PIC MİKRODENETLEYİCİLER İÇİN GERÇEK ZAMANLI İŞLETİM SİSTEMİ

HÜSEYİN ÇOTUK

YÜKSEK LİSANS TEZİ

BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI

TOBB EKONOMİ VE TEKNOLOJİ ÜNİVERSİTESİ

FEN BİLİMLERİ ENSTİTÜSÜ

MART 2008

ANKARA

_____

Prof. Dr. Yücel ERCAN

Müdür

Bu tezin Yüksek Lisans derecesinin tüm gereksinimlerini sağladığını onaylarım.

_____

Prof. Dr. Ali YAZICI

Anabilim Dalı Başkanı

Hüseyin ÇOTUK tarafından hazırlanan PIC MİKRODENETLEYİCİLER İÇİN GERÇEK ZAMANLI İŞLETİM SİSTEMİ adlı bu tezin Yüksek Lisans tezi olarak uygun olduğunu onaylarım.

_____ _____

Dr. Cengiz ERBAŞ
Tez Yrd. Danışmanı

Doç. Dr. Y.Murat ERTEN
Tez Danışmanı

Tez Jüri Üyeleri

Başkan :Doç. Dr. Mehmet Önder EFE _____

Üye : Doç. Dr. Y. Murat ERTEN _____

Üye : Yrd. Doç. Dr. Oğuz ERGİN _____

TEZ BİLDİRİMİ

Tez içindeki bütün bilgilerin etik davranış ve akademik kurallar çerçevesinde elde edilerek sunulduğunu, ayrıca tez yazım kurallarına uygun olarak hazırlanan bu çalışmada orijinal olmayan her türlü kaynağa eksiksiz atıf yapıldığını bildiririm.

Hüseyin ÇOTUK

Üniversitesi : **TOBB Ekonomi ve Teknoloji Üniversitesi**
Enstitüsü : **Fen Bilimleri Enstitüsü**
Anabilim Dalı : **Bilgisayar Mühendisliği**
Tez Danışmanı : **Doç. Dr. Yusuf Murat ERTEN**
Tez Yrd. Danışmanı : **Dr. Cengiz ERBAŞ**
Tez Türü ve Tarihi : **Yüksek Lisans – Mart 2008**

Hüseyin ÇOTUK

PIC MİKRODENETLEYİCİLER İÇİN GERÇEK ZAMANLI İŞLETİM SİSTEMİ (PICOS)

ÖZET

Günümüzde otomobillerde, görüntü sistemlerinde, cep telefonlarında, iletişim cihazlarında, biyomedikal uygulamalarda, endüstriyel otomasyon sistemlerinde gömülü sistemlerin uygulama alanları giderek yaygınlaşmaktadır. Gömülü sistemler üzerinde çalışmak üzere hazırlanan yazılımlar, donanım kısıtlarına rağmen genelde gerçek-zamanlı çalışırlar. Geleneksel yazılım sistemleri; gömülü sistemlerin gerek kısıtlı sistem kaynakları, gerek çok değişken uygulama gereksinimleri gerekse gerçek zaman kısıtları nedeniyle gömülü sistem ihtiyaçlarına cevap verememektedir. Genel amaçlı işletim sistemlerinin de gömülü sistemlerde kullanım kısıtları, gömülü sistemler için özel tasarlanmış gerçek zamanlı işletim sistemi geliştirilmesi ihtiyacını doğurmuştur. 1980'den sonra birçok devre elemanının aynı yonga içerisinde yer almaya başlaması ile mikrodenetleyici kavramı oluşmaya başlamıştır. Mikrodenetleyiciler sayesinde gömülü sistemlerin hem maliyetleri düşmüş hem de boyutlarında ciddi bir küçülme yaşanmıştır. Bu çalışmada; kolay bulunabilir ve ekonomik olmaları, geliştirme ortamının internet üzerinden veya üreticiden istendiğinde ücretsiz olarak elde edilebilmesi, çok geniş bir kullanıcı kitlesine sahip olmaları, oldukça basit sıfırlama, saat sinyali ve güç devreleri gerektirmeleri PIC mikrodenetleyicilerin seçilmesini sağlamıştır. Proje Microchip MPLAB tümleşik geliştirme ortamında, ANSI C uyumlu MPLAB C18 derleyicisiyle geliştirilmiştir. Gömülü sistemlerde kullanılmak amacıyla tasarlanan PICOS, 8-bit PIC ailesinin en gelişmiş serisi olan PIC18Fxxx serisi ile 16-bitlik PIC24xxx ve 32-bitlik PIC32xxx serileri için geliştirilmiş gerçek zamanlı bir işletim sistemidir. PICOS, geliştirilmesi esnasında istenirse çağrı üstünlüğü prensibi istenirse işbirlikçi prensip ile çalışabilecek şekilde tasarlanmıştır. Oldukça küçük, basit ve kullanımı son derece kolaydır. Tüm fonksiyonlar ve değişkenler Türkçe olarak tanımlanmıştır. Çok düşük RAM, ROM bellek kullanımı ve işlemci yükü getirmektedir. Oldukça basit bir çekirdek yapısına sahiptir.

Anahtar Kelimeler: Gerçek zamanlı işletim sistemi, gömülü sistemler, PIC mikrodenetleyiciler

University : **TOBB University of Economics and Technology**
Institute : **Institute of Natural and Applied Sciences**
Science Programme : **Computer Engineering**
Supervisor : **Associate Professor Dr. Yusuf Murat ERTEN**
Assistant Supervisor : **Dr. Cengiz ERBAŞ**
Degree Awarded and Date : **M. Sc. – March 2008**

Hüseyin ÇOTUK

REAL TIME OPERATING SYSTEM FOR PIC MICROCONTROLLERS

ABSTRACT

Embedded systems are widely used for many applications like mobile phones, mp3 players, image processing, communication systems, biomedical applications and industrial control systems. As they are designed for specific tasks, embedded systems' sizes are quite small. Because of their small sizes, microcontrollers are very suitable for embedded systems. Despite having hardware constraints, embedded systems have software that commonly operate in real-time. Embedded systems have different requirements compared to traditional software systems, such as, limited resources, variable application reqiurements and real time constraints, traditional software systems do not fit their requirements. In addition, classical operating systems do not ensure usage criteria as well. Therefore, in order to achieve these goals, designing a real time operating system for embedded systems is the purpose of this project. Due to their popularity, economic prices, easiness to get and use, strong support by the vendor, basic reset and clock circuits, PIC microcontrollers are chosen as the hardware platform of this project. With MPLAB integrated development environment, ANSI C compatible MPLAB C18 compiler is used for developing the project. PIC Operating System (PICOS) is intended to be used for embedded systems and compatible with PIC 18FXXX, PIC24XXX and PIC32XXX families. Typical PIC applications in industry are based on cooperative scheduling. PICOS supports preemptive scheduling as well as cooperative scheduling. It is quite small, basic and easy to use. It has very little RAM, ROM and CPU usage. Kernel structure is also very basic as well.

Keywords: Real time operating systems, embedded systems, pic microcontrollers

TEŞEKKÜR

Çalışmalarım boyunca değerli yardım ve katkılarıyla beni yönlendiren hocam sayın Doç. Dr. Y. Murat ERTEN'e, yine kıymetli tecrübelerinden faydalandığım sayın Dr. Cengiz ERBAŞ'a, Prof. Dr. Ali YAZICI'ya ve TOBB Ekonomi ve Teknoloji Üniversitesi Bilgisayar Mühendisliği Bölümü ile Elektrik ve Elektronik Mühendisliği Bölümü öğretim üyelerine teşekkürü bir borç bilirim.

İÇİNDEKİLER

ÇİZELGELERİN LİSTESİ

ŞEKİLLERİN LİSTESİ

KISALTMALAR

Kısaltmalar	Açıklama
ANSI	American National Standards Institute
CAN	Controller Area Network
CMOS	Complementary metal–oxide–semiconductor
EEPROM	Electrically Erasable Programmeble Read Only Memory
HF	High Frequency
ICPROG	Integrated Circuit Prpgrammer
ICSP	In Circuit Serial Programming
I^2C	Inter Integrated Circuit
LCD	Liquid Crystal Display
LED	Light Emitting Diode
PIC	Peripheral Interface Controller
PICOS	PIC Operating System
PLC	Programmable Logic Controller
PROM	Programmable Read Only Memory
RAM	Random Access Memory
RISC	Reduced Instruction Set Computer
ROM	Read Only Memory
SPI	Serial Peripheral Interface Bus
USART	Universal Synchronous/Asynchronous Receiver Transmitter
USB	Universal Serial Bus

BÖLÜM 1

1. GİRİŞ

1.1. Çalışmanın Amacı

Gömülü sistemler herhangi bir sistemin içinde yer alan ve o sisteme akıllılık özelliği katan elektronik donanım ve yazılımdan oluşan bütünü ifade etmektedir. Sözü edilen yazılımlar, bilgisayarlarımızdaki genel amaçlı yazılımlardan farklı olarak, kullanıcıyla doğrudan değil dolaylı etkileşimde bulunan ve genellikle tek bir görevi yerine getiren yazılımlardır. Bu görev daha çok mekanik bir eylemi ifade etmekle birlikte, mekanik olmayan unsurları da içerebilir. Her gömülü sistemde bir işlemci, bir bellek ve diğer yardımcı birimler bulunur. İşlemcisi olan her birim için de bir işletim sisteminden bahsetmek gerekmektedir.

Dünyada üretilen mikroişlemcilerin yaklaşık %98'inin gömülü sistemlerde kullanıldığı, gömülü sistem gereksiniminde her yıl 200 milyon artış yaşandığı ileri sürülmektedir. Artışın bu hızla devam etmesi halinde yılda 5-6 milyar adet mertebesinde olan ihtiyacın 10 yıl içerisinde 10 milyar adet mertebelerine ulaşacağı öngörülmektedir. Ortalama 10 $ üzerinden hesaplandığında yalnızca donanım maliyeti 50-60 milyar $'lık bir büyüklüğü ifade etmektedir [1].

Çoğu programcı, bir sonsuz döngü içerisinde ana uygulamayı çalıştıran ve kritik uygulamalar için kesme kullanan geleneksel yöntemlerle programlamaya alışıktır. Bu tür sistemler ön plan-arka plan sistemler olarak da isimlendirilir. Zira, kesmeler her şeyden daha öncelikli çalıştıkları için ön planda yer alır. Ana döngü ise arka planda kesmeler aktif değilse çalışmasına devam eder. Ancak uygulama büyüklükleri ve karmaşıklık arttıkça bu sistemlerin planlanması, yönetimi ve bakımı oldukça zor hale gelmeye başlar. İşte bu aşamada çoklu görev yürütebilen bir işletim sistemi önem kazanmaktadır. İşletim sistemi sayesinde işlemci gücü sadece bir görev için kullanılmayacak, farklı zamanlarda farklı görevlere odaklanabilecektir. Böylece geleneksel sistemlerde olduğu gibi bekleme gerektiren bir durumda işlemci gücü atıl kalmayacak, bu esnada bir görev bekletilirken diğer bir görev çalıştırılacaktır.

Birçok gömülü sistem uygulamasında işlem daha önceden belirlenmiş zaman dilimi içerisinde bitirilmemişse, sistem işlevsel olarak düzgün çalışmış olsa bile başarısız kabul edilir. Yani başarı zamana bağımlıdır. Bu nedenle gömülü sistemlerde kullanılacak işletim sisteminin gerçek zamanlı olması tercih edilmektedir. Kullanım kolaylıkları nedeniyle gömülü sistemler üzerinde bazı uygulamalarda artık mikroişlemcilerin yerini mikrodenetleyiciler almış ve günümüzde en kolay elde edilebilen, üzerinde uygulama geliştirilmesi en basit olan ve en fazla tercih edilen mikrodenetleyici PIC olduğu için bu projede bu mikrodenetleyicilerin kullanılması uygun bulunmuştur.

Özetle, bu çalışmada PIC mikrodenetleyiciler üzerinde çalışan gerçek zamanlı, çoklu görev yürütümü yapabilen, olay güdümlü, çağrı üstünlüğü (preemptive) ve işbirlikçi (cooperative) prensiplere göre çalışabilen, sistem kaynaklarını olabildiğince kısıtlı kullanan bir işletim sistemi geliştirilmesi hedeflenmiştir. Bu işletim sistemi iyi tasarlandığı takdirde, öncelikli uygulamalar diğerlerinin önüne geçecek, bu da uygulamalardaki verimliliği arttıracaktır. Böylelikle bazı bayrakları ve sayıcıları test etmek, olayları yoklamak için komut çevrimlerini harcamak yerine işlemci gücünün gerektiği şekilde en uygun yerde kullanabilmesi sağlanacaktır.

Bu kapsamda, 2. bölümde genel bilgiler başlığı altında gömülü sistemler ve uygulama alanlarından, mikrodenetleyiciler ve tarihçelerinden, PIC mikrodenetleyicilerden ve kullanılan PIC 18F452 mikrodenetleyicisinden, 3. bölümde kullanılan geliştirme araçlarından, 4. bölümde PIC mikrodenetleyiciler için geliştirilen gerçek zamanlı işletim sistemi PICOS ve özelliklerinden, son olarak da 5. bölümde çalışmanın sonuçları ve yapılacak çalışmalardan bahsedilecektir.

BÖLÜM 2

2. GENEL BİLGİLER

2.1. Gömülü Sistemler

Gömülü sistemler, önceden belirlenmiş donanım üzerinde istenilen görevleri yapmak üzere hazırlanmış aygıtlardır. Sadece belirlenen amacı yerine getirmek için tasarlanmaları, maliyetlerinin düşük olmasını ve karmaşık olmayan yapıları yaygın kullanılmalarını sağlamaktadır. Günümüzde otomobillerde, görüntü sistemlerinde, cep telefonlarında, iletişim cihazlarında, biyomedikal uygulamalarda ve benzer birçok alanda gömülü sistemlere rastlanmaktadır [2].

İlk gömülü sistem MIT laboratuarlarında Charles Stark Draper tarafından geliştirilen Apollo kılavuz bilgisayarıdır. Daha sonra ilk kütlesel gömülü sistem üretiminin 1961 yılında yapılmasından sonra birimlerin fiyatları kendi fiyatlarının % 0,3'üne kadar düşmüştür.

1950'den sonra tümleşik devre teknolojisinde meydana gelen gelişmeler bir çok yeniliğin önünü açmaya başlamıştır. 1980'den sonra ise birçok devre elemanının aynı yonga içerisinde yer almaya başlaması ile mikrodenetleyici kavramı oluşmaya başlamıştır. Bu sayede gömülü sistemlerin hem maliyetleri düşmüş hem de boyutlarında ciddi bir küçülme yaşanmıştır. Aynı zamanda çok düşük güç tüketimine sahip olmaları daha geniş uygulama alanlarına yayılmalarına yardımcı olmuştur. Günümüzde gömülü sistemlerin bazı uygulama alanları aşağıda sıralanmıştır:

- Endüstriyel otomasyon (PLC) sistemleri
- Ağ cihazları (anahtarlar, yönlendiriciler, v.b.)
- Uydu cihazları
- Cep telefonları
- Ölçü aletleri (multimetre, analizör, v.b.)
- Tıbbi cihazlar

- Beyaz eşyalar (çamaşır makinesi, bulaşık makinesi, buzdolabı)
- Bilgisayar yazıcıları
- Fotokopi, faks makinesi gibi iletişim cihazları
- Her türlü otomasyon sistemleri

Gömülü sistemler üzerinde çalışmak üzere hazırlanan yazılımlar, donanım kısıtlarına rağmen genelde gerçek-zamanlı çalışırlar. Gömülü sistemlerin genelde ya işletim sistemleri yoktur ya da özelleşmiş gömülü işletim sistemleri bulunabilir. Bunlara genelde gerçek-zamanlı işletim sistemleri adı verilir. Gömülü sistemlerde sabit diskin yerine yeniden yazılabilir bir programlama belleği, klavyenin yerine basit bir tuş takımı, bilgisayar monitörünün yerine küçük bir LCD ekran kullanılabilir [3].

Gömülü sistemler üzerindeki yazılımlar bazen yıllarla ifade edilebilecek kadar uzun süre hatasız çalışacak şekilde tasarlanmalıdır. Bu zaman zarfında, donanımda meydana gelebilecek hataların tespit edilmesi daha kolaydır. Ancak yazılım hataları daha zor ayıklanabilir. Ayrıca yazılım, donanımı tekrar başlatabilecek şekilde tasarlanmalıdır çünkü gömülü sistemler insanların ulaşamayacağı yerlerde çalışmak zorunda olabilir. Aksi halde yazılımdaki bir aksaklıkta sistemin çalışması durur ve tespit edilene kadar kayıplar yaşanabilir. Böyle durumlarda genelde yazılımdaki önlemlerin yanı sıra yazılımı destekleyen donanımsal bazı tedbirlerden de söz edilebilir. Bekçi köpeği zamanlayıcısı (watchdog timer) yazılımda meydana gelen bir hata durumunda sistemi baştan başlatarak işlemin aksamasını önleyebilir [4].

Gömülü sistemlerin tasarımlarında kullanılan mikroişlemci ya da mikrodenetleyicilerin çalışabilmeleri için genelde sistem tasarımlarında bir başlangıç kodu bulunur. Bu kod başlangıçta bütün kesmeleri devre dışı bırakır, sistemi kontrol eder ve uygulamayı başlatır. Yazılım sürecinde kullanıcıya işaret veren çeşitli denetlemeler kullanılabilir. En yaygın yöntem LED'leri kullanmaktır. Sistem çok kapsamlıysa daha farklı araçlar kullanılabilir.

Gömülü sistemler, kısıtlı sistem kaynakları, gerçek zamanlılık ve çok değişken uygulama gereksinimlerine sahip olmaları ile geleneksel yazılım sistemlerinden ayrılmaktadır [5].

Gömülü sistemler için gerçekleştirilen işletim sistemleri, genel amaçlı işletim sistemlerinden farklı tasarım amaçlarına ve servislere sahiptirler. Verimli bir kaynak yönetimi ve gerçek zamanlı sistem algoritmaları içeren gömülü işletim sistemleri, giderek modern yazılım kavramlarının sunmuş olduğu avantajlara gereksinim duymaktadır. Büyük ve karmaşık olan günümüz gömülü sistem uygulamalarının ihtiyaçlarını karşılayacak olan gömülü işletim sistemleri, iyi bir sistem başarımına sahip olmak ve verimli kaynak yönetimi algoritmaları içermek dışında yeniden yapılandırılabilme, taşınabilme ve dağıtıklık gibi modern yazılım özelliklerini destekleyebilmelidir [6].

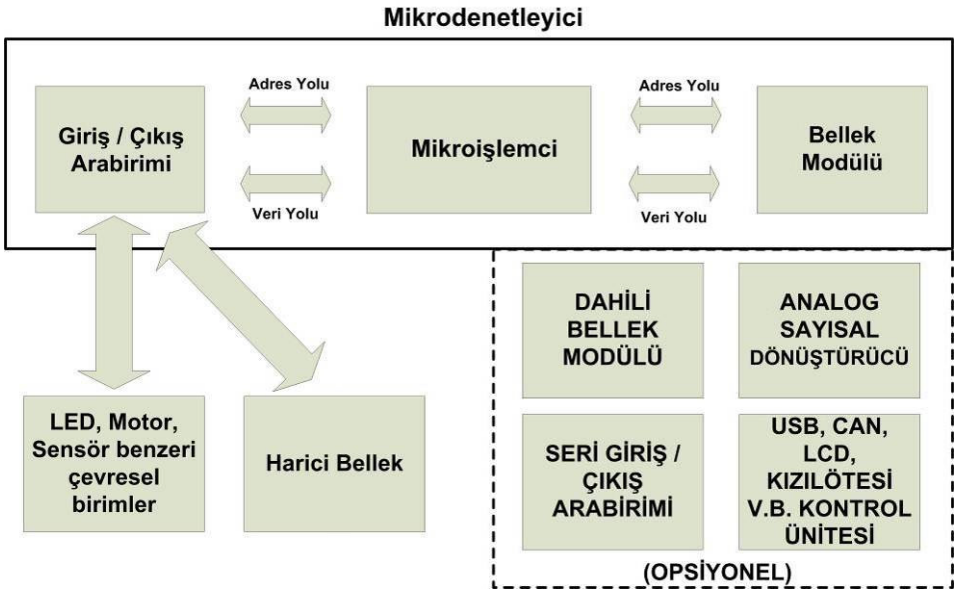
2.2. Mikrodenetleyiciler

Mikrodenetleyiciler, mikroişlemcilere en az giriş-çıkış arabirimleri ile bellek modülü eklenerek tek bir yonga içerisinde birleştirilen ve günümüzde özellikle gömülü sistemlerde oldukça yaygın olarak kullanılan kontrol aygıtlarıdır. Mikrodenetleyiciler, giriş-çıkış arabirimleri ve hafıza modülü yanında analog-sayısal dönüştürücü, seri giriş-çıkış arabirimi, darbe sinyali çıkışı, USB bağlantı kontrolcüsü, kızıl ötesi ve LCD ekran sürücüsü gibi uygulamaların karmaşıklığını azaltan, oldukça pratik modülleri de barındırabilir. Bu modüller seçilen mikrodenetleyici ailesi ve modeline göre farklılık gösterirler (Şekil 2.1).

Genelde her uygulama kullanıcı ile etkileşim gerektirdiği için hemen hemen tüm uygulamalarda giriş çıkış arabirimlerine ihtiyaç duyulmaktadır. Bu arabirimler kendisine gelen sinyale göre işlem yapabildiği gibi gerçekleştirdiği işlem sonucunda harici bir sinyal de üretebilirler. Daha açık bir ifade ile, mikrodenetleyici kullanıcıdan gelen bir isteğe veya sensor gibi başka bir bileşende meydana gelen değişikliğe göre bir işlemi gerçekleştirebilirken bu işlemin sonucunda kullanıcıya cevap olarak bir sinyal üretebilir, bu sinyal ile de gerekli bileşenler sürülerek anlamlı bir sonuç elde edilebilir. Aynı işlem mikroişlemci ile yapılmak istenildiğinde harici

giriş-çıkış arabirimleri ve mutlaka kullanılması gereken bir hafıza modülü mikroişlemci ile ilişkilendirilmelidir. Bu ise hem uygulamanın karmaşık hale gelmesine hem de maliyetin artmasına neden olacaktır. Aynı şekilde analog-sayısal dönüştürücü gibi ihtiyaç duyulan her modül benzer dezavantajları beraberinde getirecektir.

Mikrodenetleyicilerin mikroişlemcilere tercih edilmesinin başlıca nedenleri arasında karmaşıklığı azaltmaları, baskılı devre gerektirmemeleri, maliyeti düşürmeleri, oldukça düşük güç tüketimine sahip olmaları, az yer kaplamaları, pratik kullanım ve uygulama geliştirme kolaylıkları sayılabilir.



Şekil 2.1. Mikrodenetleyici Yapısı

Bu avantajlar sayesinde artık bazı özel uygulamalar dışında mikroişlemciler yerini mikrodenetleyicilere bırakmış, iletişim sistemlerinde, görüntü işlemede, biyomedikal uygulamalarda, birçok endüstriyel kontrol sisteminde ve çoğu gömülü uygulamada mikrodenetleyiciler vazgeçilmez devre elemanları haline gelmiştir [7].

Günümüzde en çok bilinen mikrodenetleyici üreticileri arasında Microchip, Intel, Scenix, Atmel, Texas Instruments, NEC, Motorola, Zilog yer almaktadır. Her üretici

kendi ürettiği çiplere bir isim vermekte ve barındırdığı modül, hafıza büyüklüğü gibi özelliklerine göre alt sınıflara ayırmaktadır.

Herhangi bir uygulama geliştirilirken kullanılacak modüller, gereken hafıza büyüklüğü, dahili zamanlayıcı ve kesme kaynakları gibi özellikler listelenmeli ve gerekli araştırmalar yapılarak uygun çip seçilmelidir [8]. Ürün seçerken dikkat edilmesi gereken bazı modüller ve dâhili özellikler aşağıdaki gibi sıralanabilir:

- Çalışma frekansı
- Osilatör tipi (RC, XT, HF gibi)
- Analog-sayısal dönüştürücü modülü
- Giriş-çıkış arabirimleri sürme kapasitesi (akım büyüklükleri)
- Analog ve sayısal olarak ayarlanabilen giriş/çıkış arabirimi ve uç sayısı
- Seri giriş/çıkış arabirimi (senkron, asenkron)
- Desteklenen haberleşme protokolleri (I^2C, SPI, USART, PSP gibi)
- Darbe sinyali çıkışı
- Harici kesme
- Zamanlayıcı tabanlı kesme
- Dâhili programlama belleği büyüklüğü
- Harici bellek arabirimi
- Dâhili bellek tipi seçenekleri (ROM, EPROM, PROM ve EEPROM)
- Dâhili RAM seçeneği ve büyüklüğü
- Dâhili zamanlayıcı sayısı ve büyüklükleri

- USB, CAN, Ethernet bağlantı kontrolcüsü, LCD ekran sürücüsü, kızılötesi modülü, v.b.

Kullanılabilen modülleri ve yaygın kullanımları göz önüne alınarak bu projede Microchip firması tarafından üretilen PIC mikrodenetleyiciler kullanılmıştır.

2.3. PIC Mikrodenetleyiciler

Arizona Microchip Technology, ilk olarak General Instruments tarafından üretilmiş olan Harvard mimarisine sahip mikrodenetleyicilerin üretimine devam etmiş ve bu ürün "Peripheral Interface Controller" kelimelerinin baş harflerinden oluşan PIC adını almıştır.

İlk olarak 12-bit komut seti ile tasarlanan ve toplam 33 komuta sahip olan PIC16C5X serisini pazara çıkaran Microchip, azaltılmış komut seti (Reduced Instruction Set Computer –RISC) mimarisi ile kullanışlı, ucuz ve hızlı işlemci üretmeye başlamıştır. Bir sefer yazılabilen (ROM) bellek kullanılan bu ürünler C kodunu almışlardır. 512 ve 2048 bayt program belleği, 25-73 bayt veri belleğine sahip ürünler 18 veya 28 uca sahip entegre devreler olarak piyasaya sürülmüşlerdir. Daha sonra üretilen PIC12CXXX serisi aynı özelliklere sahip olup sadece 8 uca sahiptir (Şekil 2.2).

PIC16C54 PIC16C54 PIC12F675

Şekil 2.2. PIC 12 ve 16 aileleri

Daha sonra ortaya çıkan analog-sayısal dönüştürücüler, daha fazla kesme ve programlama alanı, 16-bit sayıcı ve zamanlayıcı gibi ihtiyaçlar ve gelişen teknoloji sonrasında 1992 yılında 14-bit komut setine sahip PIC16CXXX ailesinin üretimine başlanmıştır. Bu ailedeki komut sayısı ise 35 adet olup önceki seri ile uyumlu tasarlanmıştır (Şekil 2.3).

PIC16F84 PIC16F877A

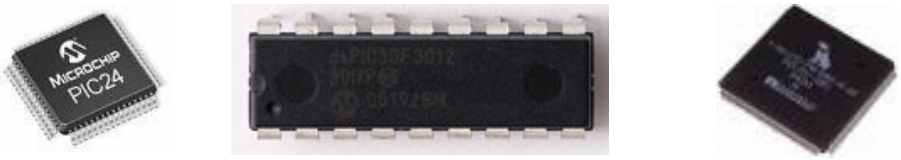
Şekil 2.3. PIC 16F serisi

1997 yılında çarpma yapan aritmetik işlem ünitesine ve gelişmiş arabirim özelliklerine sahip PIC17CXXX serisi pazara sunulmuştur. 1999 yılında ise genişletilmiş 16-bit komut setine sahip PIC18CXXX serisinin üretimine başlanmıştır. Bu seride komut sayısı 77'ye çıkmış ve yüksek seviyeli programlama dillerine yeterince destek verilmiştir. Birden fazla programlanabilen programlama belleğine sahip ürünler F kodu ile isimlendirilmiştir (Şekil 2.4) [9].

PIC18F452 I/P PIC18F452 I/L

Şekil 2.4. PIC 18F452

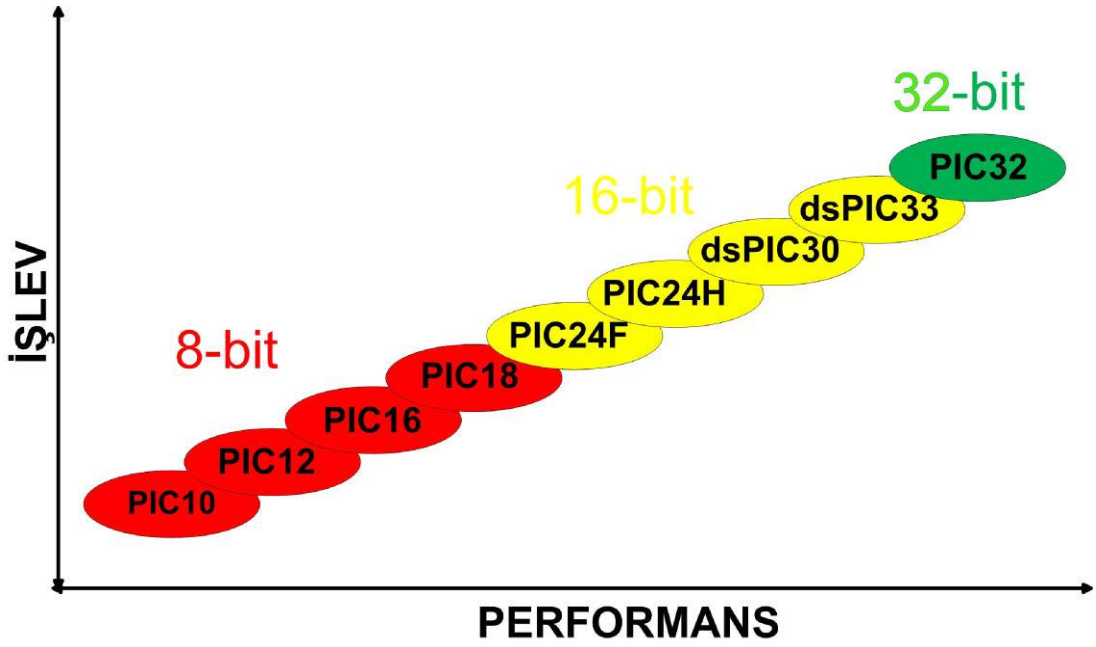
Tüm bu 8-bit'lik mikrodenetleyicilerden sonra 16-bit'lik PIC24XXX ve DSPIC30XXX serisi ile 32-bit'lik PIC32XXX serisi üretilerek pazara sunulmuştur (Şekil 2.5).



PIC24 PIC30 PIC32

Şekil 2.5. PIC24, PIC30 ve PIC32 aileleri

Uzun yıllar 8-bit'lik mikrodenetleyici üreten Microchip, günümüzde 8-bit'lik, 16-bit'lik ve 32-bit'lik mikrodenetleyici ve EEPROM hafıza üretmektedir (Şekil 2.6). Arizona eyaletinde iki, Tayland ve Tayvan'da da birer tane olmak üzere toplam dört fabrika ile kendi alanında dünyada söz sahibidir [10].



Şekil 2.6. PIC işlev – performans çizelgesi

PIC mikrodenetleyiciler kolayca temin edilebilir ve oldukça ekonomiktir. Geliştirme ortamı MPLAB IDE internet üzerinden veya Microchip'ten istendiğinde ücretsiz olarak elde edilebilir. Çok geniş bir kullanıcı kitlesine sahip olan PIC mikrodenetleyiciler, temel elektronik bilgisine sahip herkesin kolayca bulabileceği basit devre elemanları kullanarak yapılabilen bir donanımla programlanabilir. Oldukça basit sıfırlama, saat sinyali ve güç devreleri gerektiren PIC mikrodenetleyicilerin tüm bu özellikleri bu projede seçilmelerini sağlamıştır. Bunun yanında PIC ailesi içerisinde 32 KB büyüklüğünde yeniden yazılabilir belleğe sahip olması ve gelişmiş özellikleri ile PIC18F452 tercih edilen mikrodenetleyici olmuştur.

2.3.1. PIC18F452 ve Genel Özellikleri

PIC18F452, 16-bit komut kümesine sahip PIC ailesi içerisinde en gelişmiş mikrodenetleyicidir. 32 KB yeniden yazılabilir belleğe sahip PIC18F452'de tek kelimelik 16384 komut kullanılabilmektedir. 1536 bayt büyüklüğünde RAM bellek ve 256 bayt büyüklüğünde elektrik kesintilerinden etkilenmeyen EEPROM bellek kullanılabilirken çalışma frekansı 40 MHz'e kadar çıkabilmektedir. 18 farklı kesme kaynağına sahip mikrodenetleyicide kesmelere farklı öncelikler atanabilmektedir. 3 adet 8 uçlu, 1 adet 7 uçlu ve 1 adet 3 uçlu olmak üzere toplam 5 adet giriş çıkış portu bulunmaktadır. 16-bit genişlikte komutlara ve 8-bit genişlikte veri yoluna sahiptir. 8 bit iki değişkeni çarpabilen bir donanım çarpıcının yanında 4 adet zamanlayıcıya, 2 adet karşılaştırma modülüne sahiptir. Sadece 75 komut bulunduran bir komut seti sayesinde programlanabilen PIC18F452, C derleyiciler için tasarlanmış ve optimize edilmiş mimarisi ile yüksek performanslara erişebilmektedir. 25 mA akım değerine kadar çıkışları sürebilmektedir. Üç adet dâhili kesme ucu bulunmakta (INT0, INT1, INT2) ve bu kesmelere farklı öncelikler atanabilmektedir. İkinci bir osilatör opsiyonu ve iki adet yakalama, karşılaştırma ve darbe genişlik modülatörü bulunmaktadır. Senkron seri port sayesinde, üç uç ile SPI ve I^2C protokolleri kullanılabilmektedir.

PIC 18F452 adreslenebilir USART modülü sayesinde RS-232 ve RS-485 protokolleri desteklenmektedir. 10-bit çözünürlükte 8 kanal analog sayısal dönüştürücü sayesinde çok hızlı örnekleme oranı ve uyku modunda dönüştürebilme sağlanmaktadır.

Yeniden yazılabilir programlama belleği 100.000 kez silinip yazılabilirken EEPROM belleği 1.000.000 kez silinip yazılabilmektedir. Programlama belleği ve EEPROM belleği 40 yıl boyunca içerisindeki verileri koruyabilmektedir. Programlama belleği sigortalar sayesinde korumaya alınabilmekte, sadece iki uç yardımı ile devre üzerinden çıkarmadan mikrodenetleyici programlanabilmektedir.

CMOS teknolojisi sayesinde düşük güç tüketimi ve hızlı bellek erişimine sahiptir. 2.0 V'tan 5.5 V'a kadar çıkabilen esnek çalışma geriliminin yanında geniş çalışma

sıcaklıkları sayesinde endüstride çok geniş bir alanda kullanılmaktadır. PLCC, TQFP, DIP ve DIP/SOIC gibi çok çeşitli kılıflarda üretilmektedir [11].

EK A'da üretimi yapılan PIC18F452 kılıfları ve PIC18F452 giriş / çıkış uçları yer almaktadır. Her giriş çıkış ucunun hangi amaçla kullanıldığı Çizelge A.1 ile açıklanmaktadır.

2.3.1.1. Kristal Osilatör İçin Kondansatör Seçimi

PIC 18F452 mikrodenetleyicilerinde saat stabilizasyonu sağlamak amacıyla kullanılan kondansatörlerin seçimi önem kazanmaktadır. Düşük güçlü (DG), kristal (KR) ve yüksek hızlı (YH) modlarında aşağıdaki kondansatör değerleri üretici tarafından test edilmekle birlikte önerilmektedir (Çizelge 2.1).

Çizelge 2.1. Kristal osilatör için kondansatör değerleri

Mod	Frekans	Kondansatör 1	Kondansatör 2
DG	32.0 KHz	33 pF	33 pF
	200 KHz	15 pF	15 pF
KR	200 KHz	22-68 pF	22-68 pF
	1.0 MHz	15 pF	15 pF
	4.0 MHz	15 pF	15 pF
YH	4.0 MHz	15 pF	15 pF
	8.0 MHz	15-33 pF	15-33 pF
	20.0 MHz	15-33 pF	15-33 pF
	25.0 MHz	15-33 pF	15-33 pF

2.3.1.2. Bellek Yerleşimi

Gelişmiş mikrodenetleyici cihazlarında üç ayrı hafıza bloğu bulunmaktadır. Bunlar:

- Kullanıcı tarafından kullanılan program belleği
- Dinamik olarak yenilenen RAM belleği

- Elektrik kesintilerinden etkilenmeyen dâhili EEPROM belleğidir.

Veri ve program belleği için ayrı ayrı veri yolu kullanılması bu bloklara eş zamanlı erişimi sağlamaktadır.

2.3.1.3. Program Belleği Yerleşimi

21-bit uzunluğundaki komut adres yazmacı, 2 MB büyüklüğündeki program belleğini adresleyebilmektedir. 0x0000h adresinde sıfırlama vektörü, 0x0008h ve 0x0018h adreslerinde de kesme vektörleri bulunmaktadır.

2.3.1.4. RAM Bellek Yerleşimi

RAM bellek 16 adet 256 bayttan oluşan bloktan meydana gelir. RAM bellek, özel fonksiyon yazmaçları ve genel amaçlı yazmaçlardan oluşur ve bu alanda bulunan her yazmaç 12-bit uzunluğunda adrese sahip olup 4096 bayta kadar genişleyebilmektedir. Komut seti ve mimari tüm bloklar arasında işlemlere izin vermektedir.

2.3.1.5. EEPROM Belleği

EEPROM belleği normal çalışma voltajları içerisinde okunabilir ve yazılabilir durumdadır. Okuma ve yazma işlemleri için önceden tanımlanmış yazmaçlar kullanılır. Bu yazmaçlardan birinde okunup yazılacak veri yer alırken bir diğerinden bellekte erişilecek alanın adresi tutulmaktadır.

2.3.1.6. Kesmeler

PIC18F452 bir çok kesme kaynağına ve bu kesmelere yüksek veya düşük şeklinde atanabilen önceliklere sahiptir. Yüksek öncelikli kesme vektörü 0x0008h adresinde, düşük öncelikli kesme vektörü 0x0018h adresinde yer alır. Yüksek öncelikli kesmeler, o anda devam eden düşük öncelikli kesme varsa bu kesmeyi durdurur, kendi işini tamamlar ve düşük öncelikli kesmeye geri döner. INT0 kesmesinin dışında tüm kesmelerin benzer şekilde üç adet durum bayrağı vardır. Bunlar:

- Kesmenin meydana geldiğini gösteren bayrak
- Kesmeyi etkinleştiren bayrak

- Kesmenin önceliğini (düşük veya yüksek) belirleyen bayraktır.

2.3.1.7. Giriş / Çıkış Portları

PIC18F452 PortA, PortB, PortC, PortD ve PortE olmak üzere 5 adet giriş / çıkış portuna sahiptir. Sayısal veya analog genel amaçlı giriş / çıkış ucu, analog-sayısal dönüştürücü giriş ucu, yakalama-karşılaştırma giriş / çıkış ucu, haberleşme protokolleri giriş / çıkış ucu gibi amaçlarla kullanılan bu portların sahip olduğu uçlar ile ilgili detaylı bilgi Ek A'da Çizelge A.1'de verilmiştir. Bu portlar, genel anlamda uygulamanın kullanıcı ile etkileşimini sağlayan arayüzlerdir.

2.3.1.8. Zamanlayıcı Modülleri

PIC18F452, 4 adet zamanlayıcı modülüne sahiptir. Zamanlayıcı 0, 8-bit veya 16-bit zamanlayıcı ya da sayıcı olarak kullanılabilir. Zamanlayıcı değeri her saat frekansında artabileceği gibi bir katsayı sayesinde birkaç saat frekansında da artabilmektedir. Zamanlayıcı 1, 16-bit zamanlayıcı ya da sayıcı olarak kullanılabilirken karşılaştırma-yakalama modülü vasıtasıyla sıfırlanabilmektedir. Zamanlayıcı 2, 8-bit zamanlayıcı ve 8-bit periyot yazmacı olarak kullanılabilir. Zamanlayıcı 3 ise 16-bit zamanlayıcı ya da sayıcı olarak kullanılabilmekle birlikte karşılaştırma-yakalama modülü sayesinde sıfırlanabilmektedir.

2.3.1.9. Analog Sayısal Dönüştürücü

PIC18F452, 10-bit çözünürlükte 8 kanal analog sayısal dönüştürücüye sahiptir. Belirlenen değer aralıkları içerisindeki analog sinyalleri 10 bit sayısal verilere çevirir.

2.4. İşletim Sistemleri

İşletim sistemleri, tasarlanan donanımı kullanılabilir yapan, sistem kaynaklarının paylaşımını ayarlayan, uygulama geliştirme ortamı hazırlayan ve tüm bunların yönetimini sağlayan çekirdekten oluşan yazılımlardır.

İşletim sistemleri donanımın kullanım zorluklarını bir anlamda gizlemekte ve güvenlik nedeniyle doğrudan donanıma erişmeyi engellemektedir. Bunun yerine donanım ile yazılım arasında köprü görevini üstlenen sistem çağrıları kullanılmaktadır. Sistem çağrıları kütüphaneler şeklinde tanımlanır ve kullanıcı programları bu kütüphaneleri kullanarak donanıma erişirler [12,13].

İşletim sisteminin en önemli işlevlerinden birisi kaynak yönetimidir. Bir sistemin ana kaynakları genel olarak işlemci birimi, bellek, giriş/çıkış aygıtları ve veri olarak sınıflandırılabilir. Kaynakları yöneterek en verimli şekilde kullanılmalarını sağlayan işletim sistemi sistemin her an kullanılabilir olmasını amaçlamaktadır [14].

İşletim sistemleri, kaynakların görev veya kullanıcılara atanması ve gerektiğinde paylaştırılması, görevlerin işleyiş sırasının belirlenmesi, giriş-çıkış işlemlerinin yerine getirilmesi gibi işlevleri üstlenirler.

İşletim sistemi başlatıldığında genelde çekirdekteki program sistem belleğine alınır. Buradan ihtiyaç duyulan diğer programlar ana belleğe taşınır. Bellekte devamlı yer alan programlar yerleşik programlardır. Bu programlar tüm diğer birimleri denetler ve yönetirler. Diğer programlar gerektikçe belleğe alınırlar. Yerleşik programlar meydana gelen kesmeleri ele alır, işlemleri ve kaynakları yönetir, sistem durumunu ve verileri kontrol ederler.

Kaynak yönetimi, hangi kaynağın, kim tarafından ne zaman kullanılacağını kontrol etme işlemidir. Özellikle çok kullanıcılı sistemlerde önem kazanan bu işlev aynı anda kaynaklara erişmek isteyen kullanıcıların erişimini en iyi şekilde düzenler. Karşılıklı içerleme veya dışarlama (mutually inclusive or exclusive) gibi yöntemlerle anlaşılarak kaynakların kullanıcılar tarafından ortak kullanımı sağlanır. Kesmeler ise zamanlayıcının taşması, bir giriş-çıkış ucunun seviye değiştirmesi, seri porttan veri

alınması gibi meydana gelen özellikli olaylar sonrasında program akışının önceden belirlenmiş bir bellek adresine yönlendirilmesi işlemidir. Önceden belirlenen bu adreste istenilen işlevi gerçekleştiren kodlama yapılır, bu kod çalıştırıldıktan sonra program akışı tekrar kaldığı yerden devam eder. Bahsi geçtiği gibi birçok kaynaktan meydana gelebilen kesmelerin aynı anda meydana geldiklerinde hangisinin işletileceğine karar verebilmek için öncelikleri vardır. Her kesme kaynağı kullanılan mimariye ve mikroişlemciye göre farklı bir önceliğe sahiptir ya da bu öncelikler yazılım seviyesinde kullanıcı tarafından tanımlanabilir [15].

İşletim sisteminin görevlerinden birisi de düzenli aralıklarla sistemi denetlemektir. Böylece meydana gelebilecek olağanüstü durumlar algılanarak çözümü sağlanabilir. İşlem ya da başka bir ifade ile görev, bir program bloğundan oluşan ve belli bir işlevi yerine getiren kod parçasıdır. İşletim sistemleri genelde birden fazla programı aynı anda çalıştırabilecek yapıda tasarlanır. Daha doğrusu işlemci aynı anda aslında sadece bir programı çalıştırmaktadır, ancak belli zaman aralıkları ile diğer programları da çalıştırması hepsini aynı anda çalıştırıyor izlenimi vermektedir. İşletim sistemlerinde görevler arasındaki geçişleri çekirdeğin bir parçası olan çizelgeleyici yerine getirir. İşletim sistemi bunların yanında verilerin girişi, girilen verilerin belleğe aktarılması, oradan da çıkış ünitelerine gönderilmesi gibi görevleri de yerine getirir [16].

2.4.1. İşletim Sistemi Türleri

2.4.1.1. Ana Çatı (Main Frame) İşletim Sistemleri

Yoğun giriş/çıkış işlemi gerektiren çok sayıda görevin bulunduğu işletim sistemleridir. Örnek : OS/390

2.4.1.2. Sunucu İşletim Sistemleri

Sunucu makineler üzerinde çalışan ağ üzerinden çok sayıda kullanıcıya hizmet veren işletim sistemleridir. Örnek: Unix, Windows 2000

2.4.1.3. Çok İşlemcili İşletim Sistemleri

Çok işlemcili bilgisayarlarda, paralel sistemlerde, birden fazla birbirine bağlı bilgisayar sistemlerinde kullanılan işletim sistemleridir.

2.4.1.4. Kişisel Bilgisayar İşletim Sistemleri

Genellikle ofis veya ev kullanıcıları için kolay ve etkin kullanım sağlayan işletim sistemleridir. Örnek: Windows 98, 2000, XP, MacOS, Linux.

2.4.1.5. Gerçek Zamanlı İşletim Sistemleri

Zaman kısıtlarının önemli olduğu sistemlerde, endüstriyel kontrol sistemlerinde kullanılan işletim sistemleridir. Örnek: VxWorks, QNX

2.4.1.6. Gömülü İşletim Sistemleri

Avuç içi bilgisayarlar, TV, cep tel gibi özel amaçlı kullanılan işletim sistemleridir. Örnek: PalmOS, Windows CE

2.4.1.7. Akıllı Kart İşletim Sistemleri

Kredi kartı boyutlarında üzerinde işlemci bulunan bir veya birkaç işlev yüklü çoğunlukla özel sistemler için tasarlanmış işletim sistemleridir [17].

Gömülü sistemlerde genelde zaman kısıtları ön plana çıktığı bu projede için gerçek zamanlı işletim sistemleri ele alınmıştır.

2.4.2. Gerçek Zamanlılık Kavramı

Gerçek zamanlılık, işletimin başarısının zamana da bağımlı olmasıdır. Daha önce de belirtildiği gibi sadece işin tamamlanması başarı olarak kabul edilmez. Daha önceden belirlenmiş zaman dilimi içerisinde bitirilmemişse, düzgün çalışmış olsa bile işletim başarısız kabul edilir. Bazı uygulama alanları; uzay uygulamaları, tıbbi uygulamalar, savunma sanayi sistemleri, endüstriyel otomasyon sistemleri olarak sayılabilir [18].

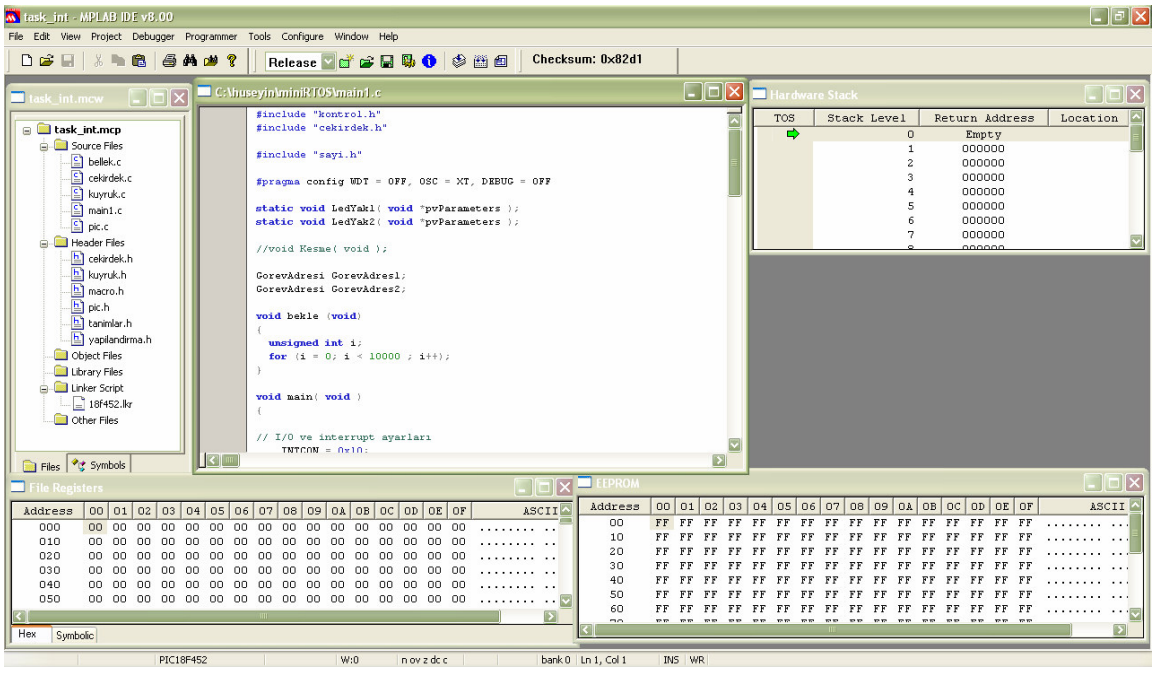
BÖLÜM 3

3. GELİŞTİRME ARAÇLARI

Bu projenin geliştirilmesi esnasında geliştirme ortamı olarak Microchip MPLAB IDE, derleyici olarak MPLAB PIC C18, seri programlayıcı olarak JDM programlayıcısı, USB programlayıcı olarak Bren8ner programlayıcısı, bilgisayar yazılımı olarak ICProg ve USBurn yazılımları kullanılmıştır.

3.1. MPLAB Yerleşik Geliştirme Ortamı

MPLAB yerleşik geliştirme ortamı, Microchip firmasının PIC ve dsPIC ürünleri ile gömülü uygulamalar geliştirilmesine izin veren, her PIC için tasarlanan komut setlerini destekleyen ve ücretsiz olarak dağıtılan bir araç setidir. 32-bit bir Windows uygulaması olarak çalışan MPLAB IDE, birçok ücretsiz araç ile kullanılabilmekte olup çok güçlü bir hata ayıklama modülüne sahiptir (Şekil 3.1). Kullanışlı bir grafik arayüze sahip MPLAB IDE'nin kullanımı oldukça basittir [19].



Şekil 3.1. MPLAB Yerleşik Geliştirme Ortamı

3.2. MPLAB PIC C18 Derleyicisi

MPLAB, PIC için tasarlanmış çevirici dili (assembly) destekler. C gibi yüksek seviyeli dillerde uygulama geliştirmek için farklı seçenekler mevcuttur. MPLAB C18, Microchip tarafından PIC18 ve daha gelişmiş serilerde kullanılmak üzere geliştirilmiş ve oldukça iyi optimize edilmiş bir C derleyicisidir. Ticari uygulamalar için ücretli olan bu yazılımın öğrenciler için hazırlanmış sürümü 3 ay denenebilmektedir. Bu çalışma, 3 aylık deneme sürümü ile gerçekleştirilmiştir. C18, ANSI X3.159-1989 standardından türetilmiş bir derleyici olup tamamen PIC mimarisinin gelişmiş özelliklerini en iyi şekilde kullanacak şekilde tasarlanmıştır. 32-bit bir Windows konsol uygulaması olan C18, MPLAB tümleşik geliştirme ortamı ile birlikte kullanılabilmektedir (Şekil 3.2) [20].

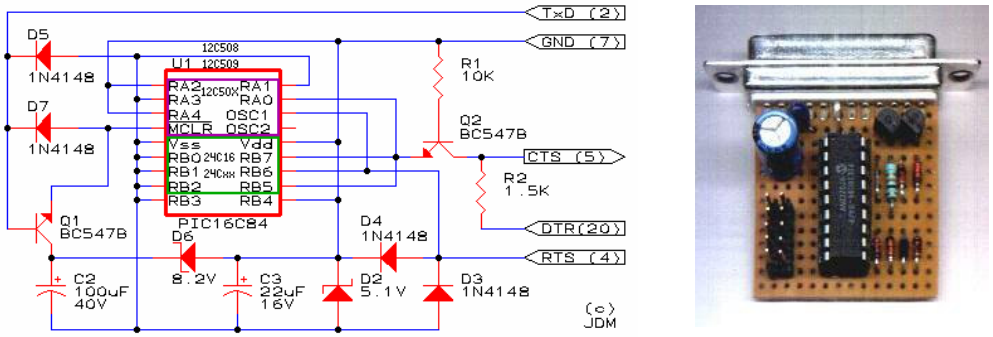
MPLAB C18 derleyicisi temel olarak aşağıdaki özelliklere sahiptir:

- ANSI '89 uyumluluğu
- MPLAB tümleşik geliştirme ortamı ile birlikte oldukça pratik kullanım
- Kod seviyesinde hata ayıklama
- PWM, SPI, I^2C, UART, USART, matematik kütüphanelerini kapsayan geniş bir kütüphane desteği
- Gerektiğinde çevrimiçi kullanılabilen çevirici dil (assembly) desteği

Şekil 3.2. PIC C18 mcc komutu

3.3. JDM Seri PIC Programlayıcısı

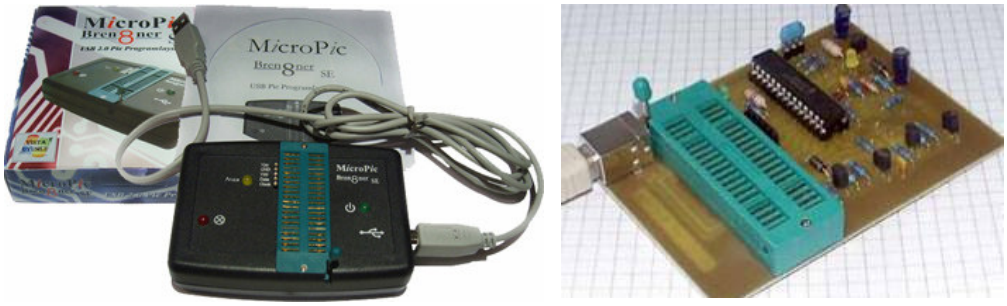
JDM, PIC mikrodenetleyicileri, 24CXX ve 24LCXX tipi I^2C EEPROM'ları çok kolay şekilde programlayan, yapılması son derece basit bir PIC programlayıcısıdır. Devre ilave bir güç kaynağı gerektirmemekte, gereken enerjiyi seri porttan almaktadır. Aşağıda bu devrenin donanım şeması yer almaktadır (Şekil 3.3) [21].



Şekil 3.3. JDM Seri PIC Programlayıcısı

3.4. Bren8ner USB PIC Programlayıcısı

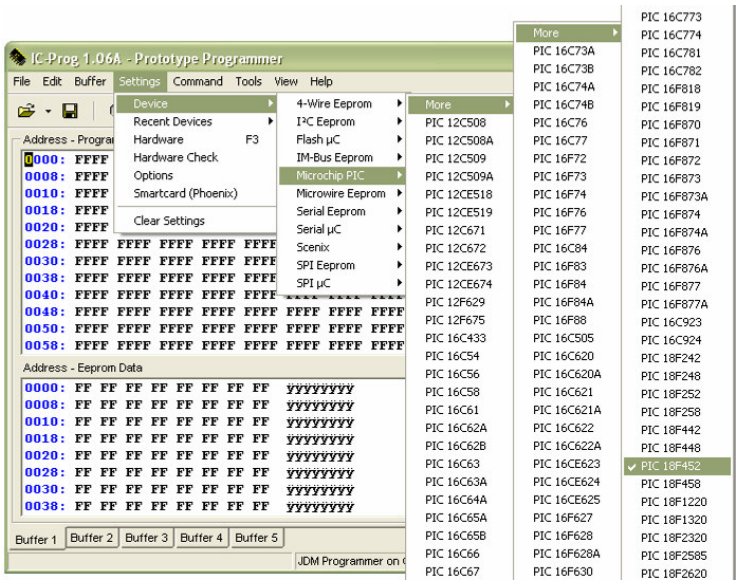
Bren8ner, PIC mikrodenetleyicileri sorunsuzca, bir kaç adımda programlayan üstelik USB2.0 hızıyla çalışan bir programlayıcıdır. Üzerinde bulunan ZIF soket yardımı ile kolayca PIC mikrodenetleyiciler programlayıcıya yerleştirilip çıkartılabilir. ZIF soket sayesinde hassas olan entegre bacaklarına hiçbir zarar gelmez. Ayrıca üzerinde bulunan ICSP bağlantı noktası ile ZIF sokete uymayan veya devreden çıkartılmak istenmeyen PIC mikrodenetleyiciler kolayca programlanabilir. Bu ürün Microchip firmasının ürettiği 173 ayrı model PIC programlayabilmektedir. Programlanabilecek PIC ler; 12F6xxx serisi, tüm 18Fxxx ve 18Fxxxx, tüm 16Fxx ve 16Fxxx ve dsPIC30Fxxxx serileridir [22,23]. Minimum sistem gereksinimleri; PII 233Mhz, 64MB Ram, 10MB boş alan, boş USB bağlantı noktası, Windows (98/me/NT/2000/XP/VISTA) işletim sistemidir (Şekil 3.4).



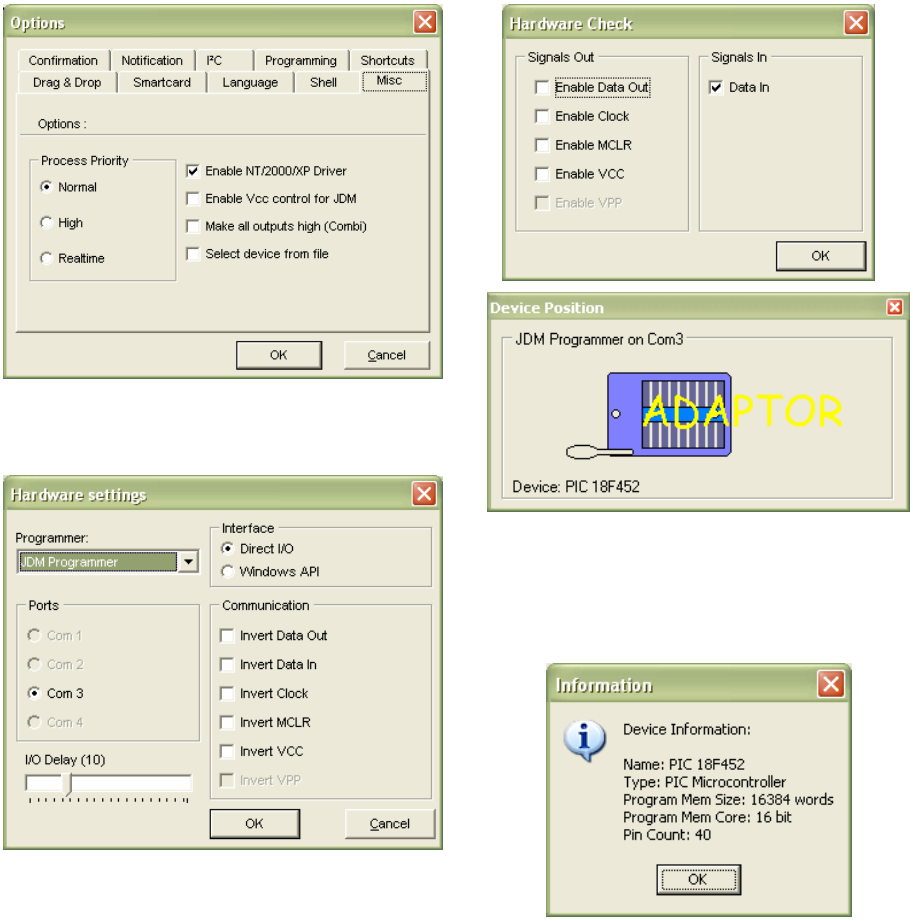
Şekil 3.4. Bren8ner USB PIC Programlayıcısı

3.5. ICProg Programlama Yazılımı

ICProg, yazılan programın derlenmesi ile meydana çıkan hex uzantılı dosyaların mikrodenetleyici, EEPROM gibi entegre devrelere yüklenmesi için JDM ve benzeri birkaç donanım ile çalışan yazılımdır (Şekil 3.5) [24].



Şekil 3.5. ICProg yazılımı



Şekil 3.6. ICProg kullanıcı seçimleri

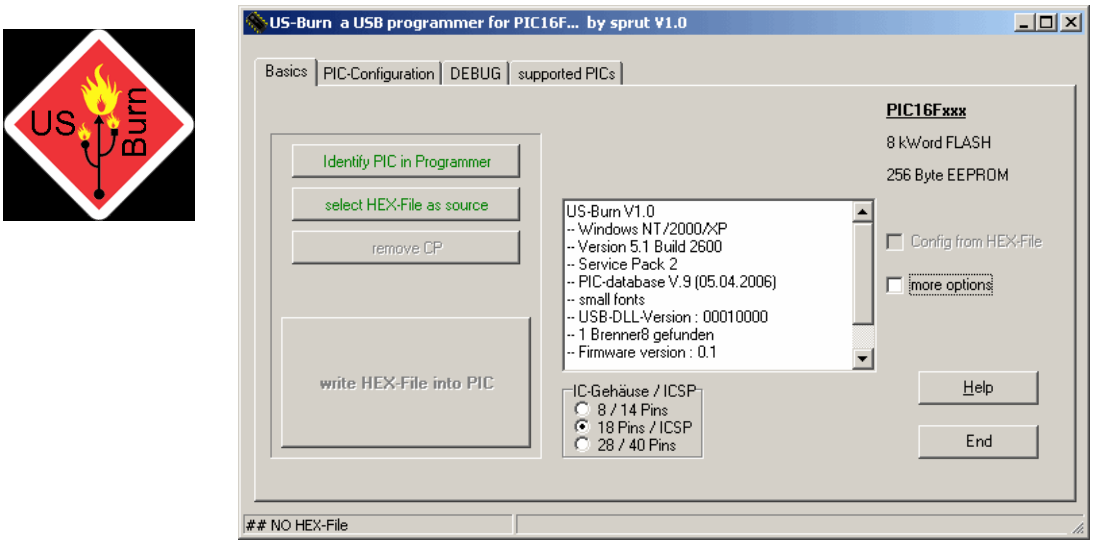
ICProg, menülerinde yer alan ayarlar sayesinde bazı seçimlere izin verir (Şekil 3.6). Bu opsiyonlar sayesinde kullanılan COM port seçilebilir, programlama uçları evrilebilir, donanım denetimi yapılabilir veya kullanılacak sürücü etkinleştirilebilir. ICProg aşağıda listelenen programlayıcı donanımlarını desteklemektedir:

- JDM Programlayıcı (Ludipipo)
- Conquest Programlayıcı
- TAFE Programlayıcı
- TAIT 'Classic' Programlayıcı
- Parallel TAIT Programlayıcı

- Fun-card Programlayıcı
- SCHAER Programlayıcı
- ProPic II Programlayıcı
- STK200 Programlayıcı
- AN589 Programlayıcı
- WILLEPRO Programlayıcı
- Fluffy2 Programlayıcı
- DL2TM Programlayıcı
- ER1400 Programlayıcı

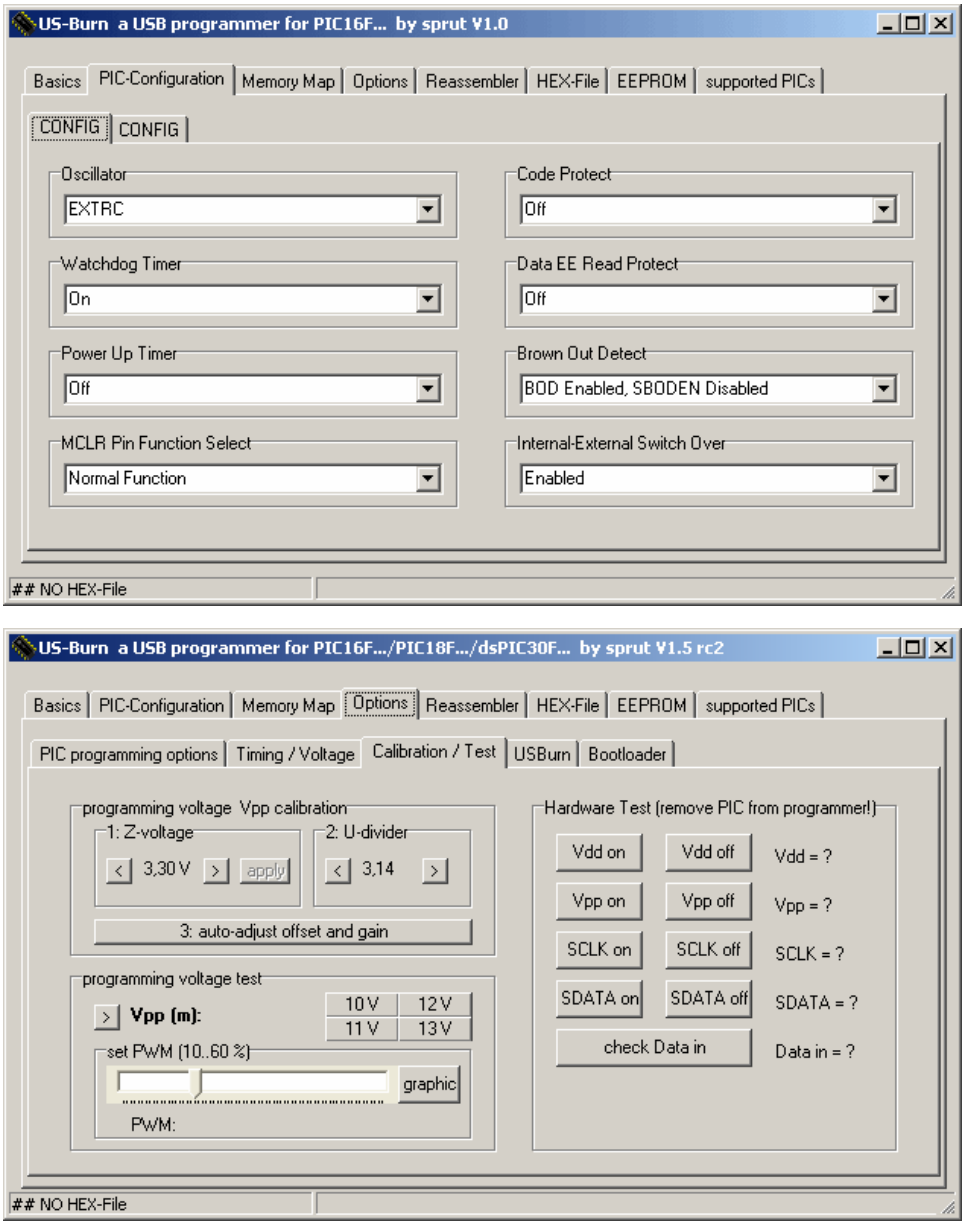
3.6. USBurn Programlama Yazılımı

USBurn, Bren8ner, Bren9ner, Bren8nerMini gibi donanımlar ile çalışan ve daha önceden derlenip hazırlanmış hex dosyaları seçip PIC üzerine yazmaya yarayan bir bilgisayar yazılımıdır (Şekil 3.7) [25].



Şekil 3.7. USBurn yazılımı

USBurn aşağıdaki gibi bir çok kişiselleştirme seçeneğine sahiptir (Şekil 3.8).



Şekil 3.8. USBurn kişiselleştirme seçenekleri

BÖLÜM 4

4. PIC Mikrodenetleyiciler İçin Gerçek Zamanlı İşletim Sistemi (PICOS)

4.1. PICOS - PIC Mikrodenetleyiciler İçin Gerçek Zamanlı İşletim Sistemi

Genel amaçlı işletim sistemlerinin gömülü sistemlerdeki kullanım kısıtları, gömülü sistemler için özel tasarlanmış gerçek zamanlı işletim sistemi geliştirilmesi ihtiyacını doğurmuştur. Bu noktadan hareketle, gömülü sistemlerde kullanılmak amacıyla tasarlanan PICOS, 8-bit PIC ailesinin en gelişmiş serisi olan PIC18Fxxx serisi ile 16-bitlik PIC24xxx ve 32-bitlik PIC32xxx serileri için geliştirilmiş gerçek zamanlı bir işletim sistemidir. PIC12xxx ve PIC16xxx serileri düşük hafıza boyutları, yazmaç limitleri gibi kısıtları nedeniyle ele alınmamıştır.

PICOS, geliştirilmesi esnasında istenirse çağrı üstünlüğü prensibi istenirse işbirlikçi prensip ile çalışabilecek şekilde tasarlanmıştır. Oldukça küçük, basit ve kullanımı son derece kolaydır. Tüm fonksiyonlar ve değişkenler Türkçe olarak tanımlanmıştır. Kodlar, MPLAB C18 derleyicisi ile tamamen ANSI C uyumlu yazılmıştır.

Oluşturulabilecek görev sayısı kullanılan cihazın hafızası ile sınırlıdır. Hafıza izin verdiği sürece istenilen sayıda görev oluşturulabilir. Kullanılabilecek öncelik seviyeleri programcı tarafından belirlenebilmektedir. Benzer şekilde yapılabilecek tanımlamalar ile yığıt büyüklükleri, görevleri durdurma ve erteleyebilme özellikleri kullanılıp kullanmayacağına karar verilebilmektedir. Çok düşük RAM, ROM bellek kullanımı ve işlemci yükü getirmektedir. Oldukça basit bir çekirdek yapısına sahiptir. Bu çalışmada iletiler, ileti kuyrukları ve semaforlar proje kapsamına alınmamıştır.

Bu aşamada aşağıdaki bölümlerde öncelikle PICOS'un de sahip olduğu bazı temel işletim sistemi kavramlarından söz edilecek ve PICOS işletim sisteminin bu özellikleri nasıl sağladığı belirtilecektir.

4.2. İşletim Sistemi Kavramları

4.2.1. Görev

Çoklu programlamalı ya da çoklu-işlemcili bir ortamda sistem tarafından bir iş öğesi olarak ele alınan bir ya da daha çok komut dizisidir. Aşağıda basit görev örnekleri verilmiştir:

- LCD ekran üzerinde bir mesaj göstermek
- Bir uyarı geldiğinde ışıklı ikaz vermek

4.2.2. Kesme

Program akışının durdurulup daha önceden belirlenmiş bir program kodunun (Interrupt Service Routine) çalıştırılmasına neden olan dâhili veya harici olaylardır. Örnek olarak;

- Bir butona basılması
- Bir zamanlayıcının taşması
- Bir giriş/çıkış ucunun durum değiştirmesi

verilebilir.

4.2.3. Görev Önceliği

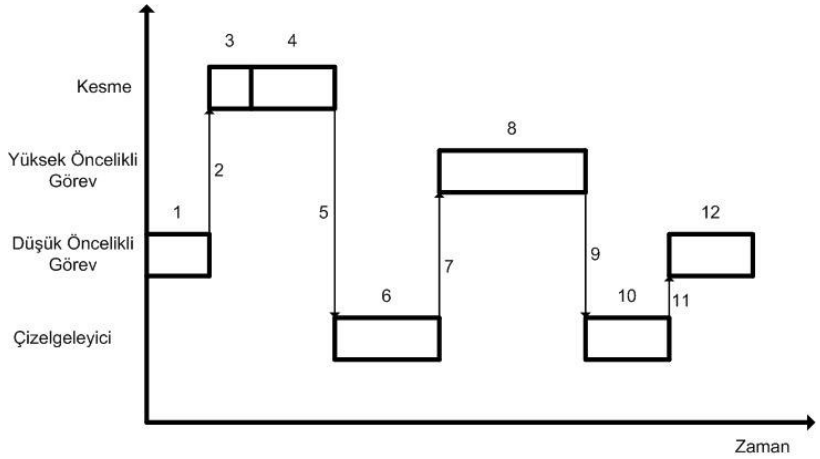
Görevlere farklı değerlerde öncelik atamak durumu ciddi anlamda değiştirmektedir. Böylece çalışma zamanında her an en önemli görevin çalışması garanti altına alınır. Görev önceliği, bir görevin diğerlerine göre önemini belirler. Değişken veya sabit olabilir. Öncelikler:

- Yüksek
- Orta
- Düşük
- Nümerik değerler

ile ifade edilebilir.

4.2.4. Çağrı Üstünlüğü Prensibiyle Çalışan Sistem (Preemptive)

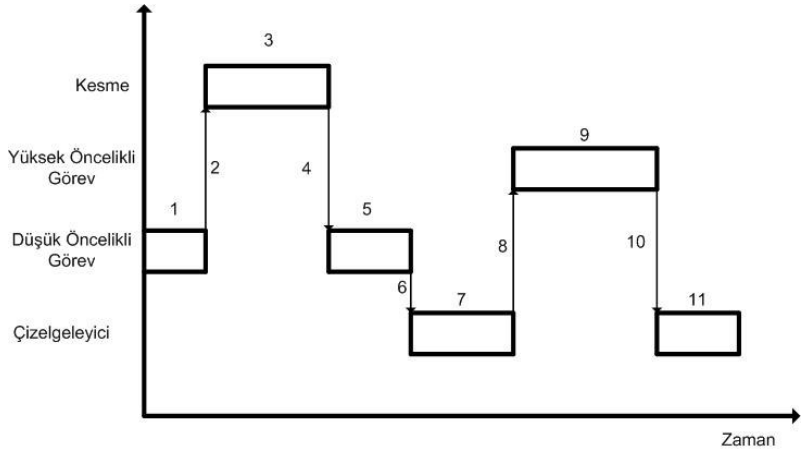
Bir görev kesme nedeniyle durdurulduğunda ve diğer görev çalışır hale geldiğinde programlayıcının herhangi bir komut kullanmasına gerek kalmadan program akışının diğer göreve geçmesidir. Görevler arası geçiş çizelgeleyici tarafından otomatik yapılır. Bu çalışma yöntemi Şekil 4.1'de gösterilmiştir.



Şekil 4.1. Çağrı üstünlüğü prensibine dayanan görev değişimi

4.2.5. İşbirlikçi Sistem (Cooperative)

İşletim sisteminin, arka planda beklemekte olan görevlere ancak ön planda yürümekte olan görevin boş zamanlarında işletim şansı verdiği çoklu görevli işletim şeklidir. İşbirlikçi sistemde görevler arası geçiş programcının kontrolündedir (Şekil 4.2).



Şekil 4.2. İşbirlikçi görev değişimi

Şekil-4.1 çağrı üstünlüğü yöntemi ile çalışan bir çizelgeleyicinin çalışmasını göstermektedir. {1} anında düşük öncelikli bir görev çalışırken {2} anında bir kesme meydana gelir ve {3} süresi boyunca tüm yazmaçları ve yığıt değerlerini de kapsayan içerik saklanır. {4} süresince kesmede tanımlanan komutlar işletilir. Bu arada daha yüksek öncelikli bir görev çalıştırılabilir konuma gelmiş ve beklemeye başlamıştır. Kesme rutini biter bitmez {5} anında çizelgeleyiciye geçiş yapılır. Çizelgeleyici {6} anında çalıştırılabilir tüm görevleri kontrol eder ve uygun konumdaki en yüksek öncelikli göreve geçiş yapar{7}. Yüksek öncelikli görev {8} süresi boyunca tanımlanmış komutlarını işletir ve biter bitmez {9} anında çizelgeleyiciye döner. Çizelgeleyici {10} anında, daha önce düşük öncelikli görev için saklamış olduğu içeriği geri yükler ve kaldığı yerden devam etmesi için düşük öncelikli göreve {11} anında geri döner. {12} boyunca düşük öncelikli görev işletimini tamamlar. Çizelgeleyici benzer şekilde çalışmasına devam eder [26].

Şekil-4.2 ise işbirlikçi görev değişimi yöntemi ile çalışan bir çizelgeleyiciyi temsil etmektedir. Bir önceki örnekte olduğu gibi düşük öncelikli bir görev çalışırken {1} meydana gelen kesmeden{2,3} sonra yüksek öncelikli göreve geçiş sağlanır. Ancak aradaki fark kesme rutini işletilir işletilmez geçiş yapılmaz. Kesme tamamlandığında program akışı kaldığı yerden yani düşük öncelikli görevden{4,5} devam eder. Düşük öncelikli görev tamamlanırken kullanıcının verdiği bir komut ile çizelgeleyiciye geçiş yapılır{6,7} ve çizelgeleyici yüksek öncelikli göreve program akışını devreder{8,9}. Yüksek öncelikli görev işletimi tamamladığında tekrar çizelgeleyiciye döner{10,11} ve program akışı bu şekilde devam eder.

4.2.6. Çekirdek

İşletim sisteminin donanım özkaynaklarını atamak gibi en temel görevleri yerine getiren kısmıdır. Sistemin çağrı üstünlüğü ile ya da işbirlikçi yöntem ile çalıştırılması çekirdek tarafından yürütülür. Çekirdek, o anda çalışan görevin ya da bir sonraki çalışacak görevin en yüksek öncelikli görev olmasını garanti eder.

4.2.7. Olay

Bir görev tarafından beklenen bir durumun gerçekleşmesidir. Programın herhangi bir yerinde olayın gerçekleştiği duyurulabilir, böylece programın diğer kısımları haberdar edilebilir.

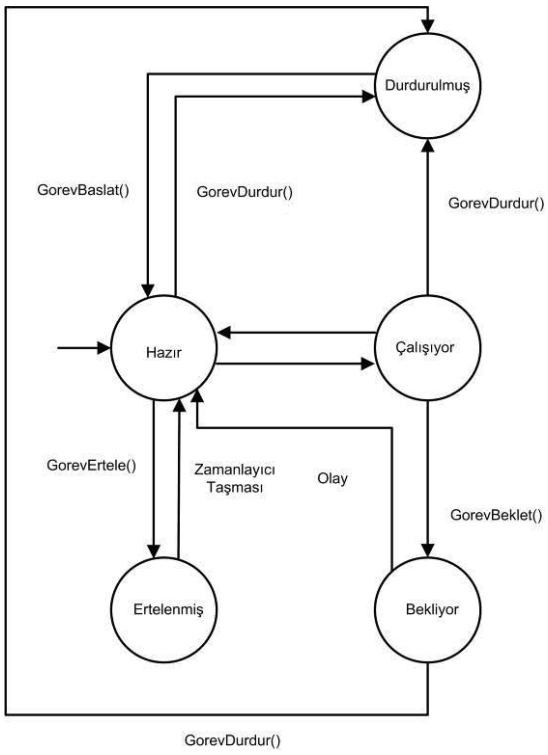
- Butona basılması
- Hata oluşması
- Ortak kullanılan bir kaynağın boşa çıkması
- Zamanlayıcının taşması
- Bir RS-232 karakterin alınması veya iletilmesi

4.2.8. Görev Durumu

Sistemde ele alınmış bir görevin o anda ne yaptığını belirtir.

- **Çalışıyor:** Herhangi bir anda program akışını devam ettiren, kendi işletimini sürdüren görev çalışıyor konumdadır. Aynı anda sadece bir tek görev çalışıyor konumda olabilir.
- **Hazır:** Herhangi bir anda her an çalışabilir konumda olan görevlerdir. Daha açık bir ifadeyle beklemeye alınmış, ertelenmiş, durdurulmuş ve çalışıyor konumda olmayan, her an çalışmaya hazır halde bekleyen görevlerdir.
- **Ertelenmiş:** Daha önceden çalışan ancak durdurulan ve tekrar çalışmak için bir zamanlayıcının taşmasını bekleyen görevlerdir. Ayarlanan süre tamamlandığında görev tekrar hazır konuma geçer.
- **Bekliyor:** Durdurulmuş ve çalışmak için bir olayın meydana gelmesini bekleyen görevlerdir.
- **Durdurulmuş:** Daha önceden çalışan fakat bir daha başlatılana kadar durdurulmuş konumdaki görevleri ifade eder.

Görev durumları Şekil 4.3'te gösterilmiştir.



Şekil 4.3. Görev durumları

4.2.9. İşletim Sistemi

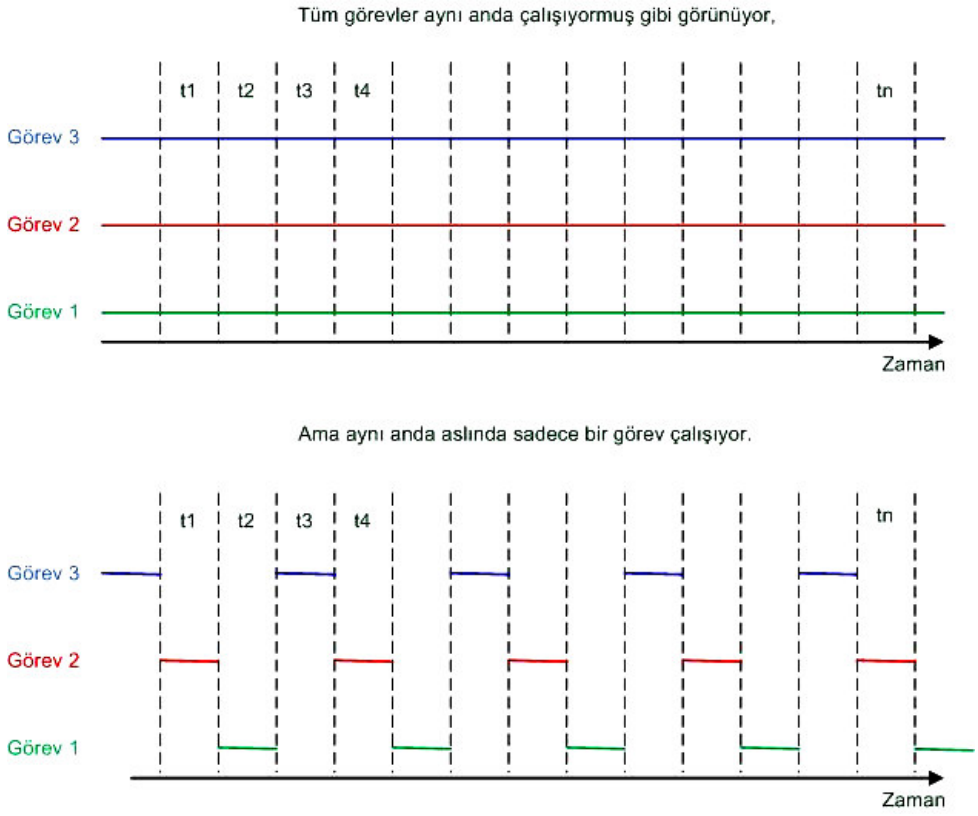
Görevleri ve olayları yürütmek için çekirdeği, zamanlayıcıları ve tüm ilgili yazılımları diğer bir deyişle servisleri barındırır. Gerçek zamanlı işletim sistemleri, kritik işlemlerin belli bir zaman aralığında ve doğru bir şekilde yapılmasını sağlayan sistemlerdir. Gerçek zamanlı işletim sistemleri ile geliştirilen uygulamalar ise tüm görevlerin, veri yapılarının, kesmelerin ve servislerin bulunduğu uygulamalardır.

4.2.10. Çoklu Görev Yürütümü

Çoklu görev yürütümünden bahsedebilmek için eş zamanlı çalışır görünen birden fazla görevin ve bu görevlerin arasındaki geçişlerde işlemci kontrolünü ve kaynak yönetimini yürüten bir mekanizmanın bulunması gerekir. Bu görevi PICOS üzerinde çizelgeleyici üstlenir. Belli gereksinimler karşılandığında bir görevi durdurup diğerini çalıştırır. Gerçek zamanlı bir işletim sisteminin çoklu görev yönetiminin en iyi şekilde kullanılması için görevlerin, işlemcinin herhangi bir zaman diliminde işlem gücünü en iyi kullanacak şekilde planlanması ve tanımlanması gerekir.

Görevler uygun şekilde tanımlandığı takdirde geriye kalan işi çizelgeleyici yürütecektir.

Çoklu görev yürütümü sayesinde işletim sistemi, karmaşık uygulamaların daha küçük ve yönetilebilir görevlere bölünebilmelerine izin verir. Böyle parçalara ayırma, yazılımın test edilmesini, iş bölümü yapılabilmesini ve kodun yeniden kullanılabilmesini kolaylaştırır. Geleneksel yazılımlardaki zamanlama ve sıralama problemleri uygulama kodundan kaldırılabilir, bu sorumlulukları artık işletim sistemi üstlenir.



Şekil 4.4. Çoklu görev yürütümü

Geleneksel işlemciler aynı anda sadece bir görev çalıştırabilir ancak çoklu görev yürüten bir sistemin bu görevler arasında çok hızlı bir biçimde geçiş yapması aynı anda hepsi çalışıyormuş gibi görünmesini sağlar. Şekil 4.4'te her görev farklı bir renkle gösterilerek bu durum sembolize edilmiştir [27].

4.2.11. Çizelgeleme

Çizelgeleyici çekirdeğin bir parçası olup herhangi bir anda hangi görevin çalışacağına karar veren bileşendir. Bir görev, çekirdek tarafından durdurulabileceği gibi kendisini durdurmayı da seçebilir. Bu durum, görev bir süre beklemeye geçtiğinde, bir olayın meydana gelmesini beklemeye başladığında ortaya çıkar. Durdurulmuş veya bekleyen bir görev işlemeyeceği için herhangi bir işlemci gücü harcamaz.

4.2.12. Görevler arası Değişim

Sistemin birden fazla görevi yürütebilmesi amacıyla birinden öbürüne geçerek her birini kısa sürelerle çalıştırmasıdır. Bu esnada tüm global saklayıcılar ve yığıt değerleri daha sonra kullanılmak üzere depolandığı için "içerik geçişi" olarak da adlandırılır.

Bir görev çalışırken işlemci gücü tüketir, yazmaçlara, RAM ve ROM belleğe erişir. Her programın bu kaynakları kullanabileceği düşünülerek görevler arası değişim esnasında tüm bu kaynaklar saklanır ve bu kaynakların tümü kısaca içerik olarak adlandırılır. Bir görev birkaç komutun bir araya gelmesiyle oluşur. Çekirdeğin bir görevi ne zaman durduracağı, ne zaman tekrar çalıştıracağı belli değildir. İki yazmacı toplayan aşağıdaki komut dizisini ele alalım.

```
{1}   addlw 0x7D
{2}   addwf YAZMAC1, 1
{3}   addlw 0x3B
{4}   addwf YAZMAC2, 1
{5}   movwf YAZMAC1
{6}   addwf YAZMAC2,1
```

{5} numaralı komutun işletilmesinden hemen sonra tam {6} numaralı komut işletilmeden çekirdeğin bu komut dizisine sahip olan görevi durdurduğunu ve başka bir göreve geçiş yapacağını düşünelim. Çalışacak olan görev, global olarak tanımlanan bu yazmaç değerlerini değiştirebilir. Bu durumda durdurulan görev yazmaç değerlerinin değiştiğini bilemeyeceği için tekrar çalışır hale geldiğinde değişen yazmaç değerlerleri ile yapılan toplama işlemi hatalı sonuç verecektir. Aynı

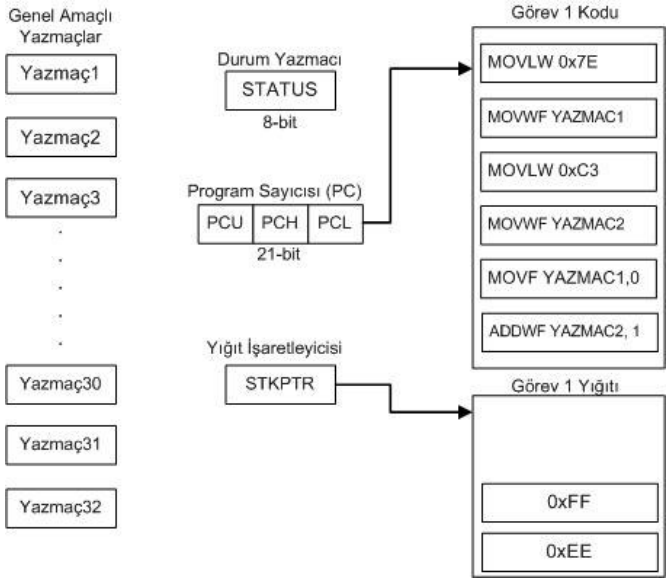
durum yığıta depolanan ve geri yüklenen değerler için de geçerlidir. Bu nedenle işletim sistemi çekirdeği her göreve ait içeriği durdurulduğunda ya da daha doğru bir deyişle arka plana alındığında saklamak ve tekrar çalıştırıldığında geri yüklemekle yükümlüdür. Bu işlem görevler arası değişim olarak adlandırılır.

4.2.13. Gerçek Zamanlı Uygulamalar

Gerçek zamanlı işletim sistemlerinin amacı, diğerleri gibi sadece istenilen fonksiyonun başarı ile çalışması değil belli bir zaman dilimi içerisinde sonuçlanmasıdır. Örneğin bir savunma sisteminde bir hedefe kilitlenmiş bir füzeyi hedefe varmadan önce imha edecek sistemin gerçek zamanlı olması gerekir. Hedefe ulaştıktan sonra füzenin imha edilmesinin bir anlamı kalmayacaktır. Gerçek zamanlı bir işletim sistemi gereken süre içerisinde cevap vermeli, çekirdek de zaman kısıtlarını sağlamalıdır.

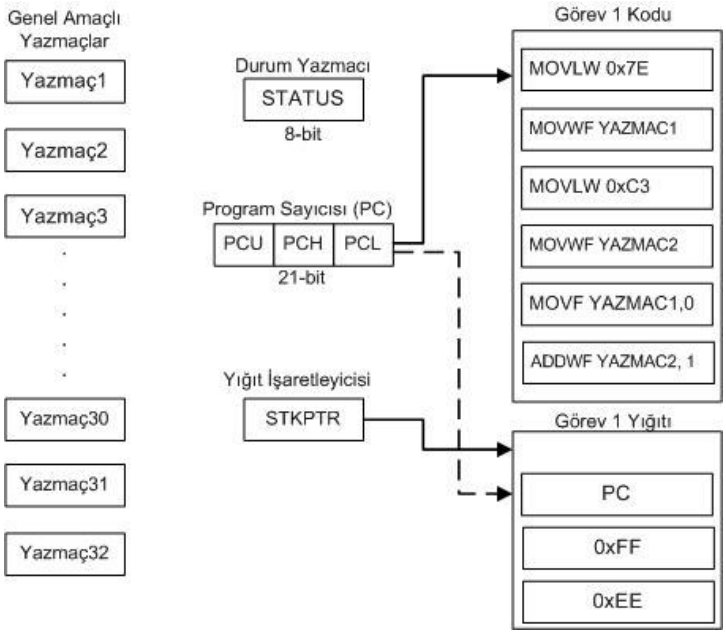
4.3. İşletim Sistemi Uygulaması

Uygulama aşağıda basit bir örnek ile açıklanacaktır. İlk konumda Görev 1 çalışmaktadır. Görev 2 ise daha önceden çalışıyorken durdurulmuş ve bu göreve ait tüm içerik saklanmıştır. Şekil 4.5'te işletim sistemi kesmesinden önce içerik durumları gösterilmektedir. Yazmaçlar Görev 1'e ait içerik değerlerini tutmaktadır.



Şekil 4.5. Görev 1 çalışıyorken içerik

İşletim sistemi kesmesi gerçekleştiğinde tam Görev 1 "MOVLW 0x7E" komutunu işletirken Görev 1'e ait program sayıcısı değerini aynı göreve ait yığıt içerisine attıktan sonra işletim sistemi kesmesinin başlangıç koduna atlar (Şekil 4.6).



Şekil 4.6. Program sayıcısı değerinin kesme tarafından yığıta atılması

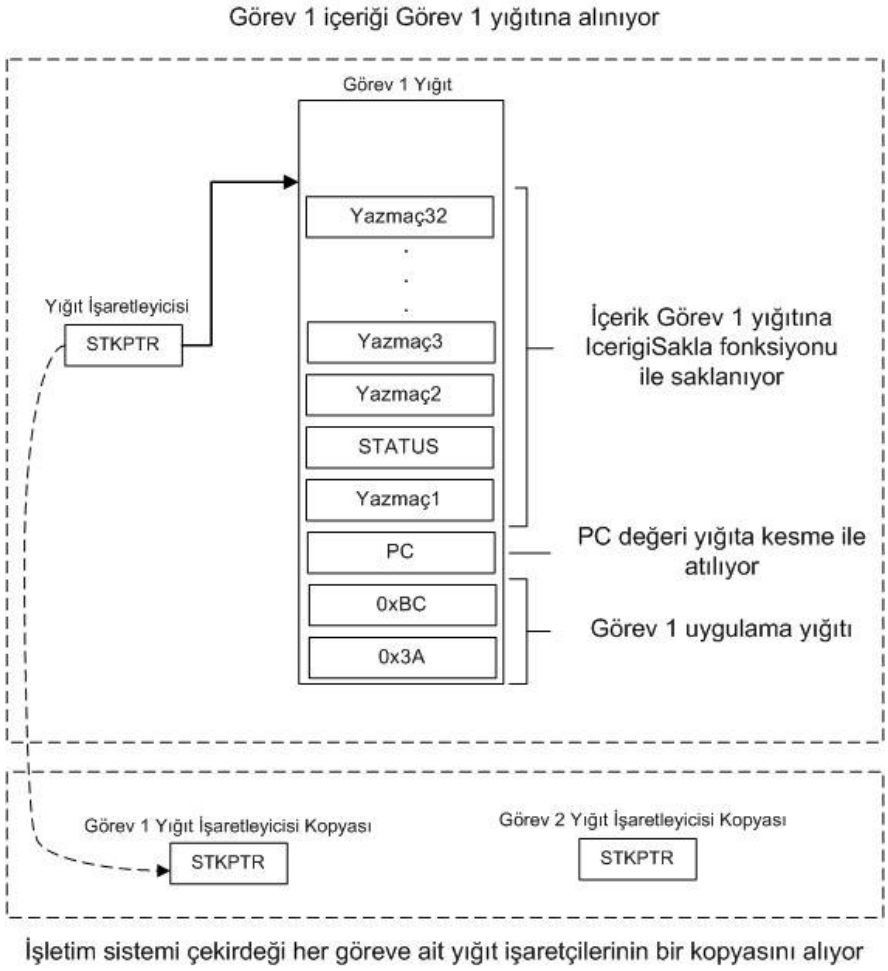
İşletim sistemi kesmesinin kodu ise aşağıdakine benzer olmalıdır.

```
/* Isletim sisteminin periyodik kesmesi (tick interrupt) icin komut
dizisi. */
void SayilariTopla( void )
{
    GorevDevret();
    asm volatile ( "retfie" );
}
/*-------------------------------------------------*/
void GorevDevret( void )
{
    IcerigiSakla();
    TikleriArttir();
    GorevDegistir();
    IcerigiGeriYukle();
    asm volatile ( "return" );
}
/*-------------------------------------------------*/
```

SayilariTopla() fonksiyonu GorevDevret() fonksiyonunu çağırır. GorevDevret() fonksiyonu IcerigiSakla() fonksiyonu ile tüm içeriği Görev 1'in yığıtına yükler.

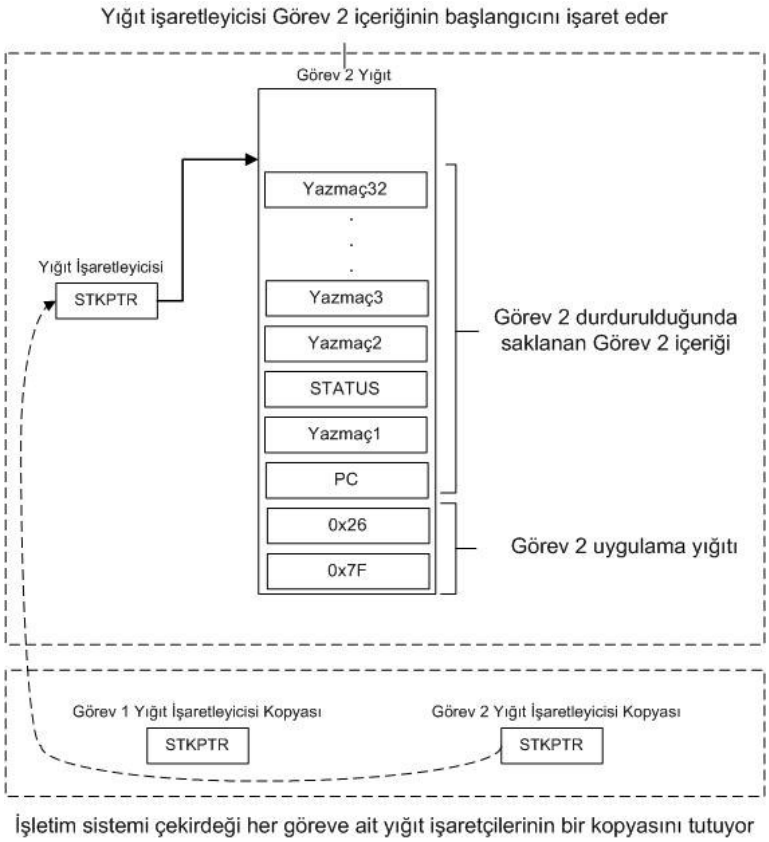
Gorev 1'e ait yığıt işaretçisi kendi içeriğinin en başında bulunan adresi gösterir. İcerigiSakla() görevini yığıt işaretleyicisinin gösterdiği adresi yedekleyerek tamamlar (Şekil 4.7).

Daha önceden çalıştırılıp durdurulmuş olan Görev 2'nin yığıt işaretleyicisinin gösterdiği adresin kopyası çekirdek tarafından tutulmaktadır.



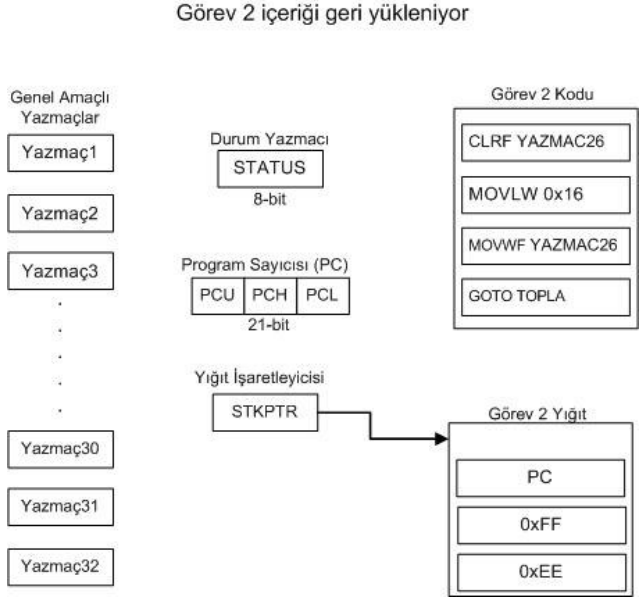
Şekil 4.7. Görev 1 içeriğinin Görev 1 yığıtına atılması

TikleriArttir() fonksiyonu Görev 1 içeriği saklandıktan sonra çalışır ve bu tikler belli bir sayıya ulaşınca Görev 2 aktif duruma gelir. Görev 2, Görev 1'den daha yüksek önceliğe sahip olduğundan GorevDegistir() fonksiyonu, görevleri kontrol ettiğinde işlemci gücünü kullanma hakkını Görev 2'ye verir (Şekil 4.8).



Şekil 4.8. Görev 2 içeriği

İcerigiGeriYukle() fonksiyonu ise daha önceden saklanmış olan Görev 2 içeriğini ilgili işlemci yazmaçlarına yükler. Sadece program sayıcısı yığıtta kalır (Şekil 4.9).



Şekil 4.9. Görev 2 içeriğinin geri yüklenmesi

GorevDevret() işini bitirdikten sonra kaldığı yerden devam etmesi için SayılarıTopla() fonksiyonuna geri döner. Bu fonksiyonun kaldığı yerde işletilecek ilk komut ise "RETFIE" komutudur. RETFIE komutu, yığıtta bulunan sıradaki değerin kesme meydana geldiğinde saklanan dönüş değeri olduğunu varsayar.

Sistem kesmesi meydana geldiğinde hemen Görev 1 dönüş adresi yığıtta saklanır. Ancak yığıt işaretçisi kesme sırasında değiştirilmiş ve Görev 2'yi göstermektedir. Bu nedenle "RETFIE" komutu yığıtdan dönüş adresini aldığında Görev 2 çalışmaya başlayacaktır.

Sistem kesmesi Görev 1'i çalışırken kesmiş, fakat devamında Görev 2'ye dönmüştür. Böylece görevler arasındaki geçiş tamamlanmıştır.

4.4. PICOS Yapılandırma Ayarları

Geliştirilen gerçek zamanlı işletim sisteminde yer alan yapılandırma ayarları sayesinde, sistem çağrı üstünlüğü prensibiyle ya da işbirlikçi prensip ile çalıştırılabilir. Ayrıca sistem kesmesi zamanı, saat frekansı, tanımlanabilecek en yüksek öncelik seviyesi, toplam hafıza büyüklüğü, öncelik atayabilme özelliği, öncelik okuyabilme özelliği, görev silme, görev durdurma, görev bekletme, görev erteleme gibi birçok sistem karakteristiği kullanıcı tarafından tanımlanabilir. Tüm bu yapılandırma ayarları ve detayları EK B'de yer almaktadır.

4.5. Bellek Kullanımı

İşletim sistemi çekirdeği, her görev oluşturulduğunda bu göreve bir RAM bellek atamaktadır. Çekirdek bellek kullanma ihtiyacı duyduğunda kullanılan bir fonksiyon sayesinde gerekli bellek ayrılmaktadır. Bellek kullanımını bellek.c dosyası içerisinde yer alan kod kontrol etmektedir. Oldukça basit olarak tasarlanan bellek yönetimi birçok görevi aynı anda kullanmaya elverişli şekilde tasarlanmıştır. Kullanılan algoritma bir dizi kullanmakta ve bu diziyi bellek kullanma isteği geldikçe alt gruplara bölmektedir. Toplam bellek büyüklüğü "Toplam_Hafiza_Buyuklugu" parametresi ile ayarlanabilmektedir. Algoritma rastgele olmayan bir yapıda

çalışmaktadır. Yani bir bloğa gidip dönmek her zaman eşit zaman almaktadır. Bir çok uygulama için bu bellek yönetim şeklinin ideal olduğu düşünülmektedir.

4.6. PICOS Fonksiyonları

PICOS'a işletim sistemi karakteristiği, bir dizi fonksiyon ile kazandırılmıştır. Bu fonksiyonlar ve açıklamaları EK C'de yer almaktadır.

4.7. Örnek Uygulama

Aşağıda PICOS ile gerçekleştirilen bir örnek uygulama ve ayrıntılı açıklamalar yer almaktadır.

```
(1)#include "kontrol.h"
(2)#include "cekirdek.h"
(3)#include "sayi.h"

(4)#pragma config WDT = OFF, OSC = XT, DEBUG = OFF

(5)static void LedYak1( void *pvParameters );
(6)static void LedYak2( void *pvParameters );

(7)GorevAdresi GorevAdres1;
(8)GorevAdresi GorevAdres2;

(9)void bekle (void)
{
(10)  unsigned int i;
(11)  for (i = 0; i < 10000 ; i++);
}

(12)void main( void )
{
(13)INTCON = 0x10;
(14)INTCON2 = 0x80;
(15)RCONbits.IPEN = 0;
(16)INTCONbits.GIEH = 1;
(17)PORTB = 0;
(18)TRISB = 1;
(19)GorevOlustur( LedYak1, "LED1", MINIMUM_YIGIT_BUYUKLUGU, NULL,
BosGorevOnceligi+2, &GorevAdres1 );
(20)GorevOlustur( LedYak2, "LED2", MINIMUM_YIGIT_BUYUKLUGU, NULL,
BosGorevOnceligi+1, &GorevAdres2 );
(21)CokluGorevBaslat();
}

(22)static void LedYak1( void *pvParameters )
{
(23)for( ;; )
  {
(24) bekle();
```

```
    {
(25)  PORTBbits.RB4 = !( PORTBbits.RB4 );
    }
(26)  bekle();
(27)  GorevDurdur( NULL );
  }
}

(28)static void LedYak2( void *pvParameters )
{
(29)for( ;; )
  {
(30)  bekle();
      {
(31)    PORTBbits.RB7 = !( PORTBbits.RB7 );
      }
(32) bekle();                                                    }
}

(33)void Kesme (void)
{
(34)if (INTCONbits.INT0IF)
   {
(35)INTCON = 0x50;
(36)KesmedenGorevBaslatma (GorevAdres1);
(37)bekle();
(38)INTCONbits.GIEH = 1;
(39)INTCONbits.INT0IF = 0;
    }
}
```

Örnek Uygulama

(1), (2) ve (3) satırlarında ön tanımlı #include komutu ile tanımlamaları içeren kontrol.h, cekirdek.h ve sayi.h dosyaları projeye eklenmiştir. (4) satırında ise kullanılan mikrodenetleyicinin konfigürasyon bitleri tanımlanmıştır. Bekçi köpeği zamanlayıcısı (Watchdog timer) pasif, osilatör tipi kristal ve hata ayıklama seçeneği de pasif olarak seçilmiştir.

(5) ve (6) satırlarında kullanılacak iki görevin ismi tanımlanmıştır. (7) ve (8) satırlarında ise bu iki göreve atanacak adresler seçilmiştir. (9), (10) ve (11) satırlarında bekle isimli bir fonksiyon tanımlanmış ve bu fonksiyon çağırıldığında program akışı 10 ms boyunca bekletilmiştir.

(12) satırında ana döngü tanımlanmıştır. (13), (14), (15) ve (16) satırlarında harici bir kesme tanımlanmış, bunun için kesme kontrol bitleri, öncelik atama bitleri ve

global kesme bitlerine gerekli değerler atanmıştır. Bu harici kesme harici bir buton aracılığı ile kullanılacaktır. (17) ve (18) satırlarında B portuna ait RB0 ucu giriş, diğer uçlar çıkış olarak ayarlanmış ve çıkış uçlarının başlangıç değerleri 0 olarak atanmıştır. (19) ve (20) satırlarında LedYak1 ve LedYak2 isimli iki görev oluşturulmuştur. Bu görevlerden ilki ikinci öncelik seviyesinde çalışırken diğeri birinci öncelik seviyesinde başlatılmıştır. Görevlere uygun yığıt büyüklükleri, açıklamalar ve adresler verilmiştir. (21) satırı ise çoklu görev yürütümünü başlatır. Böylece çekirdek çalışmaya başlar ve uygun durumdaki görevi çalıştırırken diğer görevlerin de durumlarını, öncelik seviyelerini kontrol eder.

(22) satırında LedYak1 görevi tanımlanmıştır. (23), (24), (24), (25), (26) ve (27) satırlarında ise bu görev içerisinde sonsuz bir döngüde önce program akışının insan gözüyle fark edilebilmesi için bir süre beklendikten sonra B portunun RB4 ucunun çıkış değeri değiştirilir ve tekrar edilen bir gecikme sonrasında bu görev durdurulmaktadır. GorevDurdur fonksiyonuna parametre olarak boş bir adres verildiği için kendisini durdurur.

(28) satırında ise LedYak2 görevi tanımlanmıştır. (29), (30), (31) ve (32) satırlarında sonsuz bir döngü içerisinde insan gözünün algılayabileceği frekansta B portunun RB7 ucunun çıkış değeri değiştirilmektedir.

(33) satırında kullanıcı tarafından tanımlanabilecek kesme rutini için ön tanımlı kesme fonksiyonu kullanılmıştır. (34), (35), (36), (37), (38) ve (39) satırlarında ise bu kesme fonksiyonu içerisinde önce kesme kaynağı kontrol edilmektedir. Kesme kaynağı RB0 ucunda kullanılan butonun 0 konumundan 1 konumuna geçişi değil ise kesme fonksiyonundan çıkılmaktadır. Eğer kaynak bu buton ise, önce tekrar kesme oluşabilmesi için gerekli bayraklar temizlenip gerekli kontrol bitleri ayarlandıktan sonra kesme içerisinden LedYak1 isimli görev başlatılmaktadır.

Uygulamaya bir bütün olarak bakıldığında, başlangıç aşamasında kullanılacak görevler oluşturulduktan ve gerekli tüm tanımlamalar yapıldıktan sonra gerçek

zamanlı çekirdek çoklu görev yürütümüne başlamaktadır. Bu aşamada düzenli periyotlar ile görevlerin hangi konumda oldukları ve öncelikleri kontrol edilerek hangi görevin çalışacağına karar verilmektedir. Bu örnekte daha yüksek önceliğe sahip olan birinci görev olan LedYak1 öncelikle çalışmakta ve tanımlanan içeriği gereği RB4 ucunun çıkış değerini değiştirerek kendi kendini durdurmaktadır. Bu durumda çekirdek bir dahaki kontrolü esnasında hazır görevlere baktığında sadece ikinci görev olan LedYak2 bulunduğundan program akışı bu göreve geçmektedir. Bu görevde sonsuz bir döngü içerisinde RB7 ucunun çıkış değeri belli aralıklarla değiştirilmektedir. Program akışı RB0 ucunda yer alan buton tarafından tetiklenen kesme oluşuncaya kadar bu şekilde devam eder. RB0 ucuna bağlı butona basıldığında buton 0 konumundan 1 konumuna geçer ve ön tanımlı kesme fonksiyonu devreye girer. Kesme içerisinde tanımlanan içerik gereği LedYak1 görevi kesme içerisinden tekrar çalıştırılır. Böylece çekirdeğin bir dahaki kontrolünde LedYak1 hazır konuma geçeceğinden ve bu görev daha yüksek önceliğe sahip olduğundan program akışı tekrar bu göreve geçecektir. Görüldüğü üzere görevler arası geçişlerde programcı herhangi bir komut kullanmadığı için sistem çağrı üstünlüğü prensibine göre çalışmaktadır. Tüm bu geçişler esnasında tanımlı tüm içerik saklanmakta ve görev çalışmaya başladığında geri yüklenmektedir. Her iki görev birden çalışabilir konumda olduğunda daha yüksek önceliğe sahip olan LedYak1 görevi çalışmaktadır.

BÖLÜM 5

5. SONUÇLAR VEYAPILACAK ÇALIŞMALAR

5.1. Sonuçlar

Bu çalışmada PIC18F452 mikrodenetleyicisi üzerinde özellikle gömülü sistemlerde kullanılmak üzere, istenildiğinde çağrı üstünlüğü istenildiğinde işbirlikçi prensip ile çalışabilen, gerçek zamanlı, oldukça küçük, basit ve kullanımı son derece kolay, tüm fonksiyonları ve değişkenleri Türkçe olarak tanımlanmış, çok düşük RAM, ROM bellek kullanımı ve işlemci yükü getiren ve oldukça basit bir çekirdek yapısına sahip bir işletim sistemi tasarlanmıştır. Bu işletim sistemi, yapılan çalışmalar sonrasında tavsiye edilen biçimde [28] bir dizi teste tabi tutulmuş ve çoklu görev yürütme, görevler arası geçiş, içerik saklama ve geri yükleme, öncelik tabanlı görev çalıştırma, çağrı üstünlüğü prensibi ile çalışma, işbirlikçi prensip ile çalışma gibi özellikleri bu testler ile doğrulanmıştır. Bu sayede programcı, geleneksel yöntemlerde olduğu gibi tek bir ana döngü içerisinde programı geliştirmek yerine uygulama işlevlerini görevlere atayabilir. Her görev içerisinde belli işlevler tanımlanır, bu görevler çalışma önceliklerine ve çalışma zamanlarına göre oluşturulup çalıştırıldığında işletim sistemi çekirdeği ve çizelgeleyici kontrolü sağlayacaktır. Olay güdümlü geliştirilecek uygulamalarda ise her olay bir görev ile tanımlanarak ve bazı görevlerin bu olayları beklemesi sağlanarak birbirine bağımlı birçok olaydan oluşan ve karmaşık görünen birçok uygulama basitçe gerçeklenebilir. PICOS, aynı yazmaç yapısına sahip olan PIC18XXX, PIC24XXX ve PIC32XXX ailelerinin tamamında kullanılabilmektedir.

Performans ile ilgili yapılan deneylerde PICOS'un iki göreve sahip basit uygulamalarda, 16 KB program belleğine sahip olan PIC 18F452 üzerinde yaklaşık % 35 civarında bellek tükettiği saptanmıştır. Görevler arası geçişler ise 432 komut çevrimine karşılık gelmektedir. Yığıt büyüklüğü 105 iken yapılan bu ölçümde PIC 18F452 kullanılmıştır. Ölçümler 4 MHz kristal osilatör ile yapılmış olup PIC için çevrimiçi komut işletim süresi harici frekansın ¼'üne karşılık geldiğinden bir komut 1 us sürede işletilmektedir. Dolayısıyla bahsi geçen deney ortamında görevler arası geçiş 432 us sürmektedir. Ancak bu değer platforma ve yığıt büyüklüğüne göre

değişiklik göstermektedir [29]. Seçilen mikrodenetleyiciye göre değişebilen görevler arası geçiş süresi, yığıt büyüklüğü ile doğru orantılı olarak değişim göstermektedir. Düşük yığıt büyüklükleri daha kısa görevler arası geçiş süresine, büyük yığıt büyüklükleri ise daha uzun görevler arası geçiş süresine karşılık gelmektedir. Kesmelerin işletilmesinde meydana gelen kesme gecikmesinin ölçümü yine aynı ortamda yapılmış ve 3 komut çevrimine karşılık gelen bir kesme gecikmesi ölçülmüştür. Benzer şekilde hesaplanarak kesme gecikmesi 3 us olarak tespit edilmiştir.

5.2. Yapılacak Çalışmalar

Bu çalışmada bahsi geçen temel işletim sistemi fonksiyonları sağlanırken işletim sistemine ekstra esneklik ve kullanım kolaylığı getiren semaforlar, iletiler ve ileti kuyrukları projede ele alınmamıştır. Gelecek çalışmalarda görevler arası haberleşmeyi sağlayan semaforların, iletilerin ve ileti kuyruklarının PICOS'a eklenmesi, böylelikle kullanılabilirliğin arttırılması amaçlanmaktadır.

KAYNAKLAR

[1] Akademik Bilişim 2008 tam bildiri metinleri erişim sayfası: http://ab.org.tr/ab08/sunum/110.ppt, erişim tarihi: 8 Şubat 2008.

[2] Wilmshurst, T., Designing Embedded Systems with PIC Microcontrollers, *Elsevier*, Newnes, University of Derby, UK, 2007.

[3] Curtis, K.E., Embedded Multitasking, *Elsevier*, Newnes, AZ, USA, 2006.

[4] Teknohaber sitesinde yer alan "Gömülü Sistemler" başlıklı makalenin erişim adresi: http://www.teknohaber.net/makale.php?id=50412, erişim tarihi: 13 Ekim 2007.

[5] Wecott, T., Applied Control Theory For Embedded Systems, *Elsevier*, Newnes, Oregon City, USA, 2006.

[6] Yıldırım, S., Kantarcı, A., Gömülü Sistemler İçin Yeni Bir İşletim Sistemi: eGİS, IEEE Explore, 13(7), 1-4, 2007.

[7] FSF forum sitesindeki "PIC hakkında her şey" başlıklı makalenin erişim adresi: http://www.fsf.cc/pic-hakk-nda-t37/index.html, erişim tarihi: 13 Ekim 2007.

[8] Okyay, M.O., 2004, A Portable Real Time Operating System for Embedded Platforms, *Y.Lisans Tezi*, İzmir İleri Teknoloji Enstitüsü, İzmir.

[9] Vikipedi özgür ansiklopedi sayfasında yer alan "PIC'lerin ortaya çıkışı" isimli içeriğin erişim adresi: http://tr.wikipedia.org/wiki/PIC, erişim tarihi: 21 Ekim 2007.

[10] PIC mikrodenetleyici üreticisi Microchip firmasının internet sitesi erişim adresi: http://www.microchip.com, erişim tarihleri: 21 Ekim 2007, 27 Kasım 2007, 2 Aralık 2007, 2 Ocak 2008, 10 Ocak 2008, 21 Ocak 2008.

[11] PIC 18FXX2 serisinin katalogunun internet erişim adresi: http://ww1.microchip.com/downloads/en/DeviceDoc/39564c.pdf, erişim tarihleri: 2 Aralık 2008, 10 Ocak 2008.

[12] Daniel, Mosse, Olafur Gudmundsson, and Ashok K. Agrawala, Prototyping Real Time Operating Systems: a Case Study, IEEE Explore , 4(7), 144-154, 1991.

[13] Yanbing, Lit, Miodrag, Potkonjakt, and Wayne, Wolf, Real-Time Operating Systems for Embedded Computing, IEEE Explore, 15(10), 388-392, 1997.

[14] İstanbul Teknik Üniversitesi öğretim üyesi Binnur KURT'un "İşletim Sistemleri" ders notlarının yer aldığı internet sayfasının adresi: http://www3.itu.edu.tr/~bkurt/Courses/os/lecture1.pdf, erişim tarihi: 27 Kasım 2007.

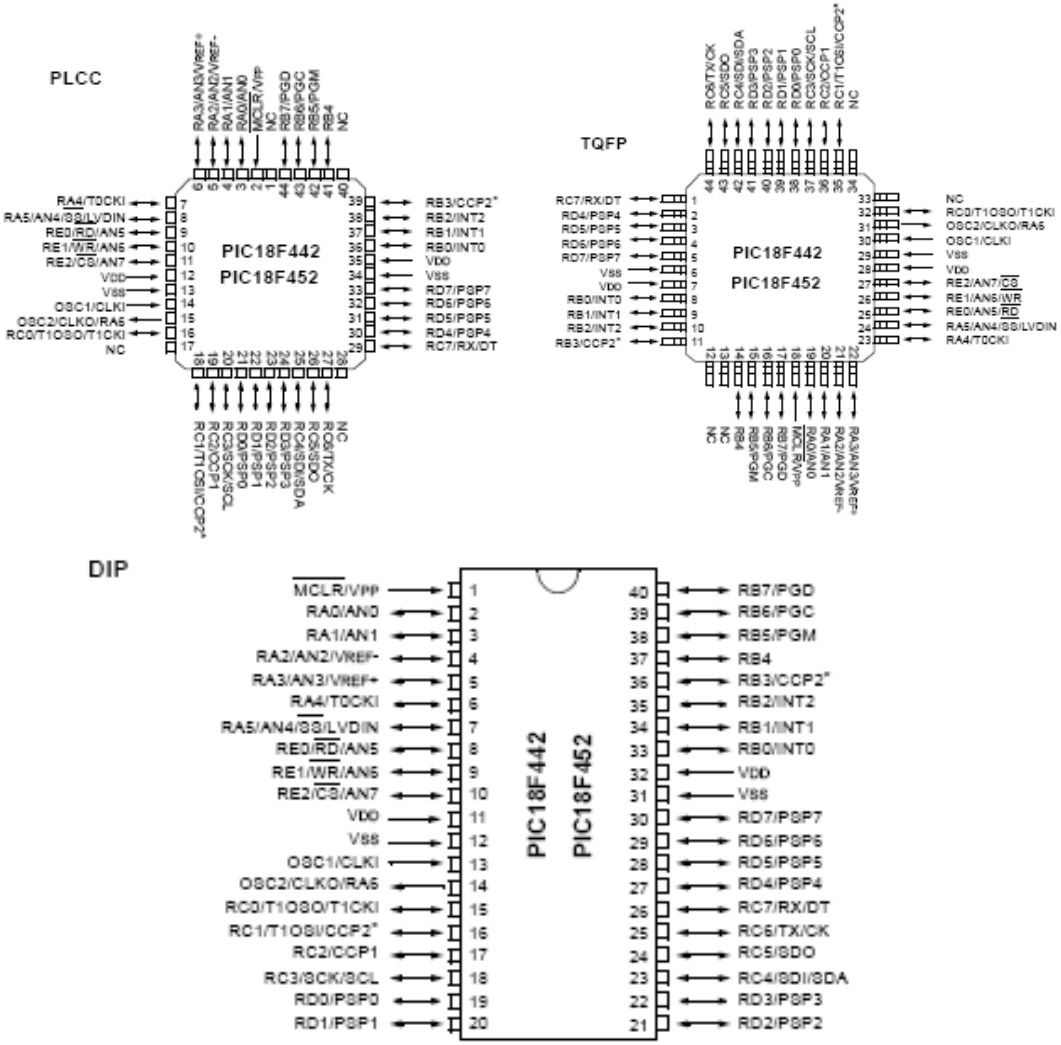
[15] Erciyeş, K., 1989, Dağıtık Bir İşletim Sistemi İçin Gerçek Zamanlı, Çok Görevli Bir Çekirdek Tasarımı, *Doktora Tezi*, Ege Üniversitesi Fen Bilimleri Enstitüsü, İzmir.

[16] Franz, J. Rammig, Marcelo, Götz, Tales, Heimfarth, Peter, Janacik, Simon, Oberthür, Real-time Operating Systems for Self-coordinating Embedded Systems, IEEE Explore, 15(5), 8pp, 2006.

[17] Gazi Üniversitesi öğretim üyesi Mustafa KÜÇÜKALİ'nin "İşletim Sistemi" ders notlarının yer aldığı internet sitesinin adresi: http://w3.gazi.edu.tr/~kmustafa/islt/site_menu.htm, erişim tarihi: 27 Kasım 2007.
[18] Büyükkardeş, S., 2004, A Real Time Operating System for 8051 Microcontrollers, *Y.Lisans Tezi*, Dokuz Eylül Üniversitesi, İzmir.
[19] PIC üreticisi Microchip firmasının yerleşik geliştirme ortamı MPLAB'ın yer aldığı internet sitesinin adresi: http://www.microchip.com/stellent/idcplg?IdcService=SS_GET_PAGE&nodeId=1406&dDocName=en019469, erişim adresi: 15 Ekim 2007.
[20] PIC üreticisi Microchip firmasının yerleşik geliştirme ortamı MPLAB ile çalışan PIC C18 derleyicisin bulunduğu internet sayfasının adresi: http://www.microchip.com/stellent/idcplg?IdcService=SS_GET_PAGE&nodeId=1406&dDocName=en010014, erişim tarihi: 15 Ekim 2007.
[21] JDM seri PIC programlayıcısının internet sitesinin adresi: http://www.jdm.homepage.dk/newpic.htm, erişim adresi: 15 Ekim 2007.
[22] Bren8ner USB PIC programlayıcısının internet sitesinin adresi: http://www.sprut.de/electronic/pic/projekte/brenner8/index.htm, erişim tarihi: 22 Ekim 2007.
[23] Bren8ner programlayıcısını satan Bartatek firmasının internet sitesinin adresi: http://www.bartatek.com/documents/urunler/picprg/brn8se/brn8se.htm, erişim tarihi: 25 Ekim 2007.
[24] JDM programlayıcı ile birlikte kullanılan IC-Prog bilgisayar yazılımının internet sitesinin adresi: http://www.ic-prog.com/, erişim tarihi: 15 Ekim 2007.
[25] Bren8ner USB programlayıcı ile birlikte kullanılan USBurn yazılımının internet sitesinin adresi: http://www.sprut.de/electronic/soft/usburn/usburn.htm, erişim tarihi: 22 Ekim 2008.
[26] Pumpkin firması tarafından üretilen Salvo işletim sisteminin kullanım kılavuzunun yer aldığı internet sayfasının adresi: http://pumpkininc.com/content/doc/manual/SalvoUserManual.pdf, erişim adresi: 28 Ekim 2007.
[27] FreeRTOS işletim sisteminin çalışma prensibinin yer aldığı internet sayfasının adresi: http://www.freertos.org/implementation/index.html, erişim tarihi: 23 Aralık 2008.
[28] Manthos A. Tsoukarellas, Systematically Testing a Real-Time Operating System, IEEE Explore, 15(10), 50-60, 1995.
[29] Carsten, Böke, Marcelo, Götz, Tales, Heimfarth, Configurable Real-Time Operating Systems and Their Applications, IEEE Explore, 17(1), 148-155, 2003.
[30] Dr. Cengiz ERBAŞ ile 16.11.2007 tarihinde ASELSAN tesislerinde yapılan görüşme.

EKLER

EK A: PIC18F452 KILIFLARI VE GİRİŞ / ÇIKIŞ UÇLARI



Şekil A.1. PIC 18F452 kılıfları

Çizelge A.1. PIC18F452 giriş/çıkış uçları

Uç İsmi	Uç Numarası	Tipi	Tampon Tipi	Açıklama
MCLR/VPP	1	Giriş	ST	Sıfırlama ucu / yüksek voltaj ICSP programlama ucu
OSC1/CLKI	13	Giriş	ST	Kristal Osilatör veya harici saat girişi

OSC2/CLKO/RA6	14	Giriş/Çıkış	ST/CMOS	Kristal Osilatör, harici saat çıkışı veya genel amaçlı giriş/çıkış ucu
RA0/AN0	2	Giriş/Çıkış	TTL	Sayısal giriş/çıkış veya analog giriş 1
RA1/AN1	3	Giriş/Çıkış	TTL	Sayısal giriş/çıkış veya analog giriş 2
RA2/AN2/VREF-	4	Giriş/Çıkış	TTL	Sayısal giriş/çıkış, analog giriş 3 veya Anolog/Sayısal Voltaj girişi
RA3/AN3/VREF+	5	Giriş/Çıkış	TTL	Sayısal giriş/çıkış, analog giriş veya Anolog/Sayısal Voltaj girişi
RA4/T0CKI	6	Giriş/Çıkış	TTL	Sayısal giriş/çıkış, analog giriş veya Zamanlayıcı 0 harici saat girişi
RA5/AN4/SS/LVDIN	7	Giriş/Çıkış	TTL	Sayısal giriş/çıkış, analog giriş 4 veya düşük voltaj algılama
RB0/INT0	33	Giriş/Çıkış	TTL	Sayısal giriş/çıkış, harici kesme 0
RB1/INT1	34	Giriş/Çıkış	TTL	Sayısal giriş/çıkış, harici kesme 1
RB2/INT2	35	Giriş/Çıkış	TTL	Sayısal giriş/çıkış, harici kesme 2
RB3/CCP2	36	Giriş/Çıkış	TTL	Sayısal giriş/çıkış, karşılaştırma 2 girişi
RB4	37	Giriş/Çıkış	TTL	Sayısal giriş/çıkış, değişiklikte kesme biti
RB5/PGM	38	Giriş/Çıkış	TTL	Sayısal giriş/çıkış, düşük voltaj programlama ICSP ucu
RB6/PGC	39	Giriş/Çıkış	TTL	Sayısal giriş/çıkış, ICD ve ICSP saat ucu
RB7/PGD	40	Giriş/Çıkış	TTL	Sayısal giriş/çıkış, ICD ve ICSP veri ucu
RC0/T1OSO/T1CKI	15	Giriş/Çıkış	ST	Sayısal giriş/çıkış, zamanlayıcı 1 osilatör veya harici saat girişi
RC1/T1OSI/CCP2	16	Giriş/Çıkış	ST	Sayısal giriş/çıkış, zamanlayıcı 1 osilatör girişi, yakalama 2 girişi, karşılaştırma 2 çıkışı, PWM 2 çıkışı
RC2/CCP1	17	Giriş/Çıkış	ST	Sayısal giriş/çıkış, yakalama 1 girişi, karşılaştırma 1 çıkışı, PWM 1 çıkışı
RC3/SCK/SCL	18	Giriş/Çıkış	ST	Sayısal giriş/çıkış, senkron seri saat giriş/çıkışı
RC4/SDI/SDA	23	Giriş/Çıkış	ST	Sayısal giriş/çıkış, SPI veri girişi, I^2C veri giriş/çıkışı
RC5/SDO	24	Giriş/Çıkış	ST	Sayısal giriş/çıkış, SPI veri çıkışı
RC6/TX/CK	25	Giriş/Çıkış	ST	Sayısal giriş/çıkış, USART asenkron iletim, USART senkron saat
RC7/RX/DT	26	Giriş/Çıkış	ST	Sayısal giriş/çıkış, USART asenkron alım, USART senkron veri
RD0/PSP0	19	Giriş/Çıkış	ST/TTL	Sayısal giriş/çıkış, paralel ikincil port veri ucu
RD1/PSP1	20	Giriş/Çıkış	ST/TTL	Sayısal giriş/çıkış, paralel ikincil port veri ucu
RD2/PSP2	21	Giriş/Çıkış	ST/TTL	Sayısal giriş/çıkış, paralel ikincil port veri ucu
RD3/PSP3	22	Giriş/Çıkış	ST/TTL	Sayısal giriş/çıkış, paralel ikincil port veri ucu
RD4/PSP4	27	Giriş/Çıkış	ST/TTL	Sayısal giriş/çıkış, paralel ikincil port veri ucu
RD5/PSP5	28	Giriş/Çıkış	ST/TTL	Sayısal giriş/çıkış, paralel ikincil

				port veri ucu
RD6/PSP6	29	Giriş/Çıkış	ST/TTL	Sayısal giriş/çıkış, paralel ikincil port veri ucu
RD7/PSP7	30	Giriş/Çıkış	ST/TTL	Sayısal giriş/çıkış, paralel ikincil port veri ucu
RE0/RD/AN5	8	Giriş/Çıkış	ST/TTL	Sayısal giriş/çıkış, okuma kontrolü. analog giriş 5
RE1/WR/AN6	9	Giriş/Çıkış	ST/TTL	Sayısal giriş/çıkış, yazma kontrolü, analog giriş 6
RE2/CS/AN7	10	Giriş/Çıkış	ST/TTL	Sayısal giriş/çıkış, devre seçme ucu, analog giriş 7
V_{DD}	12,31	Güç	-	Toprak referans ucu
V_{SS}	11,32	Güç	-	Doğru akım besleme ucu

ST: Schmitt Tetikleyicisi

EK B: PICOS İŞLETİM SİSTEMİ FONKSİYONLARI

GorevOlustur

Yeni bir görev oluşturmak için kullanılan fonksiyondur. Oluşturulan görev çalışmaya hazır görevler listesine eklenir.

Kullanımı:

GorevOlustur (GorevKodu, GorevAdi, YigitDerinligi, ParametreIsaretcisi, OncelikSeviyesi, GorevAdresi)

isaretci: Göreve ait kodlamanın yapıldığı fonksiyonu işaret eden parametredir.

GorevIsmi: Göreve verilen açıklayıcı isim.

YigitBuyuklugu: Görev yığıtının tutabileceği değişken sayısını gösteren parametredir. 32-bitlik bir yığıt kullanılıyorsa ve yığıt büyüklüğü 20 olarak tanımlandıysa yığıt için 80 bayt bellek ayrılacaktır.

ParametreIsaretcisi: Oluşturulan görev için kullanılacak parametreleri gösteren işaretçidir.

OncelikSeviyesi: Görevin hangi öncelik seviyesinde çalışacağını ifade eder.

GorevAdresi: Görevin kullandığı adresi gösteren parametredir.

GorevSil

Oluşturulmuş bir görevi silmek için kullanılır. Bu fonksiyonu kullanabilmek için Gorev_Silme_Ozelligi parametresinin 1 olarak tanımlanması gerekir. Silinen görev tüm listelerden çıkarılacaktır.

Kullanımı:

GorevSil(GorevAdresi)

GorevAdresi: Silinecek görevin adresini ifade eder. NULL olarak kullanılırsa içerisinde bulunulan görev silinecektir.

GörevErtele

Bir görevi, ifade edilen tik sayısı kadar ertelemek için kullanılır. Kullanılabilmesi için Gorev_Erteleme_Ozelligi parametresinin 1 olarak tanımlanmış olması gerekir. Görevin bloke olarak bekleyeceği süre tik çalışma hızına bağlıdır.

Kullanımı:

GorevErtele(ErtelemeTikSayisi)

ErtelemeTikSayisi: Tik periyodu cinsinden görevin erteleneceği süreyi gösterir.

GorevBeklet

Bir görevi, belirtilen bir zamana kadar bekletmek için kullanılır.

Kullanımı:

GorevBeklet(OncekiUyanmaZamani, ZamanArtisi)

OncekiUyanmaZamani: Görevin bir önceki bekletmeden çıkma zamanını gösteren işaretçiyi ifade eder.

ZamanArtisi: Arttırılacak zaman periyodunu gösterir.

GorevBeklet fonksiyonunun, GorevErtele fonksiyonundan önemli bir farkı vardır. GorevErtele fonksiyonu kullanılırken uyanma zamanı çağrıldığı ana bağlı bir zaman ile belirtilirken, GorevBeklet fonksiyonunda tam olarak belirtilen bir uyanma zamanı söz konusudur. GorevBeklet fonksiyonu için daha önceden meydana gelmiş bir uyanma zamanı belirtilmişse hiç beklemeden hemen çalışacaktır. Bu nedenle GorevBeklet fonksiyonu ile durdurulan bir görev, belli periyotlar ile uyanma zamanını yeniden hesaplamak durumundadır.

GorevOnceligiAl

Bir görevin o anda hangi öncelik seviyesinde çalıştığını geri döndürmeye yarayan fonksiyondur.

Kullanımı:

GorevOnceligiAl(GorevAdresi)

GorevAdresi: Önceliği okunacak görevin adresini ifade eder. NULL olarak kullanılırsa içerisinde bulunulan görevin önceliği okunacaktır.

GorevOnceligiAta

Bir görevin çalıştığı öncelik seviyesini belirtilen öncelik seviyesine getirmeye yarayan fonksiyondur.

Kullanımı:

GorevOnceligiAta(GorevAdresi, YeniOncelik)

GorevAdresi: Öncelik atanacak görevin adresini ifade eder. NULL olarak kullanılırsa içerisinde bulunulan görevin önceliği atanacaktır.

YeniÖncelik: Öncelik atanacak görevin hangi öncelik seviyesinde çalışacağını belirleyen parametredir.

GorevDurdur

Çalışmakta olan bir görevi durdurmaya yarayan fonksiyondur. Durdurulan görev tekrar çalışabilir konuma gelene kadar çalışan görevler listesinde yer almayacaktır.

Kulanımı:

GorevDurdur(GorevAdresi)

GorevAdresi: Durdurulacak görevin adresini ifade eder. NULL olarak kullanılırsa içerisinde bulunulan görev durdurulacaktır.

GorevBaslat

GorevDurdur fonksiyonu ile durdurulmuş bir görevi tekrar başlatmaya yarayan bir fonksiyondur. Başlatılan görev çalışmaya hazır görevler listesine eklenecektir.

Kullanımı:

GorevBaslat(GorevAdresi)

GorevAdresi: Başlatılacak görevin adresini ifade eder. NULL olarak kullanılırsa içerisinde bulunulan görev başlatılacaktır.

KesmedenGorevBaslat

Durdurulan bir görevi kesme içerisinden başlatabilmek için kullanılan bir fonksiyondur. Başlatılan görev GorevBaslat fonksiyonunda olduğu gibi çalışmaya hazır görevler listesine eklenir.

Kullanımı:

KesmedenGorevBaslat(GorevAdresi)

GorevAdresi: Başlatılacak görevin adresini ifade eder.

CizelgeleyiciyiBaslat

Görevler arası geçişi gerçekleştiren çizelgeleyicinin başlamasını sağlayan fonksiyondur. Çekirdekte yer alan tikleri işleterek hangi görevin ne zaman çalışacağına karar verilir.

Kullanımı:

CizelgeleyiciyiBaslat()

Bu fonksiyon sayesinde çizelgeleyici, görevlerden birisi CizelgeleyiciyiSonlandir fonksiyonunu çağırana kadar arka planda çalışmaya devam edecektir.

CizelgeleyiciyiSonlandir

Görevler arası geçiş yapılmasını durdurmak için daha önceden başlatılmış olan çizelgeleyiciyi sonlandırır. Böylece çekirdeğin tikleri işletmesi de askıya alınmış olmaktadır.

Kullanımı:

CizelgeleyiciyiSonlandir()

CizelgeleyiciyiSonlandir fonksiyonu çağırıldığında çekirdek tarafından kullanılan tüm kaynaklar serbest bırakılır.

GorevleriDurdur

Çalışan görev dışındaki tüm görevleri durdurmaya yarayan fonksiyondur. Bütün çekirdek uygulamalarını durdurur ancak tüm kesmeler aktif kalır.

Kullanımı:

GorevleriDurdur()

Bu fonksiyon çağırıldığında aktif olan görev başka bir göreve geçiş riski olmadan GorevleriBaslat fonksiyonu çağırılana kadar çalışabilir.

GorevleriBaslat

Çalışan görev dışında, durdurulmuş olan görevleri tekrar başlatmak için kullanılır. Tüm çekirdek uygulamaları tekrar çalışır hale gelir.

Kullanım:

GorevleriBaslat()

MevcutGorevAdresiniAl

Çalışmakta olan görevin sahip olduğu adresi döndüren fonksiyondur. Bu fonksiyonun çalışması için "MevcutGorevAdresiniAlmaOzelligi" parametresinin 1 olarak tanımlanmış olması gerekmektedir.

Kullanım:

MevcutGorevAdresiniAl()

MevcutGorevSayisiniAl

Gerçek zamanlı çekirdek tarafından yönetilen toplam görev sayısını döndüren fonksiyondur. Durdurulmuş, ertelenmiş, hazır, çalışan tüm görevleri döndürür. Silinen fakat serbest bırakılmayan görevler de bu sayıya dâhildir.

Kullanım:

MevcutGorevSayisiniAl()

GorevListeleriniBaslat()

Mevcut görevleri durumlarına ve yığıt kullanımlarına göre listeler. Bunun için Gorev_Bekletme_Ozelligi ve Gorev_Erteleme_Ozelligi parametrelerinin 1 olarak tanımlanmış olması gerekmektedir. Bloke görevler 'B' ile, hazır görevler 'H' ile, silinen görevler 'S' ile ve durdurulan görevler 'D' ile gösterilir.

Kullanım:

GorevListeleriniBaslat()

EK C: PICOS YAPILANDIRMA AYARLARI

Calisma_Prensibi: İşletim sisteminin çağrı üstünlüğüne göre mi yoksa işbirlikçi yönteme göre mi çalışacağını belirler. 1 olarak tanımlanırsa çağrı üstünlüğüne göre 0 olarak tanımlanırsa işbirlikçi yönteme göre çalışır.

Os_Saat_Hizi: Bu parametre sistem tik kesmesi için tanımlanmıştır. Bu kesme ise zamanı ölçmek için kullanılır. Dolayısıyla parametre ne kadar yüksek tanımlanırsa zaman o kadar hassas ölçülebilecektir. Fakat aynı zamanda yüksek bir değer, çekirdeğin fazla işlemci zamanı tüketmesi anlamına gelmekte ve verimliliği düşürmektedir. Bir uygulamada birden çok görev aynı önceliği paylaşabilir. Böyle bir durumda çekirdek, işlemci gücünü her tik sonrasında bu görevler arasında paylaştırır ve sırasıyla geçiş yapar. Daha yüksek bir seçim böyle bir durumda her göreve verilen zaman aralığını da daraltmış olur.

Saat_Frekansi: İşlemcinin çalışma frekansını ifade eder. Özellikle zamanlayıcı gibi zamana bağlı bileşenlerin doğru çalışması için gereklidir. Bu değer işlemciye bağlanan osilatör değerine bağlıdır.

En_Yuksek_Oncelik_Seviyesi: Uygulamada kullanılabilecek en büyük görev öncelik seviyesini tanımlamaya yarar. Bu değer uygulamada tanımlanır ve programın yazılması esnasında bu değerden daha büyük öncelik seviyesi kullanılamaz.

Minimum_Yigit_Buyuklugu: Hiçbir iş yapmayan boş bir görevin ne kadar yığıt kullanacağını tanımlamak için kullanılır. İşletim sistemi yapısı ve PIC mimarisi göz önüne alınarak en az 100 olarak tanımlanmalıdır.

Toplam_Hafiza_Buyuklugu: Çekirdek tarafından kullanılabilecek azami RAM bellek büyüklüğünü ifade eder.

Maksimum_Isim_Uzunlugu: Görevlere verilecek isimlerde en fazla kaç karakter kullanılabileceğini gösterir.

_16_Bit_Sıstem_Saati: İşletim sisteminde zaman tiklerle ölçülür. Tikler ise çekirdek başlatıldığından beri işletim sisteminde kaç kez sistem kesmesi meydana geldiğini gösterir. Bu değer 1 olarak tanımlanırsa 16 bit, 0 olarak tanımlanırsa 32 bit olarak kullanılır. 16 bit tik kullanılması 8 ve 16 bitlik işletim sistemlerinde verimliliği oldukça arttırır. Ancak bu değer olarak en fazla 65535 ile ifade edilme kısıdını beraberinde getirir. Bu nedenle, 100 Hz'lik bir sistem saati kullanıldığında, bir görevin durdurulması veya ertelenmesi 16 bit sayıcı ile en fazla 655 saniye olarak kullanılabilirken 32 bit sayıcı ile 42.949.673 saniye olarak kullanılabilir.

Oncelik_Atama_Ozelligi: Bu parametre 1 olarak atanırsa görevlere öncelik atanabilir, yani çalışma zamanında görev önceliği farklı bir değer ile değiştirilebilir. Ancak değer 0 olursa görev hangi öncelik ile oluşturulursa o değerde kalır.

Oncelik_Okuma_Ozelligi: Bu parametre 1 olarak atanırsa görevlerin öncelikleri okunabilir, yani çalışma zamanında görev önceliğinin değeri bir fonksiyon aracılığı ile alınabilir. Ancak değer 0 olursa görev önceliği okunamaz.

Gorev_Silme_Ozelligi: Bu parametre çalışma zamanında görev silme özelliğinin kullanılıp kullanılamayacağını ifade eder. 1 olarak tanımlanırsa görevler silinebilirken 0 olarak tanımlandığında görev silinemeyecektir.

Kaynaklari_Temizleme_Ozelligi: Bu parametre ise yine çalışma zamanında kullanılan kaynakların temizlenip temizlenemeyeceğini gösterir.

Gorev_Durdurma_Ozelligi: İşletim sisteminde tanımlanan görevlerin çalışma zamanında durdurulup durdurulamayacağını gösterir. 1 ise görevler durdurulabilir, 0 ise durdurma özelliği çalışmayacaktır.

Gorev_Bekletme_Ozelligi: İşletim sisteminde tanımlanan görevlerin çalışma zamanında bekletilip bekletilemeyeceğini gösterir. 1 ise görevler beklemeye alınabilirken 0 ise alınamayacaktır.

Gorev_Erteleme_Ozelligi: İşletim sisteminde tanımlanan görevlerin çalışma zamanında ertelenip ertelenemeyeceğini gösterir. 1 ise görevler ertelenebilir, 0 ise ertelenemez.

EK D: KAYNAK KODLARI

Main.c dosyası

```
#include "kontrol.h"
#include "cekirdek.h"

#include "sayi.h"

#pragma config WDT = OFF, OSC = XT, DEBUG = OFF

static void LedYak1( void *pvParameters );
static void LedYak2( void *pvParameters );

GorevAdresi GorevAdres1;
GorevAdresi GorevAdres2;

void bekle (void)
{
  unsigned int i;
  for (i = 0; i < 10000 ; i++);
}

void main( void )
{

// I/O ve interrupt ayarları
   INTCON = 0x10;
   INTCON2 = 0x80;
   RCONbits.IPEN = 0;
   INTCONbits.GIEH = 1;
         PORTB = 0;
   TRISB = 1;
//--------------------------

         GorevOlustur( LedYak1, "LED1", MINIMUM_YIGIT_BUYUKLUGU, NULL,
BosGorevOnceligi+2, &GorevAdres1 );
         GorevOlustur( LedYak2, "LED2", MINIMUM_YIGIT_BUYUKLUGU, NULL,
BosGorevOnceligi+1, &GorevAdres2 );
         CokluGorevBaslat();
}
/*-----------------------------------------------------------*/

static void LedYak1( void *pvParameters )
{
         for( ;; )
         {
         bekle();
         {
                  PORTBbits.RB4 = !( PORTBbits.RB4 );
         }
```

```
        bekle();
        GorevDurdur( NULL );
        }
}

/*-----------------------------------------------------------*/

static void LedYak2( void *pvParameters )
{
        for( ;; )
        {
        bekle();
        {
                PORTBbits.RB7 = !( PORTBbits.RB7 );
        }
        bekle();
        }
}


void Kesme (void)
{
        if (INTCONbits.INT0IF)
   {
          INTCON = 0x50;
                KesmedenGorevBaslat (GorevAdres1);
          bekle();
                INTCONbits.GIEH = 1;
        INTCONbits.INT0IF = 0;
                PORTB = 0;
   }

}
```

Kuyruk.c dosyası

```
#include <stdlib.h>
#include "kontrol.h"
#include "kuyruk.h"

void Kuyruk_Baslat( Kuyruk *pKuyruk )
{
	pKuyruk->pEndeks = ( Kuyruk_Ogesi * ) &( pKuyruk->Kuyruk_Sonu );

	pKuyruk->Kuyruk_Sonu.Oge_Degeri = max_gecikme;
	pKuyruk->Kuyruk_Sonu.pSonraki = ( Kuyruk_Ogesi * ) &( pKuyruk-
>Kuyruk_Sonu );
	pKuyruk->Kuyruk_Sonu.pOnceki = ( Kuyruk_Ogesi * ) &( pKuyruk-
>Kuyruk_Sonu );

	pKuyruk->Oge_Sayisi = 0;
}
/*-------------------------------------------------------------*/

void Kuyruk_Ogesi_Baslat( Kuyruk_Ogesi *pOge )
{
	pOge->pHazne = NULL;
}
/*-------------------------------------------------------------*/

void Kuyruk_Sonuna_Ekle( Kuyruk *pKuyruk, Kuyruk_Ogesi *pYeni_Oge )
{
volatile Kuyruk_Ogesi * pEndeks;

	pEndeks = pKuyruk->pEndeks;

	pYeni_Oge->pSonraki = pEndeks->pSonraki;
	pYeni_Oge->pOnceki = pKuyruk->pEndeks;
	pEndeks->pSonraki->pOnceki = ( volatile Kuyruk_Ogesi * ) pYeni_Oge;
	pEndeks->pSonraki = ( volatile Kuyruk_Ogesi * ) pYeni_Oge;
	pKuyruk->pEndeks = ( volatile Kuyruk_Ogesi * ) pYeni_Oge;

	pYeni_Oge->pHazne = ( void * ) pKuyruk;

	( pKuyruk->Oge_Sayisi )++;
}
/*-------------------------------------------------------------*/

void ListeyeEkle( Kuyruk *pKuyruk, Kuyruk_Ogesi *pYeni_Oge )
{
volatile Kuyruk_Ogesi *pYineleyici;
SistemSaatiTipi Ekleme_Degeri;

	Ekleme_Degeri = pYeni_Oge->Oge_Degeri;
```

```
	if( Ekleme_Degeri == max_gecikme )
	{
		pYineleyici = pKuyruk->Kuyruk_Sonu.pOnceki;
	}
	else
	{
		for( pYineleyici = ( Kuyruk_Ogesi * ) &( pKuyruk->Kuyruk_Sonu );
pYineleyici->pSonraki->Oge_Degeri <= Ekleme_Degeri; pYineleyici = pYineleyici-
>pSonraki )
		{

		}
	}

	pYeni_Oge->pSonraki = pYineleyici->pSonraki;
	pYeni_Oge->pSonraki->pOnceki = ( volatile Kuyruk_Ogesi * ) pYeni_Oge;
	pYeni_Oge->pOnceki = pYineleyici;
	pYineleyici->pSonraki = ( volatile Kuyruk_Ogesi * ) pYeni_Oge;

	pYeni_Oge->pHazne = ( void * ) pKuyruk;

	( pKuyruk->Oge_Sayisi )++;
}
/*------------------------------------------------------------*/

void Kuyruktan_Sil( Kuyruk_Ogesi *pSilinecek_Oge )
{
Kuyruk * pKuyruk;

	pSilinecek_Oge->pSonraki->pOnceki = pSilinecek_Oge->pOnceki;
	pSilinecek_Oge->pOnceki->pSonraki = pSilinecek_Oge->pSonraki;

	pKuyruk = ( Kuyruk * ) pSilinecek_Oge->pHazne;

	if( pKuyruk->pEndeks == pSilinecek_Oge )
	{
		pKuyruk->pEndeks = pSilinecek_Oge->pOnceki;
	}

	pSilinecek_Oge->pHazne = NULL;
	( pKuyruk->Oge_Sayisi )--;
}
/*------------------------------------------------------------*/
```

Bellek.c dosyası

```
#include <stdlib.h>
#include "kontrol.h"
#include "cekirdek.h"

#if BAYT_SIRALAMASI == 8
	#define BAYT_SIRALAMA_MASKESI ( ( size_t ) 0x0007 )
#endif

#if BAYT_SIRALAMASI == 4
	#define BAYT_SIRALAMA_MASKESI		( ( size_t ) 0x0003 )
#endif

#if BAYT_SIRALAMASI == 2
	#define BAYT_SIRALAMA_MASKESI		( ( size_t ) 0x0001 )
#endif

#if BAYT_SIRALAMASI == 1
	#define BAYT_SIRALAMA_MASKESI		( ( size_t ) 0x0000 )
#endif

#ifndef BAYT_SIRALAMA_MASKESI
	#error "Hatali BAYT_SIRALAMASI tanimi"
#endif

static struct OS_YIGIT
{
	unsigned long random;
	unsigned char yigit1[ TOPLAM_HAFIZA_BUYUKLUGU ];
} yigit;

static size_t Sonraki_Bos_Bayt = ( size_t ) 0;

void *pYerlestir( size_t Istenen_Boyut )
{
void *pGeriDon = NULL;

	#if BAYT_SIRALAMASI != 1
		if( Istenen_Boyut & BAYT_SIRALAMA_MASKESI )
		{
			Istenen_Boyut += ( BAYT_SIRALAMASI - ( Istenen_Boyut &
BAYT_SIRALAMA_MASKESI ) );
		}
	#endif

	GorevleriDurdur();
	{
		if( ( ( Sonraki_Bos_Bayt + Istenen_Boyut ) <
TOPLAM_HAFIZA_BUYUKLUGU ) &&
```

```
                        ( ( Sonraki_Bos_Bayt + Istenen_Boyut ) > Sonraki_Bos_Bayt ) )/*
Tasmayi kontrol et. */
                {
                        pGeriDon = &( yigit.yigit1[ Sonraki_Bos_Bayt ] );
                        Sonraki_Bos_Bayt += Istenen_Boyut;
                }
        }
        GorevleriBaslat();

        return pGeriDon;
}
/*-----------------------------------------------------------*/

void BellekBosalt( void *pv )
{
        ( void ) pv;
}
/*-----------------------------------------------------------*/

void Bloklari_Baslat( void )
{
        Sonraki_Bos_Bayt = ( size_t ) 0;
}
```

Cekirdek.c dosyası

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#include "kontrol.h"
#include "cekirdek.h"

#define BOSTA_YIGIT_BUYUKLUGU MINIMUM_YIGIT_BUYUKLUGU

#ifndef MAKSIMUM_ISIM_UZUNLUGU
	#define MAKSIMUM_ISIM_UZUNLUGU 16
#endif

#if MAKSIMUM_ISIM_UZUNLUGU < 1
	#undef MAKSIMUM_ISIM_UZUNLUGU
	#define MAKSIMUM_ISIM_UZUNLUGU 1
#endif

#ifndef KESMEDEN_GOREV_BASLATMA_OZELLIGI
	#define KESMEDEN_GOREV_BASLATMA_OZELLIGI 1
#endif

typedef struct grvGorevKontrolBlogu
{
	volatile unsigned char   *pYigitBasi;
	Kuyruk_Ogesi                    GenelKuyrukOgesi;
	Kuyruk_Ogesi                    OlayListeOgesi;
	unsigned char   Oncelik;
	unsigned char   *pStack;
	signed char             GorevAdi[ MAKSIMUM_ISIM_UZUNLUGU ];
} grvGKB;

grvGKB * volatile AktifGKB = NULL;

static Kuyruk HazirGorevlerListesi[ EN_YUKSEK_ONCELIK_SEVIYESI ];
static Kuyruk ErtelenenGorevListesi1;
static Kuyruk ErtelenenGorevListesi2;
static Kuyruk * volatile ErtelenenGorevListesi;
static Kuyruk * volatile Tasma_ErtelenenGorevListesi;
static Kuyruk BekleyenHazirListe;

#if ( GOREV_SILME_OZELLIGI == 1 )
	static volatile Kuyruk SonlanmayiBekleyenGorevler;
	static volatile unsigned char SilinenGorevler = ( unsigned char ) 0;
#endif

#if ( GOREV_DURDURMA_OZELLIGI == 1 )
	static Kuyruk DurdurulanGorevListesi;
#endif
```

```
static volatile unsigned char MevcutGorevSayisi         = ( unsigned char ) 0;
static volatile SistemSaatiTipi TikSayisi                       = ( SistemSaatiTipi ) 0;
static unsigned char KullanilanEnYuksekOncelik                  =
BosGorevOnceligi;
static volatile unsigned char HazirEnYuksekOncelik              = BosGorevOnceligi;
static volatile signed char CizelgeleyiciAktif                  = 0;
static volatile unsigned char CizelgeleyiciPasif        = ( unsigned char ) 0;
static volatile unsigned char AtlananTikler                     = ( unsigned char )
0;
static volatile char AtlananGecis                                       = ( char ) 0;
static volatile char TasmaSayisi                                = ( char ) 0;

#define YigitDoldurmaBayti     ( 0xa5 )

#define BlokeGorev_KRK                  ( ( signed char ) 'B' )
#define HazirGorev_KRK                  ( ( signed char ) 'H' )
#define SilinenGorev_KRK        ( ( signed char ) 'S' )
#define DurdurulanGorev_KRK             ( ( signed char ) 'D' )

#define TamponaYaz()

#define GoreviKuyrugaEkle( pGKB )

        \
{

                                                \
        if( pGKB->Oncelik > HazirEnYuksekOncelik )
                                                                                \
        {

                                                \
                HazirEnYuksekOncelik = pGKB->Oncelik;

                \
        }

                                                \
        Kuyruk_Sonuna_Ekle( ( Kuyruk * ) &( HazirGorevlerListesi[ pGKB->Oncelik ] ),
&( pGKB->GenelKuyrukOgesi ) );      \
}

#define ErtelenmisGorevleriKontrolEt()

                                \
{

                                                        \
```

```
register grvGKB *pGKB;

                                        \


                                                        \
        while( ( pGKB = ( grvGKB * ) Ilk_Kayit_Sahibini_Al( ErtelenenGorevListesi ) ) !=
NULL )                                                  \
        {

                                                        \
                if( TikSayisi < Liste_Oge_Degerini_Al( &( pGKB->GenelKuyrukOgesi ) ) )
                                                                \
                {

                                                        \
                        break;

                                                \
                }

                                                        \
                Kuyruktan_Sil( &( pGKB->GenelKuyrukOgesi ) );

                \


                                                        \
                if( pGKB->OlayListeOgesi.pHazne )

                        \
                {

                                                        \
                        Kuyruktan_Sil( &( pGKB->OlayListeOgesi ) );

                \
                }

                                                        \
                GoreviKuyrugaEkle( pGKB );

                                \
        }

                                                        \
}
```

```
#define AdrestenGKByiAl( pAdres ) ( ( pAdres == NULL ) ? ( grvGKB * ) AktifGKB : ( grvGKB * ) pAdres )

static void GKBDegiskenleriniBaslat( grvGKB *pGKB, const signed char * const GorevAdi, unsigned char Oncelik );

static void GorevListeleriniBaslat( void );

static GorevIslev( BosGorev, pvParameters );

#if ( ( GOREV_SILME_OZELLIGI == 1 ) || ( KAYNAKLARI_TEMIZLEME_OZELLIGI == 1 ) )
	static void GKByiSil( grvGKB *pGKB );
#endif

static void SonlanmayiBekleyenGorevleriKontrolEt( void );

static grvGKB *GKBveYigitiYerlestir( unsigned int YigitDerinligi );

#if ( GOREV_DURDURMA_OZELLIGI == 1 )

	static char GorevDurmusMu( const grvGKB * const pGKB );

#endif

signed char GorevOlustur( dGorevKodu vGorevKodu, const signed char * const GorevAdi, unsigned int YigitDerinligi, void *pvParameters, unsigned char Oncelik, GorevAdresi *pOlusturulanGorev )
{
signed char GeriDon;
grvGKB * pYeniGKB;

	pYeniGKB = GKBveYigitiYerlestir( YigitDerinligi );

	if( pYeniGKB != NULL )
	{
		unsigned char *pYigitBasi;

		GKBDegiskenleriniBaslat( pYeniGKB, GorevAdi, Oncelik );

		#if YigitBuyumesi < 0
		{
			pYigitBasi = pYeniGKB->pStack + ( YigitDerinligi - 1 );
		}
		#else
		{
			pYigitBasi = pYeniGKB->pStack;
		}
		#endif
```

```
                pYeniGKB->pYigitBasi = YigitiBaslat( pYigitBasi, vGorevKodu,
pvParameters );

                KritikBolgeBaslangici();
                {
                        MevcutGorevSayisi++;
                        if( MevcutGorevSayisi == ( unsigned char ) 1 )
                        {
                                AktifGKB =  pYeniGKB;

                                GorevListeleriniBaslat();
                        }
                        else
                        {
                                if( CizelgeleyiciAktif == 0 )
                                {
                                        if( AktifGKB->Oncelik <= Oncelik )
                                        {
                                                AktifGKB = pYeniGKB;
                                        }
                                }
                        }

                        if( pYeniGKB->Oncelik > KullanilanEnYuksekOncelik )
                        {
                                KullanilanEnYuksekOncelik = pYeniGKB->Oncelik;
                        }

                        GoreviKuyrugaEkle( pYeniGKB );

                        GeriDon = Gecerli;
                }
                KritikBolgeSonu();
        }
        else
        {
                GeriDon = errGerekliBellekAyrilamiyor;
        }

        if( GeriDon == Gecerli )
        {
                if( ( void * ) pOlusturulanGorev != NULL )
                {
                        *pOlusturulanGorev = ( GorevAdresi ) pYeniGKB;
                }

                if( CizelgeleyiciAktif != 0 )
                {
                        if( AktifGKB->Oncelik < Oncelik )
                        {
                                GorevDevret();
```

```
                        }
                }
        }

        return GeriDon;
}
/*-----------------------------------------------------------*/

#if ( GOREV_SILME_OZELLIGI == 1 )

        void GorevSil( GorevAdresi pSilinecekGorev )
        {
        grvGKB *pGKB;

                GorevKritikBolge();
                {
                        if( pSilinecekGorev == AktifGKB )
                        {
                                pSilinecekGorev = NULL;
                        }

                        pGKB = AdrestenGKByiAl( pSilinecekGorev );

                        Kuyruktan_Sil( &( pGKB->GenelKuyrukOgesi ) );

                        if( pGKB->OlayListeOgesi.pHazne )
                        {
                                Kuyruktan_Sil( &( pGKB->OlayListeOgesi ) );
                        }

                        Kuyruk_Sonuna_Ekle( ( Kuyruk * )
&SonlanmayiBekleyenGorevler, &( pGKB->GenelKuyrukOgesi ) );

                        ++SilinenGorevler;
                }
                GorevKritikBolgeSonu();

                if( CizelgeleyiciAktif != 0 )
                {
                        if( ( void * ) pSilinecekGorev == NULL )
                        {
                                GorevDevret();
                        }
                }
        }

#endif


#if ( GOREV_BEKLETME_OZELLIGI == 1 )
```

```
	void GorevBeklet( SistemSaatiTipi * const pOncekiUyanmaZamani,
SistemSaatiTipi ZamanArtisi )
	{
	SistemSaatiTipi UyanmaZamani;
	char GecisYapildi, Bekleme = 0;

		GorevleriDurdur();
		{
			UyanmaZamani = *pOncekiUyanmaZamani + ZamanArtisi;

			if( TikSayisi < *pOncekiUyanmaZamani )
			{
				if( ( UyanmaZamani < *pOncekiUyanmaZamani ) && (
UyanmaZamani > TikSayisi ) )
				{
					Bekleme = 1;
				}
			}
			else
			{
				if( ( UyanmaZamani < *pOncekiUyanmaZamani ) || (
UyanmaZamani > TikSayisi ) )
				{
					Bekleme = 1;
				}
			}

			*pOncekiUyanmaZamani = UyanmaZamani;

			if( Bekleme )
			{
				Kuyruktan_Sil( ( Kuyruk_Ogesi * ) &( AktifGKB-
>GenelKuyrukOgesi ) );

				ListeOgeDegeriniAyarla( &( AktifGKB-
>GenelKuyrukOgesi ), UyanmaZamani );

				if( UyanmaZamani < TikSayisi )
				{
					ListeyeEkle( ( Kuyruk * )
Tasma_ErtelenenGorevListesi, ( Kuyruk_Ogesi * ) &( AktifGKB->GenelKuyrukOgesi ) );
				}
				else
				{
					ListeyeEkle( ( Kuyruk * ) ErtelenenGorevListesi, (
Kuyruk_Ogesi * ) &( AktifGKB->GenelKuyrukOgesi ) );
				}
			}
		}
		GecisYapildi = GorevleriBaslat();
```

```
		if( !GecisYapildi )
		{
			GorevDevret();
		}
	}

#endif

#if ( GOREV_ERTELEME_OZELLIGI == 1 )

	void GorevErtele( SistemSaatiTipi ErtelemeTikSayisi )
	{
	SistemSaatiTipi UyanmaZamani;
	signed char GecisYapildi = 0;

		if( ErtelemeTikSayisi > ( SistemSaatiTipi ) 0 )
		{
			GorevleriDurdur();
			{
				UyanmaZamani = TikSayisi + ErtelemeTikSayisi;

				Kuyruktan_Sil( ( Kuyruk_Ogesi * ) &( AktifGKB->GenelKuyrukOgesi ) );

				ListeOgeDegeriniAyarla( &( AktifGKB->GenelKuyrukOgesi ), UyanmaZamani );

				if( UyanmaZamani < TikSayisi )
				{
					ListeyeEkle( ( Kuyruk * ) Tasma_ErtelenenGorevListesi, ( Kuyruk_Ogesi * ) &( AktifGKB->GenelKuyrukOgesi ) );
				}
				else
				{
					ListeyeEkle( ( Kuyruk * ) ErtelenenGorevListesi, ( Kuyruk_Ogesi * ) &( AktifGKB->GenelKuyrukOgesi ) );
				}
			}
			GecisYapildi = GorevleriBaslat();
		}

		if( !GecisYapildi )
		{
			GorevDevret();
		}
	}

#endif
/*-----------------------------------------------------------*/

#if ( ONCELIK_OKUMA_OZELLIGI == 1 )
```

```
	unsigned char GorevOnceligiAl( GorevAdresi pGorev )
	{
	grvGKB *pGKB;
	unsigned char uGeriDon;

		GorevKritikBolge();
		{
			pGKB = AdrestenGKByiAl( pGorev );
			uGeriDon = pGKB->Oncelik;
		}
		GorevKritikBolgeSonu();

		return uGeriDon;
	}

#endif

#if ( ONCELIK_ATAMA_OZELLIGI == 1 )

	void GorevOnceligiAta( GorevAdresi pGorev, unsigned char YeniOncelik )
	{
	grvGKB *pGKB;
	unsigned char MevcutOncelik, GecisGerekli = 0;

		if( YeniOncelik >= EN_YUKSEK_ONCELIK_SEVIYESI )
		{
			YeniOncelik = EN_YUKSEK_ONCELIK_SEVIYESI - 1;
		}

		GorevKritikBolge();
		{
			pGKB = AdrestenGKByiAl( pGorev );

			MevcutOncelik = pGKB->Oncelik;

			if( MevcutOncelik != YeniOncelik )
			{
				if( YeniOncelik > MevcutOncelik )
				{
					if( pGorev != NULL )
					{
						GecisGerekli = 1;
					}
				}
				else if( pGorev == NULL )
				{
					GecisGerekli = 1;
				}

				pGKB->Oncelik = YeniOncelik;
```

```
                                ListeOgeDegeriniAyarla( &( pGKB->OlayListeOgesi ), (
EN_YUKSEK_ONCELIK_SEVIYESI - ( SistemSaatiTipi ) YeniOncelik ) );

                                if( ListeIcerigi( &( HazirGorevlerListesi[ MevcutOncelik ] ),
&( pGKB->GenelKuyrukOgesi ) ) )
                                {
                                        Kuyruktan_Sil( &( pGKB->GenelKuyrukOgesi ) );
                                        GoreviKuyrugaEkle( pGKB );
                                }

                                if( GecisGerekli == 1 )
                                {
                                        GorevDevret();
                                }
                        }
                }
                GorevKritikBolgeSonu();
        }

#endif

#if ( GOREV_DURDURMA_OZELLIGI == 1 )

        void GorevDurdur( GorevAdresi pDurdurulacakGorev )
        {
        grvGKB *pGKB;

                GorevKritikBolge();
                {
                        if( pDurdurulacakGorev == AktifGKB )
                        {
                                pDurdurulacakGorev = NULL;
                        }

                        pGKB = AdrestenGKByiAl( pDurdurulacakGorev );

                        Kuyruktan_Sil( &( pGKB->GenelKuyrukOgesi ) );

                        if( pGKB->OlayListeOgesi.pHazne )
                        {
                                Kuyruktan_Sil( &( pGKB->OlayListeOgesi ) );
                        }

                        Kuyruk_Sonuna_Ekle( ( Kuyruk * ) &DurdurulanGorevListesi, &(
pGKB->GenelKuyrukOgesi ) );
                }
                GorevKritikBolgeSonu();

                if( ( void * ) pDurdurulacakGorev == NULL )
                {
```

```
                        GorevDevret();
                }
        }

#endif

#if ( GOREV_DURDURMA_OZELLIGI == 1 )

        static char GorevDurmusMu( const grvGKB * const pGKB )
        {
        char GeriDon = 0;

                if( ListeIcerigi( &DurdurulanGorevListesi, &( pGKB->GenelKuyrukOgesi )
) != 0 )
                {
                        if( ListeIcerigi( &BekleyenHazirListe, &( pGKB->OlayListeOgesi )
) != 1 )
                        {
                                if( ListeIcerigi( NULL, &( pGKB->OlayListeOgesi ) ) == 1
)
                                {
                                        GeriDon = 1;
                                }
                        }
                }

                return GeriDon;
        }

#endif

#if ( GOREV_DURDURMA_OZELLIGI == 1 )

        void GorevBaslat( GorevAdresi pBaslatilacakGorev )
        {
        grvGKB *pGKB;

                pGKB = ( grvGKB * ) pBaslatilacakGorev;

                if( pGKB != NULL )
                {
                        GorevKritikBolge();
                        {
                                if( GorevDurmusMu( pGKB ) == 1 )
                                {
                                        Kuyruktan_Sil(  &( pGKB->GenelKuyrukOgesi ) );
                                        GoreviKuyrugaEkle( pGKB );

                                        if( pGKB->Oncelik >= AktifGKB->Oncelik )
                                        {
                                                GorevDevret();
```

```
                                        }
                                }
                        }
                        GorevKritikBolgeSonu();
                }
        }

#endif

#if ( ( KESMEDEN_GOREV_BASLATMA_OZELLIGI == 1 ) && (
GOREV_DURDURMA_OZELLIGI == 1 ) )

        char KesmedenGorevBaslat( GorevAdresi pBaslatilacakGorev )
        {
        char GecisGerekli = 0;
        grvGKB *pGKB;

                pGKB = ( grvGKB * ) pBaslatilacakGorev;

                if( GorevDurmusMu( pGKB ) == 1 )
                {
                        if( CizelgeleyiciPasif == ( unsigned char ) 0 )
                        {
                                GecisGerekli = ( pGKB->Oncelik >= AktifGKB->Oncelik );
                                Kuyruktan_Sil(  &( pGKB->GenelKuyrukOgesi ) );
                                GoreviKuyrugaEkle( pGKB );
                        }
                        else
                        {
                                Kuyruk_Sonuna_Ekle( ( Kuyruk * ) &( BekleyenHazirListe
), &( pGKB->OlayListeOgesi ) );
                        }
                }
                return GecisGerekli;
        }

#endif

void CokluGorevBaslat( void )
{
char GeriDon;

        GeriDon = GorevOlustur( BosGorev, ( signed char * ) "IDLE",
BOSTA_YIGIT_BUYUKLUGU, ( void * ) NULL, BosGorevOnceligi, ( GorevAdresi * )
NULL );

        if( GeriDon == Gecerli )
        {
                KesmeleriPasiflestir();

                CizelgeleyiciAktif = 1;
```

```
                TikSayisi = ( SistemSaatiTipi ) 0;

                if( CizelgeleyiciyiBaslat() )
                {

                }
                else
                {

                }
        }
}
/*-------------------------------------------------------------*/

void CokluGorevSonlandir( void )
{
        KesmeleriPasiflestir();
        CizelgeleyiciAktif = 0;
        CizelgeleyiciyiSonlandir();
}
/*-------------------------------------------------------------*/

void GorevleriDurdur( void )
{
        KritikBolgeBaslangici();
                ++CizelgeleyiciPasif;
        KritikBolgeSonu();
}
/*-------------------------------------------------------------*/

signed char GorevleriBaslat( void )
{
register grvGKB *pGKB;
signed char GecisYapildi = 0;

        KritikBolgeBaslangici();
        {
                --CizelgeleyiciPasif;

                if( CizelgeleyiciPasif == ( unsigned char ) 0 )
                {
                        if( MevcutGorevSayisi > ( unsigned char ) 0 )
                        {
                                char GecisGerekli = 0;

                                while( ( pGKB = ( grvGKB * ) Ilk_Kayit_Sahibini_Al(  ( (
Kuyruk * ) &BekleyenHazirListe ) ) ) != NULL )
                                {
                                        Kuyruktan_Sil( &( pGKB->OlayListeOgesi ) );
                                        Kuyruktan_Sil( &( pGKB->GenelKuyrukOgesi ) );
                                        GoreviKuyrugaEkle( pGKB );
```

```
                                        if( pGKB->Oncelik >= AktifGKB->Oncelik )
                                        {
                                                GecisGerekli = 1;
                                        }
                                }

                                if( AtlananTikler > ( unsigned char ) 0 )
                                {
                                        while( AtlananTikler > ( unsigned char ) 0 )
                                        {
                                                TikleriArttir();
                                                --AtlananTikler;
                                        }

                                        GecisGerekli = 1;
                                }

                                if( ( GecisGerekli == 1 ) || ( AtlananGecis == 1 ) )
                                {
                                        GecisYapildi = 1;
                                        AtlananGecis = 0;
                                        GorevDevret();
                                }
                        }
                }
        }
        KritikBolgeSonu();

        return GecisYapildi;
}


SistemSaatiTipi xTaskGetTickCount( void )
{
SistemSaatiTipi Tikler;

        GorevKritikBolge();
        {
                Tikler = TikSayisi;
        }
        GorevKritikBolgeSonu();

        return Tikler;
}
/*-----------------------------------------------------------*/

unsigned char MevcutGorevSayisiniAl( void )
{
unsigned char GorevSayisi;
```

```
	GorevKritikBolge();
		GorevSayisi = MevcutGorevSayisi;
	GorevKritikBolgeSonu();

	return GorevSayisi;
}

inline void TikleriArttir( void )
{
	if( CizelgeleyiciPasif == ( unsigned char ) 0 )
	{
		++TikSayisi;
		if( TikSayisi == ( SistemSaatiTipi ) 0 )
		{
			Kuyruk *pTmp;

			pTmp = ErtelenenGorevListesi;
			ErtelenenGorevListesi = Tasma_ErtelenenGorevListesi;
			Tasma_ErtelenenGorevListesi = pTmp;
	TasmaSayisi++;
		}

		ErtelenmisGorevleriKontrolEt();
	}
	else
	{
		++AtlananTikler;
	}

}

#if ( ( KAYNAKLARI_TEMIZLEME_OZELLIGI == 1 ) && (
GOREV_DURDURMA_OZELLIGI == 1 ) )

	void KaynaklariTemizle( void )
	{
	unsigned int Kuyruk1;
	volatile grvGKB *pGKB;

		Kuyruk1 = ( unsigned int ) KullanilanEnYuksekOncelik + ( unsigned int ) 1;

		do
		{
			Kuyruk1--;

			while( !KuyrukBos( &( HazirGorevlerListesi[ Kuyruk1 ] ) ) )
			{
				SonrakiKayitSahibiniAl( pGKB, &( HazirGorevlerListesi[
Kuyruk1 ] ) );
				Kuyruktan_Sil( ( Kuyruk_Ogesi * ) &( pGKB-
>GenelKuyrukOgesi ) );
```

```
                                GKByiSil( ( grvGKB * ) pGKB );
                        }
                }while( Kuyruk1 > ( unsigned int ) BosGorevOnceligi );

                while( !KuyrukBos( &ErtelenenGorevListesi1 ) )
                {
                        SonrakiKayitSahibiniAl( pGKB, &ErtelenenGorevListesi1 );
                        Kuyruktan_Sil( ( Kuyruk_Ogesi * ) &( pGKB->GenelKuyrukOgesi
) );

                        GKByiSil( ( grvGKB * ) pGKB );
                }

                while( !KuyrukBos( &ErtelenenGorevListesi2 ) )
                {
                        SonrakiKayitSahibiniAl( pGKB, &ErtelenenGorevListesi2 );
                        Kuyruktan_Sil( ( Kuyruk_Ogesi * ) &( pGKB->GenelKuyrukOgesi
) );

                        GKByiSil( ( grvGKB * ) pGKB );
                }

                while( !KuyrukBos( &DurdurulanGorevListesi ) )
                {
                        SonrakiKayitSahibiniAl( pGKB, &DurdurulanGorevListesi );
                        Kuyruktan_Sil( ( Kuyruk_Ogesi * ) &( pGKB->GenelKuyrukOgesi
) );

                        GKByiSil( ( grvGKB * ) pGKB );
                }
        }

#endif

void IcerigiDegistir( void )
{
        if( CizelgeleyiciPasif != ( unsigned char ) 0 )
        {
                AtlananGecis = 1;
                return;
        }

        while( KuyrukBos( &( HazirGorevlerListesi[ HazirEnYuksekOncelik ] ) ) )
        {
                --HazirEnYuksekOncelik;
        }

        SonrakiKayitSahibiniAl( AktifGKB, &( HazirGorevlerListesi[
HazirEnYuksekOncelik ] ) );
        TamponaYaz();
```

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

void ListedekiGorevYeri( const Kuyruk * const pOlayListesi, SistemSaatiTipi
BeklenecekTikSayisi )
{
SistemSaatiTipi UyanmaZamani;

        ListeyeEkle( ( Kuyruk * ) pOlayListesi, ( Kuyruk_Ogesi * ) &( AktifGKB-
>OlayListeOgesi ) );

        Kuyruktan_Sil( ( Kuyruk_Ogesi * ) &( AktifGKB->GenelKuyrukOgesi ) );

        #if ( GOREV_DURDURMA_OZELLIGI == 1 )
        {
                if( BeklenecekTikSayisi == max_gecikme )
                {
                        Kuyruk_Sonuna_Ekle( ( Kuyruk * ) &DurdurulanGorevListesi, (
Kuyruk_Ogesi * ) &( AktifGKB->GenelKuyrukOgesi ) );
                }
                else
                {
                        UyanmaZamani = TikSayisi + BeklenecekTikSayisi;

                        ListeOgeDegeriniAyarla( &( AktifGKB->GenelKuyrukOgesi ),
UyanmaZamani );

                        if( UyanmaZamani < TikSayisi )
                        {
                                ListeyeEkle( ( Kuyruk * ) Tasma_ErtelenenGorevListesi, (
Kuyruk_Ogesi * ) &( AktifGKB->GenelKuyrukOgesi ) );
                        }
                        else
                        {
                                ListeyeEkle( ( Kuyruk * ) ErtelenenGorevListesi, (
Kuyruk_Ogesi * ) &( AktifGKB->GenelKuyrukOgesi ) );
                        }
                }
        }
        #else
        {
                        UyanmaZamani = TikSayisi + BeklenecekTikSayisi;

                        ListeOgeDegeriniAyarla( &( AktifGKB->GenelKuyrukOgesi ),
UyanmaZamani );

                        if( UyanmaZamani < TikSayisi )
                        {
                                ListeyeEkle( ( Kuyruk * ) Tasma_ErtelenenGorevListesi, (
Kuyruk_Ogesi * ) &( AktifGKB->GenelKuyrukOgesi ) );
                        }
```

```
                        else
                        {
                                ListeyeEkle( ( Kuyruk * ) ErtelenenGorevListesi, (
Kuyruk_Ogesi * ) &( AktifGKB->GenelKuyrukOgesi ) );
                        }
        }
        #endif
}

signed char OlayListesindenSil( const Kuyruk * const pOlayListesi )
{
grvGKB *pBostaGKB;
char GeriDon;

        pBostaGKB = ( grvGKB * ) Ilk_Kayit_Sahibini_Al( pOlayListesi );
        Kuyruktan_Sil( &( pBostaGKB->OlayListeOgesi ) );

        if( CizelgeleyiciPasif == ( unsigned char ) 0 )
        {
                Kuyruktan_Sil( &( pBostaGKB->GenelKuyrukOgesi ) );
                GoreviKuyrugaEkle( pBostaGKB );
        }
        else
        {
                Kuyruk_Sonuna_Ekle( ( Kuyruk * ) &( BekleyenHazirListe ), &(
pBostaGKB->OlayListeOgesi ) );
        }

        if( pBostaGKB->Oncelik >= AktifGKB->Oncelik )
        {
                GeriDon = 1;
        }
        else
        {
                GeriDon = 0;
        }

        return GeriDon;
}

void ZamanasimiDurumunuAyarla( ZamanasimiTipi * const pZamanasimi )
{
   pZamanasimi->TasmaSayaci = TasmaSayisi;
   pZamanasimi->BaslamaZamani = TikSayisi;
}

char ZamanasimiKontrol( ZamanasimiTipi * const pZamanasimi, SistemSaatiTipi * const
pBeklenecekTikSayisi )
{
char GeriDon;
```

```
        #if ( GOREV_DURDURMA_OZELLIGI == 1 )
                if( *pBeklenecekTikSayisi == max_gecikme )
                {
                        GeriDon = 0;
                }
                else
        #endif

   if( ( TasmaSayisi != pZamanasimi->TasmaSayaci ) && ( TikSayisi >= pZamanasimi-
>BaslamaZamani ) )
   {
      GeriDon = 1;
   }
   else if( ( TikSayisi - pZamanasimi->BaslamaZamani ) < *pBeklenecekTikSayisi )
   {
      *pBeklenecekTikSayisi -= ( TikSayisi - pZamanasimi->BaslamaZamani );
      ZamanasimiDurumunuAyarla( pZamanasimi );
      GeriDon = 0;
   }
   else
   {
      GeriDon = 1;
   }

   return GeriDon;
}

void Atlanan_Gecis( void )
{
        AtlananGecis = 1;
}


static GorevFonksiyonu( BosGorev, pvParameters )
{
        ( void ) pvParameters;

        for( ;; )
        {
                SonlanmayiBekleyenGorevleriKontrolEt();

                #if ( CALISMA_PRENSIBI == 0 )
                {
                        GorevDevret();
                }
                #endif

                #if ( CALISMA_PRENSIBI == 1 )
                {
                        if( MevcutListeUzunlugu( &( HazirGorevlerListesi[
BosGorevOnceligi ] ) ) > ( unsigned char ) 1 )
```

```
                        {
                                GorevDevret();
                        }
                }
                #endif

        }
}

static void GKBDegiskenleriniBaslat( grvGKB *pGKB, const signed char * const GorevAdi,
unsigned char Oncelik )
{

        strncpy( ( char * ) pGKB->GorevAdi, ( const char * ) GorevAdi, ( unsigned int )
MAKSIMUM_ISIM_UZUNLUGU );
        pGKB->GorevAdi[ ( unsigned int ) MAKSIMUM_ISIM_UZUNLUGU - ( unsigned
int ) 1 ] = '\0';

        if( Oncelik >= EN_YUKSEK_ONCELIK_SEVIYESI )
        {
                Oncelik = EN_YUKSEK_ONCELIK_SEVIYESI - 1;
        }

        pGKB->Oncelik = Oncelik;

        Kuyruk_Ogesi_Baslat( &( pGKB->GenelKuyrukOgesi ) );
        Kuyruk_Ogesi_Baslat( &( pGKB->OlayListeOgesi ) );

        ListeOgesiSahibiniAyarla( &( pGKB->GenelKuyrukOgesi ), pGKB );

        ListeOgeDegeriniAyarla( &( pGKB->OlayListeOgesi ),
EN_YUKSEK_ONCELIK_SEVIYESI - ( SistemSaatiTipi ) Oncelik );
        ListeOgesiSahibiniAyarla( &( pGKB->OlayListeOgesi ), pGKB );
}

static void GorevListeleriniBaslat( void )
{
unsigned char Oncelik;

        for( Oncelik = 0; Oncelik < EN_YUKSEK_ONCELIK_SEVIYESI; Oncelik++ )
        {
                Kuyruk_Baslat( ( Kuyruk * ) &( HazirGorevlerListesi[ Oncelik ] ) );
        }

        Kuyruk_Baslat( ( Kuyruk * ) &ErtelenenGorevListesi1 );
        Kuyruk_Baslat( ( Kuyruk * ) &ErtelenenGorevListesi2 );
        Kuyruk_Baslat( ( Kuyruk * ) &BekleyenHazirListe );

        #if ( GOREV_SILME_OZELLIGI == 1 )
        {
                Kuyruk_Baslat( ( Kuyruk * ) &SonlanmayiBekleyenGorevler );
```

```
	}
	#endif

	#if ( GOREV_DURDURMA_OZELLIGI == 1 )
	{
		Kuyruk_Baslat( ( Kuyruk * ) &DurdurulanGorevListesi );
	}
	#endif

	ErtelenenGorevListesi = &ErtelenenGorevListesi1;
	Tasma_ErtelenenGorevListesi = &ErtelenenGorevListesi2;
}

static void SonlanmayiBekleyenGorevleriKontrolEt( void )
{
	#if ( GOREV_SILME_OZELLIGI == 1 )
	{
		char ListeBos;

		if( SilinenGorevler > ( unsigned char ) 0 )
		{
			GorevleriDurdur();
				ListeBos = KuyrukBos( &SonlanmayiBekleyenGorevler );

			GorevleriBaslat();

			if( !ListeBos )
			{
				grvGKB *pGKB;

				KritikBolgeBaslangici();
				{
					pGKB = ( grvGKB * ) Ilk_Kayit_Sahibini_Al( ( (
Kuyruk * ) &SonlanmayiBekleyenGorevler ) );
					Kuyruktan_Sil( &( pGKB->GenelKuyrukOgesi ) );
					--MevcutGorevSayisi;
					--SilinenGorevler;
				}
				KritikBolgeSonu();

				GKByiSil( pGKB );
			}
		}
	}
	#endif
}

static grvGKB *GKBveYigitiYerlestir( unsigned int YigitDerinligi )
{
grvGKB *pYeniGKB;
```

```
	pYeniGKB = ( grvGKB * ) pYerlestir( sizeof( grvGKB ) );

	if( pYeniGKB != NULL )
	{
		pYeniGKB->pStack = ( unsigned char * ) pYerlestir( ( ( size_t
)YigitDerinligi ) * sizeof( unsigned char ) );

		if( pYeniGKB->pStack == NULL )
		{
			BellekBosalt( pYeniGKB );
			pYeniGKB = NULL;
		}
		else
		{
			memset( pYeniGKB->pStack, YigitDoldurmaBayti, YigitDerinligi *
sizeof( unsigned char ) );
		}
	}

	return pYeniGKB;
}

#if ( ( GOREV_SILME_OZELLIGI == 1 ) || ( KAYNAKLARI_TEMIZLEME_OZELLIGI
== 1 ) )

	static void GKByiSil( grvGKB *pGKB )
	{
		BellekBosalt( pGKB->pStack );
		BellekBosalt( pGKB );
	}

#endif


#if ( MevcutGorevAdresiniAlmaOzelligi == 1 )

	GorevAdresi MevcutGorevAdresiniAl( void )
	{
	GorevAdresi GeriDon;

		KritikBolgeBaslangici();
		{
			GeriDon = ( GorevAdresi ) AktifGKB;
		}
		KritikBolgeSonu();

		return GeriDon;
	}

#endif
```

Cs_pic.c dosyası

```
#include "kontrol.h"
#include "cekirdek.h"

#include "timers.h" //std pic c18 library

#define OsilatorOlcegi                  ( ( unsigned long ) 4 )

#define KesmeBaslangicDegeri                    0xc0

#define GlobalKesmeDegeri                               0x80

#define DegismeyenKesmeler                      0x00

#define DerleyiciBellekBuyuklugu        ( ( unsigned char ) 0x13 )

typedef void grvGKB;
extern volatile grvGKB * volatile AktifGKB;

extern volatile GorevAdresi GorevAdres1;

static void ZamanlayiciKesmesiniAyarla( void );

static void TikKesmesi( void );

int a;

void SistemKesme( void );
extern void Kesme (void);

static void DusukOncelikliKesme( void );

#define IcerigiSakla( SecilenKesmeBayraklari )
                \
{
                                                                        \
        _asm
                                                                \
                MOVFF           WREG, PREINC1
                                                                \
                MOVFF   STATUS, PREINC1
                                                        \
                MOVFF           INTCON, WREG
                                                                \
                IORLW           SecilenKesmeBayraklari
                                                        \
                MOVFF           WREG, PREINC1
                                                                \
        _endasm
        \
```

```
                                                                \
KesmeleriPasiflestir();
                                \

                                                                \
_asm
                                                                \
        MOVFF       BSR, PREINC1
                                                                \
        MOVFF       FSR2L, PREINC1
                                                                \
        MOVFF       FSR2H, PREINC1
                                                                \
        MOVFF       FSR0L, PREINC1
                                                                \
        MOVFF       FSR0H, PREINC1
                                                                \
        MOVFF       TABLAT, PREINC1
                                                                \
        MOVFF       TBLPTRU, PREINC1
                                                                \
        MOVFF       TBLPTRH, PREINC1
                                                                \
        MOVFF       TBLPTRL, PREINC1
                                                                \
        MOVFF       PRODH, PREINC1
                                                                \
        MOVFF       PRODL, PREINC1
                                                                \
        MOVFF       PCLATU, PREINC1
                                                                \
        MOVFF       PCLATH, PREINC1
                                                                \
        CLRF   FSR0L, 0
                                                                \
        CLRF   FSR0H, 0
                                                                \
        MOVFF       POSTINC0, PREINC1
                                                                \
        MOVFF       POSTINC0, PREINC1
                                                                \
        MOVFF       POSTINC0, PREINC1
                                                                \
        MOVFF       POSTINC0, PREINC1
                                                                \
        MOVFF       POSTINC0, PREINC1
                                                                \
        MOVFF       POSTINC0, PREINC1
                                                                \
```

```
        MOVFF       POSTINC0, PREINC1
                                        \
        MOVFF       POSTINC0, PREINC1
                                        \
        MOVFF       POSTINC0, PREINC1
                                        \
        MOVFF       POSTINC0, PREINC1
                                        \
        MOVFF       POSTINC0, PREINC1
                                        \
        MOVFF       POSTINC0, PREINC1
                                        \
        MOVFF       POSTINC0, PREINC1
                                        \
        MOVFF       POSTINC0, PREINC1
                                        \
        MOVFF       POSTINC0, PREINC1
                                        \
        MOVFF       POSTINC0, PREINC1
                                        \
        MOVFF       POSTINC0, PREINC1
                                        \
        MOVFF       POSTINC0, PREINC1
                                        \
        MOVFF       POSTINC0, PREINC1
                                        \
        MOVFF       POSTINC0, PREINC1
                                        \
        MOVFF       POSTINC0, PREINC1
                                        \
        MOVFF       INDF0, PREINC1
                                            \
        MOVFF       FSR0L, PREINC1
                                            \
        MOVFF       FSR0H, PREINC1
                                            \
        MOVFF       STKPTR, FSR0L
                                            \
_endasm
                                                \

                                                \
        while( STKPTR > ( unsigned char ) 0 )
                \
        {
                                                \
                _asm
                                                    \
                        MOVFF       TOSL, PREINC1
                                                    \
                        MOVFF       TOSH, PREINC1
                                                    \
                        MOVFF       TOSU, PREINC1
                                                    \
```

```
                POP                                                     \
        _endasm                                                         \
    }                                                                   \
                                                                        \
    _asm                                                                \
        MOVFF   FSR0L, PREINC1                                          \
        MOVF PREINC1, 1, 0                                              \
    _endasm                                                             \
                                                                        \
    _asm                                                                \
        MOVFF   AktifGKB, FSR0L                                         \
        MOVFF   AktifGKB + 1, FSR0H                                     \
        MOVFF   FSR1L, POSTINC0                                         \
        MOVFF   FSR1H, POSTINC0                                         \
    _endasm                                                             \
}

#define IcerigiGeriYukle()                                              \
{                                                                       \
    _asm                                                                \
        MOVFF   AktifGKB, FSR0L                                         \
        MOVFF   AktifGKB + 1, FSR0H                                     \
                                                                        \
        MOVFF   POSTINC0, FSR1L                                         \
        MOVFF   POSTINC0, FSR1H                                         \
                                                                        \
```

```
        MOVFF       POSTDEC1, FSR0L
                                                \
        MOVFF       POSTDEC1, FSR0L
                                                \
_endasm
                                                \
                                                \
STKPTR = 0;
                                                \
                                                \
while( STKPTR < FSR0L )
                                                \
{
                                                \
        _asm
                                                \
                PUSH
                                                \
                MOVF POSTDEC1, 0, 0
                                                \
                MOVWF       TOSU, 0
                                                \
                MOVF POSTDEC1, 0, 0
                                                \
                MOVWF       TOSH, 0
                                                \
                MOVF POSTDEC1, 0, 0
                                                \
                MOVWF       TOSL, 0
                                                \
        _endasm
                                                \
}
                                                \
                                                \
_asm
                                                \
        MOVFF       POSTDEC1, FSR0H
                                                \
        MOVFF       POSTDEC1, FSR0L
                                                \
        MOVFF       POSTDEC1, POSTDEC0
                                                \
        MOVFF       POSTDEC1, POSTDEC0
                                                \
        MOVFF       POSTDEC1, POSTDEC0
                                                \
```

```
MOVFF       POSTDEC1, POSTDEC0
                                \
MOVFF       POSTDEC1, POSTDEC0
                                \
MOVFF       POSTDEC1, POSTDEC0
                                \
MOVFF       POSTDEC1, POSTDEC0
                                \
MOVFF       POSTDEC1, POSTDEC0
                                \
MOVFF       POSTDEC1, POSTDEC0
                                \
MOVFF       POSTDEC1, POSTDEC0
                                \
MOVFF       POSTDEC1, POSTDEC0
                                \
MOVFF       POSTDEC1, POSTDEC0
                                \
MOVFF       POSTDEC1, POSTDEC0
                                \
MOVFF       POSTDEC1, POSTDEC0
                                \
MOVFF       POSTDEC1, POSTDEC0
                                \
MOVFF       POSTDEC1, POSTDEC0
                                \
MOVFF       POSTDEC1, POSTDEC0
                                \
MOVFF       POSTDEC1, POSTDEC0
                                \
MOVFF       POSTDEC1, POSTDEC0
                                \
MOVFF       POSTDEC1, POSTDEC0
                                \
MOVFF       POSTDEC1, POSTDEC0
                                \
MOVFF       POSTDEC1, INDF0
                                \
MOVFF       POSTDEC1, PCLATH
                                  \
MOVFF       POSTDEC1, PCLATU
                                  \
MOVFF       POSTDEC1, PRODL
                                \
MOVFF       POSTDEC1, PRODH
                                \
MOVFF       POSTDEC1, TBLPTRL
                                \
MOVFF       POSTDEC1, TBLPTRH
                                \
MOVFF       POSTDEC1, TBLPTRU
                                \
MOVFF       POSTDEC1, TABLAT
                                  \
```

```
        MOVFF       POSTDEC1, FSR0H
                                                    \
        MOVFF       POSTDEC1, FSR0L
                                                    \
        MOVFF       POSTDEC1, FSR2H
                                                    \
        MOVFF       POSTDEC1, FSR2L
                                                    \
        MOVFF       POSTDEC1, BSR
                                                    \
        MOVFF       POSTDEC1, WREG
                                                    \
    _endasm
                                                        \

                                                        \
    if( WREG & GlobalKesmeDegeri )
                        \
    {
                                                        \
        _asm
                                                    \
            MOVFF       POSTDEC1, STATUS
                                                        \
            MOVFF       POSTDEC1, WREG
                                                    \
            RETFIE      0
                                                        \
        _endasm
                                                        \
    }
                                                        \
    else
                                                    \
    {
                                                        \
        _asm
                                                    \
            MOVFF       POSTDEC1, STATUS
                                                        \
            MOVFF       POSTDEC1, WREG
                                                    \
            RETURN      0
                                                        \
        _endasm
                                                        \
    }
                                                        \
}
```

```
unsigned char *YigitiBaslat( unsigned char *pYigitBasi, dGorevKodu pKod, void
*pvParameters )
{
unsigned long uAdres;
unsigned char uBlok;

        *pYigitBasi = 0x11;
        pYigitBasi++;
        *pYigitBasi = 0x22;
        pYigitBasi++;
        *pYigitBasi = 0x33;
        pYigitBasi++;


        uAdres = ( unsigned long ) pvParameters;
        *pYigitBasi = ( unsigned char ) ( uAdres & ( unsigned long ) 0x00ff );
        pYigitBasi++;

        uAdres >>= 8;
        *pYigitBasi = ( unsigned char ) ( uAdres & ( unsigned long ) 0x00ff );
        pYigitBasi++;

        *pYigitBasi = 0x44;
        pYigitBasi++;

        *pYigitBasi = ( unsigned char ) 0x66; /* WREG. */
        pYigitBasi++;

        *pYigitBasi = ( unsigned char ) 0xcc; /* Status. */
        pYigitBasi++;

        *pYigitBasi = ( unsigned char ) KesmeBaslangicDegeri; /* INTCON */
        pYigitBasi++;

        *pYigitBasi = ( unsigned char ) 0x11; /* BSR. */
        pYigitBasi++;

        *pYigitBasi = ( unsigned char ) 0x22; /* FSR2L. */
        pYigitBasi++;

        *pYigitBasi = ( unsigned char ) 0x33; /* FSR2H. */
        pYigitBasi++;

        *pYigitBasi = ( unsigned char ) 0x44; /* FSR0L. */
        pYigitBasi++;

        *pYigitBasi = ( unsigned char ) 0x55; /* FSR0H. */
        pYigitBasi++;

        *pYigitBasi = ( unsigned char ) 0x66; /* TABLAT. */
        pYigitBasi++;
```

```
*pYigitBasi = ( unsigned char ) 0x00; /* TBLPTRU. */
pYigitBasi++;

*pYigitBasi = ( unsigned char ) 0x88; /* TBLPTRUH. */
pYigitBasi++;

*pYigitBasi = ( unsigned char ) 0x99; /* TBLPTRUL. */
pYigitBasi++;

*pYigitBasi = ( unsigned char ) 0xaa; /* PRODH. */
pYigitBasi++;

*pYigitBasi = ( unsigned char ) 0xbb; /* PRODL. */
pYigitBasi++;

*pYigitBasi = ( unsigned char ) 0x00; /* PCLATU. */
pYigitBasi++;

*pYigitBasi = ( unsigned char ) 0x00; /* PCLATH. */
pYigitBasi++;

for( uBlok = 0; uBlok <= DerleyiciBellekBuyuklugu; uBlok++ )
{
        *pYigitBasi = ( unsigned char ) uBlok;
        *pYigitBasi++;
}

*pYigitBasi = ( unsigned char ) DerleyiciBellekBuyuklugu; /* Low. */
pYigitBasi++;

*pYigitBasi = ( unsigned char ) 0x00; /* High. */
pYigitBasi++;

uAdres = ( unsigned long ) pKod;

*pYigitBasi = ( unsigned char ) ( uAdres & ( unsigned long ) 0x00ff );
pYigitBasi++;
uAdres >>= 8;

*pYigitBasi = ( unsigned char ) ( uAdres & ( unsigned long ) 0x00ff );
pYigitBasi++;
uAdres >>= 8;

*pYigitBasi = ( unsigned char ) ( uAdres & ( unsigned long ) 0x00ff );
pYigitBasi++;

*pYigitBasi = ( unsigned char ) 1;
pYigitBasi++;

return pYigitBasi;
```

```
}

char CizelgeleyiciyiBaslat( void )
{
	ZamanlayiciKesmesiniAyarla();

	IcerigiGeriYukle();

	if( PIR1bits.CCP1IF )
	{
		_asm
			goto TikKesmesi
		_endasm
	}
	( void ) TikKesmesi;

	return 1;
}

void CizelgeleyiciyiSonlandir( void )
{

}

void GecisYap( void )
{
	IcerigiSakla( DegismeyenKesmeler );

	IcerigiDegistir();

	IcerigiGeriYukle();
}

//#pragma code high_vector=0x08

#pragma code InterruptVectorHigh = 0x08

void
InterruptVectorHigh (void)
{
	_asm
	goto SistemKesme
	_endasm
}


#pragma code
//#pragma interrupt Kesme

void SistemKesme (void)
{
```

```
        T0CON = 0x98;
        INTCONbits.TMR0IE = 0;
        TMR0L = 0;
        TMR0H = 0;
        if( PIR1bits.CCP1IF )
        {
        INTCONbits.GIE = 0;
        IcerigiSakla( GlobalKesmeDegeri );
        PIR1bits.CCP1IF = 0;

        TikleriArttir();

        #if CALISMA_PRENSIBI == 1
        {
                IcerigiDegistir();
        }
        #endif
        IcerigiGeriYukle();
        INTCONbits.GIE = 1;
        }
        Kesme();
}

static void TikKesmesi( void )
{
        IcerigiSakla( GlobalKesmeDegeri );
        PIR1bits.CCP1IF = 0;

        TikleriArttir();

        #if CALISMA_PRENSIBI == 1
        {
                IcerigiDegistir();
        }
        #endif

        IcerigiGeriYukle();
}


static void ZamanlayiciKesmesiniAyarla( void )
{
const unsigned long SabitKarsilastirmaDegeri = ( ( SAAT_FREKANSI / OsilatorOlcegi ) /
OS_SAAT_HIZI );
unsigned long KarsilastirmaDegeri;
unsigned char cBayt;

        TMR1H = ( unsigned char ) 0x00;
        TMR1L = ( unsigned char ) 0x00;
        KarsilastirmaDegeri = SabitKarsilastirmaDegeri;
        CCPR1L = ( unsigned char ) ( KarsilastirmaDegeri & ( unsigned long ) 0xff );
```

```
        KarsilastirmaDegeri >>= ( unsigned long ) 8;
        CCPR1H = ( unsigned char ) ( KarsilastirmaDegeri & ( unsigned long ) 0xff );

        CCP1CONbits.CCP1M0 = 1;
        CCP1CONbits.CCP1M1 = 1;
        CCP1CONbits.CCP1M2 = 0;
        CCP1CONbits.CCP1M3 = 1;
        PIE1bits.CCP1IE = 1;

        INTCONbits.GIEL = 1;
        OpenTimer1( T1_16BIT_RW & T1_SOURCE_INT & T1_PS_1_1 &
T1_CCP1_T3_CCP2 );
}
```

Cekirdek.h dosyası

```
#ifndef CEKIRDEK_H
#define CEKIRDEK_H

#include "pic.h"
#include "kuyruk.h"

#ifdef __cplusplus
extern "C" {
#endif

typedef void * GorevAdresi;

typedef struct xZamanAsimi
{
   char TasmaSayaci;
   SistemSaatiTipi  BaslamaZamani;
} ZamanasimiTipi;

#define BosGorevOnceligi                    ( ( unsigned char ) 0 )

#define GorevDevret()                              Gecis()

#define GorevKritikBolge()          KritikBolgeBaslangici()

#define GorevKritikBolgeSonu()                     KritikBolgeSonu()

#define gorevKesmeleriPasiflestir()    KesmeleriPasiflestir()

#define gorevKesmeleriEtkinlestir()          KesmeleriEtkinlestir()

#define CizelgeleyiciBaslamadi 0
#define CizelgeleyiciCalisiyor      1
#define CizelgeleyiciDuruyor        2

signed char GorevOlustur( dGorevKodu vGorevKodu, const signed char * const GorevAdi,
unsigned int YigitDerinligi, void *pvParameters, unsigned char Oncelik, GorevAdresi
*pOlusturulanGorevler );

void GorevSil( GorevAdresi pGorev );


void GorevErtele( SistemSaatiTipi ErtelemeTikSayisi );

void GorevBeklet( SistemSaatiTipi * const pOncekiUyanmaZamani, SistemSaatiTipi
ZamanArtisi );

unsigned char GorevOnceligiAl( GorevAdresi pGorev );

void GorevOnceligiAta( GorevAdresi pGorev, unsigned char YeniOncelik );
```

```
void GorevDurdur( GorevAdresi pDurdurulacakGorev );

void GorevBaslat( GorevAdresi pBaslatilacakGorev );

char KesmedenGorevBaslat( GorevAdresi pBaslatilacakGorev );

void CokluGorevBaslat( void );

void CokluGorevSonlandir( void );

void GorevleriDurdur( void );

signed char GorevleriBaslat( void );

unsigned char MevcutGorevSayisiniAl( void );

void GorevListesi( signed char *pTamponaYaz );

inline void TikleriArttir( void );

void ListedekiGorevYeri( const Kuyruk * const pOlayListesi, SistemSaatiTipi
BeklenecekTikSayisi );

signed char OlayListesindenSil( const Kuyruk * const pOlayListesi );

void KaynaklariTemizle( void );

inline void IcerigiDegistir( void );

GorevAdresi MevcutGorevAdresiniAl( void );

void ZamanasimiDurumunuAyarla( ZamanasimiTipi * const pZamanasimi );

char ZamanasimiKontrol( ZamanasimiTipi * const pZamanasimi, SistemSaatiTipi * const
pBeklenecekTikSayisi );

void Atlanan_Gecis( void );

char CizelgeleyiciDurumunuAl( void );

#ifdef __cplusplus
}
#endif
#endif /* CEKIRDEK_H */
```

Kuyruk.h dosyası

```
#ifndef KUYRUK_H
#define KUYRUK_H

#ifdef __cplusplus
extern "C" {
#endif

struct xListeOgesi
{
	SistemSaatiTipi Oge_Degeri;
	volatile struct xListeOgesi * pSonraki;
	volatile struct xListeOgesi * pOnceki;
	void * pSahibi;
	void * pHazne;
};
typedef struct xListeOgesi Kuyruk_Ogesi;

struct MiniListeOgesi
{
	SistemSaatiTipi Oge_Degeri;
	volatile struct xListeOgesi *pSonraki;
	volatile struct xListeOgesi *pOnceki;
};
typedef struct MiniListeOgesi xMiniListeOgesi;

typedef struct Kuyruk
{
	volatile unsigned char Oge_Sayisi;
	volatile Kuyruk_Ogesi * pEndeks;
	volatile xMiniListeOgesi Kuyruk_Sonu;
} Kuyruk;

#define ListeOgesiSahibiniAyarla( pKuyruk_Ogesi, pxSahibi )		( pKuyruk_Ogesi )->pSahibi = ( void * ) pxSahibi

#define ListeOgeDegeriniAyarla( pKuyruk_Ogesi, xDeger )		( pKuyruk_Ogesi )->Oge_Degeri = xDeger

#define Liste_Oge_Degerini_Al( pKuyruk_Ogesi )		( ( pKuyruk_Ogesi )->Oge_Degeri )

#define KuyrukBos( pKuyruk )		( ( pKuyruk )->Oge_Sayisi == ( unsigned char ) 0 )

#define MevcutListeUzunlugu( pKuyruk )		( ( pKuyruk )->Oge_Sayisi )

#define SonrakiKayitSahibiniAl( pGKB, pKuyruk )
		\
```

```
{                                                                                                   \
Kuyruk * const pxSabitListe = pKuyruk;                                                              \
        ( pxSabitListe )->pEndeks = ( pxSabitListe )->pEndeks->pSonraki;                            \
        if( ( pxSabitListe )->pEndeks == ( Kuyruk_Ogesi * ) &( ( pxSabitListe )->Kuyruk_Sonu ) )    \
        {                                                                                           \
                ( pxSabitListe )->pEndeks = ( pxSabitListe )->pEndeks->pSonraki;                    \
        }                                                                                           \
        pGKB = ( pxSabitListe )->pEndeks->pSahibi;                                                  \
}


#define Ilk_Kayit_Sahibini_Al( pKuyruk )  ( ( pKuyruk->Oge_Sayisi != ( unsigned char ) 0 ) ? ( (&( pKuyruk->Kuyruk_Sonu ))->pSonraki->pSahibi ) : ( NULL ) )

#define ListeIcerigi( pKuyruk, pKuyruk_Ogesi ) ( ( pKuyruk_Ogesi )->pHazne == ( void * ) pKuyruk )

void ListeyiBaslat( Kuyruk *pKuyruk );

void Kuyruk_Ogesi_Baslat( Kuyruk_Ogesi *pOge );

void ListeyeEkle( Kuyruk *pKuyruk, Kuyruk_Ogesi *pYeni_Oge );

void Kuyruk_Sonuna_Ekle( Kuyruk *pKuyruk, Kuyruk_Ogesi *pYeni_Oge );

void Kuyruktan_Sil( Kuyruk_Ogesi *pSilinecek_Oge );

#ifdef __cplusplus
}
#endif

#endif
```

Macro.h dosyası

```
#ifndef MACRO_H
#define MACRO_H

#if( _16_BIT_SISTEM_SAATI == 1 )
        typedef unsigned int SistemSaatiTipi;
        #define max_gecikme ( SistemSaatiTipi ) 0xffff
#else
        typedef unsigned long SistemSaatiTipi;
        #define max_gecikme ( SistemSaatiTipi ) 0xffffffff
#endif

#define BAYT_SIRALAMASI                 1
#define GlobalKesmeEtkinBiti  0x80
#define YigitBuyumesi                   1
#define TikHiziMS                       ( ( SistemSaatiTipi ) 1000 / OS_SAAT_HIZI )


#define KesmeleriPasiflestir()          INTCONbits.GIEH = 0;
#define KesmeleriEtkinlestir()          INTCONbits.GIEH = 1;

#define KritikBolgeBaslangici()                 POSTINC1 = INTCON;
                \
                                                                INTCONbits.GIEH
= 0;

#define KritikBolgeSonu()                       _asm
                                \
                                                                        MOVF
        POSTDEC1, 1, 0                                  \
                                                                INTCONbits.GIEH
                                                                        \
                                                                if( INDF1 &
GlobalKesmeEtkinBiti )          \
                                                                {
                                                                        \

        KesmeleriEtkinlestir();                 \
                                                                }

extern void GecisYap( void );
#define Gecis()                         GecisYap()

#define GorevIslev( vFunction, pvParameters ) void vFunction( void *pvParameters )
#define GorevFonksiyonu( vFunction, pvParameters ) void vFunction( void *pvParameters )


#define inline

#define portNOP()                               _asm    \
```

```
                                                NOP \
                                            _endasm

#endif /* MACRO_H */
```

Pic.h dosyası

```
#ifndef PIC_H
#define PIC_H

#include "..\inc\macro.h"

#ifdef __cplusplus
extern "C" {
#endif

unsigned char *YigitiBaslat( unsigned char *pYigitBasi, dGorevKodu pKod, void
*pvParameters );

void *pYerlestir( size_t xSize );
void BellekBosalt( void *pv );
void Bloklari_Baslat( void );

char CizelgeleyiciyiBaslat( void );

void CizelgeleyiciyiSonlandir( void );

#ifdef __cplusplus
}
#endif

#endif /* PIC_H */
```

Sayi.h dosyası

```
#ifndef SAYI_H
#define SAYI_H

void vStartIntegerMathTasks( unsigned char Oncelik );
char xAreIntegerMathsTaskStillRunning( void );

#endif
```

Tanimlar.h dosyası

```
#ifndef TANIMLAR_H
#define TANIMLAR_H

typedef void (*dGorevKodu)( void * );

#define Gecerli                                    ( 1 )
#define Gecersiz                                   ( 0 )
#define errKuyrukBos                               ( 0 )
#define errKuyrukDolu                              ( 0 )

#define errGerekliBellekAyrilamiyor    ( -1 )
#define errCalistirilacakGorevYok                  ( -2 )
#define errKuyrukBloke                             ( -4 )
#define errKuyrukGecis                             ( -5 )

#endif /* TANIMLAR_H */
```

Yapilandirma.h dosyası

```
#ifndef YAPILANDIRMA_H
#define YAPILANDIRMA_H

#include <p18cxxx.h>

#define CALISMA_PRENSIBI                        1 //PREEMPTION
#define OS_SAAT_HIZI                            ( ( SistemSaatiTipi ) 1000 )
//SYSTEM TICK RATE
#define SAAT_FREKANSI                           ( ( unsigned long ) 4000000
)
#define EN_YUKSEK_ONCELIK_SEVIYESI      ( ( unsigned char ) 4 )
#define MINIMUM_YIGIT_BUYUKLUGU                 ( 105 )
#define TOPLAM_HAFIZA_BUYUKLUGU                 ( ( size_t ) 1024 )
#define MAKSIMUM_ISIM_UZUNLUGU                  ( 4 )

#define _16_BIT_SISTEM_SAATI            1

#define ONCELIK_ATAMA_OZELLIGI                  1
#define ONCELIK_OKUMA_OZELLIGI                  0
#define GOREV_SILME_OZELLIGI            1
#define KAYNAKLARI_TEMIZLEME_OZELLIGI   0
#define GOREV_DURDURMA_OZELLIGI                 1
#define GOREV_BEKLETME_OZELLIGI                 1
#define GOREV_ERTELEME_OZELLIGI                 1

#endif /* YAPILANDIRMA_H */
```

Kontrol.h dosyası

```
#ifndef INC_FREERTOS_H
#define INC_FREERTOS_H

#include <stddef.h>

#include "tanimlar.h"
#include "yapilandirma.h"
#include "pic.h"

#ifndef CALISMA_PRENSIBI
	#error Eksik Tanimlama:  CALISMA_PRENSIBI tanimlanmali
#endif

#ifndef ONCELIK_ATAMA_OZELLIGI
	#error Eksik Tanimlama:  ONCELIK_ATAMA_OZELLIGI tanimlanmali
#endif

#ifndef ONCELIK_OKUMA_OZELLIGI
	#error Eksik Tanimlama:  ONCELIK_OKUMA_OZELLIGI tanimlanmali
#endif

#ifndef GOREV_SILME_OZELLIGI
	#error Eksik Tanimlama:  GOREV_SILME_OZELLIGI tanimlanmali
#endif

#ifndef KAYNAKLARI_TEMIZLEME_OZELLIGI
	#error Eksik Tanimlama:  KAYNAKLARI_TEMIZLEME_OZELLIGI tanimlanmali
#endif

#ifndef GOREV_DURDURMA_OZELLIGI
	#error Eksik Tanimlama:  GOREV_DURDURMA_OZELLIGI  tanimlanmali
#endif

#ifndef GOREV_BEKLETME_OZELLIGI
	#error Eksik Tanimlama:  GOREV_BEKLETME_OZELLIGI tanimlanmali
#endif

#ifndef GOREV_ERTELEME_OZELLIGI
	#error Eksik Tanimlama:  GOREV_ERTELEME_OZELLIGI tanimlanmali
#endif

#ifndef _16_BIT_SISTEM_SAATI
	#error Eksik Tanimlama:  _16_BIT_SISTEM_SAATI tanimlanmali
#endif

#ifndef MevcutGorevAdresiniAlmaOzelligi
	#define MevcutGorevAdresiniAlmaOzelligi 0
#endif
#endif
```

ÖZGEÇMİŞ

Kişisel Bilgiler

Soyadı, adı : ÇOTUK, Hüseyin

Uyruğu : T.C.

Doğum tarihi ve yeri : 23.06.1980 İstanbul

Medeni hali : Evli

Telefon : 0 (312) 292 40 26

Faks : 0 (312) 292 40 25

e-mail : hcotuk@etu.edu.tr

Eğitim

Derece	Eğitim Birimi	Mezuniyet tarihi
Lisans	Süleyman Demirel Üniversitesi/Elektronik ve Haberleşme Müh	2002

İş Deneyimi

Yıl	Yer	Görev
2004-	TOBB Ekonomi ve Teknoloji Üniversitesi	Bilişim Teknolojileri Müdürü
2003-2004	Elektronik Cihaz Sanayi Tic.Ltd.Şti.	Ar-ge Mühendisi

Yabancı Dil

İngilizce

Yayınlar

Bildiriler

Çotuk, H., IEEE 802.1x ve Radius protokolleri ile kimlik doğrulama uygulamaları, İnternet Haftası, TOBB Ekonomi ve Teknoloji Üniversitesi, Ankara, Aralık 2006.