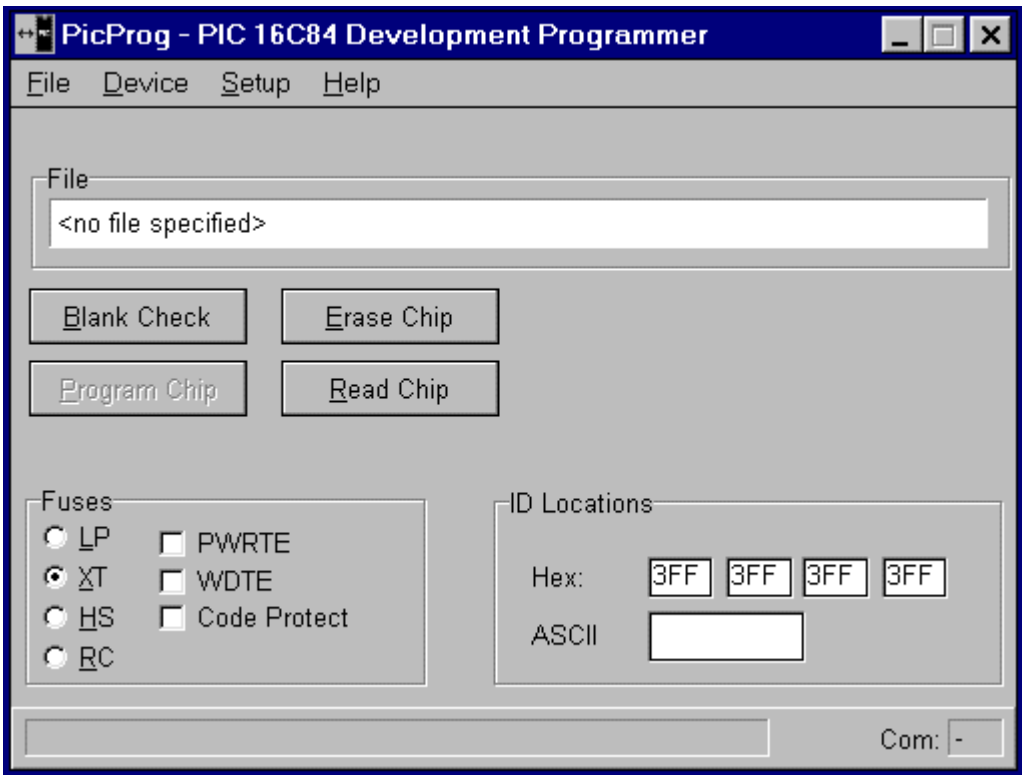


Başarı ile compile edilmiş dosyayı, pic'e aktarmak için kullanılan program picprog dur, 16f84 için başka programlar da mevcuttur. Programlayıcılar da paralel veya seri porttan çalışabilirler. Biz burada picprog ve seri porttan çalışan Easypic programlayıcısını kullanacağız.

Picprog küçük bir dosyadır bu dosyayı uygun bir klasör altına koyabilirsiniz, bu program 331 Kb lık bir application dosyasıdır. Bu dosyayı da masa üstüne taşıyınız. Elinizde bir HEX file varsa picprog penceresinden o file'ı bulunuz ve önce Com port ayarını (bu ayar bir kez yapılırsa hep aynı kalır) yapınız. Daha sonra gerekli ayarları yaparak pic'i programlayınız.



Com port genelde bilgisayarınızın konfigürasyonuna bağlı olarak Com1 veya Com 2 dir.



PicProg program penceresinden hex file bulunur. xt işaretlenir ve program butonuna basılır. Programlama hardware'i ne ileride tekrar döneceğiz.

4.2. Assembler Direktifleri

Kod yazılımında kullanılan komutlara (ADDLW, MOVWF gibi) ek olarak, kod yazılımım hızlandırabilecek assembler direktifleri vardır. Assembler direktifi, kaynak kod içine, kontrol etmek, direktif vermek veya zaman kazanmak için yazılan ve yazılımın işletimine hiç etkisi bulunmayan bir komuttur. Assembler direktifleriyle çalışmanın sonuçlan, ileride .LST dosyasında görülecektir. Bu komutlardan bazıları define ve org'u içerir. MPASM assembler kullanıcı rehberinde tam bir liste verilmiştir. Ancak en faydalı olanları burada anlatacağız :

≠DEFİNE: Bir dizinin , birkaç dizinin yerini tutması için kullanılır. Örneğin port ve bit bilgileri bir tek isim altında toplanır ve program boyunca bu isim kullanılır.

≠define Switchl porta,0 ; bit O'ı portA'da tanımlar

≠define Z-bit status,2; sıfir bitini statüs yazmacında tanımlar

Böylelikle yukarıdaki isimler aşağıdaki gibi kullanılmış olur.

Bu , hummalı bir şekilde kod yazımıyla uğraşırken yanlış bit veya port'u yazma veya test etme olasılığını ortadan kaldırmaya yardımcı olur.

PAGE Program bölümlerinin ayrı ayrı yazdırılmasını sağlamak amacıyla sayfayı dışarı çıkmaya zorlar.

LINES Sayfa başına yazdırılacak satır sayısını ayarlar. Standardın dışında kağıtlar kullanılıyorsa yararlı bir komuttur.

C Yazdırma işleminde sütun genişliğini belirler.

C=80, A4 sayfalarda ;
C=132 ise geniş yazıcılarda kullanılır.
INCLUDE Harici dosyalan program içine çeker. Önceden tanımlanmış makroları, işlemci için tüm genel denklik ve yazmaç tanımlarını veya geniş bir tabloyu (örneğin herhangi bir dönüşüm tablosu) içeren bir dosya kurulumunuz varsa bu komutu kullanabilirsiniz. Bu komutun kullanımı

Include "PICREG.H" veya
Include "C:\PIC\macros.asm" şeklinde olabilir.

RES Programda kullanılmak üzere bellek konumlarını rezerve eder. Programın başında yazmaç denkliklerini tanımlamanın bir yolu da şudur:

```
Templ      equ   Och
Temp2      equ   Odh
Count      equ   Oeh
```

Bunun yerine şöyle de yazılabilir:

```
Org    Och
Templ  resi
Temp2  resi
Count  resi
```

Bu yazmaçları tekrar tanımlamadan yazmaç listesine etiketler eklemenizi mümkün kılar.

IF ELSE ENDIF komutu

Bu direktif programın isteğe göre ayarlanmasının assemble etme aşamasında gerçekleştirilmesini olanaklı kılar. Program assemble edildiğinde stop bit'i, parite ve bit sayılarının seçilebildiği veri iletişiminde veya değişik PIC 'lerde reset vektörünü

sıfırlamasını seçebilmek buna örnek olarak gösterilebilir. Böylelikle kullanılmayan kod devre dışı bırakılır ve tüm programın boyutu küçülür.

Bir veri iletişim programında , seçim öncelikle en önemli (Most Şignificant Bit) veya en az önemli bit'in çıktısının alınmasıysa , programın tamamıyla farklı olması gerekir. Ancak, MSB'nin l veya O olması için gerçekleştirilen her bir testte IF ELSE ENDIF komutları konulması mümkün olabilir.

Örnek IF MSB
rrf xmit_reg.f ; MSB=1 ise bu satır derlenecektir.
ELSE
rlf xmit_reg.f ; değilse bu satır derlenecektir.

ORG İfadeyi izleyen kodun kökenini belirler. Programın, kodun bir kısmının (örneğin kesme vektörü) veya belleğin bir bloğun nerede başlayacağını belirler

LIST İşlemci, sayfa başına satır sayısı, sütun genişliği ve programın yazıldığı format (radıks) hakkındaki bilgileri assembler'a aktarmak için kullanılır.

Örnek:
list C=80, n=55, p=16c84, r=hex Makrolar Makro kullanımı, bir kütüphaneden INCLUDE komutu aracılığıyla sık kullanılan rutinleri çağırarak , kod yazımından tasarruf etmenin bir yoludur. Aynı bilgiyi tekrar tekrar yazmamanızı ve program yazımında tutarlılığı sağlar. Bir makro kütüphanesi bir test etme, led'lerin yanıp sönmesi, a/d dönüştürme rutinleri, port kurulumları E2 rutinleri vb. içerebilir ve bunlar yalnızca bir satırla çağrılabilir.

Örnek;
Set_A0 macro; makro başını tanımlar
ENDIF
bcf adcon0,0
bcf adcon0, l

endm ; bu da sondur.
Bu satırlar aşağıdaki satırları yazabilmenize izin verir:
set_OA ; kanalı seçer
cali canv ; alt rutini dönüştürmek için çağrı yapar.

Böylelikle belirli bir a/d kanalı için hangi bit'lerin kurulması veya kaldırılması gerektiğini hatırlamaya gerek kalmaz. Yukarıdaki satırlar assemble edildiğinde .LST dosyasında şöyle görülür:

```
bcf    adcon0.0
bcf    edcon0,1
call    conv.
```

4.3. 16F84'te Kullanılan Komutlar

1- #define : Uygulamalarımızda #define komutunu daha çok port bitlerine değişken isimler atamada kullanacağız. Bu şekilde yazdığımız programlar da anlaşılır bir hal alacaktır.
Örnek:

```
#define buton    porta,3    ; Bu komutla -PortA,3- yazacağımız yerlerde -buton- yazmamızda aynı işi görecektir. Böylelikle programımızı kontrol ederken -PortA,3- ün ne olduğunu araştırmamıza gerek kalmayacak ve bunun -buton- olduğunu anlayacağız.
```

2- clrf : Sonrasına yazılan portu tamamen sıfırlar.
Örnek:

```
clrf    Porta    ; A Portunu sıfırlar. Genellikle programın başına yazılmasında; fayda vardır.
```

3- bsf : Portların çıkışına mantık 1 çıkışı vermeyi sağlar.
Örnek:

```
bsf    Porta,3    ; A portunun 3. bitini mantık 1 yükler.
```

4- bcf : Portların çıkışlarını mantık 0 (sıfır) yapar.

Örnek:

```
bcf    Portb,5     ; B portunun 5. bitini 0 (sıfır) yapar.
```

5- btfss : Program akışında portun belirtilen bitinin mantık 1 olup olmadığını kontrol etmekte kullanılır.

Örnek:

```
main    btfss     Porta,2    ; A portunun 2. biti mantık '1' ise sonraki komutu
                                ;atla.
        goto      main       ; Porta,2 mantık '0' ise -main- etiketli bölüme geri
                                ;dön.
        bsf       Portb,6    ; B portunun 6. bitini çıkış mantık 1 yap.
        .....     ........
```

Bu komut setinde A portu mantık 1 olana kadar sürekli 1. ve 2. satır arasında döngüye girer. A portunun 2. biti mantık 1 olduğunda, program 2. satırı uygulamayıp daha sonraki satırı işlemeye başlayacaktır.

6- btfsc : btfss (5. komut) ile aynı özellikte, ancak mantık 0 (sıfır) seviyesini kontrol eder.

7- goto : Programda istenen yere gidilip işlemlerin oradan itibaren yürütülmesini sağlar.

Örnek :

```
main   bsf     Portb,4   ; B portunun 4. bitini 1 yap.
       .....   ......    ; Buraya bir geciktirme rutini (Bkz. 10. madde)
                          ;yerleştirilebilir.
       bcf     Portb,4   ; B portunun 4. bitini 0 (sıfır) yap
       goto    main      ; -main- etiketli satıra gider ve işlem oradan sonra
                          ;tekrar devam eder.
```

8- equ : Bu komut PortA, PortB, ve yazmaçların adreslerinin atanmasını sağlar.

Örnek ***:*** PortA nın PIC16F84 teki adresi 05 tir.

```
porta   equ   05   ;Bu komutla PortA nın PIC içerisinde bulunduğu adresin 05
olduğunu tanımlamış olduk.
portb   equ   06   ; Bu komutlada PortB nin adresini tanımlamış olduk.
```

9- movlw-monwf : Bu komutlarla PIC içerisinde bulunan -W- yazmacına (Değişkenine) değer atanır. Programın ilerleyen aşamalarında bu değer okunup portlara atanabilir. Gerekli görüldüğü yerlerde bu -W- yazmacının değeri arttırılıp-eksiltilerek değişik değerlerin portlara atanmasını da sağlarlar

Örnek:

```
movlw    80H    ;  W  yazmacına  80H  değeri  yüklenir.
movwf    trisb  ; W yazmacındaki 80H değeri PortB ye yüklenmiş oldu.
```

10- Geciktirme rutinleri : Geciktirme rutinleri; programın işleyişi esnasında, herhang ibir işlemden sonra belli bir süre beklemesini istediğimiz zamanlarda kullanmamız gereken zamanlayıcı alt programlarıdır.

Örnek:

Bir yürüyen ışık devresinde, her ledin yanmasından sonraki satıra geciktirme rutini yerleştirmezsek ledlerin hepsini aynı anda yanıyormuş gibi görürüz. Bunu önlemek için ledleri yakan her program satırından sonra bir geciktirme rutini eklememiz gerekir. Aşağıda basit bir örnek göreceksiniz. Butür alt rutinler genellikle programın sonuna yazılır.

```
.....      .....
.....      .....
delay0     equ      0DH ; delay1 yazmacını 0DH adresinde tanımladık.
delay1     equ      0EH  ; delay2 yazmacı 0EH adresinde.
```

```
delay2    equ      0FH  ; delay3 yazmacı 0FH adresinde.
.........  .....     .....
.........  .....     .....  ; Ana programın işlediği program satırları.
          bsf      portb,1  ; Portb nin 1. bitini mantık '1' yap.
          call     DELAY    ; DELAY etiketli alt rutini çağır.
          bcf      portb,1  ; Portb nin 1. bitini mantık '0' yap.
          bsf      portb,2  ; Portb nin 2. bitini '1' yap.
          call     DELAY    ; DELAY etiketli alt rutini çağır.
          bcf      portb,2  ; portb nin 2. bitini '0' yap.
          .....    .........  ; Program bu şekilde devam edebilir.
DELAY     movlw    .4       ; W yazmacına .4 değeri atandı.
          movwf    delay0   ; W yazmacındaki .4 değeri delay0
                            ;yazmacına atandı.
D0        movlw    .200     ; W yazmacına .200 değeri atandı.
          movwf    delay1   ; W yazmacındaki .200 değeri delay1
                            ;yazmacına atandı.
D1        movlw    .200     ; W yazmacına .200 değeri atandı.
          movwf    delay2   ; W yazmacındaki .200 değeri delay2
                            ;yazmacına atandı.
D2        decfsz   delay2,F ; delay2 değerini azalt. '0' olduysa sonraki
                            ;komutu atla.
          goto     D2       ; D2 etiketli satıra geri dön.
          decfsz   delay1,F ; delay1 değerini  azalt '0' olduysa
                            ;sonraki komutu atla.
          goto     D1       ; D1 etiketli satıra git.
          decfsz   delay0   ; delay0 değerini azalt '0' olduysa sonraki
                            ;komutu atla.
          goto     D0       ; D0 etiketli satıra git.
          retlw    00       ; rutinin çağrıldığı satırın alt satırına
                            ;dön.
```

Bu program grubunda önce portb nin 2. biti çıkışı '1' oluyor, ardından -call DELAY- komutuyla geciktirme rutini çağrılıp çalıştırılıyor. DELAY rutininde 3 ayrı döngü kurulmuştur. Her döngüde -delayX- değerleri azaltılıp '0' değerine kadar düşmesi sağlanmaktadır. -delayX- değerlerinin hepsi '0' değerine ulaştığında program -retlw 00- komutuyla -call DELAY- komutunun altındaki satıra geri dönüp işlemlerin o satırdan itibaren işlemesini sağlamaktadır.

11- decfsz : Sonrasına yazılan değişkenin değerini azaltıp '0' olup olmadığını kontrol eder.

Örnek: Aşağıdaki örnek tek döngülü bir geciktirme (Bekleme) rutinidir.

```
DELAY   movlw   .200      ; W yazmacına .200 değeri atandı.
        movwf   delay0    ; W yazmacındaki .200 değeri delay0 yazmacına atandı.
D0      decfsz  delay0    ; delay0 değerini azalt '0' olduysa sonraki komutu atla.
        goto    D0        ; D0 etiketli satıra git.
        goto    x         ; -x- etiketli satıra git.
```

12- ADDWF komutu: W ile f nin toplanması

Syntax: [etiket] ADDLW f,d

Değişkenler: 0<k<127

d € [0,1]

işlem: (W) + (k) → (hedef)

Etkilenenler: C, DC, Z

Dekodu:

00	0111	Dfff	Ffff

Tanımlama: W yazmacının içeriğini, 'f yazmacına ekler. Eğer 'd' O değerliğinde ise sonuç W yazmacında saklanır. Eğer 'd' l ise, sonuç 'f yazmacına geri yüklenir.

Örnek :

ADDWF FSR, O

Emirden önce;

W = 0x17 FSR = OxC2

Emirden sonra;

W = OxD9 FSR = OxC2

13- ANDWF komutu: W ile f nin AND lenmesi

Syntax: [etiket] ANDWF f,d

Değişkenler: $0 \leq k \leq 127$

işlem: (W) .AND. f→ (hedef)

Etkilenenler: Z

Dekodu:

00	0111	Dfff	Ffff

Tanımlama: W yazmacının içeriği ile T yazmacı ile AND lenir. Eğer 'd' değeri O ise sonuç W yazmacına yazılır, l ise sonuç geri T yazmacına yazılır.

Kelimeler: 1

Döngüler: 1

Örnek:

ADDWF FSR,0

Emirden önce;

W = 0x17

FSR = OxC2

Emirden sonra;

W = 0x17

FSR = 0x02

14-COMF komutu: f 'i tümler

Syntax: [etiket] COMF f,d

Değişkenler: O ≤ f ≤ 127

işlem: d € [0.11]

(f) → (hedef)

Etkilenenler: Z

Dekodu :

00	1001	dfff	ffff

Tanımlama: T yazmacının içeriğini tümler. Eğer 'd' 0 değerini içerirse sonuç W de saklanır. Eğer 'd' l değerini içeriyorsa ise sonuç T de saklanır.

Kelimeler: l

Döngüler: l

Örnek:

COMF REG1,0

Emirden önce;

REGl = 0xl3

Emirden sonra;

REG1 = 0x13

W = 0xEC

15- DECF komutu: f i bir azaltmak

Syntax: [etiket] DECF f,d

Değişkenler: O < f <, 127

İşlem: d € [0,1]

(f)-l→ (hedef)

Etkilenenler: Z

Dekodu:

00	0001	dfff	ffff

Tanımlama: f yazmacını bir azaltır. Eğer 'd' 0 değerini içerirse sonuç W de saklanır. Eğer 'd' l değerini içeriyorsa ise sonuç f de saklanır.

Kelimeler: 1

Döngüler: 1

Örnek:

DECF CNT, 1

Emirden önce;

CNT = 0x01

Z = 0

Emirden sonra;

CNT = 0x00

Z =1

16- INCF komutu: F yi bir artırmak

Syntax: [etiket] INC f,d

Değişkenler: O ≤ f ≤ 127

d € [0,1]

İşlem: (f) + 1 → (hedef)

Etkilenenler: Z

Dekodu:

00	1010	dfff	ffff

Tanımlama: T yazmacının içeriği bir artırılır. Eğer 'd' 0 değerini içerirse, sonuç W yazmacına yerleştirilir, eğer 'd' nin değeri 1 ise, sonuç T yazmacına yazılır.

Kelimeler: 1

Döngüler: 1

Örnek:

INCF CNT, 1

Emirden önce;

CNT = 0Xff

Emirden sonra;

CNT = 0x00

Z=1

17- RETFIE Komutu: Interrupt (kesme) dan geri dönme

Syntax: [etiket] RETFIE

Değişkenler: Yok

işlem: TOS → (PC)

Etkilenenler: yok

Dekodu:

00	0000	0000	1001

Tanımlama: Stack (yığın) poplanır ve yığının en üstündeki veri (TOP) PC ye yazılır. Global Interrupt Enable (Genel kesme açıcı) bit tarafından kesmeler kaldırılır. Bu 2 döngülü bir emirdir.

Kelimeler: l

Döngüler: 2

Örnek:

RETFIE

Emirden sonra;

PC = TOS

GIE = l

18-RETLW Komutu: W ye değer döndürme

Syntax: [etiket] RETLWk

Değişkenler: O < k < 255

işlem: (k) → (W)

(TOS) →(PC)

Etkilenenler: Yok

Dekodu:

11	01xx	kkkk	kkkk

Tanımlama: W yazmacına 8 bitlik 'k' değeri yüklenir. Program counter yığının en üst değeri ile doldurulur (adres döndürülür). Bu komut 2 döngülüktür.

Kelimeler: l

Döngüler: 2

Örnek:

```
CALL  TABLE ; W tablonun offset değerini içerir
; W şimdi bir tablo değerine sahiptir.
 TABLE
        ADDWF  PC ; W = offset
        RETLWkl ; tablo başı
        RETLWk2 ;
        .
        .
        .
        RETLWkn ; tablonun sonu
```

Emirden önce;

W = 0x07

Emirden sonra;

W = k7 nin değeri

19- LRETURN Komutu: Alt programdan dönme

Syntax: [etiket] RETURN

Değişkenler: Yok

İşlem: TOS → (PC)

Etkilenenler: Yok

Dekodu:

00	0000	0000	1000

Tanımlama: Altprogramdan dönüşü sağlar. Stack (yığın) poplanır ve yığının en üstündeki veri (TOP) PC ye yazılır. Bu 2 döngülü bir emirdir.

Kelimeler:1

Döngüler: 2

Örnek:

```
RETURN Emirden sonra;
        PC = TOS
```

20- XORLW Komutu W ile bir değerin XOR lanması

Syntax: [etiket] XORLW k

Değişkenler: O < f < 255

İşlem: (W). XOR. k -» (W)

Etkilenenler:Z

Dekodu:

11	1010	kkkk	kkkk

Tanımlama: W yazmacının içeriği, 8 bitlik 'k' değeri ile XOR lanır. Sonuç W yazmacına yerleştirilir.

Kelimeler: l

Döngüler: l

örnek:

XORLW OxAF

Emirden önce;

W = OXB5

Emirden sonra;

W = OxlA

21- INCFSZ Komutu f'i bir artırır, eğer 0 ise atlar

Syntax: [etiket] INCFSZ f,d

Değişkenler: 0 < f < 27

d e [0,1]

İşlem: (f) + l→ (hedef); eğer sonuç = 0 ise atla

Etkilenenler: Yok

Dekodu:

00	1111	dfff	ffff

Tanımlama: T yazmacının içeriğini bir artırılır. Eğer 'd' O değerini içerirse sonuç W de saklanır. Eğer 'd' l değerini içeriyorsa ise sonuç T de saklanır. Eğer sonuç O ise bir sonraki emire atlanır. O emir erine bir NOP icra ettirilir, bu komut 2 döngülüdür.

Kelimeler: l

Döngüler:l (2)

Örnek:

HERE INCFSZ CNT , l

GOTO LOOP

CONTINUE •

Emirden önce;

PC = address HERE

Emirden sonra;

CNT = CNT + l

ifCNT =0,

PC = address CONTINUE

ifCNT ≠ 0,

PC = address HERE+1

22- IORWF W ile f nin OR lanması

Syntax: [etiket] IORWF f,d

Değişkenler: 0< f < 127

d € [0,1]

İşlem: (f) → (hedef)

Etkilenenler: Z

Dekodu:

00	0100	dfff	ffff

Tanımlama: W yazmacının içeriği 'f yazmacının içeriği ile OR'lamr. Eğer 'd' O ise sonuç W yazmacında saklanır. Eğer 'd' l ise sonuç 'f yazmacına yazılır.

Kelimeler: l

Döngüler: l

örnek:

IORWF RESULT,0

Emirden önce;

RESULT = 0X13

W = 0x91

Emirden sonra;

RESULT = 0X13

W = 0x93

23- MOVF Komutu f yi hareket ettir

Syntax: [etiket] MOVF f,d

Değişkenler: O < f < 127

d € [0,1]

İşlem: (f) → (hedef)

Etkilenenler: Z

Dekodu:

00	1000	dfff	ffff

Tanımlama: f yazmacının içeriği hedef d ye hareket ettirilir. Eğer d=0 ise, hedef W yazmacıdır. Eğer d=l ise, hedef f yazmacının kendisidir. d=l ise, Z durum bayrağı set edildiğinde, f yazmacının testi kullanılır.

Kelimeler: l

Döngüler:l

Örnek:

MOVF FSR, 0

Emirden sonra;

W = FSR yazmacındaki değer

24- MOVWF W yi f ye hareket ettirme

Syntax: [etiket] MOVWF f

Değişkenler: 0<f<127

İşlem: (W) → (f)

Etkilenenler: Yok

Dekodu:

00	0000	1fff	Ffff

Tanımlama: W yazmacından yazmaca veri transferi

Kelimeler: 1

Döngüler: 1

Örnek:

MOVWF OPTION

Emirden önce;

OPTION = 0xFF

W = 0x4F

Emirden sonra;

OPTION = 0x4F

W= 0x4F

25-NOP İşlemsiz

Syntax: [etiket] NOP

Değişkenler: Yok

İşlem: İşlemsiz

Etkilenenler: Yok

Dekodu:

00	0000	0xx00	0000

Tanımlama: İşlem yapılmaz

Kelimeler: 1

Döngüler: 1

Örnek: NOP

26- RLF f yi sola Carry üzerinden kaydırır

Syntax: [etiket] RLF f,d

Değişkenler: 0<f<127

d e [0,1]

işlem: Aşağıdaki yapıya bakın.

Etkilenenler: C

Dekodu:

00	1100	dfff	ffff

Tanımlama: T yazmacının içeriğindeki bir bit sola Carry (taşma) bayrağını üzerinden dönderilir. Eğer 'd' O ise sonuç W yazmacında, l ise sonuç 'f yazmacına yazılır.

Kelimeler: 1

Döngüler: 1

Örnek:

RRF REG1.0

Emirden önce;

REG1 C = 1110 0110

C = o

Emirden sonra;

REG1 = 1110 0110

W = 0111 0011

C = 1

27- SUBWF Komutu: W den f çıkarılır

Syntax: [etiket] SUBWF f,d

Değişkenler: O < k < 255

işlem: (f) - (W) →(hedef)

Etkilenenler: C, DC, Z

Dekodu:

00	0010	dfff	Ffff

Tanımlama: W yazmacından T yazmacı (2'lik ters alma metodu ile) çıkarılır. Eğer d 0 ise sonuç W yazmacında saklanır, l ise T yazmacında saklanır.

Kelimeler: 1

Döngüler: l

Örnekl:

SUBWF REG1.1

Emirden önce;

REG1 = 3

W = 2

C = ?

Emirden sonra;

REG1 = 1

W = 2

C = 1: sonuç pozitif ise

Örnek 2:

Emirden önce;

REG1 = 2

W = 2

C = ?

Emirden sonra;

REG1 = 0

W = 2

C = 1: sonuç sıfır ise

28- SWAPF f takas edilir

Syntax: [etiket] SWAPF f,d

Değişkenler: 0 < f < 127

d e [0,1]

İşlem: (f<3:0>) → (hedef<7:4>)

(f<7:4>)→ (hedef < 3:0 >)

Etkilenenler: Yok

Dekodu:

00	1110	dfff	Ffff

Tanımlama: T yazmacının düşük değerlikli parçası ile yüksek değerlikli parçası yer değiştirilir. Eğer 'd' 0 ise sonuç W yazmacında, 1 ise T yazmacında saklanır.

Kelimeler: 1

Döngüler: 1

Örnek:

SWAPFF REG,0

Emirden önce;

REG1 = 0XA5

Emirden sonra;

REG1 = 0XA5

W = 0x5A

29- XORWF Komutu: W ile f nin XOR lanması

Syntax: [etiket] XORWF kf,d

Değişkenler: O < f < 127

İşlem: (W) . XOR. (f) → (hedef)

Etkilenenler: Z

Dekodu:

00	0110	dfff	Ffff

Tanımlama: W yazmacı ile f yazmacının XOR lanması. Eğer 'd' O ise sonuç W yazmacında, l ise 'f2 yazmacında saklanacaktır.

Kelimeler: l

Döngüler: 1

Örnek:

XORWF REG l

Emirden önce;

REG = 0xAF

W = 0XB5

Emirden sonra;

REG = 0xlA

W = 0xB5

30- BCF F in bitlerini sıfırlar

Syntax: [etiket] BCF f,b

Değişkenler:0 < f < 127

0<b<7

işlem: O → (f<b>)

Etkilenenler: Yok

Dekodu:

01 000bb bfff Ffff

Tanımlama:f yazmacındaki 'b' bitlerini sıfırlar, temizler.

Kelimeler: l

Döngüler: l

Örnek:

BCF FLAG_REG, 7

Emirden önce;

FLAG_REG = 0xC7

Emirden sonra;

FLAG_REG = 0x47

31-SLEEP komutu: Bekleme Konumuna Giriş

Sentaks: [yafta] SLEEP

Çalıştırıcılar: Yok

işlem: OOh -» WDT,

0->WDTönölçer,

1->TO

0->PD

Etkilenen Statü: TO, PD

Kod Çözümü: 00 0000 0110 0011

Tanım:Güç kapatma statüsü biti (PD) temizlenmiştir. Mola statüsü biti (TO) ayarlanmıştır. Koruyucu Zamanlayıcı ve onun ön ölçeri temizlenmiştir, işlemci, sarkacı durdurarak SLEEP moduna sokulmuştur.

Kelime: l

Döngü: l

Örnek

SLEEP

32- BSF Komutu: F in bitlerine bir yükler, setler.

Syntax: [etiket] BSF f,b

Değişkenler: 0 < f < 127

0<b<7

işlem: l → (f<b>)

Etkilenenler: Yok

Dekodu:

01	01bb	bfff	Ffff

Tanımlama: T yazmacındaki 'b' bitlerini setler, 'l' ler.

Kelimeler: l

Döngüler: l

Örnek:

BSF FLAG_REG, 7

Emirden önce;

FLAG_REG = 0x0A

Emirden sonra;

FLAG REG = 0x8A

33- BTFSC Komutu:f biti test edilir eğer 0 ise atlanır

Syntax: [etiket] BTFSC f,b

Değişkenler: 0<f<127

0<b<7

işlem: atla eğer (f<b>) = 0

Etkilenenler: Yok

Dekodu

01	10bb	bfff	Ffff

Tanımlama: Eğer 'f yazmacmdaki 'b' i O ise bir sonraki emire atlama yapılır. Eğer 'b' O ise, derleyici geçerli emri atlar ve onun yerine bir NOP (no operation) çalıştırır, bu emir 2 döngülüktür.

Kelimeler: l

Döngüler: l (2)

Örnek:

```
HERE  BTFSC FLAG,0
FALSE GOTO        PROCESS_CODE
```

TRUE

Emirden önce;

PC = address HERE

Emirden sonra;

Eğer FLAG < l > = 0,

PC = address TRUE

EğerFLAG<l>=l,

PC= address FALSE

34-BTFSSF komutu biti test edilir, Eğer l ise atlanır

Syntax: [etiket] BTFSS f,b

Değişkenler: 0 < f < 127

0<b<7

İşlem: atla eğer (f<b>) = l

Etkilenenler: Yok

Dekodu

01	11bb	bfff	Ffff

Tanımlama: Eğer f yazmacındaki 'b' i l ise bir sonraki emire atlama yapılır. Eğer 'b' l ise, derleyici geçerli emri atlar ve onun yerine bir NOP (no operation) çalıştırır, bu emir 2 döngülüktür.

Kelimeler: l

Döngüler: l (2)

Örnek:

HERE BTFSC FLAG, l

FALSE GOTO PROCESS_CODE

TRUE

Emirden önce;

PC = address HERE

Emirden sonra;

EğerFLAG<l> = 0,

PC = address TRUE

EğerFLAG<l> = 1

PC= address FALSE

35- ADDIAV komutu: Bir değer ile W nin toplanması

Syntax: [etiket] ADDLW k

Değişkenler:0 <k < 255

işlem (W)+k→W

Etkilenenler: C, DC, Z

Dekodu:

11 111x kkkk Kkkk

Tanımlama: W yazmacının içeriği 'k' harfi ile gösterilen 8 bitlik değerle toplanır ve sonuç geri W yazmacına yazılır.

Kelimeler: l

Döngüler: l

Örnek:

ADDLW 0x15

Emirden önce;

W = 0x10

Emirden sonra;

W = 0x25

36-ANDLW W ile f nin AND lenmesi

Syntax: [etiket] ANDWF f,d

Değişkenler: 0 < k < 127

işlem: (W) .AND. f→ (hedef)

Etkilenenler: Z

Tanımlama:W yazmacının içeriği ile T yazmacı ile AND lenir. Eğer 'd' değeri O ise sonuç W yazmacına yazılır, l ise sonuç geri 'f yazmacına yazılır.

Kelimeler: 1

Döngüler: 1

Örnek:

ADDWF FSR,0

Emirden önce;

W = 0x17

FSR = 0xC2

Emirden sonra;

W = 0x17

FSR = 0x02

37- CALL Alt programı çağırmak

Syntax: [etiket] CALL k

Değişkenler: O < f < 2047

İşlem: (PC) + l →TOS

k→ (PC<10:0>)

(PCLATH<4:3>)

Etkilenenler: yok

Tanımlama: Altprogram çağrılır, îlk önce, geri döndürülen (PC+1) adresi yığına atılır. Sonra o an geçerli olan 11 bitlik adres PC <10:0> bitlerine yerleştirilir. PC'nin üstteki bitleri PCLATH dan yüklenir. CALL, 2 döngülük bir emirdir.

Kelimeler: l

Döngüler: 2

Örnek:

HERE CALL THERE

Emirden önce;

PC = address THERE

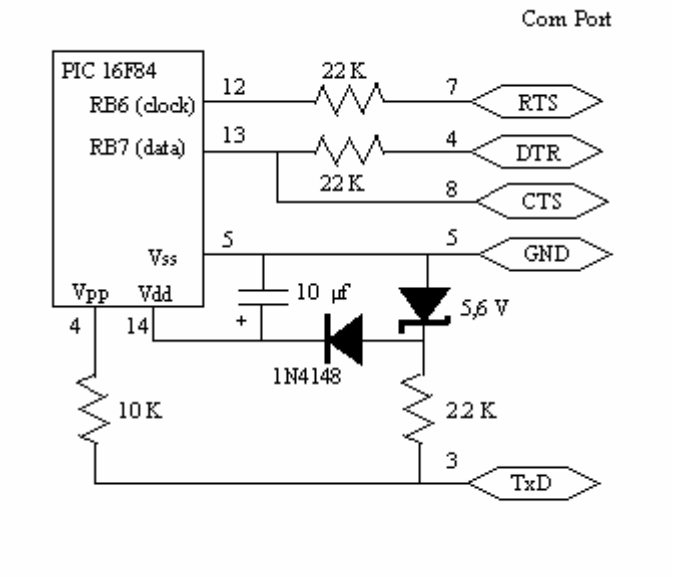
Emirden sonra;

PC= address HERE

4.4. PIC Programlayıcı

Bir adet DB9 dişi Konnektör ve üç adet dirençten oluşmaktadır. Bu devre çok basit olup PC ‘nin COM portunu kullanır. Devrenin gerçekleştirilmesi aynı zamanda minimum bir PIC Sistemi yapmamızı gerektirdiğinden iki işi bir defada halletmiş

olacağız. Şekildeki devreyi kurarak hem bir programlayıcı hem de bir PIC tabanlı Bilgisayar Sistemimiz olacak. Bir PIC İşlemcisinin çalışması için bir adet kristal, 2 adet direnç, 3 adet kondansatör ve 14 numaralı bacağa +5Volt , 5 numaralı bacağa toprak vermek yeterlidir. Geri kalan bacaklar giriş ve çıkış portlarıdır. Devrede kullanılan zener diyot Vdd 5 volt gerilimini sabit tutmaktadır.



Şekil 4.1. PIC 16F84 Programlayıcısının devre şeması

BÖLÜM 5. FREKANSMETRE

5.1.Giriş

Piyasada çeşitli yöntemler kullanarak yapılmış değişik tiplerde frekansmetreler satılmaktadır. Analog ibreli, titreşimli, sayısal, programlarla yapılmış olanlar bunlardan bazısıdır. Bu çalışmada yukarda belirtilen yöntemler dışında PİC16f84 mikro denetleyicisi kullanarak bir frekansmetre oluşturacağız.

5.2.Tanımlar

Frekansmetre ile ilgili bütün konularda frekansın tanımını esas almak gerekmektedir. Aynı şekilde sayısal (dijital) frekansmetre yapmak için de frekansın tanımını devre olarak aynen uygulamak gerekmektedir.

FREKANS: 1 saniyedeki periyot sayısına frekans denir.
PERYOT: Tam devir yapmış dalga bir periyotluktur (devirliktir). Devrini tamamlayan dalgaya periyot denir.
PULSE: Yarım periyota pals denir. 1 periyotta birisi pozitif, diğeri negatif olmak üzere iki adet pals vardır.
Frekansın birimi: HERTZ veya SAYKIL olarak belirtilir.

Yukarıdaki temel tanımlar bir frekansmetre yapmak için yeterlidir. Şimdi frekansmetreyi açıklayabiliriz

FREKANSMETRE: 1 saniyedeki periyot sayısını ölçen ölçü aletlerine frekansmetre denir.

5.3. Frekans Metrelerin Çalışma Prensibi

Frekans metre yapmak için 2 saniyelik, eşit palslı ve kare dalga bir periyot kullanılması yeterlidir. Bu periyotun pozitif ve negatif palsları birbirine eşit olmalıdır. Bu 2 saniyelik periyotun 1'er saniyelik iki adet palsından istediğimizi kullanarak bir frekans metre tasarımı yapabiliriz. Ayrıca yardımcı palslara da ihtiyacımız vardır. Bu yardımcı palslar, göstergeleri sayma esnasında tutma ve sayma işlemi bittiğinde sayacıları resetleme (sıfırlamak) olmak üzere iki adet olması gerekmektedir. Biz burada temel ölçüm devresini esas alıyoruz. Değişik aksesuar ve görüntü düzenleri için daha fazla palslar kullanılabilir. 2 saniyelik periyotta kullanacağımız pals değerini, öğrenmek istediğimiz sayıcılara girecek olan frekans için anahtar görevi gören devreyi çalıştıracaktır.

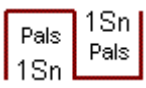
5.3.1. Periyotların Sayılması

Salınımın olduğu bir teli bir anahtar kullanarak 1 saniye için açıp kapatırsak ve telden bu 1 saniyede geçen salınım miktarını sayarsak o telden o anda geçen periyot miktarını öğrenmiş oluruz. Böylece frekansı öğrenmiş oluruz.

İlk Aşama: Referans Frekansı ve Özellikleri:

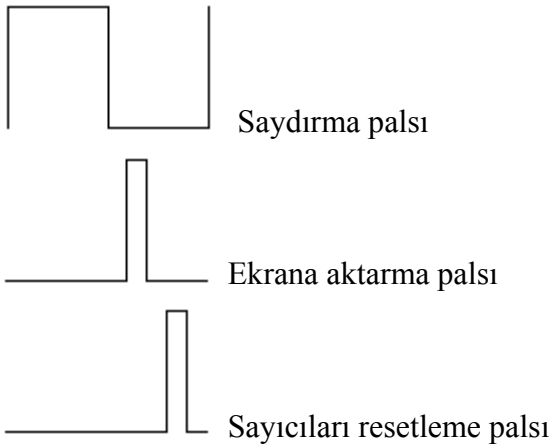
Öncelikle referans frekansını elde etmek gerekmektedir. Bu, 2 saniyelik, eşit palslı ve kare dalga bir periyottur. Frekansı 1/2 hertzdir.

Bu referans frekansını ne şekilde elde ettiğimiz önemli değildir. Temel şart, kararlı ve kullanacağımız palsın 1 saniyesinin kesin olması gerekmektedir. Bu temel şart yerine getirilemezse ölçüm hatalı ve karasız olur. Bu istenen bir durum değildir. Bu frekansta kararlılığı sağlamak için kristal kontrollü osilatör devresi hazırlamak gerekmektedir. Bu tür devreler, TTL veya CMOS entegrelerle yapılabilir. Bizim vereceğimiz örnek CMOS, 4000 serisi entegrelerle olacaktır.



Şekil 5.1. Referans frekansı.

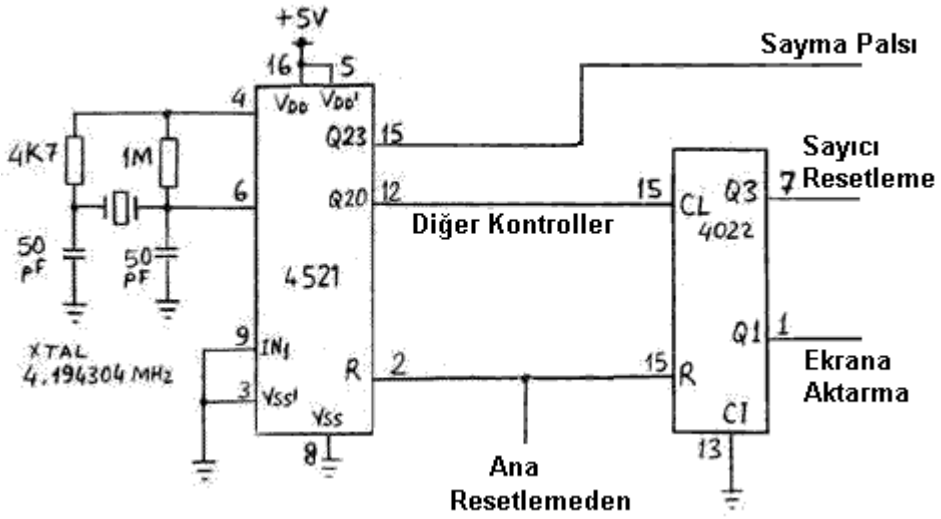
Referans frekansı ile saydırma palsı, ekrana aktarma palsı ve sayıcıları resetleme palsı elde edilmesi gerekmektedir.



Şekil5.2: Referans frekansı ile elde edilmesi gereken palslar.

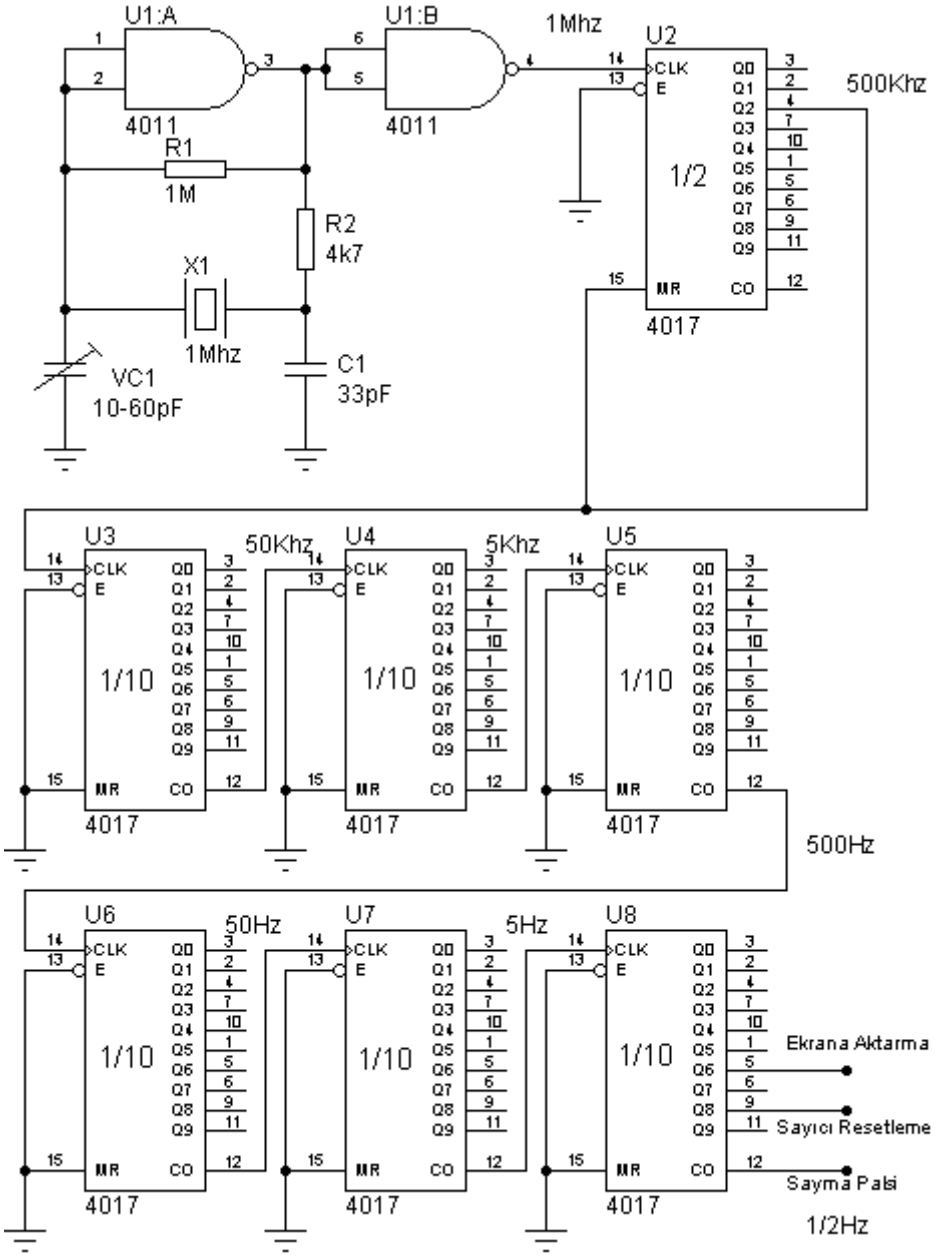
5.4. Örnek Frekansmetre Uygulamaları

Örnek 1: 4521 entegresi ve 4.194304 Mhz'lik kristal kullanılarak referans frekansı devresi yapılır. 4521 entegresi kristal osilatör devresi kullandığı gibi RC osilatör devresi de kullanır. Osilatörden elde edilen frekansı 18'den 24'e kadar bölerek çıkışa verir. Biz Q23 çıkışını (15 nolu uç) kullanarak kristalin frekans değerini 23'e bölerek 1/2 Hz'lik referans frekansını elde ediyoruz. 4521 entegresinin Q20 çıkışını (12 nolu uç, 4 Hz) diğer kontrolleri üretmek için 4022 entegresinin 15 nolu Clock (saat) giriş ucuna bağlıyoruz. 4022 bölücü/sayıcı entegresinin 7 nolu bacağındaki Q3 çıkışından sayıcıları resetleme için ve 1 nolu bacağındaki Q1 çıkışından sayıcıların saydığı değeri ekrana aktarma uçlarını alıyoruz. 4521'in 2 nolu ve 4022'nin 15 nolu reset uçlarını ise frekansmetrenin genel reset (sıfırlama) ucuna bağladığımız gibi şaseye de bağlayabiliriz. Bu devrede, 4022'nin sayıcı resetleme ve ekrana aktarma uçlarını kullanacağımız sayıcı entegresine ve sayıcıların dizaynına göre seçilmelidir.



Şekil 5.3: 4521 entegresi ve 4.194304 Mhz'lik kristal kullanılan referans frekansı devresi

Örnek 2: 4011 entegresi kapılarıyla 1Mhz kristal kullanarak kristal kontrollü osilatör yapılır ve elde edilen 1 Mhz'lik frekans önce 2'ye bölünerek 500KHz elde edilir, sonra 500 KHz 10'a bölünerek 50 Khz elde edilir. 50 Khz 10'a bölünerek 5 Khz elde edilir. 5 Khz 10'a bölünerek 500 Hz elde edilir. 500 Hz 10'a bölünerek 50 Hz elde edilir. 50 Hz 10'a bölünerek 5 Hz elde edilir. Elde edilen 5 Hz'lik frekans son olarak bir defa daha 10'a bölünerek istediğimiz palsları elde ederiz. Bütün bölme işlemleri için 4017 entegresi kullanırız. Aşağıdaki şekilde uygulama örneği görülmektedir. Bu devrede, 4022'nin sayıcı resetleme ve ekrana aktarma uçlarını kullanacağımız sayıcı entegresine ve sayıcıların dizaynına göre seçilmelidir.

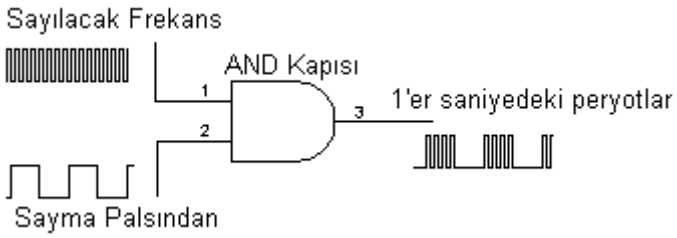


Şekil 5.4: 4011 ve 4017 ile gerçekleştirilen referans frekası devresi

İkinci Aşama: Periyotları kontrol altına alarak geçirmek

Örnek 1: Bunun için her bir palsı 1 saniye olan referans frekansı elde edildikten sonra, bu referans frekansı ile periyotların geçişi kontrollü olarak yapılır. Böylece frekans sayılmaya hazır hale getirilir. Yani 1 saniye geçirilen frekans 1 saniye bekletilir. Frekansın geçişi sırasında sayma işlemi yapılır, bekleme sırasında

göstergelere aktarılır ve sayıcılar sıfırlanarak tekrar sayıma hazır hale getirilir. Düzenek böyle çalışır. Bu düzeneğin çalışması için referans frekansına ihtiyaç vardır. Bu referans frekansı ile ölçeceğimiz frekansı kıyaslayarak sonuca varırız. Bu türlü bir işlem için bir AND kapısı kullanılabilir.



Şekil 5.5: Referans frekansı ile sayılacak frekansın elde edilmesi.

Şekil 5.5'te kullanılan AND kapısı yerine 4053 entegresi de kullanılabilir.

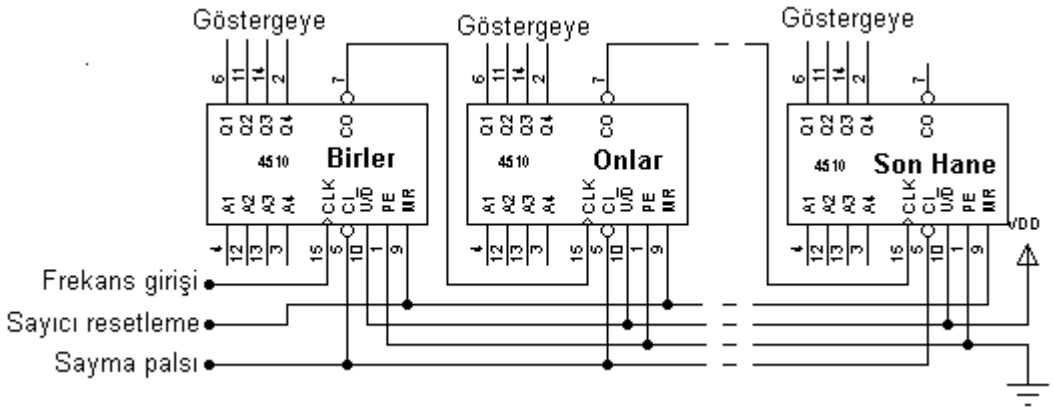
Örnek 2: Örnek 1'deki işlemi sayıcıların üzerinde gerçekleştirilir. Sayma palsı ile 4510 sayıcı entegresi kullanıyorsanız, Clock Inhibit girişini kontrol edilir. 4518 entegresi kullanıyorsanız, Clock girişi kontrol edilir. Diğer sayıcı entegrelerinin kullanımı durumunda yine benzer işlemler yapılır.

Üçüncü Aşama: Frekansı Saymak

Frekansı saymak için herhangi bir sayıcı entegresi kullanmak yeterli. Burada amaca uygun sayıcı entegresi tercih etmekte fayda vardır. Sayıcı entegreyi sayma yetkilendirmesi için referans frekansından gelen sayma palsı ve sayma işlemini gerçekleştirmiş olan sayıcının sonraki sayma işlemi için sıfırlanmasını sağlayacak reset uçları kullanılır. Sayılan değer göstergelere aktarılmak üzere gösterge sürücü entegrelere aktarılır. Sayılan frekansın ilk girdiği entegre birler hanesini oluşturur. Diğerleri sırasıyla onlar hanesi, yüzler hanesi şeklinde gider.

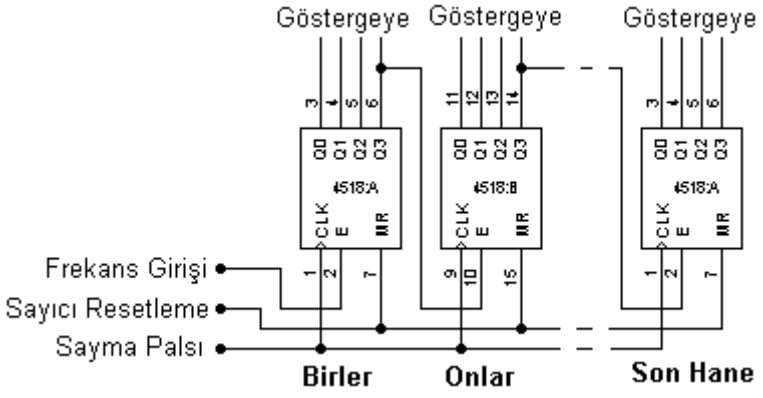
Örnek 1: 4510 entegresi kullanarak frekansı saydırmak. Bu devrede sayılacak frekans 4510'un CLOCK girişine bağlanıyor. RESET uçları birleştirilerek referans devresinden gelen resetleme ucuna bağlanıyor. Sayma palsı ucu ise, CLOCK

INHIBIT ucuna bağlanıyor. İlk entegrenin CARRY OUT çıkışı sonraki entegrenin CLOCK ucuna bağlanıyor. Q1, Q2, Q3, Q4 çıkışları ise gösterge sürücü entegrelere gidiyor.



Şekil 5.6: 4510 entegresi ile yapılan temel frekans sayma devresi.

Örnek 2: 4518 entegresi kullanarak frekansı saydırmak. Bu devrede sayılacak frekans 4518'un ENABLE girişine bağlanıyor. RESET uçları birleştirilerek referans devresinden gelen resetleme ucuna bağlanıyor. Sayma palsı ucu ise, CLOCK ucuna bağlanıyor. İlk entegrenin Q3 çıkışı sonraki entegrenin ENABLE ucuna bağlanıyor. Q1, Q2, Q3, Q4 çıkışları ise gösterge sürücü entegrelere gidiyor.

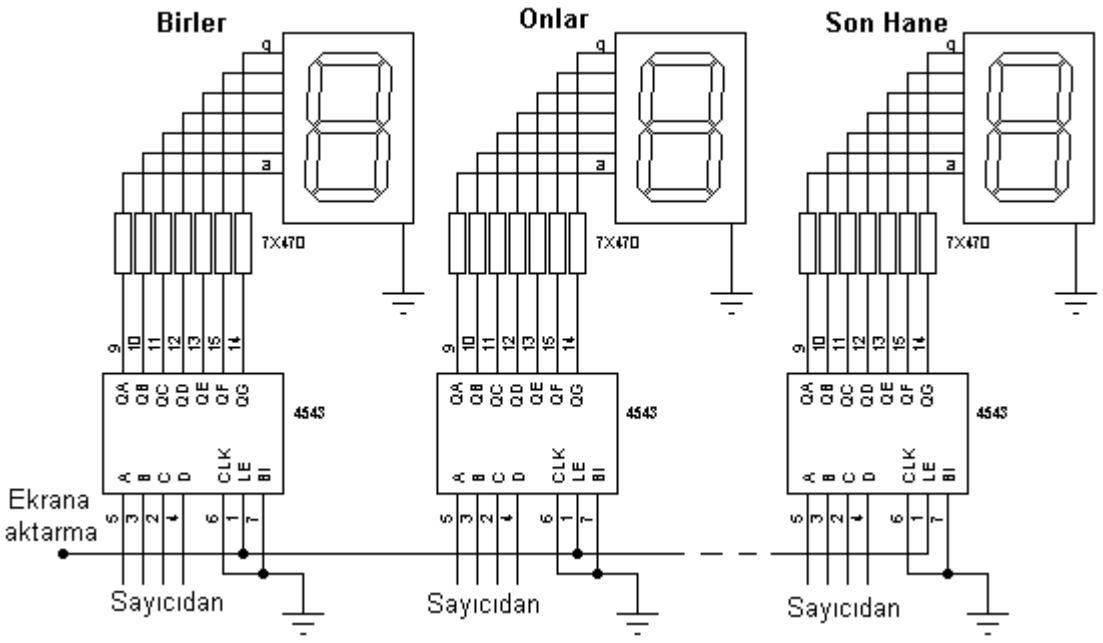


Şekil 5.7: 4518 entegresi ile yapılan temel frekans sayma devresi.

Dördüncü Aşama: Sayılan Frekansı Göstermek

Gösterge için seçtiğimiz tip göstergeye sürücü olan devremizin dizaynını etkiler. Gösterge olarak LED display kullanılabilir. Bu durunda 4543 entegresi bence en ideal entegredir. Hem KATOD display, hem ANOD display ve hem de likit kristal displayi sürer. 4511 entegresi veya diğer entegreler bu bölüm için fazlasıyla yeterlidir.

4543'ün 6 ve 7 nolu uçlarını şaseye bağlıyoruz. LATCH DİSABLE (1 nolu uç) ucunu ise referans frekansından gelen ekrana aktarma ucuna bağlıyoruz. QA....QG uçlarını displeye bağlıyoruz.



Şekil 5.8:4543 entegresi ile led displey sürme devresi.

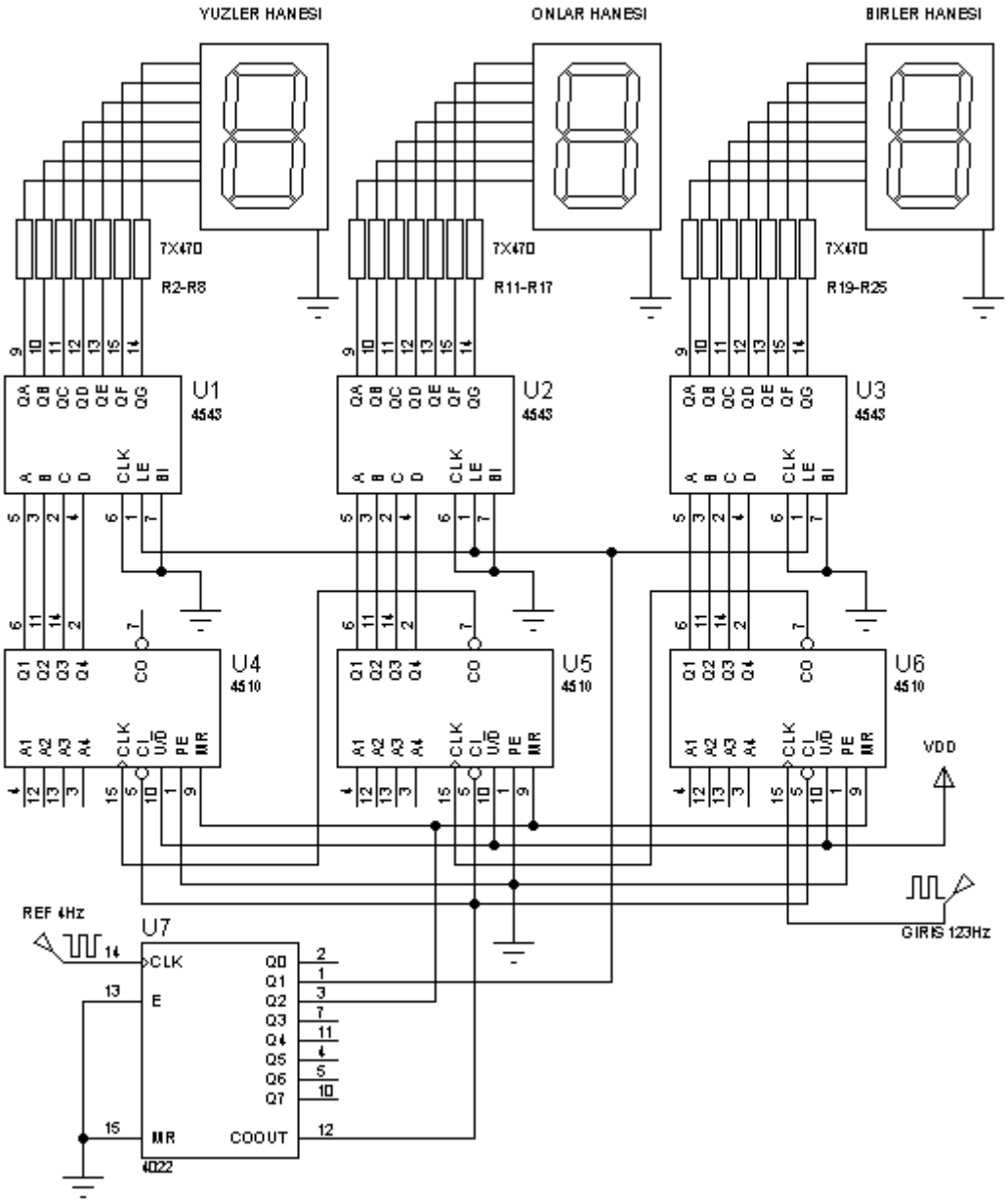
Beşinci Aşama: Sayacağımız frekansı devremiz ile uyumlu hale getirmek için ön yükselteç (Preamplifikatör)

Frekansmetrelerde sayılacak olan giriş sinyali, öncelikle çek düşük gerilim değerine çekilir. Bu şekil 7'deki devrede iki adet 1N4148 diyotları ile yapılır. Sonraki aşamada sinyale FET (PN4303, BF245) vasıtasıyla çok yüksek kazanç uygulanır ve kare dalga

şekil elde edilir. Son aşamada ise, taransistörlü (BC337, BC547) kazanç devresiyle sürülerek kullanılır. Şekil 5.7'de en basit preamplifikatör devresi görülmektedir. Bu devre düşük frekansta yeterli gelmektedir. Girişine şebeke voltajı uygulanabilmektedir. Şebeke gibi gürültülü veya parazit yüklü bir noktanın frekansını ölçmek istediğinizde istenmeyen parazitler frekansmetrenin preamplifikatörünün hassasiyetine bağlı olarak yanlış sonuçlar verebilir. Mesela 50Hz'lik şebeke frekansı 2 kat ya da 4 kat fazla gösterebilir. Bu durumu dikkate alarak Preamplifikatör hazırlamakta fayda var. Bizim devremizde, bu sorunu çözmek için BC337 transistorunun Beysi ile şase arasına 100nF değerinde kondansatör ilave etmek yeterli olmaktadır. Böylece parazitleri söndürmüş oluyoruz. Amacınıza uygun olabilecek değeri seçerek bu gibi sorunları çözebilirsiniz.

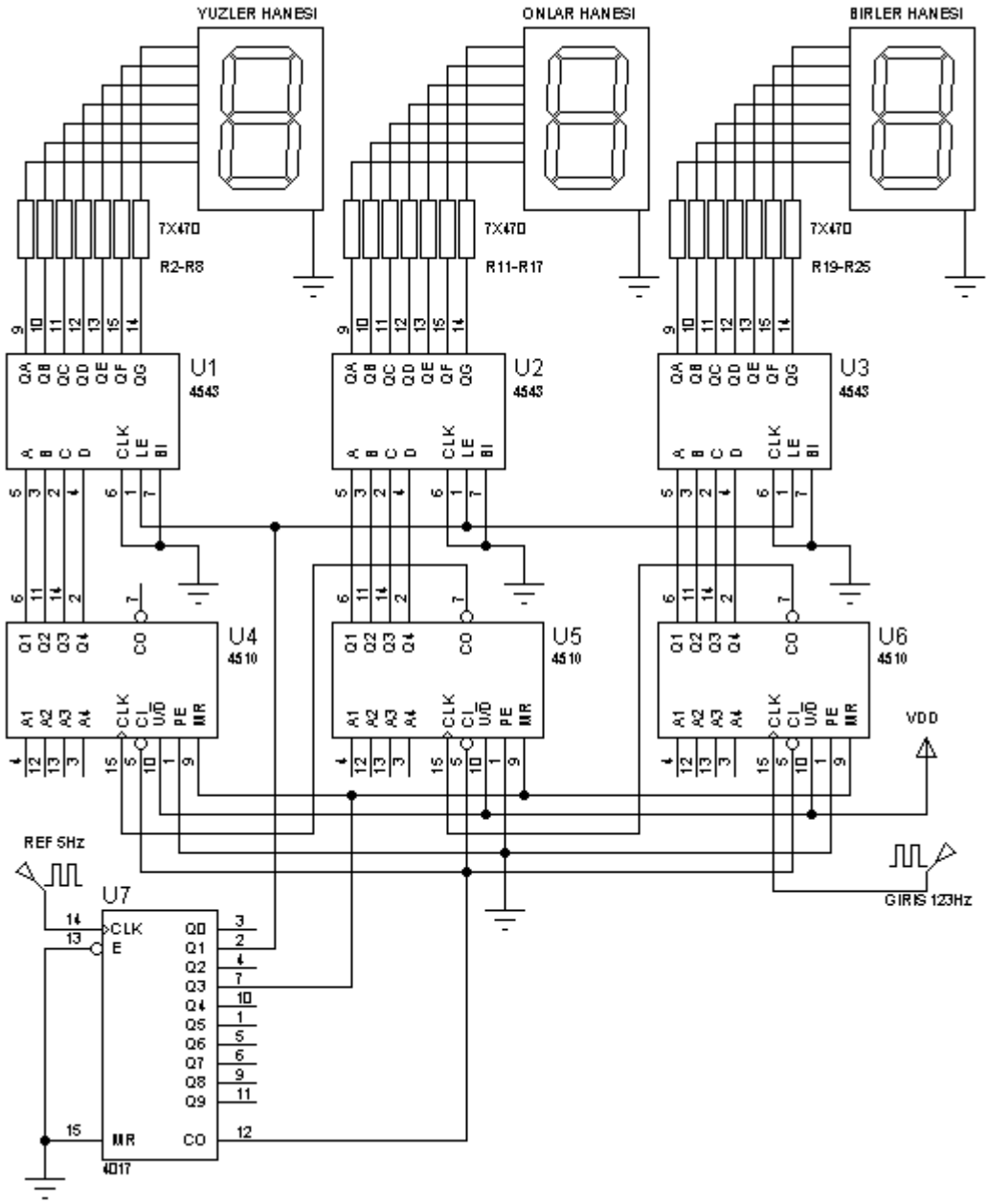
5.4.1.Frekansmetre Dizaynı ve Devre Örnekleri

Devre 1: Bu devre şekil 5.3'teki 4521 entegresi ile yapılan referans osilatörü ile çalışmaya uygun olarak tasarlanmıştır. 4521'in Q20 (12 nolu ucu) çıkışı 4022'nin CLOCK (14 nolu uç) ucuna bağlanacak. Bu uçtaki frekans 4 Hz'dir



Şekil 5.9: 4510, 5022 ve 4543 entegreleri ile düzenlenmiş frekansmetre prensip şeması.

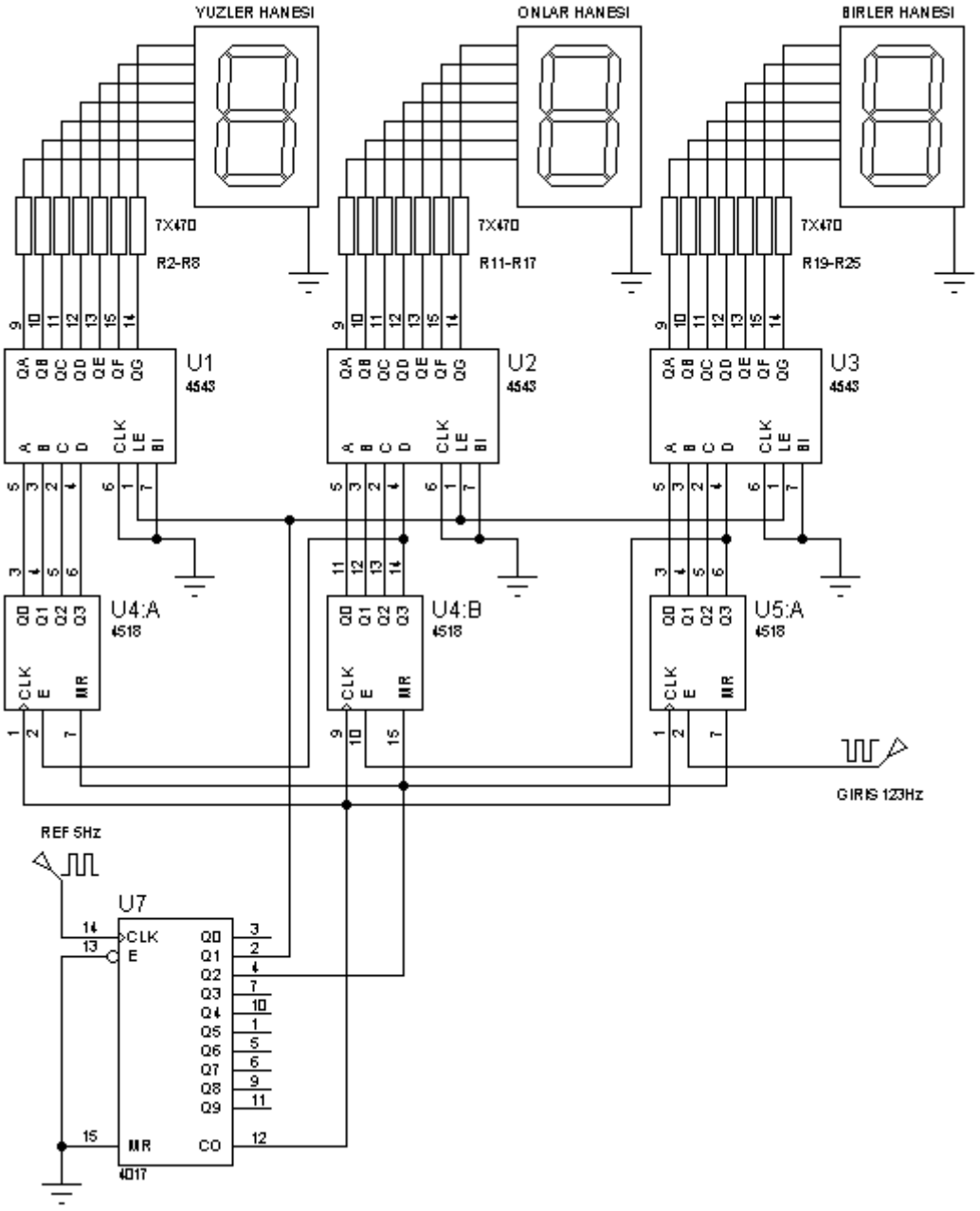
Devre 2: Bu devre şekil 4'deki 4017 entegresi ile yapılan referans osilatörü ile çalışmaya uygun olarak tasarlanmıştır. 4017'nin (U7) CARRY OUT (12 nolu ucu) çıkışı 4017'nin CLOCK (14 nolu uç) ucuna bağlanacak. Bu uçtaki frekans 5 Hz'dir



Şekil 5.10: 4510 4017 ve 4543 entegreleri ile düzenlenmiş frekansmetre prensip şeması.

Devre 3: Bu devrede sayıcı entegre olarak 4518 entegresi kullanılmıştır. 4518 sayıcı entegresi içerisinde iki adet BCD sayıcı olduğundan kullanılacak entegre sayısı azalmaktadır. Çalışması açısından 4510 sayıcı entegresi kullanılan devreden bir farkı bulunmamaktadır. Bu devre şekil 4'deki 4017 entegresi ile yapılan referans osilatörü ile çalışmaya uygun olarak tasarlanmıştır. 4017'nin (U7) CARRY OUT (12 nolu ucu) çıkışı 4017'nin CLOCK (14 nolu uç) ucuna bağlanacak. Bu uçtaki frekans 5 Hz'dir.

Bu devrede kullanılan 4017 entegresi (U7) yerine 4022 entegresi kullanarak çalıştırılabilir. Bu değişiklik için referans frekansı osilatörü olarak şekil 3'teki devre kullanılır. Bu uygulama için, 4521'in Q20 (12 nolu ucu) çıkışı 4022'nin CLOCK (14 nolu uç) ucuna bağlantısı yapılacak. Bu uçtaki frekans 4 Hz'dir.

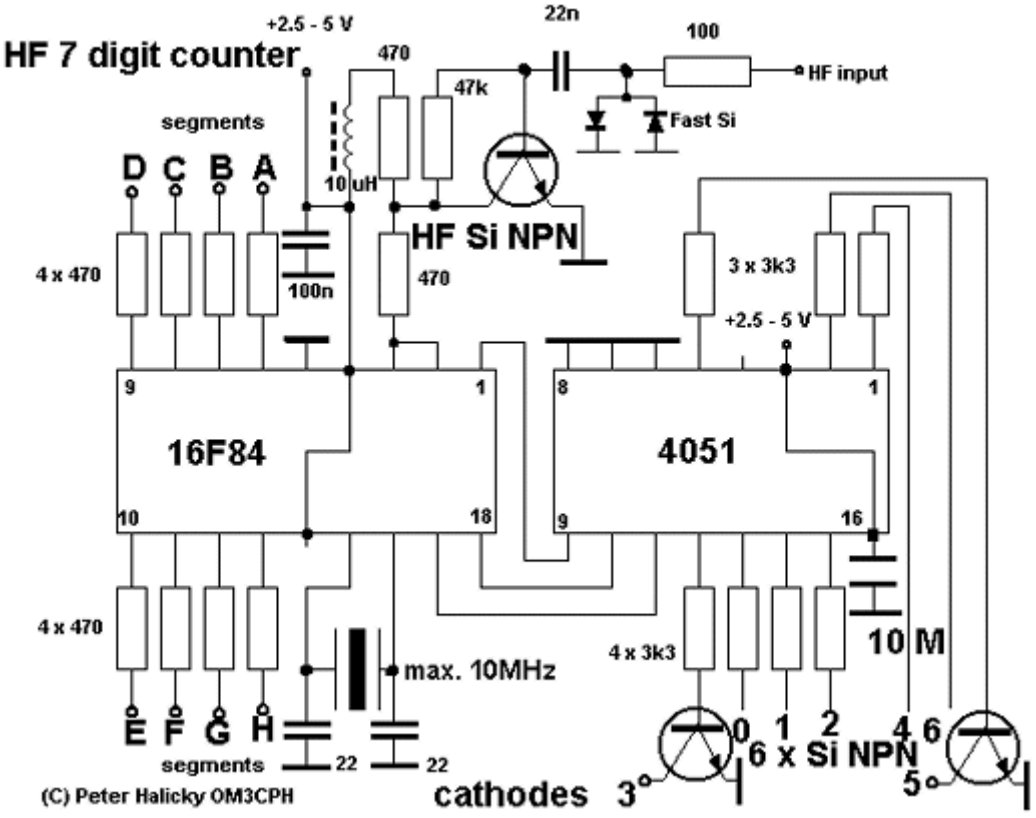


Şekil 5.11: 4510 ve 4543 entegreleri ile düzenlenmiş frekansmetre prensip şeması.

5.5. PIC 16f84 ile Frekansmetre Uygulaması

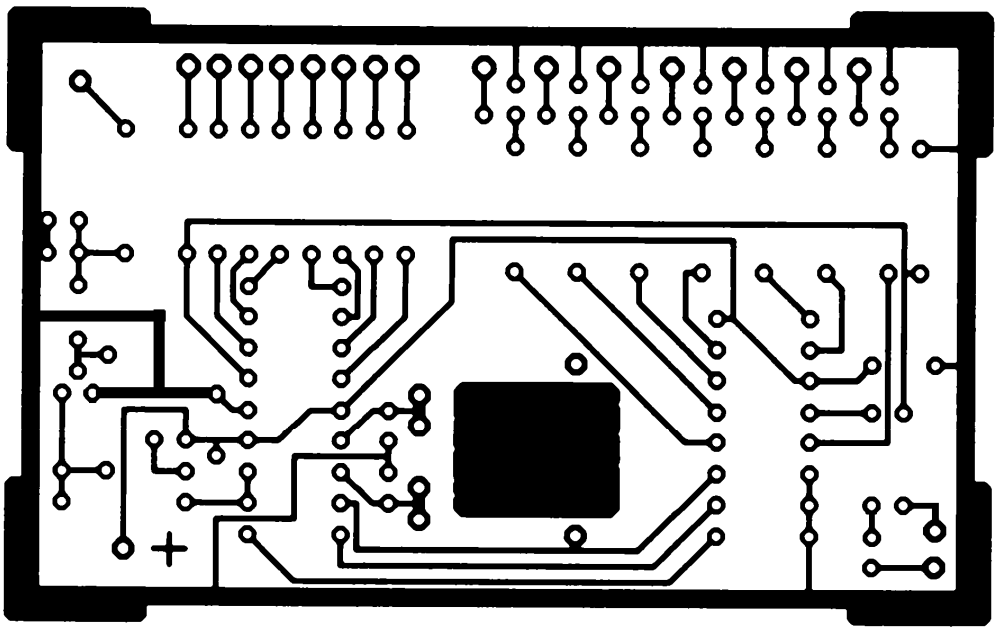
Frekans sayıcı şemadan da görüleceği gibi temel olarak 16f84 ve 4051 den oluşmaktadır. Bunlar dışında transistorler ve birkaç pasif eleman bulunmaktadır.

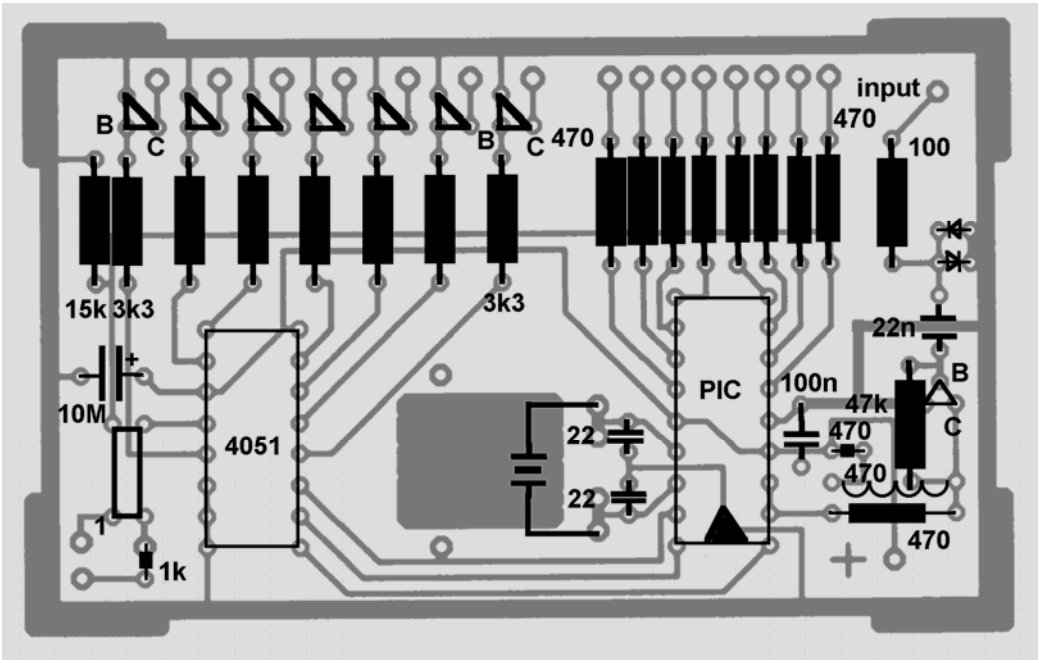
5.5.1. Devre Şeması



Frekans sayıcı HF bir amatör cihazın frekansını görmek için tasarlanmıştır, ölçüm aralığı10hz – 35 Mhz dir. Bu aralık girişine bir prescaler (örn: U664B) konularak 1-2 Ghz'e kadar kolayca çıkarılabilir.

5.5.2 Baskı Devre Şeması





Devrede bulunan PIC 16F84 hem ölçüm hem de göstergelerin sürülmesi işinden sorumludur. 4051 ise pic bacak sayısı yeterli olmadığından çoğullayıcı olarak her display elemanını pic komutlarına göre adresler. Belki dikkatinizi çekmiştir burada kullanılan kristal 10Mhz'dir tabii Pic de 10Mhzlik **PIC16F84 10 I/P** tipi olmalıdır

gibi bir kanı oluşabilir aslında doğrusuda budur ama hiç gerek yok ufak bir kandırmacayla 4 Mhzlik piyasada bol bulunan PIC16F84 – 04 I/P ile bu işi halledebiliriz.

Yapmamız gereken tek şey programlama öncesi programlayıcı yazılımında OSC seçeneğinde XT yerine HS yani Hızlı seçeneğini seçmektir. Bunu yaparsak sorunsuz bir 10 Mhz'lik pic elde ederiz.

Pic bacaklarındaki A, B, C, D, E, F, G harfleri display sisteminde segmentlere gidecek ve 0, 1, 2, 3, 4, 5, 6 sayıları ise ilgili displayin ortak katoduna gidecektir.

Sistem multiplex sürme sistemini kullanmaktadır dolayısıyla 7 gösterge elemanının aynı olan segmentleri birbirlerine bağlanmalı katotları ise ayrı olarak rakam grubuna sırayla soldan sağa gitmelidir.

Besleme voltajı olarak 2.5 – 5.0 volt denmektedir ama en sağlıklı sonuç +5V ile alınır.

Nokta sabit olarak Mhz displayinden sonraki ilk nokta olarak yakılmalıdır. Bu bir direnç (270 ohm) ile ilgili bacak +5V asılarak sağlanır…

5.5.3.Frekansmetrenin Assembler Listesi

```
       ; ------------------------------------------------------------
-------
       IndF        equ        00h         ; Indirect addr. register
Timer0      equ        01h         ; TMR0 register   - RP0=0
OptionR     equ        01h         ; Option register - RP0=1
PC          equ        02h         ; Program counter
Status      equ        03h         ; Status register
FSR         equ        04h         ; Pointer register
PortA       equ        05h         ; Port A - RP0=0
TrisA       equ        05h         ; Tris A - RP0=1
PortB       equ        06h         ; Port B - RP0=0
TrisB       equ        06h         ; Tris B - RP0=1
;PortC      equ        07h         ; Port C - 16C54+84 not
```

```
EEData      equ         08h         ; 16C84
EEAdr       equ         09h         ; 16C84
PCLath      equ         0Ah         ; 16C84
IntCon      equ         0Bh         ; 16C84
Index       equ         0Ch         ; dummy register
Count       equ         0Dh         ; inkremental register
Help        equ         0Eh         ; dummy register
LED0        equ         0Fh
LED1        equ        010h
LED2        equ        011h
LED3        equ        012h
LED4        equ        013h
LED5        equ        014h
LED6        equ        015h
LED7        equ        016h
TimerH      equ        017h         ; higher byte of SW counter
LowB        equ        018h         ; low byte of resulted frequency
MidB        equ        019h         ; middle byte of resulted
frequency
HigB        equ        01Ah         ; high byte of resulted frequency
       Temp        equ       01Bh         ; temporary register
HIndex      equ        01Ch         ; index register
LEDIndex    equ        01Dh         ; LED pointer
; ------------------------------------------------------------------
------
                                    ; timing loop values
                                    ; must be from 1 to 255!!!
T1          equ        .70          ; rough timing loop
T2          equ        .3           ; timing loop
T3          equ        .20          ; fine timing loop
                                    ; values for 4 194 kHz
       ; ------------------------------------------------------------
-------------
;
; Measuring period is 100 000 us.
; Procesor cycle is T = 4/fx us   [MHz], fx is Xtal frequency
; Number of procesor cycles per measuring period:
;        N = 100 000/T procesor cycles
;        N = fx x 100 000/4 = 25 000 x fx
```

```
; The main steps of measuring period:
;        1. start measurement,
;        2. precode decimal value of digit to segments,
;        3. if it's 5th digit set decimal point,
;        4. output to PortB,
;        5. output digit number to PortA
;           (numbers from left to right are 6543210),
;        6. test TMR0 overflow bite, if YES increase TimerH,
;        7. leave digit to light,
;        8. increase digit number,
;        9. if <7 goto 2,
;       10. else zero digit number,
;           decrease counter and goto 2,
;       11. stop measurement,
;       12. shift out precounter content,
;       13. precode 3-byte value into 7 decimal numbers,
;       14. goto 1
;
; ------------------------------------------------------------------
------
; Total timing formula:
; N = 25 000 x fx = 60x[(36 + 3xT1 + X)x7 + 2 + 3xT2 + Y] + 19 +
3xT3 + Z
; where T1,T2,T3 are initial values of timing loops,
;       X, Y, Z  are additional tunig NOPs.
;
; ------------------------------------------------------------------
------
W           equ       0           ; destination is accumulator
F           equ       1           ; register
; ------------------------------------------------------------------
------
                                        ; Flag bits:
CF          equ       0           ; Carry
DC          equ       1           ; DC
ZF          equ       2           ; Zero
RP0         equ       5
RP1         equ       6
IRP         equ       7
```

```
       ; ------------------------------------------------------------
-------------
            org        0
       Start       clrf       Index
            clrf       LEDIndex
            clrf       LED0
            clrf       LED1
            clrf       LED2
            clrf       LED3
            clrf       LED4
            clrf       LED5
            clrf       LED6
            clrf       LED7
            clrf       LowB
            clrf       MidB
            clrf       HigB
            bsf        Status,RP0
            movlw      b'00010000' ; RA0..RA3 outputs
            movwf      TrisA       ; RA4 input
            movlw      b'00000000' ; RB0..RB7 outputs
            movwf      TrisB
            clrwdt                 ;
            movlw      b'00100111' ; Prescaler -> Timer0,
            movwf      OptionR     ; 1:256, rising edge
            bcf        Status,RP0  ;
            goto       Go
       ;------------------------------------------------------------
------------
; 3 byte substraction of the constant from the table which sets
carry if
; result is negative
;------------------------------------------------------------------
-----
Subc24      clrf       Temp        ; it will temporary save CF
            movf       Index,W     ; pointer to low byte of constant
            movwf      HIndex      ; W -> HIndex
            call       DecTable    ; W returned with low byte of
constant
            bsf        Status,CF   ; set CF
```

```
            subwf       LowB,F      ; LowB - W -> LowB
                                    ; if underflow -> CF=0
            btfsc       Status,CF
            goto        Step1
            bsf         Status,CF
            movlw       1
            subwf       MidB,F      ; decrement MidB
                                    ; if underflow -> CF=0
            btfsc       Status,CF
            goto        Step1
                   bsf         Status,CF
            movlw       1
            subwf       HigB,F      ; decrement HigB
            btfsc       Status,CF   ; if underflow -> CF=0
            goto        Step1
            bsf         Temp,CF     ; set CF
        Step1       decf        HIndex,F
            movf        HIndex,W    ; pointer to middle byte of const
            call        DecTable
            bsf         Status,CF
            subwf       MidB,F      ; MidB - W -> MidB
            btfsc       Status,CF   ; if underflow -> CF=0
            goto        Step2
            bsf         Status,CF
            movlw       1
            subwf       HigB,1      ; decrement HigB
            btfsc       Status,CF   ; if underflow -> CF=0
            goto        Step2
            bsf         Temp,CF     ; set CF
        Step2       decf        HIndex,F
            movf        HIndex,W    ; pointer to middle byte of
constatnt
            call        DecTable
            bsf         Status,CF
            subwf       HigB,F      ; HigB - W -> HigB
            btfsc       Status,CF   ; if underflow -> CF=0
            goto        ClearCF
            bsf         Status,CF
            goto        SubEnd
```

```
ClearCF     rrf        Temp,CF     ; CF -> Status
SubEnd      retlw      0
       ; ------------------------------------------------------------
------------
; 3 byte addition of the constant from the table which sets carry if
; result overflows
; ------------------------------------------------------------------
------
Addc24      clrf       Temp        ; register for temporary storage
of CF
            movf       Index,W     ; pointer to lower byte of const
into W
            movwf      HIndex      ; save it into HIndex
            call       DecTable    ; W contains low byte of const
            bcf        Status,CF   ; clear CF
            addwf      LowB,1      ; W + LowB -> LowB
            btfss      Status,CF   ; test overflow
            goto       Add2
            bcf        Status,CF   ; clear CF
            movlw      1
            addwf      MidB,F      ; increment MidB
            btfss      Status,CF
            goto       Add2
            bcf        Status,CF
            movlw      1
            addwf      HigB,F      ; increment HigB
            btfss      Status,CF   ; test overflow
            goto       Add2
            bsf        Temp,CF     ; store CF
Add2        decf       HIndex,F    ; pointer to middle byte into W
            movf       HIndex,W
            call       DecTable
            bcf        Status,CF
            addwf      MidB,1      ; W + MidB -> MidB
            btfss      Status,CF
            goto       Add3
            bcf        Status,CF   ; clear CF
            movlw      1
            addwf      HigB,1      ; increment HigB
```

```
            btfss       Status,CF
            goto        Add3
            bsf         Temp,CF
Add3        decf        HIndex,F    ; pointer to higher byte into W
            movf        HIndex,W
            call        DecTable
            bsf         Status,CF
            addwf       HigB,F      ; W + HigB -> HigB,
            btfss       Status,CF
            goto        ClarCF
            bsf         Status,CF
            goto        AddEnd
ClarCF      rrf         Temp,CF     ; CF -> Status
AddEnd      retlw       0
       ; ------------------------------------------------------------
-------------
; Tables for 3 byte constants
; ------------------------------------------------------------------
------
; Table of decades
; ------------------------------------------------------------------
------
       DecTable    addwf       PC,F        ; W + PC -> PC
            retlw       0           ; 10
            retlw       0           ;
            retlw       0Ah         ;
            retlw       0           ; 100
            retlw       0           ;
            retlw       064h        ;
            retlw       0           ; 1 000
            retlw       03h         ;
            retlw       0E8h        ;
            retlw       0           ; 10 000
            retlw       027h        ;
            retlw       010h        ;
            retlw       01h         ; 100 000
            retlw       086h        ;
            retlw       0A0h        ;
```

```
            retlw      0Fh         ; 1 000 000
            retlw      042h        ;
            retlw      040h        ;
       ; ------------------------------------------------------------
-------------
; Conversion BCD -> 7 segments
; ------------------------------------------------------------------
------
       LEDTable    addwf      PC,F        ; W + PC -> PC
            retlw      b'00111111' ; ..FEDCBA = '0'
            retlw      b'00000110' ; .....CB. = '1'
            retlw      b'01011011' ; .G.ED.BA = '2'
            retlw      b'01001111' ; .G..DCBA = '3'
            retlw      b'01100110' ; .GF..CB. = '4'
            retlw      b'01101101' ; .GF.DC.A = '5'
            retlw      b'01111101' ; .GFEDC.A = '6'
            retlw      b'00000111' ; .....CBA = '7'
            retlw      b'01111111' ; .GFEDCBA = '8'
            retlw      b'01100111' ; .GF..CBA = '9'
            retlw      b'10000000' ; H.......  = '.'
       ; ------------------------------------------------------------
-------------
; Table for RF shift
; example: 10.7 MHz is set as 1 070 000 = 10 53 B0 hex
; ------------------------------------------------------------------
------
       MFTable     addwf      PC,F
            retlw      010h
            retlw      053h
            retlw      0B0h
       ; ------------------------------------------------------------
-------------
; Routine for conversion of 3 byte number into 7 digits
; ------------------------------------------------------------------
------
Go          movlw      6*3-1       ; pointer to dec. table
            movwf      Index       ; 6*3-1 -> Index
            movlw      9           ; maximum of substractions
            movwf      Count       ; 9 -> Count
```

```
                clrf       Help
                movlw      6
            movwf      LEDIndex
Divide      call       Subc24      ; substract untill result is
negative,
            btfsc      Status,CF   ; add last substracted number
            goto       Add24       ; next digit
            incf       Help,F
            decf       Count,F
            btfss      Status,ZF
            goto       Divide
            movlw      3
            subwf      Index,F
            goto       Next
Add24       call       Addc24
            movlw      03h
            subwf      Index,F
Next        movlw      9
            movwf      Count
            movlw      LED1        ; LED1 -> W
            addwf      LEDIndex,W  ; LED1 + LEDIndex -> W
            movwf      Temp
            decf       Temp,F      ; LEDIndex+LED1-1 -> TEMP
            movf       Temp,W
            movwf      FSR         ; W -> FSR
            movf       Help,W      ; Help -> W
            clrf       Help        ; save result at LEDx
            movwf      IndF        ; W -> LED(6..1)
            decf       LEDIndex,F
            movlw      1
            addwf      Index,W
            btfss      Status,ZF
            goto       Divide
            movf       LowB,W
            movwf      LED0        ; the rest -> LED0
       ; ------------------------------------------------------------
--------------
; registers LED0..LED6 are filled with values
```

```
; ------------------------------------------------------------------
-------
                  clrf       TimerH
                  clrf       Timer0
            nop
            nop
                 clrf       LEDIndex
            movlw      .60         ; set initial counter value
            movwf      Index       ; 60 -> Index
                  clrf       IntCon      ; global INT disable,
Timer0 INT disable
                                 ; clear Timer0 overflow bite
       ; ------------------------------------------------------------
-------------
; Start measurement:  RA3 + RA4 set input
; ------------------------------------------------------------------
------
            movlw      b'00010000' ; all ports set L, RA4 set H
            movwf      PortA
            bsf        Status,RP0
            movlw      b'00011000' ; RA0..RA2 output,RA3,RA4 input
            movwf      TrisA
            bcf        Status,RP0
       ; ------------------------------------------------------------
--------------
; 7-step cycle of digits
; ------------------------------------------------------------------
-------
LEDCycle    movlw      LED0
            addwf      LEDIndex,W  ; LED1 + LEDIndex -> W
            movwf      FSR         ; W -> FSR
            movf       IndF,W      ; LED(0..6) -> W
            call       LEDTable    ; W contains segments
            movwf      Temp        ; test for decimal point
            movlw      5
            bsf        Status,ZF
            subwf      LEDIndex,W
            btfss      Status,ZF
            goto       NoDot
```

```
            bsf         Temp,7
NoDot       movf        Temp,W
            movwf       PortB       ; segments -> PortB
            movf        LEDIndex,W  ; LEDIndex -> W
            nop
            movwf       PortA       ; digit number -> PortA
       ; ------------------------------------------------------------
-------------
; Test for TMR0 overflow
; ------------------------------------------------------------------
------
            btfss       IntCon,2
            goto        DoNothing
            incf        TimerH,F    ; YES! Increment SW counter
            bcf         IntCon,2    ; clear overflow bite
            goto        O_K
DoNothing   nop
            nop
            nop
       ; ------------------------------------------------------------
-------------
; The first timing loop 2+3*T1+X procesor cycles
; ------------------------------------------------------------------
------       movlw      T1
            movwf       Temp
Pause       decfsz      Temp,F
            goto        Pause
            nop
;            nop                ; X times NOP
;            nop
       ; ------------------------------------------------------------
-------------
           incf        LEDIndex,F
            movlw       7           ; is 7th?
            bcf         Status,ZF
            subwf       LEDIndex,W
            btfss       Status,ZF
            goto        LEDCycle    ; next digit
            nop
```

```
       ; ------------------------------------------------------------
-------------
; The second timing loop 2+3*T2+Y procesor cycles
; ------------------------------------------------------------------
------
            movlw      T2
            movwf      Temp
Again       decfsz     Temp,F
            goto       Again
            nop
            nop                    ; Y times NOP
            nop
       ; ------------------------------------------------------------
-------------
            clrf       LEDIndex
            decfsz     Index,F
            goto       LEDCycle    ; next 7xLED
            nop
       ; ------------------------------------------------------------
-------------
; The third timing loop 2+3*T3+Z procesor cycles
; ------------------------------------------------------------------
------
               movlw      T3
              movwf      Temp
       EndPause    decfsz     Temp,F
            goto       EndPause
            nop
                   nop                     ; Z times NOP
            nop
       ; ------------------------------------------------------------
-------------
; Final test for TMR0 overflow
; ------------------------------------------------------------------
------
            btfss      IntCon,2
            goto       Nothing2Do
            incf       TimerH,F
            bcf        IntCon,2
```

```
            goto       Nx
Nothing2Do  nop
            nop
            nop
       ; ------------------------------------------------------------
-------------
; Stop the measurement
; ------------------------------------------------------------------
------
       Nx          clrw
            movwf      PortB
            movlw      b'00010000' ; RA0..RA3 = 0
            movwf      PortA       ; W -> PortA

                   bsf        Status,RP0
            movlw      b'00010000' ; RA0..RA3 output
            movwf      TrisA       ; RA4 input
            bcf        Status,RP0
       ; ------------------------------------------------------------
-------------
; Analyse precounter and store counted value in registers
; ------------------------------------------------------------------
------
            movf       Timer0,W
            movwf      MidB        ; TMR0 -> MidB
            movf       TimerH,W
            movwf      HigB        ; TimerH -> HigB
            clrf       Temp
CountIt     incf       Temp,F
            bsf        PortA,3     ; _| false impulz
            bcf        PortA,3     ;    |_
            bcf        IntCon,2
            movf       Timer0,W    ; actual Timer0 -> W
            bcf        Status,ZF
            subwf      MidB,W
            btfsc      Status,ZF
            goto       CountIt
            incf       Temp,F
            comf       Temp,F
```

```
            incf        Temp,F
            incf        Temp,W
            movwf       LowB
            goto        Go          ;  start new cycle
       ; ------------------------------------------------------------
-------------
                   org        0
            end
```

KAYNAKLAR

[1] PEATMAN,JOHN B., Desing with PIC microcontrollers.

[2] IOVİNE , JOHN, PIC microcontroller Projectbook.

[3] BELL AND HOWELL, Microprocessor Architecture, Programing and Applications.

[4] TOCCİ, R.J. ,Digital Systems , Prenciples and Applications

[5] METZGER, D.L. ,Microcomputer Elektronics.

[6] ADALI , EŞREF , Mikroişlemciler ve Mikrobilgisayarlar.

[7] ALTINBAŞAK, ORHAN, Mikrodenetleyiciler PIC Programlama , Altaş yayıncılık , İstanbul , 2000

[8] KARAKAŞ,HAKAN, İleri PIC 16F84 Uygulamalar -1, Altaş Yayıncılık ,İstanbul , 2002

[9] TOPALOĞLU, N. , Mikroişemciler ve Assembly Dili, Seçkin Yayınevi, Ankara, 1999

[10] GÜMÜŞKAYA , H. , Mikroişemciler ve 8051 Ailesi ,Alfa Basım Yayın Dağıtım Ltd. Şti., İstanbul , 1999

[11] GÜMÜŞKAYA,H.,Mikroişlemciler ve Bilgisayarla,Alfa Basım Yayım Dağıtım Ltd.Lti.,İstanbul,1999.

[12] UFFENBECK, J.,Microcomputers and Microprocessor , Prentice- Hall , 1991

[13] ŞAHİN, HİKMET,PIC Programlama Teknikleri ve PIC16F877A,Altaş Yayıncılık,İstanbul,2006

[14] www.antrak.org.tr/gazete/051998/barbar.htm

[15] www.antrak.org.tr/gazete/081999/erek.htm

ÖZGEÇMİŞ

İsmail Zafer TANRIVERDİ, 09.08.1977 tarihinde, Tokat'ta doğdu. İlk okulu Sinop'ta orta okulu Ankara Atatürk Anadolu Lisesi ve liseyi Tokat Anadolu Lisesinde tamamladı. 1995 yılında Sakarya Üniversitesi Mühendislik Fakültesi, Elektrik Elektronik Mühendisliği Bölümünü kazandı ve 2000 yılında mezun oldu.

1999-2000 yılları arasında Goodyear lastikleri A.Ş' de yazılım uzmanı olarak görev yaptı. 2000 yılında T.C Sakarya Üniversitesi Geyve Meslek Yüksek Okulunda Öğretim Görevlisi olarak göreve başladı. 2001 yılında Bilgisayar Programcılığı Program başkanlığına atandı.

2003-2005 yılları arasında TÜVASAŞ (Türkiye Vagon Fabrikası) AR-GE daire başkanlığında Elektronik Mühendisi olarak görev yaptı. Eş zamanlı olarak Sakarya meslek Yüksek Okulu Endüstriyel Elektronik, Elektrik ve Bilgisayar programcılığı bölümlerinde öğretim görevlisi olarak görev yaptı.

Halen bu görevine devam etmekte olan Tanrıverdi, 2000 yılında Sakarya Üniversitesi Fen Bilimleri Enstitüsü Elektrik-Elektronik anabilim dalında yüksek lisans son sınıf öğrencisidir. Evli ve Sakarya merkezde ikamet etmektedir.