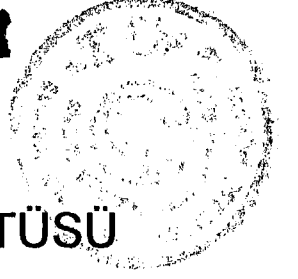


171044

T.C.



GEBZE YÜKSEK TEKNOLOJİ ENSTİTÜSÜ
MÜHENDİSLİK VE FEN BİLİMLERİ ENSTİTÜSÜ

MOBİL ROBOT KONTROLLERİ İÇİN AÇIK
PLATFORM TASARIMI

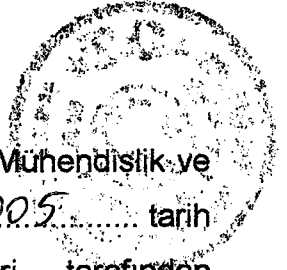
Oğuz ÖKTEM
YÜKSEK LİSANS TEZİ

ELEKTRONİK MÜHENDİSLİĞİ
ANABİLİM DALI

TEZ DANIŞMANI
Yrd. Doç. Dr. Erkan ZERGEROĞLU

GEBZE

2005



Oğuz Öktem'in tez çalışması, G.Y.T.E. Mühendislik ve Fen Bilimleri Enstitüsü Yönetim Kurulu'nun 24/01/2005 tarih ve 2005/4 sayılı kararıyla oluşturulan jüri tarafından Elektronik Mühendisliği Anabilim Dalında YÜKSEK LİSANS tezi olarak kabul edilmiştir.

JÜRİ

ÜYE : Yrd. Doç. Dr. Erkan Zergeroğlu
(Tez Danışmanı)



ÜYE : Yrd. Doç. Dr. Serdar S. Erdem
S. S. Ece

ÜYE : Yrd. Doç. Dr. Abdülkadir Balıks
A. Balıks

ONAY

G.Y.T.E. Mühendislik ve Fen Bilimleri Enstitüsü Yönetim Kurulu'nun 10.03.2005 tarih ve 2005/11 sayılı kararı.

İmza / Mühür

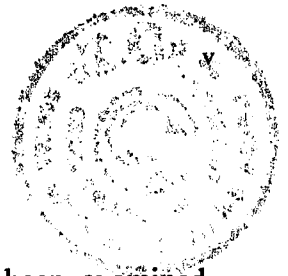

Yrd. Doç. Dr. Ahmet ÇAPOĞLU
GYTE Mühendislik ve Fen Bil.
Enstitüsü Müdür Yardımcısı



ÖZET

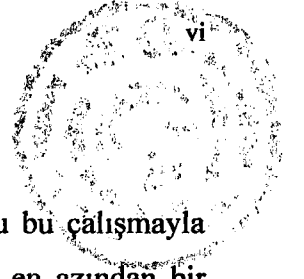
Bu çalışmada tekerlekli mobil robotların kontrolü incelendi. Regülasyon ve iz sürme problemleri ele alındı. Gerilim, devir ve oyunkolu modu olmak üzere üç farklı modda çalışabilen üç tekerlekli bir mobil robot platformu tasarlanıp geliştirildi. Platformun denetiminin RS-232 protokolü aracılığıyla Windows işletim sistemi tabanlı bir bilgisayardan yapılabilmesi için C++ dilinde yazılım geliştirildi. Robot platformu üzerindeki iki DC motoru kontrol edip sürecekt elektronik devre tasarlanıp gerçekleştirildi. Mikrodenetleyici olarak Microchip firmasının ürettiğı PIC16F877 kullanıldı. Mikrodenetleyicinin her bir motorun devir sinyalinı sayabilmesi, bilgisayardan gelen bilgileri yakalayabilmesi, bilgisayara her bir motorun devir bilgisini gönderebilmesi ve bilgisayardan aldığı hız ve yön bilgilerini motor sürücüyeye gönderebilmesi için makine dilinde program yazıldı. Mekanik aksam tasarlanıp ürettilirildi. Son olarak gerçekleştirilen mobil robot platformu üzerinde çalışacak mühendisler'e önerilerde bulunuldu.

SUMMARY



In this study, the control of wheeled mobile robots has been examined. Regulation and tracking problems have been treated. A platform of mobile robot with three wheels that can be controlled in three different modes as voltage, speed and joystick modes has been designed and developed. A C++ based software has been developed in order to control the platform from a Windows operating system based computer via RS-232 protocol. The electronic circuit that controls and drives two DC motors on the robot platform has been designed and developed. As microcontroller, PIC 16F877 produced by Microchip has been used. A program in PIC Assembly language has been written such that the microcontroller can count the speed signal of every motor, capture signals coming from the computer, send the speed information of every motor to the computer and send the information of speed and direction of every motor received from the computer to motor drivers. The mechanical part has been designed and fabricated. And finally some suggestions to engineers who will work on the mobile robot platform have been made.

TEŞEKKÜR



Bir ürün tasarlayıp gerçekleştirmenin ne kadar zor olduğunu bu çalışmayla çok iyi anladım. Çalışmam boyunca doğrudan veya dolaylı olarak en azından bir fikrini benimle paylaşarak destek olan insanlara teşekkür etmeden başlamam herhalde bu çalışma tam olarak amacına ulaşmış olmaz. Çalışmamın başlangıcında bana büyük destek veren ancak zamansız kaybettiğimiz Sn. Adil SAĞLAM'a, tez danışmanım Sn. Yrd. Doç. Dr. Erkan ZERGEROĞLU'na, Sn. Ahmet KARAGÖZOĞLU'na, Sn. Neyir KARAGÖZOĞLU'na, Sn. Muharrem EKE'ye, Sn. Semih ÖZLER'e, Sn. Hasan TOPTAŞ'a, Sn. Hüseyin TOPTAŞ'a, Sn. Mithat AK'a, Sn. Ferhat ESER'e, Sn. Erten KANATLAR'a, Sn. Ertunç ERKOÇ'a, Sn. Murat ÜNDEY'e, Sn. İsmail DEMİRCAN'a, Sn. Emre DİNDAR'a, Sn. Ali KESKİN'e, Sn. Emre ERTUĞRUL'a, Sn. Onur AKKOZAK'a, Sn. Zafer KOÇER'e, Sn. Vedat DAĞ'a, Sn. Cihan HAKİMOĞLU'na ve OYAK-RENAULT Otomobil Fabrikaları'na, ayrıca çalışmam süresince maddi manevi her türlü desteği aldığım Annem Sn. Hayriye ÖKTEM'e, Babam Sn. Haldun ÖKTEM'e ve Kardeşim Sn. Ceyhun ÖKTEM'e teşekkürü borç bilirim.

İÇİNDEKİLER DİZİNİ



ÖZET	iv
SUMMARY	v
TEŞEKKÜR	vi
İÇİNDEKİLER DİZİNİ	vii
ŞEKİLLER DİZİNİ	ix
RESİMLER DİZİNİ	ix
1. GİRİŞ	1
1.1. Amaç	1
1.2. Mobil Robotlar	1
1.3. Mobil Robot Örnekleri	2
1.3.1. PowerTec 26	3
1.3.2. B21	4
1.3.3. Nomad 200	5
1.3.4. Measure Pathfinder Micro Rover	6
2. TEKERLEKLİ MOBİL ROBOTLARIN KONTROLÜ	7
2.1. Giriş	7
2.2. Kinematik Model	7
2.3. Regülasyon Problemi	8
2.4. İz Sürme Problemi	10
3. MOBİL ROBOT PLATFORMUNUN TANITIMI VE ÖZELLİKLERİ	12
4. DONANIM	14
4.1. Motorlar	14
4.2. Motor Sürücü	14
4.3. Mikrodenetleyici	19
4.4. Elektronik Devre Şeması	17
4.5. Mekanik Aksam	17
5. YAZILIM	21
5.1. Mikrodenetleyici Yazılımı	21
5.1.1. PWM Üreteçle Motor Gerilimi Kontrolü	21
5.1.2. Yön Kontrolü	23

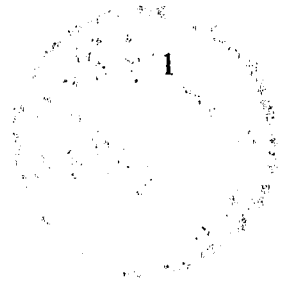
5.1.3. Devir Sayma	23
5.1.4. Bilgisayar ile Haberleşme	28
5.2. Bilgisayar Yazılımı	32
5.2.1. Giriş	32
5.2.2. Bilgisayar ile Mikrodenetleyici Arasında Veri Alışverişi	
Algoritması	34
5.2.3. Gerilim Modu	40
5.2.4. Devir Modu	41
5.2.5. Oyunkolu (Joystick) Modu	43
6.SONUÇ/ÖNERİLER	48
KAYNAKLAR	49
ÖZGEÇMİŞ	50
EKLER	
EK-1 PIC 16F877 Timer0 Birimi Veri Kitapçığı	
EK-2 PIC 16F877 Timer1 Birimi Veri Kitapçığı	
EK-3 PIC 16F877 Timer2 Birimi Veri Kitapçığı	
EK-4 PIC 16F877 PWM Birimleri Veri Kitapçığı	
EK-5 PIC 16F877 USART Birimi Veri Kitapçığı	
EK-6 PITTMAN GM9236S019 DC Motor Veri Kitapçığı	
EK-7 RF Modem Veri Kitapçığı	

ŞEKİLLER DİZİNİ

<u>Sekil</u>	<u>Sayfa</u>
2.1. Mobil Robotun Koordinat Düzleminde Analizi	8
4.1. Motor Sürücü Devre	16
4.2. PIC16F877 Pin Diyagramı	17
4.3. Elektronik Devre	18
5.1. PWM Üretimi Blok Diyagramı	22
5.2. PWM Sinyal Analizi	23
5.3. Devir Sayımı için Motor Seçme Devresi	26
5.4. Ayarlar Ekran Görüntüsü	33
5.5. Bilgisayar ile Mikrodenetleyici Arasındaki Veri Haberleşmesi Akış Diyagramı	35
5.6. Gerilim ve Devir Modları Ortak Ekran Görüntüsü	40
5.7. Oyunkolu Modu Ekran Görüntüsü	43

RESİMLER DİZİNİ

<u>Resim</u>	<u>Sayfa</u>
1.1. PowerTec 26	3
1.2. B21	4
1.3. Nomad 200	5
1.4. Mesure Pathfinder Micro Rover	6
4.1. Mobil Robot	20



1.GİRİŞ

1.1.Amaç

Bu çalışmanın amacı tekerlekli mobil robot algoritmalarının denenmesine imkan veren akademik amaçlı ve düşük maliyetli bir tekerlekli mobil robot platformunun tasarlanıp gerçekleştirilmesidir. Ayrıca platformun gelişime açık olmasına dikkat edilmiştir. Bunun için platform bilgisayardan kontrol edilecek şekilde tasarlanmıştır. Böylece mikroişlemcinin gücünden faydalanmak mümkün olmuştur.

1.2.Mobil Robotlar

Medeniyet tarihinin eski devirlerinden beri mühendisler insanların büyük emekler sarfederek yaptıkları işleri kolaylaştırmanın yollarını aramışlardır. Bu arayışlarında başarılı olduklarının birçok örneği bulunmaktadır. Ancak içlerinden bazıları vardır ki insanların yaptıkları işleri kolaylaştırmanın ötesinde insanların işlerini devralacak aygıtların tasarlanabilirliği üzerine kafa yormuşlardır. Örnek vermek gerekirse bu düşünceyi ilk dile getiren bilim insanı olarak kabul edilebilecek 12. yüzyılda yaşamış Ebû'l İzz El Cezeri ismi uygun olur. Dünya bilim tarihi açısından bugünkü sibernetik ve robot biliminde çalışmalar yapan ilk bilim adamı olan Ebû'l İzz El Cezeri, çalışmalarını Artukoğulları Sultanı için yazdığı Kitab'ül-Cami Beyn'el İlmi ve el-Ameli'en Nafi fi Sinaati'l Hiyel (Mekanik Hareketlerden Mühendislikte Faydalanmayı İçeren Kitap) [8] adlı eserinde ortaya koydu. 50'den fazla cihazın kullanım esaslarını, yararlanma olanaklarını çizimlerle gösterdiği bu olağanüstü kitapta Cezeri, "Tatbikata çevrilmeyen her teknik ilmin, doğru ile yanlış arasında kalacağını" söyler. Bu kitabın orijinali günümüze kadar ulaşamadıysa da, bilinen 15 kopyasından 10'u Avrupa'nın farklı müzelerinde, 5 tanesi Topkapı ve Süleymaniye kütüphanelerinde yer almaktadır. Cezeri'nin mekanik olarak tasarladığı, günümüzde ise elektroniğin ve bilgisayar biliminin olanaklarından yararlanılarak gerçekleştirilen bu tip aygıtları Çek oyun yazarı Karel Capek'in 1920 yılında yayınlanan Rossum'un Evrensel Robotları [9] adlı oyununda geçen ve çekçe işçi anlamına gelen Robot kelimesiyle adlandırmak kabul görmüştür. Buna göre bir

aygıta robot denilebilmesi için sağlaması gereken koşulları belirlemek ihtiyacı doğmuştur. Robotlar için birçok tanım mevcut olmasına rağmen bunlardan en bilineni ünlü bilim-kurgu yazarı Isaac Asimov'unkidir. Asimov 1942 yılında yayınlanan robotik kelimesinin ilk defa kullanıldığı Runaround [10] adlı eserinde kendi adıyla anılan üç adet yasa tanımlamıştır. Bir, iki ve üç numaralı yasalara daha sonra sıfır numaralı yasayı eklemiştir. Bu yasalar şöyledir:

- Sıfır numaralı yasa- Robot insanlığa zarar vermez ve zarar görmesine seyirci kalmaz.
- Bir numaralı yasa- Sıfır numaralı yasayla çelişmediği sürece robot insana zarar vermez ve zarar görmesine seyirci kalmaz.
- İki numaralı yasa- Robot insanın emirlerine itaat etmeye, birinci yasayla çelişmediği sürece zorunludur.
- Üç numaralı yasa- Birinci ve ikinci yasalarla çelişmediği sürece, robot kendi varlığını korumaya zorunludur.

Yasaların tanımlanışından da anlaşılacağı üzere bu yasalara yenileri eklenebilir.

Uzayda tekerlek üzerinde veya bacak benzeri mekanik uzuvlarla yer değiştirebilen robotlar mobil robot olarak isimlendirilir. Mobil robotların bir çok kullanım alanı vardır. Bunlardan bazıları bomba imha, temizlik, bedenen engelli insanlara destek, oyun, endüstride malzeme taşıma, denizaltı araştırmaları, uzay araştırmaları, askeri uygulamalar, güvenlik, tarım, madencilik, akademik araştırmalar olarak sayılabilir. Bu çalışmada tekerlekli mobil robotlar üzerine yoğunlaşmıştır.

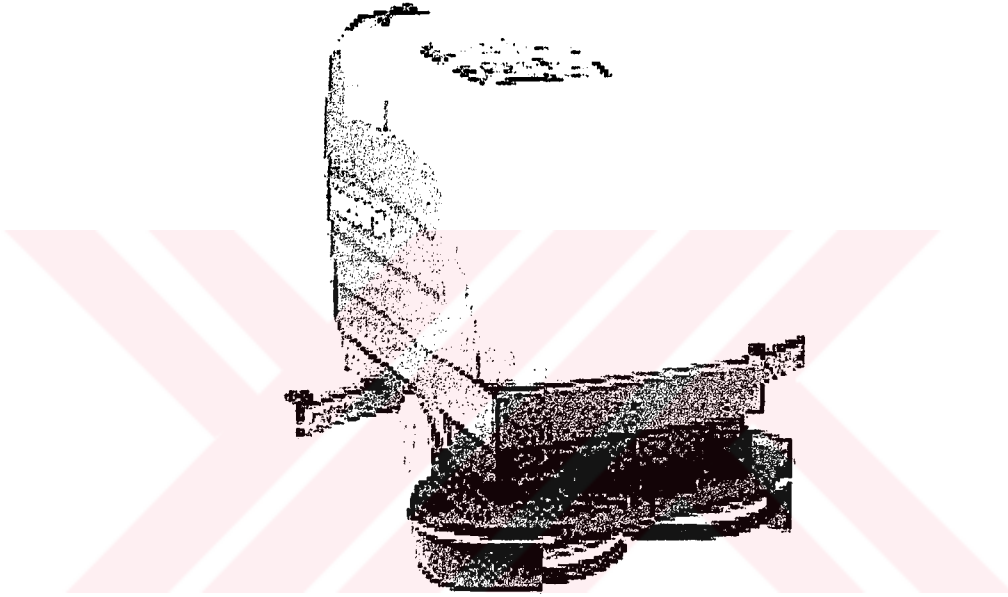
1.3.Mobil Robot Örnekleri

Bu bölümde dört örnek mobil robot kısaca tanıtılmıştır:

- **PowerTec 26**
- **B21**
- **Nomad 200**
- **Mesure Pathfinder Micro Rover**

1.3.1.PowerTec 26

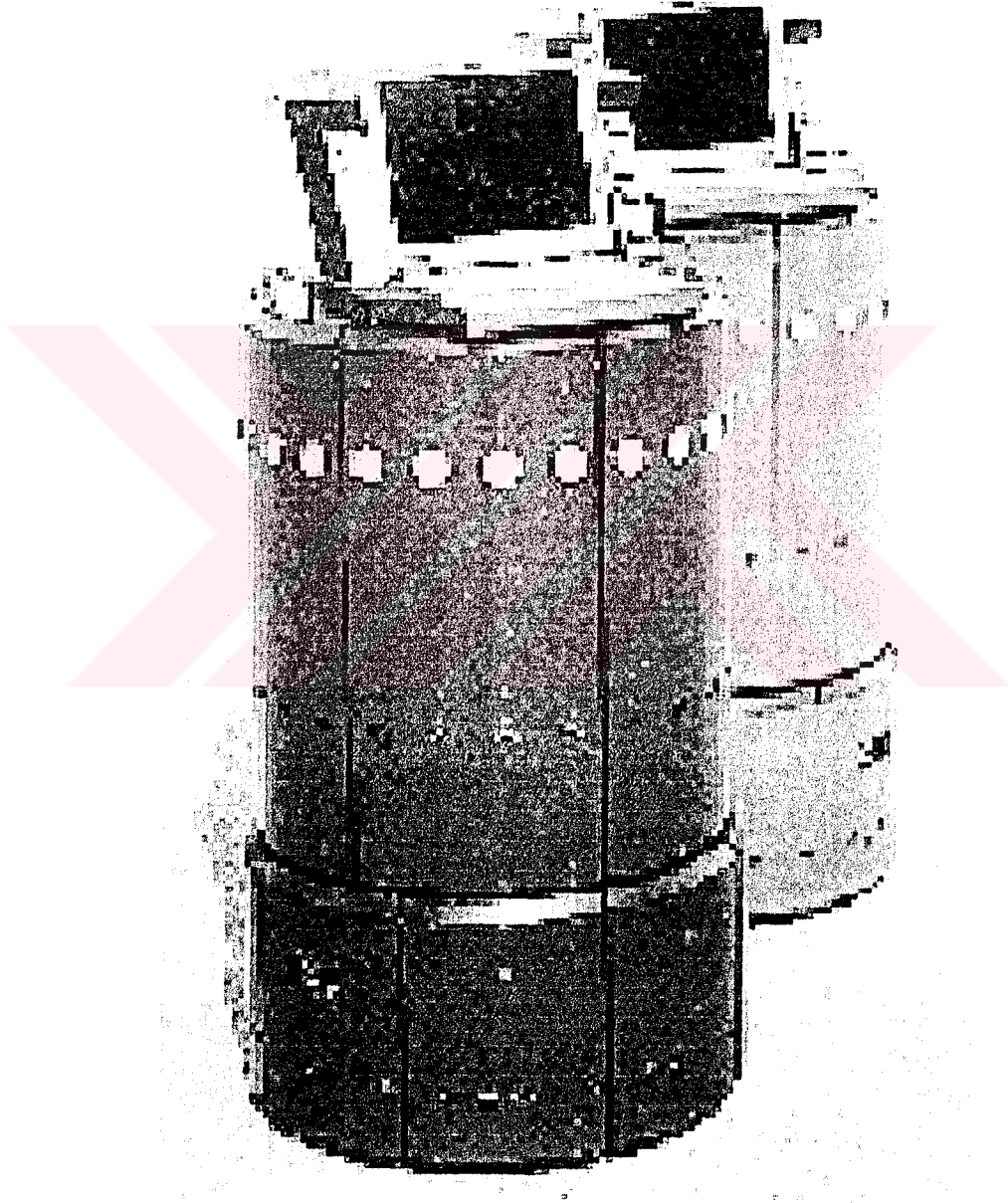
Windsor Industries tarafından üretilmiş bir temizlik robotudur. 6,6 volt 305 Ah aküsü şarj edildikten sonra durmaksızın 30000 metrekare alanı temizleyebilir. 2,5 metre/saniye hıza ulaşabilen robot 21 galon kapasiteli bir temizlik sıvısı tankına sahiptir. LASER yansıtıcı ile yolunu bulur.



Resim 1.1. PowerTec 26

1.3.2.B21

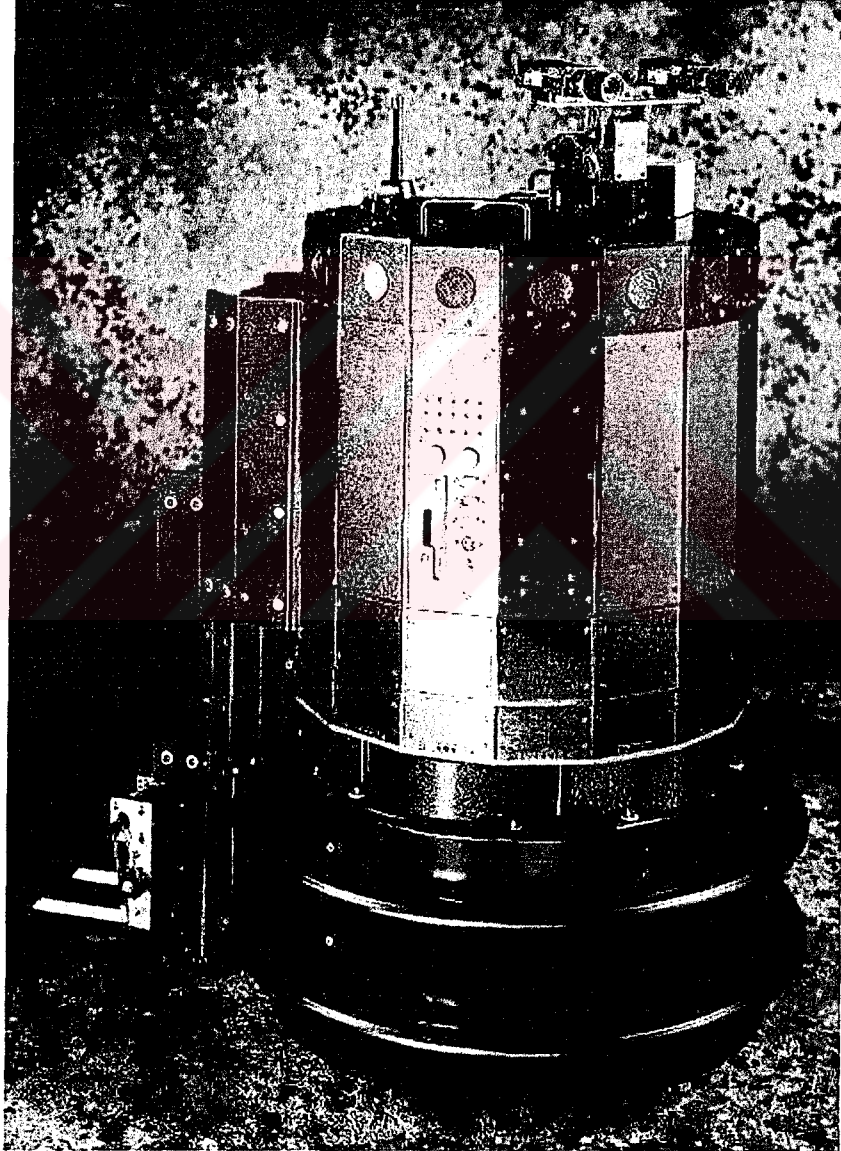
B21 akademik arařtırmalarda kullanılmak üzere ticari amaçla üretilmiř bir mobil robottur. Üreticisi Real World Interface (RWI) olup dört tekerleğinin dışında dört eksenli bir robot kola sahiptir. Saniyede yarım metreye yakın hızda hareket edebilir. Linux işletim sistemi tabanlıdır.



Resim 1.2. B21

1.3.3.Nomad 200

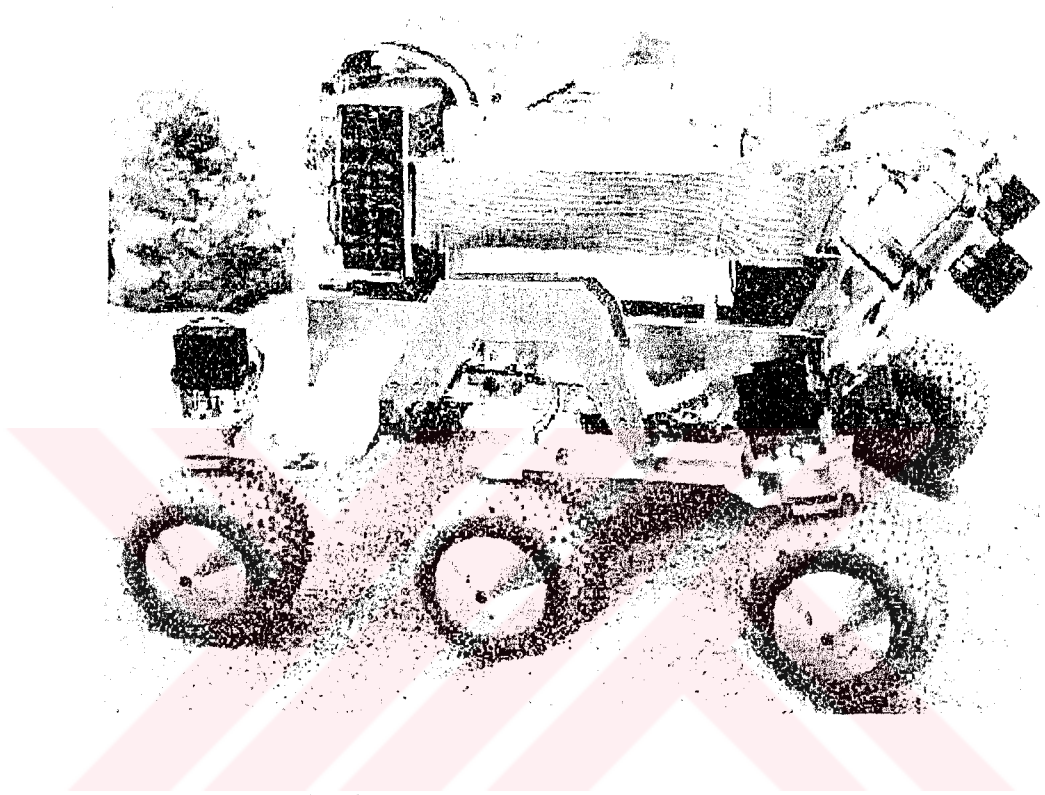
Üç tekerlekli Nomad 200 de B21 gibi akademik arařtırmalara yöneliktir. Üretici firma Nomadic Technologies'dir. Donanımı algılayıcı ve motor kontrolünü ve bilgisayarla haberleşmeyi destekler. Yazılımı ise grafik arabirimi, robot simülatörü, hareket planlama ve kontrol kütüphaneleri, algılayıcı bilgisi yorumlama kütüphaneleri içerir. Saniyede doğrusal olarak 30 santimetre, açısız olarak ise 60 derece hareket edebilir.



Resim 1.3. Nomad 200

1.3.4.Mesure Pathfinder Micro Rover

JPL (Jet Propulsion Laboratory) mobil robot grubu tarafından geliştirilmiştir. Engeli yüzeylerde hareket edecek şekilde tasarlanmıştır. Gezegen yüzeyleri bu tip robotlarla incelenir. Üzerinde LASER tarayıcı bulunmaktadır.



Resim 1.4. Mesure Pathfinder Micro Rover

2.TEKERLEKLİ MOBİL ROBOTLARIN KONTROLÜ

2.1.Giriş

Tekerlekli mobil robotların kontrolü iki temel problemin olası çözümlerinin uygulanması olarak ele alınabilir. Bunlar regülasyon ve iz sürme problemleridir. Regülasyon , mobil robotun mevcut kartezyen konum ve yöneliminin verilen sabit bir konum ve yönetime değişmeye zorlanmasından ibarettir. Başka bir deyişle belli bir konum ve yönelimdeki mobil robotun verilen başka bir konuma belli bir yönelimde kalacak şekilde gitmesini sağlamaktır. İz sürme ise regülasyonda istenen konum ve yönelim değişiminin zamanın belirli fonksiyonları olarak sürekli değişmesidir ki bu durumda robotun verilen bir yörüngeyi izleyeceği açıktır. Mobil robotun regülasyonunu veya iz sürmesini sağlamak için uygun kontrolörler tasarlanmalıdır. Bu bölümde her iki problem ele alınarak her biri için birer kontrolör örneği verilecektir. Ancak öncelikle tekerlekli mobil robotlar için bir kinematik model kurulmalıdır.

2.2.Kinematik Model

Tekerlekli mobil robotlar için kinematik model aşağıdaki gibi verilir:

$$\frac{dq}{dt} = S(q) v \quad (2.1)$$

Burada

$$q = [x_c \ y_c \ \theta]^T \quad (2.2)$$

ve

$$\frac{dq}{dt} = \left[\frac{dx_c}{dt} \ \frac{dy_c}{dt} \ \frac{d\theta}{dt} \right]^T \quad (2.3)$$

olur. Şekil 2.1.'de de görüldüğü üzere x_c ve y_c mobil robotun ağırlık merkezinin kartezyen koordinatları olup θ da yönelim açısıdır. $\frac{dx_c}{dt}$ ve $\frac{dy_c}{dt}$ ise v_x olarak

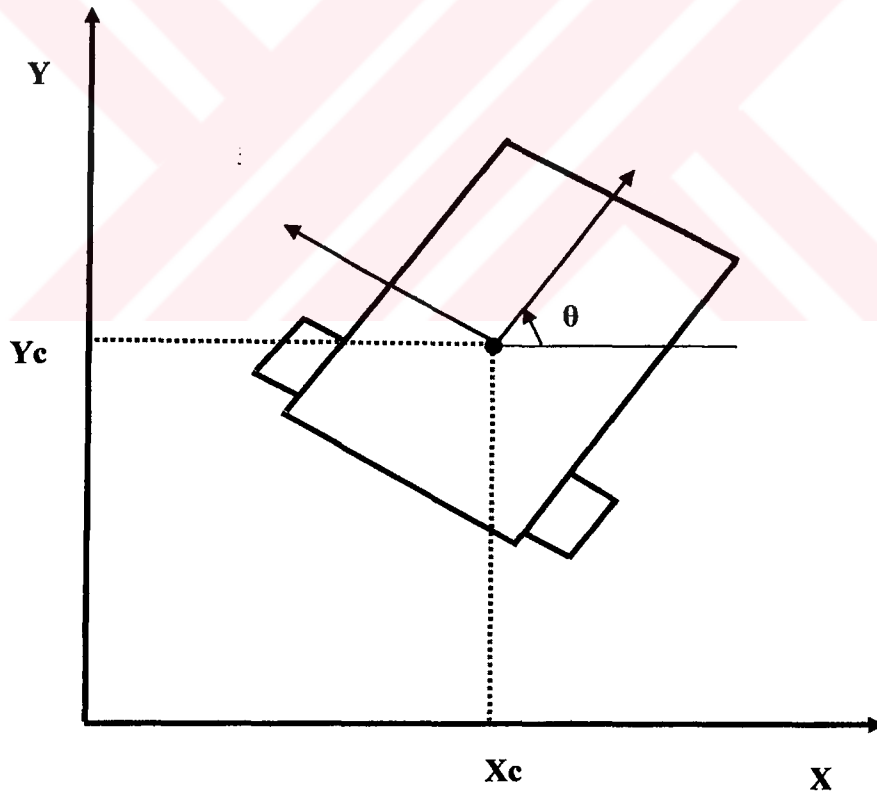
gösterilen doğrusal hızın kartezyen bileşenleri olup $\frac{d\theta_c}{dt}$ açısal hızdır. $S(q)$ matrisi aşağıdaki gibi tanımlanır:

$$S(q) = \begin{bmatrix} \cos \theta & 0 \\ \sin \theta & 0 \\ 0 & 1 \end{bmatrix} \quad (2.4.)$$

Son olarak da hız vektörü

$$v = \begin{bmatrix} v_1 & v_2 \end{bmatrix}^T = \begin{bmatrix} v_1 & \frac{d\theta}{dt} \end{bmatrix}^T \quad (2.5.)$$

olur. Burada v_1 ve v_2 mobil robotun doğrusal ve açısal hızlarıdır. Bu modelde ağırlık merkezi ile dönme merkezinin çakıştığı varsayılmıştır.



Şekil 2.1. Mobil Robotun Koordinat Düzleminde Analizi

2.3.Regülasyon Problemi

Regülasyon probleminde kontrolün amacını modelleyebilmek için

$$x' = x_c - x_{rc} \quad y' = y_c - y_{rc} \quad \theta' = \theta_c - \theta_{rc} \quad (2.6.)$$

tanımlanır. Burada $x'(t), y'(t)$ ve $\theta'(t)$ mobil robotun mevcut konum ve yönelimiyle ulaşması istenilen referans konum ve yönelimleri arasındaki farktır. Kapalı çevrim hata sisteminin tanımlanmasını kolaylaştırmak açısından aşağıdaki gibi bir dönüştürme kullanılır [1]:

$$\begin{bmatrix} e_1 \\ e_2 \\ e_3 \end{bmatrix} = \begin{bmatrix} \cos \theta & \sin \theta & 0 \\ -\sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x' \\ y' \\ \theta' \end{bmatrix} \quad (2.7.)$$

Burada e_1, e_2 ve e_3 konum ve yönelim regülasyon hata fonksiyonlarından türetilen zaman bağımlı yardımcı hata fonksiyonlarıdır. Eşitlik 2.7.'nin zamana göre türevi alınıp Eşitlik 2.1. ve 2.5 kullanılarak

$$\begin{bmatrix} \frac{de_1}{dt} \\ \frac{de_2}{dt} \\ \frac{de_3}{dt} \end{bmatrix} = \begin{bmatrix} v_1 + v_2 e_2 \\ -v_2 e_1 \\ v_2 \end{bmatrix} \quad (2.8.)$$

olduğu gösterilebilir.

Regülasyon problemini çözmeye yönelik bir kontrolör örneği aşağıda verilmiştir [1]:

$$\begin{bmatrix} v_1 \\ v_2 \end{bmatrix} = \begin{bmatrix} -k_1 e_1 \\ -k_2 e_3 + e_2^2 \sin(t) \end{bmatrix} \quad (2.9.)$$

Burada k_1 ve k_2 pozitif sabit kontrol kazançlarıdır. Eşitlik 2.9.'daki v_1 ve v_2 değerleri Eşitlik 2.8.'de yerine konulursa aşağıdaki kapalı çevrim hata sistemi elde edilir:

$$\begin{bmatrix} \frac{de_1}{dt} \\ \frac{de_2}{dt} \\ \frac{de_3}{dt} \end{bmatrix} = \begin{bmatrix} -k_1 e_1 + v_2 e_2 \\ -v_2 e_1 \\ -k_2 e_3 + e_2^2 \sin(t) \end{bmatrix} \quad \text{Eşitlik 2.10.}$$

2.4. İz Sürme Problemi

İz sürme probleminin regülasyon probleminden farkı referans kartezyen konum ve yöneliminin zamanın fonksiyonu olarak değişmesidir. Regülasyon probleminde olduğu gibi burada da

$$x' = x_c - x_{rc} \quad y' = y_c - y_{rc} \quad \theta' = \theta_c - \theta_{rc} \quad (2.11.)$$

tanımlanır. Regülasyon probleminden farklı olarak burada rc indisli konum ve yönelim tanımlayıcıları zamanın fonksiyonu olarak değişmektedirler. Tekerlekli mobil robota verilen yörüngeyi izletebilmek için öncelikle aşağıda verilen dinamik yörünge denklemini sağlayacak bir referans robot tanımlanır:

$$\frac{dq_r}{dt} = S(q_r) v_r \quad (2.12.)$$

Burada

$$v_r = [v_{1r} \quad v_{2r}]^T \quad (2.13.)$$

olacak şekilde zaman bağımlı referans doğrusal ve açısal hızı verir.

Açık çevrim hata sistemini tanımlamak için Eşitlik 2.7.'nin zamana göre türevi alınıp Eşitlik 2.1., 2.5., 2.11 ve 2.12 kullanılarak aşağıdaki matrisel denklem elde edilir:

$$\begin{bmatrix} \frac{de_1}{dt} \\ \frac{de_2}{dt} \\ \frac{de_3}{dt} \end{bmatrix} = \begin{bmatrix} v_1 + v_2 e_2 - v_{1r} \cos e_3 \\ -v_2 e_1 + v_{1r} \sin e_3 \\ v_2 - v_{2r} \end{bmatrix} \quad (2.14.)$$

Burada $v_{1r} = 0$ ve $v_{2r} = 0$ olduğunda regülasyon probleminde verilen açık çevrim hata sisteminin (Eşitlik 2.8.) elde edildiğine dikkat edilmelidir.

İz sürme problemini çözmek için Eşitlik 2.14. ile verilen modele uygun bir kontrolör tasarlamak gerekmektedir. Aşağıda [1] örnek bir kontrolör verilmiştir:

$$\begin{bmatrix} v_1 \\ v_2 \end{bmatrix} = \begin{bmatrix} -k_1 e_1 + v_{1r} \cos e_3 \\ \frac{\sin e_3}{e_3} e_2 - k_2 e_3 + v_{2r} \end{bmatrix} \quad (2.15.)$$

Burada k_1 ve k_2 pozitif sabit kontrol kazançlarıdır. Eşitlik 2.15.'deki v_1 ve v_2 değerleri Eşitlik 2.14.'de yerlerine konulursa aşağıdaki kapalı çevrim hata sistemi elde edilir:

$$\begin{bmatrix} \frac{de_1}{dt} \\ \frac{de_2}{dt} \\ \frac{de_3}{dt} \end{bmatrix} = \begin{bmatrix} v_2 e_2 - k_1 e_1 \\ -v_2 e_1 + v_{1r} \sin e_3 \\ \frac{\sin e_3}{e_3} e_2 - k_2 e_3 \end{bmatrix} \quad (2.16.)$$

3.MOBİL ROBOT PLATFORMUNUN TANITIMI VE ÖZELLİKLERİ

Bu çalışmada mobil robot algoritmalarının test edilebileceği bir platform tasarlanmış ve gerçekleştirilmiştir. Esnekliği sağlamak amacıyla platform Windows tabanlı bir bilgisayar tarafından kontrol edilebilecek şekilde tasarlanmıştır. Sistem iki adet DC motor, her motor için bir elektronik motor sürücü katı, bilgisayardan gelen komutları değerlendirip motor sürücüleri kontrol eden mikrodenetleyici temelli bir kontrol katı, makine dilinde yazılmış mikrodenetleyici yazılımı, bir tanesi pasif diğer ikisi motorlara bağlı olmak üzere üç tekerlek, mekanik aksam, 12 V şarj edilebilir akü, RS-232 bağlantısı ve C++ dilinde yazılıp Microsoft Visual C++ 6.0 derleyicisi ile derlenmiş kontrol yazılımından oluşmaktadır. Mikrodenetleyici olarak Microchip firmasının ürettiği PIC 16F877 kullanılmıştır. Osilatör frekansı olarak 4 MHz seçilmiştir. PIC 16F877 mikrodenetleyicisi beş adet port, iki adet darbe genişlik modülasyonu (PWM) modülü, evrensel senkron-asenkron alıcı-verici (USART), üç adet zamanlayıcı/sayıcı, 8 K program hafızası ve 368 bayt veri belleği gibi özelliklere sahip olması dolayısıyla bu tip bir çalışma için en düşük maliyetli çözümlerden biridir.

Motor kontrol devresi her bir motorun hızını ve yönünü kontrol eder. Sağa veya sola dönüşler motorlara farklı gerilimlerin ve/veya farklı yönlerin verilmesiyle sağlanır. Motorlara uygulanacak gerilimler mikrodenetleyicinin darbe genişlik modülasyonu modülleri tarafından ayarlanır. Robot platformu üç değişik modda kumanda edilebilir. Bunlar oyunkolu modu, gerilim modu ve devir modudur. Motorlar oyunkolu modunda farenin hareketlerine göre hareket ederken gerilim modunda doğrudan motorlara verilecek gerilim ile orantılı değerler girilmek suretiyle hareket ettirilir. Devir modunda ise kullanıcı tarafından motorların her birinin dönmesi istenilen devir girilir ve devir ayarlama rutini istenen devirler elde edilene kadar motor gerilimlerini değiştirir. Bilgisayar ile mikrodenetleyiciyi haberleştirmek için RS-232 arabirimi kullanılmıştır. Haberleşme hızı 9600 bit/saniye olarak seçilmiştir. Bilgisayar yazılımı her 10 milisaniyede bir mikrodenetleyiciye motorların gerilim değerlerini ve eğer değişiklik varsa yön bilgilerini gönderir. Bunun karşılığında mikrodenetleyici de her bir motorun devir

bilgisini bilgisayara gönderir. Bu karşılıklı gönderim işleminin düzgün işleyebilmesi amacıyla basit bir protokol yazılmıştır. Bu protokole sadık kalınarak kolaylıkla robota daha sonradan eklenebilecek algılayıcıların algıladıkları değerler de bilgisayara gönderilebilir.



4.DONANIM

4.1.Motorlar

Motor olarak iki adet PITTMAN GM9236S019 12 VDC motor kullanılmıştır. Dişli oranı 19.7:1 olup üzerinde gömülü optik devir sayma modülü bulunmaktadır. Devir sayma modülünün çalışabilmesi için 5 V ile beslenmesi gerekmektedir. İki devir sayma çıkışı beraber $500 \times 4 = 2000$ CPR (Count per Revolution) ölçüm duyarlılığı sağlamaktadır. Ancak bu çalışmada her bir motorun sadece bir çıkışı kullanılmıştır. Bunun nedeni iki çıkışın kullanılarak devir bilgisinin elde edilmesi için bu iş için özel tasarlanmış oldukça pahalı tümleşik devrelere ihtiyaç olmasıdır. Bunun yerine devir sayım işlemini gerçekleştirebilmek için mikrodenetleyicinin imkanları kullanılmıştır. Yazılım kısmında ayrıntılarıyla bahsedilen devir sayımında mikrodenetleyicinin 0 ve 1 numaralı zamanlayıcı/sayıcı modüllerinden yararlanılmıştır.

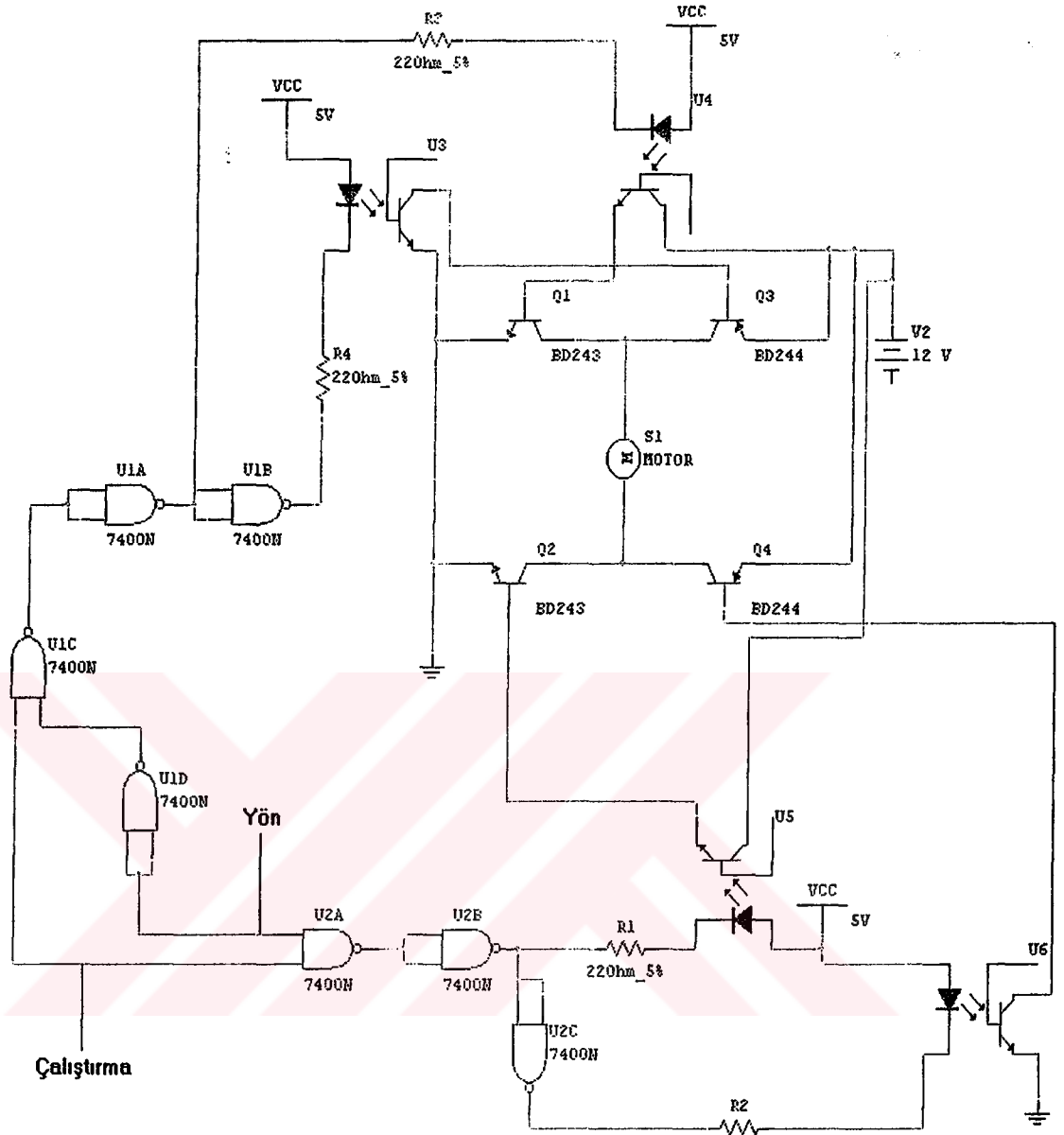
4.2.Motor Sürücü

Motor sürücü olarak transistörlü bir devre tasarlanmıştır. H köprü tipindeki sürücü iki adet BD243 (NPN), iki adet de BD244 (PNP) güç transistörlerinden oluşur. Devrenin giriş katındaki lojik devre mikrodenetleyiciden gelen çalıştırma ve yön seçimi sinyallerini değerlendirip çıkıştaki güç transistörlerinin iletim veya kesime girmesini sağlamaktadır. Lojik kat ile güç katı arasında izolasyonu sağlamak amacıyla dört adet 4N25 optokuplörü bulunmaktadır. Devre şeması Şekil 4.1.'de verilmiştir. Devrenin çalıştırma girişine 0 Volt geldiğinde yön girişine gelen gerilim ne olursa olsun U3 ve U6 optokuplörleri kesimde kalır. Bu durumda U3 ve U6'nın sürdüğü iki PNP güç transistörü de kesimde kalacağından motor üzerinden akım akmaz. Dolayısıyla da motor hareket etmez. Ters durumda yani çalıştırma girişine 5 Volt geldiğinde yön girişinde de 5 Volt olduğunu varsayarsak U4 ve U6 optokuplörlerinin LED'leri enerjileneceğinden bunların sürdüğü Q1 ve Q4 transistörleri de ilettime geçer. Bu sırada U3 ve U5 optokuplörleri dolayısıyla da Q3 ve Q2 transistörleri kesimdedir. Motor üzerinden Q4'den Q1'e doğru akım akacağı açıktır. Motor belli bir yönde dönmeye başlar. Yön girişindeki sinyalin 0 Volt olduğu düşünülürse bu sefer Q1 ve Q4 transistörleri kesimdeyken Q3 ve Q2

transistörleri iletimde olacağından motor üzerinden Q3'den Q2'ye doğru akım akacaktır. Bu da şemada da açıkça görüldüğü üzere yön girişinin 5 Volt olduğu duruma göre akımın ters yönde akması anlamına gelir. Artık motor diğer yönde dönmektedir.

Motorun dönme hızı çalışma girişine mikrodenetleyiciden darbe genişlik modülasyonu (PWM) sinyal gönderilmesiyle ayarlanabilir. Motor hızının PWM'le kontrolü Bölüm 7.1'de ayrıntılı olarak açıklanmıştır.

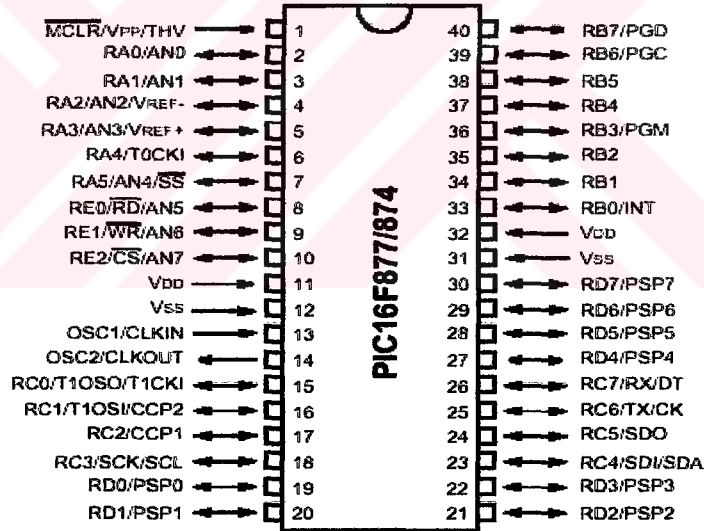




Şekil 4.1. Motor Sürücü Devre

4.3.Mikrodenetleyici

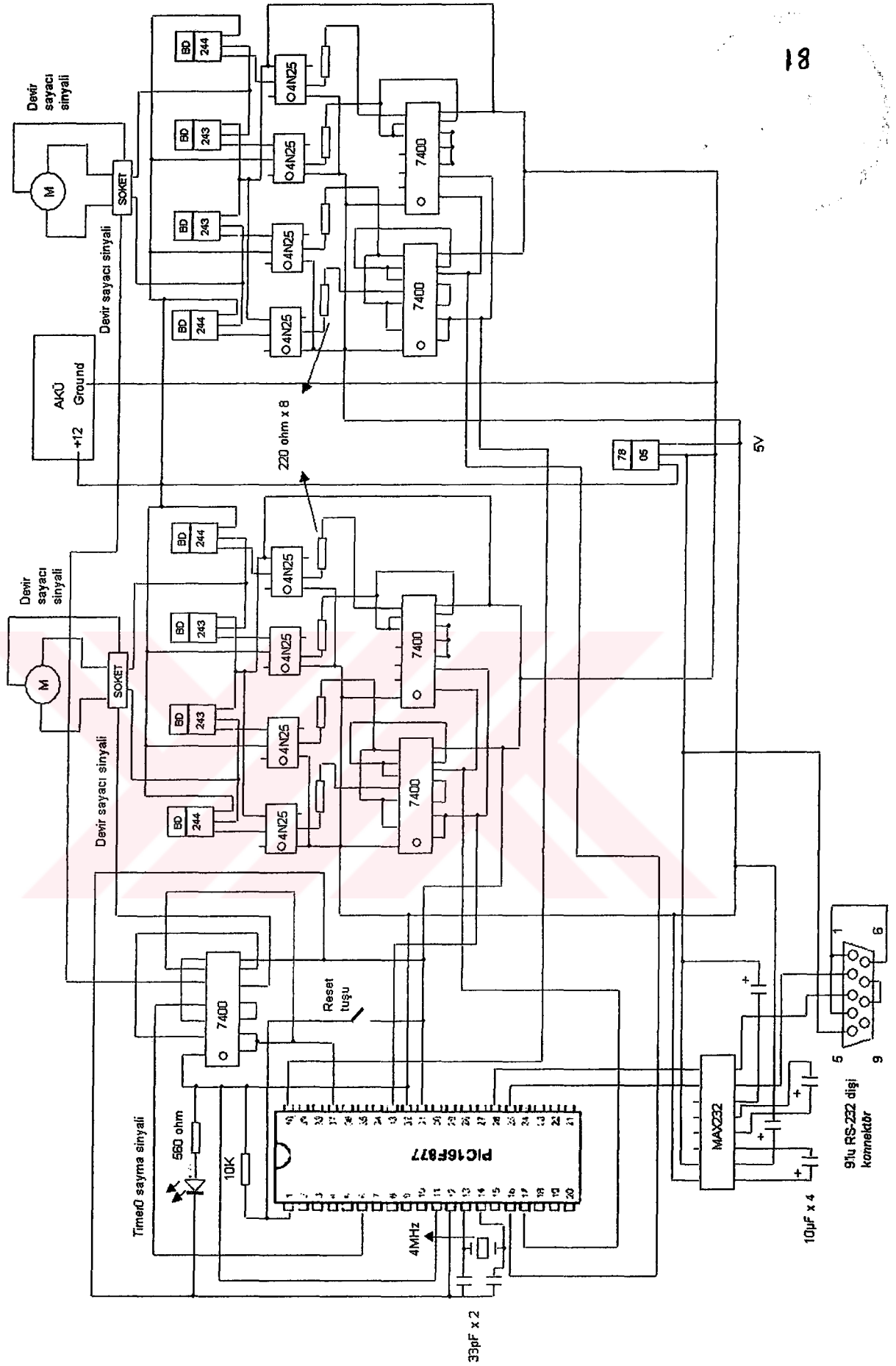
Mikrodenetleyici olarak daha önce de belirtildiği üzere Microchip firmasının ürettiği PIC 16F877 kullanılmıştır. Motor hızlarını kontrol etmek için mikrodenetleyicinin iki adet darbe genişlik modülasyonlu (PWM) sinyal üreteç modülü idealdir. Ayrıca bu mikrodenetleyicinin sahip olduğu Evrensel Senkron-Asenkron Alıcı-Verici (USART) bilgisayar ile RS-232 protokolü aracılığıyla haberleşmeyi sağlar. Motor devirlerinin sayımı Timer0 ve Timer1 modülleri ve Timer1'in taşma durumunda ürettiği kesme aracılığıyla yapılır. Yön sinyalleri ve bir adet devri sayılacak motorun seçimini sağlayan sinyal için PORTB'nin üç ucu çıkış olarak kullanılmıştır. 40 bacaklı mikrodenetleyicinin pin diagramı Şekil 4.2'de verilmiştir. Çalışma frekansı maksimum 20 MHz olmakla beraber bu çalışmada 4 MHz olarak seçilmiştir. Bu frekansta dallanma komutları haricindeki tüm komutlar 1, dallanma komutları ise 2 mikrosaniyede işlenir.



Şekil 4.2. PIC16F877 Pin Diyagramı

4.4.Elektronik Devre Şeması

Açık devre şeması takip eden sayfada verilmiştir. Tümleşik devrelerin bacak numaraları daire işaretinden bulunabilir. Daire işaretli tarafın alt kısmı 1 numara, üst tarafı da en yüksek numaralı pindir. Her bir motor soketi üzerinde motorlara bütünleşik devir sayacı devresini besleyen 5V ve toprak bulunmaktadır.



Şekil 4.3. Elektronik Devre

4.5.Mekanik Aksam

Robotun mekanik aksamı ince profilden imal edilmiştir. Motorların doğrudan bağlandığı 2 adet 150 mm çaplı tekerlek, 1 adet de motor gerilimleri arasındaki farka göre serbestçe yön değiştiren 80 mm çaplı pasif tekerlek üzerinde hareketini sağlar. Enerji, merkeze yerleştirilip sabitlenmiş 2 mAh kapasiteli akü tarafından sağlanır. Akünün yerinden sökülmeden şarj edilebilmesi için kutuplarından birer adet kablo dışarıya verilmiştir. Üst kapağa bütünleşik fan devreyi soğutur. Robot Resim 4.1.'de görülebilir.





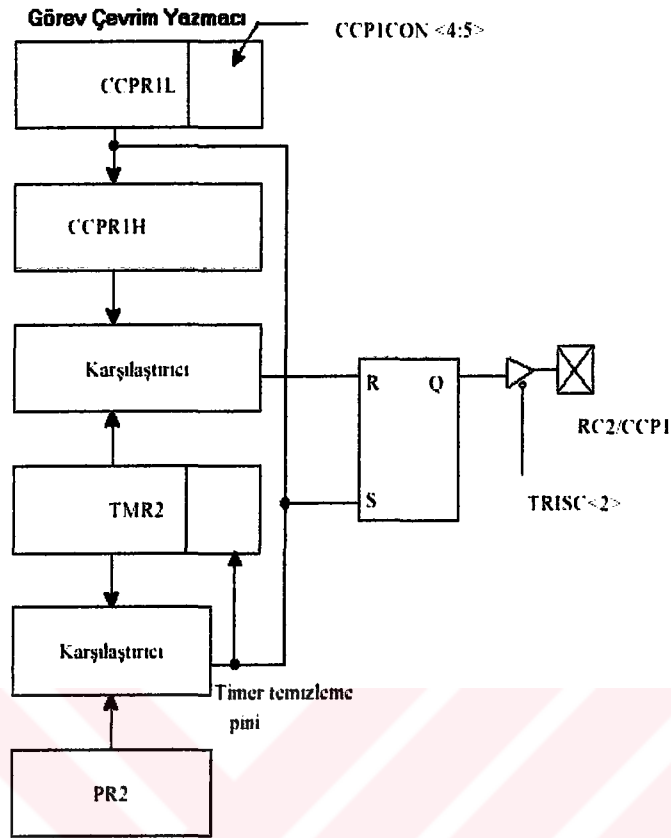
Resim 4.1. Mobil Robot

5.YAZILIM

5.1.Mikrodenetleyici Yazılımı

5.1.1.PWM Üreteçle Motor Gerilimi Kontrolü

DC motorların hızı uçlarına uygulanan gerilimle orantılı olarak değişir. Ancak mikrodenetleyici kontrollü bir sistemde lineer kuvvetlendiricilerin yüksek güç harcamaları ve yazılımda yapılmasının kolay olmaması gibi nedenlerle hız kontrolü için motor gerilimini ayarlamak pratik değildir. Bunun yerine Darbe Genişlik Modülasyonu (PWM) kullanılır. Yani motor gerilimi sabit tutulup belli bir peryot kadar sürenin belli bir parçasında motor uçlarına uygulanır. Örneğin 12 Volt gerilim motor uçlarına her 2 milisaniyenin 1 milisaniyesinde uygulanıp kalan 1 milisaniyede motor uçlarına 0 Volt uygulanırsa pratikte 6 Volt uygulanmış gibi olur. Burada 12 Volt gerilimin uygulanma süresine görev çevrimi (duty cycle) denir. Her bir PWM modülü 10 bit çözünürlükte PWM sinyali üretir. Buna göre her bir motorun uçlarına 1024 farklı gerilim değeri uygulanabilir. Uygulanacak değer CCPR1L yazmacına ve CCP1CON yazmacının 4. ve 5. bitlerine yazılır. Burada ikinci PWM modülü için yazmaç isimlerinin CCPR2L ve CCP2CON olduğuna dikkat edilmelidir. Uygulamada 1024 farklı değer 12 Volt olan motor geriliminin yazmaca yazılan değer her bir birim arttırıldığında $12/1023 \approx 11,7$ mV kadar artması anlamına gelir ki bu kadar düşük bir gerilim motor hızında ihmal edilebilecek kadar bir artışa neden olur. Ayrıca seri haberleşmede bir seferde 8 bit veri alınıp gönderilmesinden dolayı motor gerilimlerinin 10 bit olarak tutulması verilerin işlenmesi ve iletilmesini zorlaştırır. Bu nedenlerden dolayı CCP1CON ve CCP2CON yazmaçlarında tutulan iki en düşük değerlikli bit yazılımda birleştirilerek sabit tutulmakta ve motor gerilimlerinin kontrolü için yalnızca 8 bitlik yazmaçlar olan CCPR1L ve CCPR2L kullanılmaktadır. Buna göre her bir motora 256 farklı gerilim uygulanabilir. Uygulanan gerilim yazmacın onluk sistemdeki değerinin $12/255 \approx 47$ mV ile çarpılmasıyla yaklaşık olarak hesaplanabilir. Mikrodenetleyicinin PWM modülünün basitleştirilmiş blok diyagramı Şekil 5.1.'de verilmiştir.

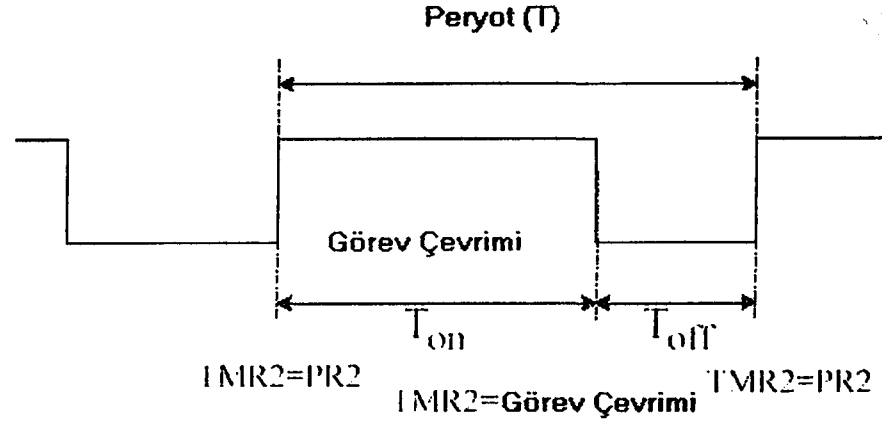


Şekil 5.1. PWM Üretimi Blok Diyagramı

PWM sinyal üretimi için öncelikle iki modül için de ayrı ayrı PWM modu seçilmelidir. Bu seçim CCP1CON ve CCP2CON yazmaçlarının 2. ve 3. bitleri birlenmelidir. Ayrıca PORTC'nin CCP1 ve CCP2 pinleri TRISC yazmacının 2. ve 1. bitlerinin sıfıra çekilmesiyle çıkış olarak seçilir. Sinyal periyodu PR2 yazmacına yazılan değer ile orantılıdır. Peryot aşağıdaki formülle hesaplanabilir:

$$T = \text{PWM periyot} = [(PR2) + 1] \cdot 4 \cdot TOSC \cdot [TMR2 \text{ (Zamanlayıcı 2) önbölücü değeri}]$$

Devrenin osilatör frekansı 4 MHz olduğundan PR2 değeri olarak 255, TMR2 önbölücü değeri olarak da 1 seçildiğinden PWM periyodu 256 mikrosaniye, frekans ise 3,906 KHz olur. Son olarak T2CON yazmacının 2. biti birlenerek TMR2 etkinleştirilir. Şekil 5.2.'de görüldüğü gibi TMR2 değeri görev çevrim değerine eşit olana kadar PWM çıkış lojik 1 (5 V) olup bu andan itibaren TMR2 PR2 değerine eşit olana kadar lojik 0 (0 V) olarak kalır. Bu durumun tek istisnası olan görev çevriminin 0 olduğu durumda PWM çıkışı tüm periyot boyunca lojik 0'dır.



Şekil 5.2. PWM Sinyal Analizi

Daha önce PWM görev çevriminin en düşük değerlikli iki biti olan CCP1CON ve CCP2CON yazmaçlarının 4. ve 5. bitlerinin program başında bire çekilerek sabit bırakıldıkları belirtilmişti. Buna göre motor gerilimleri CCPR1L ve CCPR2L yazmaçlarına yazılan değerle değişir. Bu yazmaçlara yazılacak değerler kullanıcı tarafından bilgisayardaki arabirim yazılımına girilen değerlerin seri porttan mikrodentleyiciye gönderilmesiyle alınır.

5.1.2.Yön Kontrolü

Motor hareket hızından başka yönünün de kontrol edilebilmesi gerekir. Motor sürücünün ayrıntılarıyla açıklandığı 4.2.'de yön seçimi için sürücü devrede bir giriş bulunduğu gösterilmişti. Mikrodenetleyicinin yönleri kontrol edebilmesi için her sürücüye bir sayısal port çıkışı atanmıştır. PORTB'nin 0. ve 7. bitleri sayısal çıkış olarak seçilmiş ve her biri bir motor sürücünün yön girişine bağlanmıştır. Motor yönlerinin seçimi seri porttan yön değiştirme bilgisi geldiğinde mikrodnetleyicinin ilgili port çıkışının durumunu kontrol edip terslemesiyle olur. Yani örneğin belli bir motor için yön değiştirme bilgisi geldiğinde PORTB'nin ilgili çıkışı lojik 1 ise lojik 0 yapılır.

5.1.3.Devir Sayma

Kapalı çevrim kontrol sistemlerinde kontrol edilen elemanın durumuyla ilgili bilgi alıp kontrol bloğuna göndermek gerekir. Burada kontrol edilen motor olup

motorla ilgili motor gerilimi, yön ve motor devri gibi değişkenler bulunmaktadır. Motor gerilimi ve yönü zaten kontrol devresi olan mikrodenetleyici tarafından belirlendiğinden bu bilgiler sistemde mevcuttur. Geriye bilinmeyen değişken olarak devir bilgisi kalır. Devir bilgisinin mikrodenetleyici tarafından bilinmesi mobil robot kontrolü için hayati önem taşımaktadır. Kullanılan motorlar 4.1.'de belirtildiği gibi tümleşik optik iki çıkışlı devir sayma modüllerine sahiptir. Bu tip devir sayma modüllerinin (Quadrature Encoder) çıkış sinyallerinin değerlendirilebilmesi için özel sayıcı tümleşik devreleri mevcuttur. Ancak oldukça maliyetli olan bu çözüm yerine ölçüm duyarlılığından feragat edilerek mikrodenetleyicinin 0 ve 1 numaralı zamanlayıcı/sayıcı (Timer0, Timer1) modülleri kullanılmıştır. Timer0 8 bitlik zamanlayıcı/sayıcı modülü olup burada sayıcı olarak kullanılmıştır. Timer0'ın ayarlandığı kod parçası aşağıdaki gibidir:

```
Timer0_Ayarla
    bsf STATUS,5      ;bank1'e geç
    movlw    b'11100011'
    movwf    OPTION_REG
    bcf STATUS,5      ;bank0'a geç
    clrf     SOL_MOTOR_DEVIR
    clrf     SAG_MOTOR_DEVIR
    clrf     TMR0
```

Kod 5.1. Timer0 Ayarlama

Timer0 ile ilgili ayarlar OPTION_REG yazmacı üzerinden yapılmaktadır. Burada en düşük değerlikli üç bit 011 seçilerek önbölücü değeri 1/16 olarak ayarlanmıştır. Yani Timer0'ın her bir birim artışı aslında sayma girişine her 16 sinyal geldiğinde olur. Bu değer deneme yanılma yöntemiyle en uygun önbölücü değeri olarak bulunmuştur. Zira bu değerde en duyarlı devir bilgisi alınmaktadır. Timer0 sayma girişi olan PORTA'nın 4 numaralı girişine gelen motor devir sayma modülünün bir çıkışının sinyalleri sayılmaktadır. Sayma değeri, Timer1 her kesme ürettiğinde devir bilgisini saklamak için ayrılmış bir yazmaca atılmaktadır.

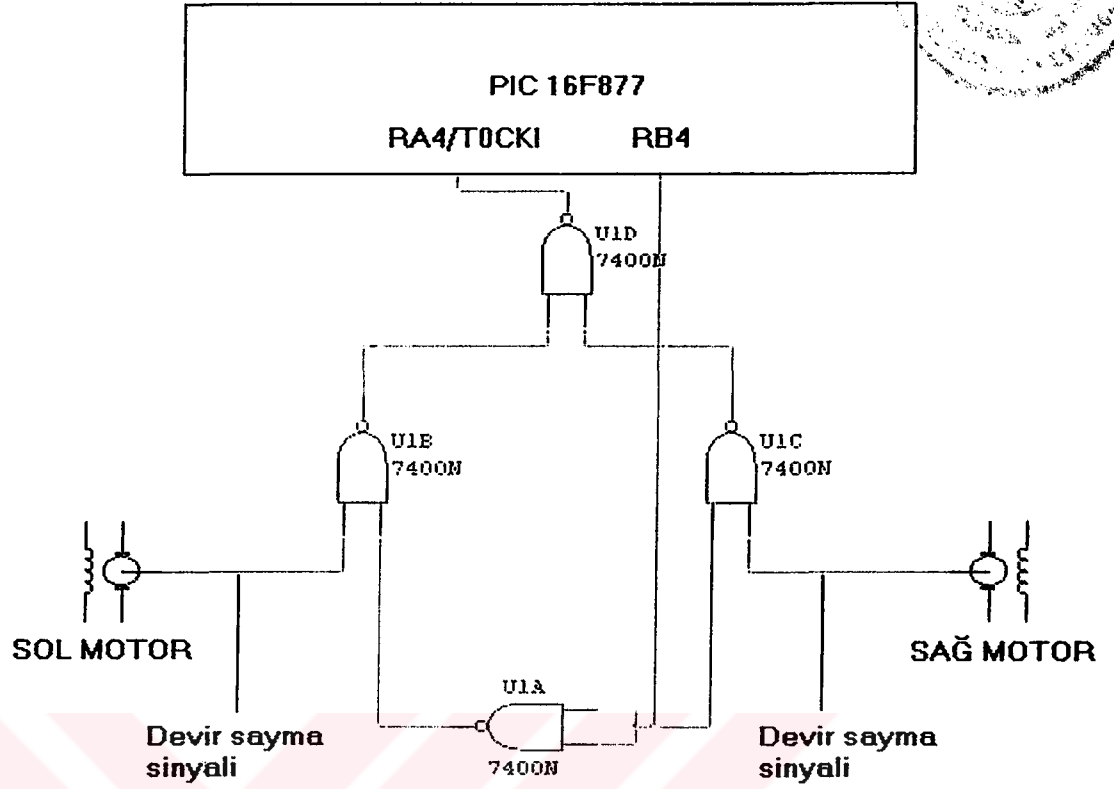
Timer1 ise 16 bitlik zamanlayıcı/sayıcı olup zamanlayıcı modunda kullanılmaktadır. Devrede Timer1'in görevi her taşma oluştuğunda kesme üretip programı kesme hizmet rutinine dallandırmaktır. Timer1 önbölücü değeri 1/1 olarak

ayarlanmıştır. Buna göre Timer1 her komut çevriminde bir artmaktadır. Bir komut çevrimi dört osilatör çevriminden oluştuğundan 16 bitlik Timer1 taşmaya her $2^{16} \times 1/4 \text{ MHz} \times 4 = 65536$ mikrosaniye sonunda ulaşır. Timer1'in ayarlandığı kod parçası aşağıda verilmiştir:

```
Timer1_Ayarla
    bcf    STATUS,5      ;bank0'a geç
    movlw  b'00000001'
    movwf  T1CON
    movlw  b'11000000'
    movwf  INTCON
    bcf    5              ;bank1'e geç
    movlw  b'00000001'
    movwf  PIE1
```

Kod 5.2. Timer1 Ayarlama

Burada çevresel kesme yazmacının ilk biti birlenerek Timer1 kesmesi etkinleştirilmiştir. Buna göre Timer1 her taşmasında diğer bir deyişle her 65536 mikrosaniyede bir kesme üretmektedir. Kesme oluştuğunda o anda Timer0'ın değerinin belli bir yazmaca kaydedildiği belirtilmişti. Fakat Timer0 tek bir girişten gelen sinyalleri saydığından iki motorun da devir sinyallerini saymak için özel bir motor seçme devresi tasarlanmıştır. Bu devre Şekil 5.3.'de görülmektedir.



Şekil 5.3. Devir Sayımı için Motor Seçme Devresi

Devri sayılacak motoru seçme devresi mikrodenetleyicinin PORTB 4 numaralı çıkışının lojik durumuna göre motorlardan birinin devir sinyallerini Timer0 girişine iletmesini sağlayan 4 NAND kapısından oluşan bir devredir. Buna göre PORTB 4 numaralı çıkış yazılımında uygun şekilde koşullandırılmalıdır. Bu işlem kesme hizmet programında yapılır. Kesme hizmet programı kodu aşağıda verilmiştir.

Kesme

```
MOVWF W_TEMP ;Copy W to TEMP register
SWAPF STATUS,W ;Swap status to be saved into W
CLRF STATUS ;bank 0, regardless of current bank, Clears IRP,RP1,RP0
MOVWF STATUS_TEMP ;Save status to bank zero STATUS_TEMP register
MOVF PCLATH, W ;Only required if using pages 1, 2 and/or 3
MOVWF PCLATH_TEMP ;Save PCLATH into W
CLRF PCLATH ;Page zero, regardless of current page
```

```

    bcf     STATUS,5           ;bank0'a geç
    movlw   b'00000000'
    movwf   T1CON
    bcf     PIR1,TMR1IF
    movlw   b'00000000'
    movwf   TMR1H
    movlw   b'00000000'
    movwf   TMR1L
    movlw   b'00000001'
    movwf   T1CON
    movf    TMR0,0             ;Timer0'ın saydığı değeri W yazmacına at
    btfss   PORTB,4
    goto    SolMotorDevir
    bcf     PORTB,4
    movwf   SAG_MOTOR_DEVIR
    goto    SagMotorDevir
SolMotorDevirbsf   PORTB,4
    movwf   SOL_MOTOR_DEVIR
SagMotorDevirclrf   TMR0      ;Timer0'ı yeniden sayım için sıfırlar

    MOVF    PCLATH_TEMP, W ;Restore PCLATH
    MOVWF   PCLATH ;Move W into PCLATH
    SWAPF   STATUS_TEMP,W ;Swap STATUS_TEMP register into W
    ;(sets bank to original state)
    MOVWF   STATUS ;Move W into STATUS register
    SWAPF   W_TEMP,F ;Swap W_TEMP
    SWAPF   W_TEMP,W ;Swap W_TEMP into W

    retfie      ;Kesmeden dönüş komutu

```

Kod 5.3. Kesme Hizmet Programı

Kesme hizmet programının başında ve sonunda ingilizce açıklamayla verilen kısım mikrodenetleyicinin veri kitapçığında önerilen kesmeye girmeden önceki STATUS ve W yazmaç durumlarını korumaya yarayan koddur. Kesme hizmet programı Timer1'i durdurur. Timer0'daki değeri devri sayılacak motorun seçilmesini sağlayan PORTB 4 numaralı bitin durumunu kontrol ederek SOL_MOTOR_DEVİR veya SAĞ_MOTOR_DEVİR yazmacına atar. Bir sonraki kesmede diğer motorun devrini saymak üzere PORTB 4 numaralı biti tersler. Timer0'ı sıfırlar. Kesmeden *retfie* komutuyla çıkar.

Timer0'ın saydığı değer başına motorun yaklaşık 1,51 devir/dakika hızında döndüğü görsel olarak ölçülmüştür. Teorik değer ise motor ve mikrodnetleyici parametrelerinden hesaplanabilir. Buna göre Timer0 önbölücü sabiti 16 olduğundan değer başına aslında en az 16 sayım yaptığı bilinmektedir. Motor üzerindeki devir sayma modülünün spesifikasyonlarından bir devir başına 500 pals ürettiği ve motorların dişli aktarım oranının (gear ratio) 19,7 olduğu da bilinmektedir. Buradan $16/(500*19,7)$ değerinin Timer0'ın saydığı değer başına motor milinin dönüş sayısı olduğu gösterilebilir. Ancak bu sayının Timer0 çalışmaya başladığı andan itibaren Timer1 kesme üretene kadar geçen zaman olan 65536 mikrosaniyede motorun dönüş miktarı olduğu unutulmamalıdır. Öyleyse bu değer doğru orantıdan $65536*10^{-6}$ sayısına bölünerek saniyedeki dönüş sayısı bulunur. Son olarak 60 ile çarpılarak dakikadaki devir sayısı olarak yaklaşık 1,487 bulunur. Görsel olarak ölçülen değer daha yüksek olması Timer0 değeri kesme hizmet programında saklanırken Timer0 önbölücü yazmacında 0 ila 15 arasında değişen değerin bilinmemesinden kaynaklanmaktadır. Buradaki hesapta önbölücü yazmacında değerin 0 olduğu varsayılmıştır. Ancak uygulamada bu değerin 0 olma ihtimali 16'da 1'dir.

5.1.4.Bilgisayar ile Haberleşme

Mikrodnetleyici seri port üzerinden RS-232 protokolüyle haberleşmeyi sağlamak üzere bir adet Evrensel Senkron-Asenkron Alıcı-Verici (USART) modülüne sahiptir. USART kullanılmadan önce RCSTA (Alım Durum ve Kontrol Yazmacı), TXSTA (Gönderim Durum ve Kontrol Yazmacı) uygun şekilde koşullandırılarak konfigüre edilir. Ayrıca TRISC yazmacının 6. ve 7. bitleri birleşmek suretiyle seri gönderme ve alma amacıyla kullanılacak olan PORTC 6. ve 7. çıkışları ayarlanır. USART aşağıdaki kod parçası yardımıyla konfigüre edilmektedir.

```
bsf    STATUS,RP0    ;bank1'e geç
movlw  0x19
movwf  SPBRG          ;Baud hızı 9600 Baud/saniye olarak ayarlanıyor
movlw  b'00100100'
movwf  TXSTA
```

```

bcf STATUS,RP0      ;bank0'e geç
movlw b'10010000'
movwf RCSTA

```

Kod 5.4. USART Konfigürasyonu

Burada hangi bitin ne anlama geldiği ekte de verilen PIC16F877 veri kitapçığının USART ile ilgili bölümü incelenerek anlaşılabilir. Özetle seri port aktifleştirilmiş, USART asenkron modda çalışacak şekilde konfigüre edilmiş, 8 bit sürekli alım ve yine 8 bit gönderim seçilmiş, gönderim işlevi etkinleştirilmiş ve baud hızının ayarlanması açısından önemli olan baud oranı seçme biti birleştirilerek yüksek hız moduna geçilmiştir. Bunların dışında baud hızının ayarı SPBRG yazmacına hexadesimal 19 değerinin yazılmasıyla gerçekleştirilir. Veri kitapçığında verilen, baud oranı seçme bitinin birleştiği durumda kullanılan formül kullanılarak baud hızı $F_{OSC}/16(X+1) = 9615$ baud/saniye bulunur. Burada $F_{OSC} 4 \cdot 10^6$ olup X, SPBRG yazmacına yazılan hexadesimal 19 değerinin onluk sistemdeki karşılığı olan 25'dir. Bu hızda yine veri kitapçığından yüzde olarak hata oranı 0,16 gibi oldukça düşük bir değerdir. Daha yüksek hızların seçilmesi mümkün olsa da bunun hata oranını artıracığı göz önünde bulundurulmalıdır. Bilgisayardan gelen verileri işleyip cevap veren program kodu aşağıda verilmiştir:

VeriAl

```

movlw 06h
andwf RCSTA,W
btfss STATUS,Z
goto VeriAlimindaHataVar

```

VeriHazirmi

```

btfss PIR1,5
return

movf RCREG,W
bcf STATUS,Z
sublw b'00001010' ;W'den 10 çıkar
btfsc STATUS,Z
call SolMotorYon
movf RCREG,W
bcf STATUS,Z
sublw b'00010100' ;W'den 20 çıkar

```

```

btfsc STATUS,Z
call SolMotorHiz

movf RCREG,W
bcf STATUS,Z
sublw b'00011110' ;W'den 30 çıkar
btfsc STATUS,Z
call SagMotorYon
movf RCREG,W
bcf STATUS,Z
sublw b'00101000' ;W'den 40 çıkar
btfsc STATUS,Z
call SagMotorHiz

return

```

SolMotorYon

```

movf RCREG,W
movwf TXREG
bsf STATUS,5 ;bank1'e geç
bsf TXSTA, TXEN
bcf STATUS,5 ;bank0'a geç
btfss PORTB,0
goto SolMotorIleri
bcf PORTB,0
goto SolMotorGeri
SolMotorIleri bsf PORTB,0
SolMotorGeri return

```

SolMotorHiz

```

movf RCREG,W
movwf TXREG
bsf STATUS,5 ;bank1'e geç
bsf TXSTA, TXEN
bcf STATUS,5 ;bank0'a geç

```

SolMotorVeriAl

```

movlw 06h
andwf RCSTA,W
btfss STATUS,Z
goto SolMotorVeriAlimindaHataVar
btfss PIR1,5
goto SolMotorVeriAl

```

```

movf    SOL_MOTOR_DEVIR,W
movwf   TXREG
bsf     STATUS,5      ;bank1'e geç
bsf     TXSTA,TXEN
bcf     STATUS,5      ;bank0'a geç

```

```

movf    RCREG,W
movwf   CCPR1L
bcf     PIR1,5
return

```

SagMotorYon

```

                movf    RCREG,W
                movwf   TXREG
                bsf     STATUS,5      ;bank1'e geç
                bsf     TXSTA,TXEN
                bcf     STATUS,5      ;bank0'a geç
                btfss   PORTB,7
                goto    SagMotorIleri
                bcf     PORTB,7
                goto    SagMotorGeri
SagMotorIleri  bsf     PORTB,7
SagMotorGeri  return

```

SagMotorHiz

```

                movf    RCREG,W
                movwf   TXREG
                bsf     STATUS,5      ;bank1'e geç
                bsf     TXSTA,TXEN
                bcf     STATUS,5      ;bank0'a geç

```

SagMotorVeriAl

```

                movlw   06h
                andwf   RCSTA,W
                btfss   STATUS,Z
                goto    SagMotorVeriAlimindaHataVar
                btfss   PIR1,5
                goto    SagMotorVeriAl

                movf    SAG_MOTOR_DEVIR,W
                movwf   TXREG
                bsf     STATUS,5      ;bank1'e geç
                bsf     TXSTA,TXEN
                bcf     STATUS,5      ;bank0'a geç

```

```

movf    RCREG,W
movwf   CCPR2L
bcf     PIR1,5
return

```

VeriAlimindaHataVar

```

bcf     RCSTA,4
bsf     RCSTA,4
goto    VeriAl

```

SolMotorVeriAlimindaHataVar

```

bcf     RCSTA,4
bsf     RCSTA,4
goto    SolMotorVeriAl

```

SagMotorVeriAlimindaHataVar

```

bcf     RCSTA,4
bsf     RCSTA,4
goto    SagMotorVeriAl

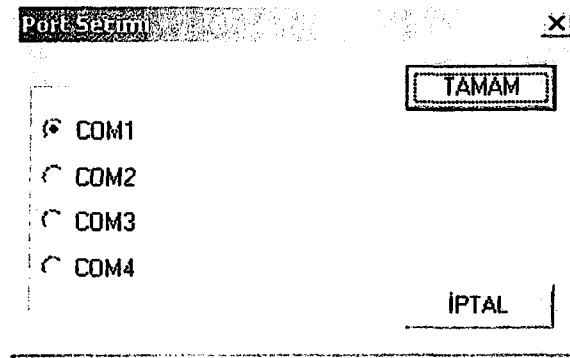
```

Kod 5.5. Bilgisayarla Haberleşme Rutini

5.2.Bilgisayar Yazılımı

5.2.1.Giriş

Bilgisayar tarafındaki yazılım C++ dili ile yazılmıştır. Derleyici olarak da Microsoft Visual C++ 6.0 kullanılmıştır. Yazılım görsel bir arabirim ve arka planda verileri işleyip mikrodnetleyiciye seri porttan gönderen ve mikrodnetleyicinin gönderdiklerini alan uygulama katmanından oluşmaktadır. Ayrıca seri portun seçimi için menüde **Kontrol** maddesi altında **Ayarlar** seçeneği bulunmaktadır. Ayarlar ekranı aşağıda görüldüğü gibidir:



Şekil 5.4. Ayarlar Ekran Görüntüsü

Seri portu seçip veri gönderen ve alan üç adet Windows API fonksiyonu yazılımda hayati önem taşımaktadır. Bunlar isminden de anlaşılacağı üzere dosya yaratıp üzerinde okuma ve yazma işlemleri yapmak için tasarlanmış olan CreateFile, ReadFile ve WriteFile fonksiyonlarıdır. CreateFile fonksiyonunun dosya adının belirtildiği parametresine string olarak seçilmek istenen port ismi girildiğinde ilgili portu yazılımsal olarak bir dosyaymış gibi yaratır. Dönüş değeri olarak Handle tipinde bir port numarası verir. ReadFile ve WriteFile fonksiyonlarına parametre olarak geçirilen bu numara ilgili porttan veri okuma ve porta veri yazmayı sağlar. Bu üç fonksiyona ait Microsoft tarafından verilen veri kitapçıkları ekte verilmiştir.

Program başladığında ilk iş olarak yazılımsal olarak port yaratır. Varsayılan değer olarak 1 numaralı port yaratılır. Eğer farklı bir port kullanılacaksa menüde **Kontrol** altında **Ayarlar**'dan ilgili port seçilir. Port yaratılması ve hemen ardından konfigürasyonu aşağıda verilen kod parçasıyla gerçekleştirilir:

```
PortNo = CreateFile(COMPort, GENERIC_READ | GENERIC_WRITE,    // Specify mode that open
device.
0,                                                            // the device isn't shared.
NULL,                                                         // the object gets a default security.
OPEN_EXISTING,                                               // Specify which action to take on file.
0,                                                            // default.
NULL);

m_bPortReady = GetCommState(PortNo, &m_dcb);
m_dcb.BaudRate = 9600;
m_dcb.ByteSize = 8;
```

```

m_dcb.Parity = NOPARITY;
m_dcb.StopBits = ONESTOPBIT;
m_dcb.fAbortOnError = TRUE;

m_bPortReady = SetCommState(PortNo, &m_dcb);

// Instance an object of COMMTIMEOUTS.
COMMTIMEOUTS comTimeOut;
// Specify time-out between character for receiving.
comTimeOut.ReadIntervalTimeout = 300;
// Specify value that is multiplied by the requested number of bytes to be read.
comTimeOut.ReadTotalTimeoutMultiplier = 300;
// Specify value is added to the product of the ReadTotalTimeoutMultiplier member.
comTimeOut.ReadTotalTimeoutConstant = 30;
// Specify value that is multiplied by the requested number of bytes to be sent.
comTimeOut.WriteTotalTimeoutMultiplier = 30;
// Specify value is added to the product of the WriteTotalTimeoutMultiplier member.
comTimeOut.WriteTotalTimeoutConstant = 30;
// Set the time-out parameter into device control.
SetCommTimeouts(PortNo,&comTimeOut);

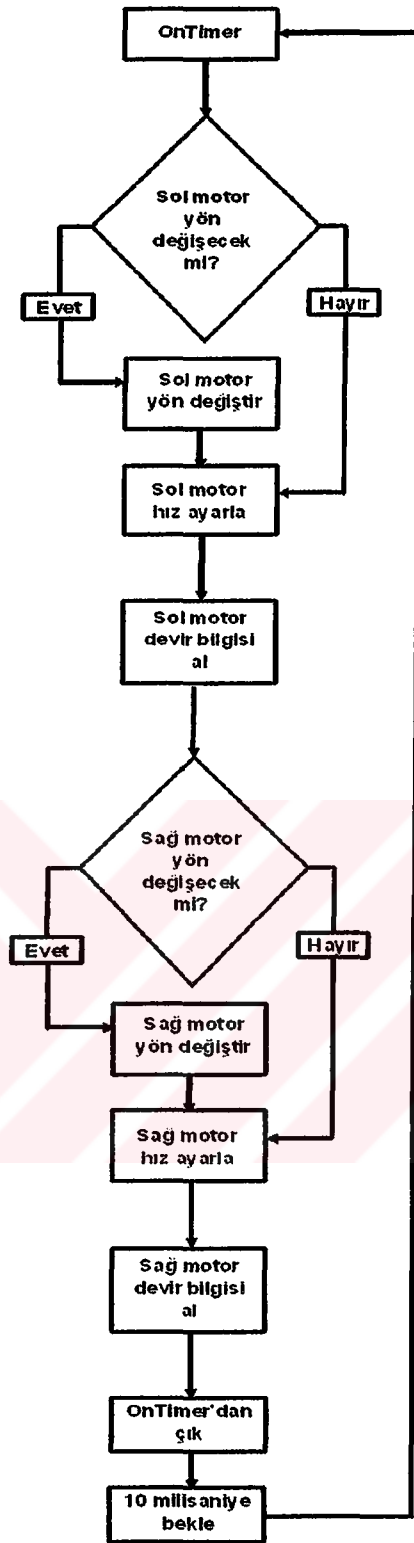
```

Kod 5.6. Port Oluşturma Kodu

Burada PortNo Handle, COMPort ise CString tipinde değişkenlerdir. Özetle port yaratılmakta ve konfigüre edilmektedir. Bu kod parçası port yaratılmak istendiği zaman aynen kullanılabilir. Gerekirse konfigürasyon değişiklikleri yapılabilir. Örneğin baud hızı farklı seçilebilir. Ancak bu durumda mikrodnetleyici yazılımında da aynı hız değişikliğinin yapılması gerektiği unutulmamalıdır. Koda ait açıklama Türkçe'ye çevrildiğinde anlamında sapmalar olabileceği düşüncesiyle orijinal İngilizce haliyle bırakılmıştır.

5.2.2.Bilgisayar ile Mikrodnetleyici Arasında Veri Alışveriş Algoritması

Bilgisayar ile mikrodnetleyici arasında veri alışveriş şekil 5.5'de verilen akış şemasında özetlenmiştir. Akış şemasından da anlaşılacağı üzere yönetim bilgisayar tarafından yapılır.



Şekil 5.5. Bilgisayar ile Mikrodenetleyici Arasındaki Veri Haberleşmesi Akış Diyagramı

Timer 10 milisaniyeye ulaştığında Windows'un WM_TIMER mesajı üretilerek OnTimer fonksiyonu çağırılır. OnTimer fonksiyonu modlara göre farklılık gösteren parçalar gözönüne alınmazsa aşağıda verilen kodu işletir:

```
void CDlgKontrol::OnTimer(UINT nIDEvent)
{
    BYTE p[1];
    LPDWORD length;
    length = new unsigned long;
    sizeBuffer = 1;
    sizeBuffer2 = 1;

    p[0] = 10;
    outputData = &p[0];
    if(sol_motor_yon_==TRUE)
    {
        if (WriteFile(PortNo, // handle to file to write to
            outputData,      // pointer to data to write to file
            sizeBuffer,      // number of bytes to write
            length, NULL) == 0) // pointer to number of bytes written
        {
            AfxMessageBox("Seri porta veri yazılamadı.");
        }

        if (ReadFile(PortNo, // handle of file to read
            inputData1,      // handle of file to read
            sizeBuffer2,     // number of bytes to read
            length,          // pointer to number of bytes read
            NULL) == 0)      // pointer to structure for data
        {
            AfxMessageBox("Seri porttan veri okunamadı.");
        }
    }

    p[0] = 20;
    outputData = &p[0];

    if (WriteFile(PortNo, // handle to file to write to
        outputData,      // pointer to data to write to file
        sizeBuffer,      // number of bytes to write
        length, NULL) == 0) // pointer to number of bytes written
```

```

{
    AfxMessageBox("Seri porta veri yazılamadı.");
}

:

if (ReadFile(PortNo, // handle of file to read
inputData1,        // handle of file to read
sizeBuffer2,       // number of bytes to read
length,            // pointer to number of bytes read
NULL) == 0)        // pointer to structure for data
{
    AfxMessageBox("Seri porttan veri okunamadı.");
}
BYTE i = *(BYTE*)inputData1;
if(i==20)
{
    p[0] = m_sol_motor_gerilim;
    outputData = &p[0];

    if (WriteFile(PortNo, // handle to file to write to
outputData,            // pointer to data to write to file
sizeBuffer,            // number of bytes to write
length,NULL) == 0)    // pointer to number of bytes written
    {
        AfxMessageBox("Seri porta veri yazılamadı.");
    }

    if (ReadFile(PortNo, // handle of file to read
inputData1,            // handle of file to read
sizeBuffer2,          // number of bytes to read
length,                // pointer to number of bytes read
NULL) == 0)           // pointer to structure for data
    {
        AfxMessageBox("Seri porttan veri okunamadı.");
    }
    m_sol_motor_devir = *(BYTE*)inputData1;
}

p[0] = 30;
outputData = &p[0];

if(sag_motor_yon_==TRUE)
{

    if (WriteFile(PortNo, // handle to file to write to

```

```

        outputData,          // pointer to data to write to file
        sizeBuffer,          // number of bytes to write
        length, NULL) == 0)   // pointer to number of bytes written
    {
        AfxMessageBox("Seri porta veri yazılamadı.");
    }

    if (ReadFile(PortNo, // handle of file to read
        inputData1,      // handle of file to read
        sizeBuffer2,      // number of bytes to read
        length,           // pointer to number of bytes read
        NULL) == 0)       // pointer to structure for data
    {
        AfxMessageBox("Seri porttan veri okunamadı.");
    }
}

```

```

p[0] = 40;
outputData = &p[0];

if (WriteFile(PortNo, // handle to file to write to
    outputData,        // pointer to data to write to file
    sizeBuffer,        // number of bytes to write
    length, NULL) == 0) // pointer to number of bytes written
{
    AfxMessageBox("Seri porta veri yazılamadı.");
}

```

```

if (ReadFile(PortNo, // handle of file to read
    inputData1,      // handle of file to read
    sizeBuffer2,      // number of bytes to read
    length,           // pointer to number of bytes read
    NULL) == 0)       // pointer to structure for data
{
    AfxMessageBox("Seri porttan veri okunamadı.");
}

```

```

i = *(BYTE*)inputData1;
if(i==40)
{
    p[0] = m_sag_motor_gerilim;
    outputData = &p[0];

    if (WriteFile(PortNo, // handle to file to write to

```

```

        outputData,      // pointer to data to write to file
        sizeBuffer,      // number of bytes to write
        length, NULL) == 0) // pointer to number of bytes written
    {
        AfxMessageBox("Seri porta veri yazılamadı.");
    }

    if (ReadFile(PortNo, // handle of file to read
        inputData1,      // handle of file to read
        sizeBuffer2,      // number of bytes to read
        length,          // pointer to number of bytes read
        NULL) == 0)      // pointer to structure for data
    {
        AfxMessageBox("Seri porttan veri okunamadı.");
    }
    m_sag_motor_devir = *(BYTE*)inputData1;

    delete length;
    sol_motor_yon_ = FALSE;
    sag_motor_yon_ = FALSE;
}

```

Kod 5.7. OnTimer Fonksiyonu

Burada her seferinde seri porttan mikrodnetleyiciye bir değeri gönderilmektedir. 5.1.4.'de bahsedilen mikrodnetleyici tarafındaki haberleşme algoritması gönderilen değerin anlamına göre ilgili alt programa dallanmaktadır. Bu değerler ondalık düzendeki karşılıkları ve anlamlarıyla şöyledir:

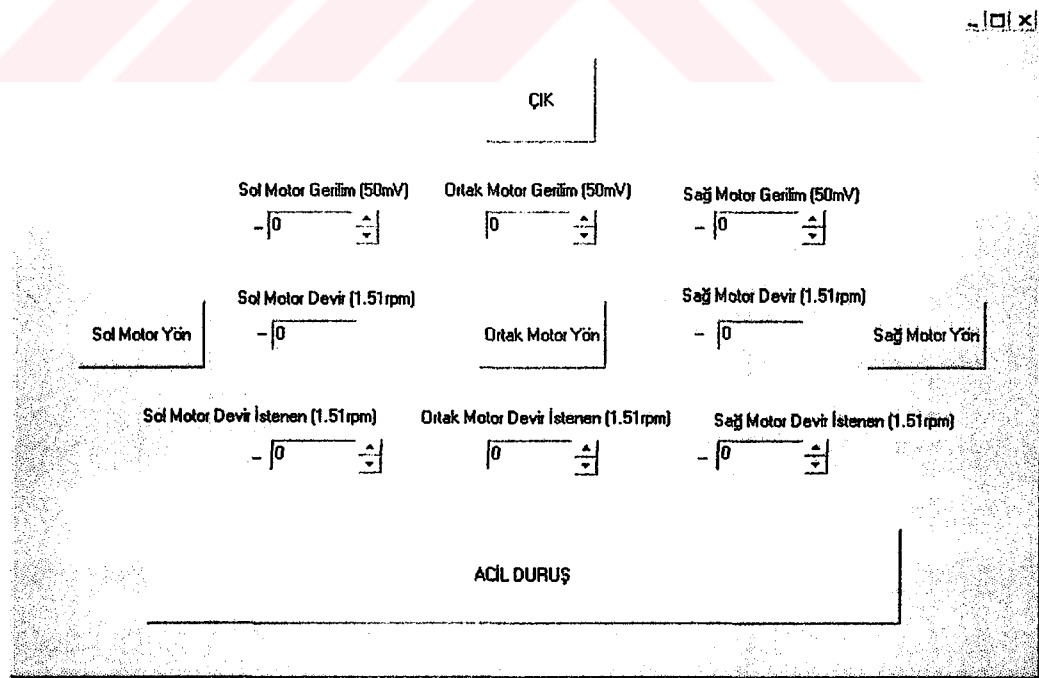
- 10 → Sol motor yönünü değiştir.
- 20 → Sol motor hız bilgisi geliyor.
- 30 → Sağ motor yönünü değiştir.
- 40 → Sağ motor hız bilgisi geliyor.

Örneğin mikrodnetleyici 10 değeri gönderildiğinde sol motorun yönünü değiştirir ve cevap olarak geriye 10 gönderir. 20 gönderilirse komutu anladığını belirten 20 değerini bilgisayara geri gönderir. 20 değerini geri alan bilgisayar bu sefer sol motor PWM yazmacına yazılacak değeri gönderir. Bu değeri alan mikrodnetleyici

bilgisayara o anda sol motor devir bilgisi olarak SOL_MOTOR_DEVIR yazmacında bulunan değeri gönderir. Benzer durum sağ motor için de geçerlidir.

5.2.3.Gerilim Modu

Mobil robot platformunun çalışma modlarından biri gerilim modudur. Menüde **Kontrol** maddesi altında **Robot Kontrol** seçeneği seçilerek açılan pencerede bulunan sol motor gerilim, sağ motor gerilim ve ortak motor gerilim girdi kutularına girilen değerlerin robota gönderilmesinden ibarettir. İki motora da aynı gerilim değerinin verileceği durumlarda ortak motor gerilim kutusu kolaylık sağlamaktadır. Bu kutulara en fazla 255 değeri girilebilir. Her girdi kutusunun sağ tarafında kendisiyle ilişkilendirilmiş bir spin kontrolü değerini fare veya klavyedeki oklar yardımıyla değiştirilmesine olanak sağlar. Ayrıca motor yönlerini, ilgili butonlara basarak tek tek veya aynı anda değiştirmek mümkündür. İki motorun da yönü aynı anda değiştirilmek istendiğinde ortak motor yön butonu kullanılabileceği gibi pencere üzerinde farenin sol tuşuna tıklanarak da aynı işlem gerçekleştirilebilir. Yön değişimi bir sonraki bölümde anlatılan devir modunda da aynıdır. Gerilim modundaki ekran görüntüsü devir moduyla ortak olmak üzere aşağıdaki gibidir:



Şekil 5.6. Gerilim ve Devir Modları Ortak Ekran Görüntüsü

5.2.4.Devir Modu

Gerilim moduyla aynı pencereyi paylaşan devir modunda ilgili kutulara motorların dönmesi istenen devir değerleri girilir. OnTimer fonksiyonu içinde bulunan devir ayarlama rutini gerçek devirle istenen deviri karşılaştırarak motor gerilimini artırır veya azaltır. Çok kısa bir süre içinde de gerçek devir değeri istenen devir değeriyle eşitlenir. Burada istenen devirle gerçek devir arasındaki farkın büyüklüğüne göre değer artımı veya azaltımı yapılır. Aradaki fark azaldıkça daha küçük artım veya azaltımlar yapılarak istenen devirle gerçek devir eşitlenmeye çalışılır. Aradaki farka göre verilecek artım ve azaltım değerleri üzerinde oynanarak optimizasyon yapılabilir. Devir ayarlama işlemini gerçekleştiren rutin aşağıda verilmiştir:

```
if(sol_motor_devir_ayarla==TRUE)
{
    if(m_sol_motor_devir_istenen>m_sol_motor_devir)
    {
        if((m_sol_motor_devir_istenen-m_sol_motor_devir)>40)
        {
            m_sol_motor_gerilim=m_sol_motor_gerilim+20;
            if(m_sol_motor_gerilim>255) m_sol_motor_gerilim=255;
        }

        else if((m_sol_motor_devir_istenen-m_sol_motor_devir)>15)
        {
            m_sol_motor_gerilim=m_sol_motor_gerilim+8;
            if(m_sol_motor_gerilim>255) m_sol_motor_gerilim=255;
        }

        else if((m_sol_motor_devir_istenen-m_sol_motor_devir)>3)
        {
            m_sol_motor_gerilim=m_sol_motor_gerilim+2;
            if(m_sol_motor_gerilim>255) m_sol_motor_gerilim=255;
        }
        else
        {
            m_sol_motor_gerilim=m_sol_motor_gerilim+1;
            if(m_sol_motor_gerilim>255) m_sol_motor_gerilim=255;
        }
    }
}
```

```

else if(m_sol_motor_devir_istenen<m_sol_motor_devir)
{
    if((m_sol_motor_devir-m_sol_motor_devir_istenen)>40)
    {
        m_sol_motor_gerilim=m_sol_motor_gerilim-20;
        if(m_sol_motor_gerilim<0) m_sol_motor_gerilim=0;
    }

    else if((m_sol_motor_devir-m_sol_motor_devir_istenen)>15)
    {
        m_sol_motor_gerilim=m_sol_motor_gerilim-8;
        if(m_sol_motor_gerilim<0) m_sol_motor_gerilim=0;
    }

    else if((m_sol_motor_devir-m_sol_motor_devir_istenen)>3)
    {
        m_sol_motor_gerilim=m_sol_motor_gerilim-2;
        if(m_sol_motor_gerilim<0) m_sol_motor_gerilim=0;
    }
    else
    {
        m_sol_motor_gerilim=m_sol_motor_gerilim-1;
        if(m_sol_motor_gerilim<0) m_sol_motor_gerilim=0;
    }
}

else    m_sol_motor_devir_istenen=m_sol_motor_devir;

UpdateData(FALSE);
}

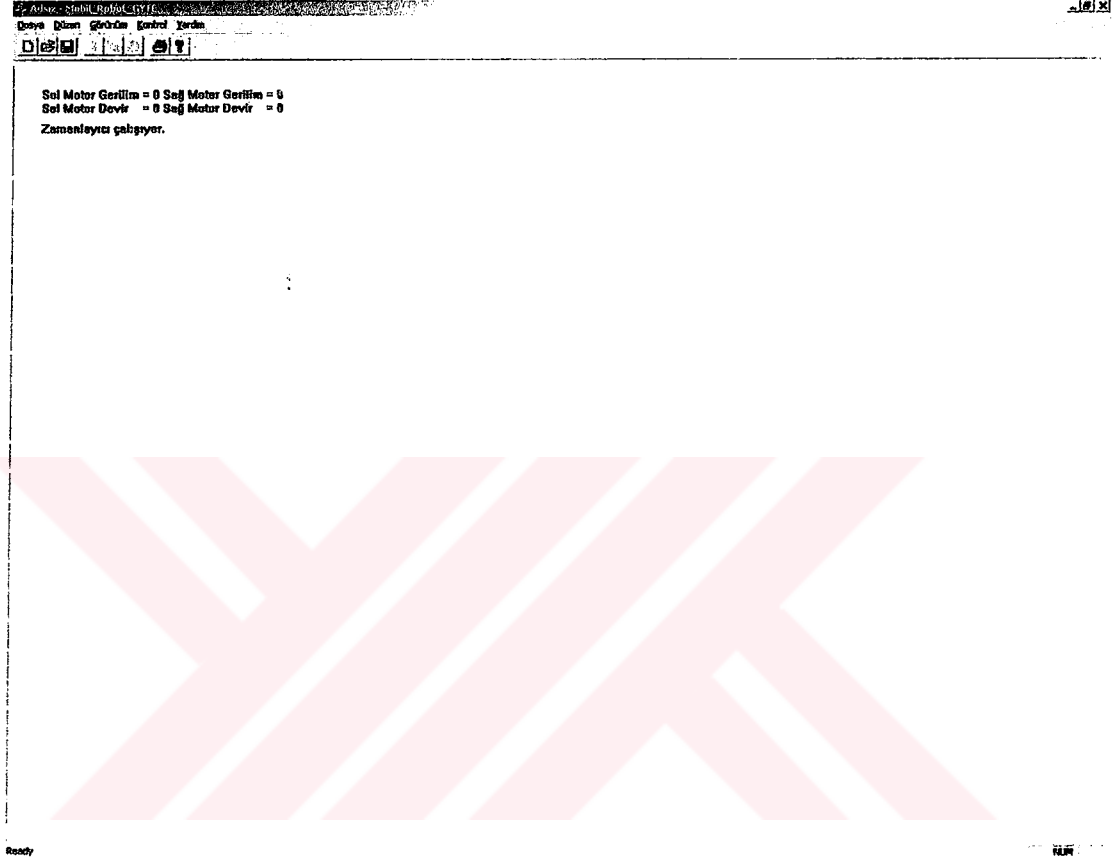
```

Kod 5.8. Devir Ayarlama Rutini

Yalnızca sol motorun devrini ayarlayan rutin verilmiş olup sağ motor için “sol” kelimelerinin “sağ” ile değiştirilmiş olmasından başka bir fark yoktur.

5.2.5.Oyunkolu (Joystik) Modu

Mobil robotun hareketlerinin bilgisayarın faresinin hareketlerine göre kontrol edildiği moddur. Ekran görüntüsü aşağıda verilmiştir:



Şekil 5.7. Oyunkolu Modu Ekran Görüntüsü

Bu modda imleç pozisyonunun değişimleri yakalanarak farenin ileri hareketinde motor gerilimleri artırılırken geri hareketinde azaltılır. Sağa veya solahareketlerde ise sapma ile orantılı olarak iki motorun gerilimleri farklılaştırır. Örneğin fare sağa hareket ettirilirse bu sırada robot ileri yönde düz hareketini sürdürmekteyse sağ motor gerilimi sol motor geriliminin belli bir oranı kadar azaltılarak sol motorun sağ motora göre daha hızlı dönmesi sağlanır. Bu durumda robot sağa doğru yönelecektir. Fare sağa çekilirken ileri doğru da götürülürse robot hızlanarak sağa dönecektir. Farenin sağa hareketi en fazla sağ motor geriliminin sol motor geriliminin negatif değerine ulaşmasına izin verir. Bu durumda robot kendi eksenini etrafında dönmelidir. Ayrıca gerilimler farenin scroll tuşu ile artırılıp

azaltılabilir. Farenin sağ tuşuna basıldığında iki motor gerilimi sıfırlanır. Sol tuş ise zamanlayıcıyı durdurarak farenin motor gerilimlerini etkilemeden hareket ettirilebilmesine imkan tanır. Bu durumdayken sol tuşa tekrar basılırsa zamanlayıcı çalışmaya başlar. Program kodu aşağıda verilmiştir:

```
CPoint Kursor_Pozisyon(0,0);
GetCursorPos(&Kursor_Pozisyon);

if(sol_motor_yon==TRUE) m_sol_motor_gerilim = -m_sol_motor_gerilim;
if(sag_motor_yon==TRUE)m_sag_motor_gerilim = -m_sag_motor_gerilim;

if((((Kursor_Pozisyon.y-Kursor_Pozisyon_Temp.y)>2)||((Kursor_Pozisyon.y-
Kursor_Pozisyon_Temp.y)<=-2))
{
    m_sol_motor_gerilim=m_sol_motor_gerilim-((Kursor_Pozisyon.y-
    Kursor_Pozisyon_Temp.y)/2);
    m_sag_motor_gerilim=m_sag_motor_gerilim-((Kursor_Pozisyon.y-
    Kursor_Pozisyon_Temp.y)/2);
    Kursor_Pozisyon_Temp.y=Kursor_Pozisyon.y;
}

if(((Kursor_Pozisyon.x-Kursor_Pozisyon_Temp.x)>20)
{
    if(m_sag_motor_gerilim>0)
    {
        if(m_sol_motor_gerilim<m_sag_motor_gerilim)
        {
            m_sol_motor_gerilim =
            m_sol_motor_gerilim+m_sag_motor_gerilim/5;
            if(m_sol_motor_gerilim>m_sag_motor_gerilim)
                m_sol_motor_gerilim = m_sag_motor_gerilim;
        }
        else
        {
            m_sag_motor_gerilim=m_sag_motor_gerilim-
            m_sol_motor_gerilim/5;
            if(m_sag_motor_gerilim<-m_sol_motor_gerilim)
                m_sag_motor_gerilim = -m_sol_motor_gerilim;
        }
    }
}
else
```

```

{
    if(m_sag_motor_gerilim<m_sol_motor_gerilim)
    {
        m_sol_motor_gerilim =
        m_sol_motor_gerilim+m_sag_motor_gerilim/5;
        if(m_sol_motor_gerilim<m_sag_motor_gerilim)
            m_sol_motor_gerilim = m_sag_motor_gerilim;
    }
    else
    {
        m_sag_motor_gerilim=m_sag_motor_gerilim-
        m_sol_motor_gerilim/5;
        if(m_sag_motor_gerilim>-m_sol_motor_gerilim)
            m_sag_motor_gerilim = -m_sol_motor_gerilim;
    }
}

Kursor_Pozisyon_Temp.x=Kursor_Pozisyon.x;
}

else if(((Kursor_Pozisyon.x-Kursor_Pozisyon_Temp.x)<=-20)
{
    if(m_sol_motor_gerilim>0)
    {
        if(m_sag_motor_gerilim<m_sol_motor_gerilim)
        {
            m_sag_motor_gerilim =
            m_sag_motor_gerilim+m_sol_motor_gerilim/5;
            if(m_sag_motor_gerilim>m_sol_motor_gerilim)
                m_sag_motor_gerilim = m_sol_motor_gerilim;
        }
        else
        {
            m_sol_motor_gerilim=m_sol_motor_gerilim-
            m_sag_motor_gerilim/5;
            if(m_sol_motor_gerilim<-m_sag_motor_gerilim)
                m_sol_motor_gerilim = -m_sag_motor_gerilim;
        }
    }
    else
    {
        if(m_sol_motor_gerilim<m_sag_motor_gerilim)
        {
            m_sag_motor_gerilim =
            m_sag_motor_gerilim+m_sol_motor_gerilim/5;

```

```

        if(m_sag_motor_gerilim<m_sol_motor_gerilim)
            m_sag_motor_gerilim = m_sol_motor_gerilim;
    }
    else
    {
        m_sol_motor_gerilim=m_sol_motor_gerilim-
        m_sag_motor_gerilim/5;
        if(m_sol_motor_gerilim>-m_sag_motor_gerilim)
            m_sol_motor_gerilim = -m_sag_motor_gerilim;
    }
}

Kursor_Pozisyon_Temp.x=Kursor_Pozisyon.x;
}

//Fare hareketleri ile orantılı olarak motorlara verilecek gerilim ve yönleri hesaplayan rutin
if(m_sol_motor_gerilim<0)
{
    m_sol_motor_gerilim = -m_sol_motor_gerilim;
    if(sol_motor_yon==FALSE)
    {
        sol_motor_yon=TRUE;
        sol_motor_yon_=TRUE;
    }
}
else
{
    if(sol_motor_yon==TRUE)
    {
        sol_motor_yon=FALSE;
        sol_motor_yon_=TRUE;
    }
}

if(m_sag_motor_gerilim<0)
{
    m_sag_motor_gerilim = -m_sag_motor_gerilim;
    if(sag_motor_yon==FALSE)
    {
        sag_motor_yon=TRUE;
        sag_motor_yon_=TRUE;
    }
}
else
{
    if(sag_motor_yon==TRUE)

```

```
{  
    sag_motor_yon=FALSE;  
    sag_motor_yon_=TRUE;  
}  
  
if(m_sol_motor_gerilim>255)  
    m_sol_motor_gerilim=255;  
if(m_sag_motor_gerilim>255)  
    m_sag_motor_gerilim=255;
```

Kod 5.9. Oyunkolu Modu Gerilim Ayarlama Rutini

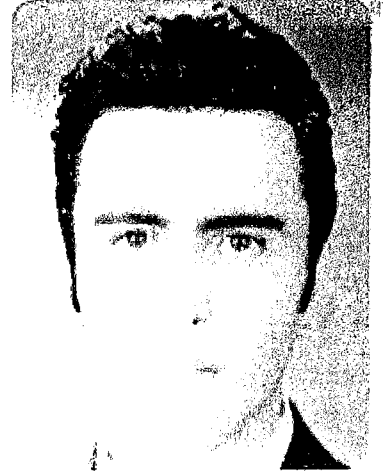


6.SONUÇLAR VE ÖNERİLER

Bu çalışmada gerçekleştirilen mobil robot platformu bilgisayar mikroişlemcisinin gücünden yararlanmaktadır. Bu da kontrol tasarımcısına büyük kolaylık sağlamaktadır. Zira bilgisayar yazılımının koduna yapılacak birkaç eklemeye istenilen kontrol algoritması denenebilir. Bilgisayarla mikrodnetleyici arasındaki veri alışveriş algoritması hız,yön ve devir bilgilerine ek olarak farklı veriler gönderip almaya uygundur. Örneğin robota eklenecek algılayıcıların algıladıkları değerler bilgisayara gönderilerek işlenebilir. Elektronik devre üzerinde değişiklik yapılarak akünün şarj durumu bilgisayardan izlenebilir. Akü gerilimi belli bir değerin altına düştüğünde robotun şarj aletine yönelmesini sağlayacak bir algılayıcı düzeneği geliştirilebilir. Mikrodnetleyici ve bilgisayar yazılımları uygun şekilde değiştirilerek motorların belli bir gerilimde belli bir süre dönüp bu sürenin sonunda başka bir gerilimde dönmeye zorlanması mümkündür. Aynı durum yön için de geçerlidir. Ancak öncelikle robotun kablo bağlantısından kurtarılması önerilir. Bunun için RF seri sayısal veri alıcı-verici devresi tasarlanabileceği gibi piyasada satılan seri iletişime uygun RF modemlerden satın alınabilir. Uygun bir örnek Ek 7.8.'de verilmiştir. Sonuç olarak tasarlanıp gerçekleştirilen üç tekerlekli mobil robot platformu esnek tasarımıyla gelişmeye açık olup mobil robot algoritmalarının denenmesi için düşük maliyetli bir çözümdür.

KAYNAKLAR

- [1] **Nonlinear Control of Wheeled Mobile Robots**, Dixon W.E., Dawson D.M., Zengeroglu E., Behal A., 2001 Springer USA
- [2] **Microsoft Visual C++ Williams, M., 1999 Sistem İstanbul**
- [3] **Endüstriyel Elektronik, Çetin K., 2002 Birsen İstanbul**
- [4] **Adım Adım PICmicro Programlama, Bodur Y., 2002 Era Ankara**
- [5] **A'dan Z'ye C Kılavuzu, Aslan K., 2002 Pusula İstanbul**
- [6] **Microchip PICmicro PIC16F877 Data Sheet 28/40-Pin 8-Bit CMOS FLASH Microcontrollers, 2001 Microchip Technology Inc. / www.microchip.com**
- [7] **Microsoft Corporation MSDN Library / www.microsoft.com/msdn**
- [8] **Kitab'ül-Cami Beyn'el İlmi ve el-Ameli'en Nafi fi Sınaati'l Hiyel (Mekanik Hareketlerden Mühendislikte Faydalanmayı İçeren Kitap), Ebû'l İz El Cezeri, 12. Yüzyıl**
- [9] **Rossum'un Evrensel Robotları, Karel Capek, 1920**
- [10] **Runaround, Isaac Asimov, 1942**



ÖZGEÇMİŞ

Oğuz Öktem 9 Haziran 1978'de İstanbul'da doğdu. 1996'da Saint-Joseph Lisesi'ni bitirdikten sonra 2001'de Yıldız Teknik Üniversitesi Elektronik ve Haberleşme Mühendisliği'nden lisans diplomasını aldı. Hastane Otomasyon Yazılımı yapan BILMED firmasında 8 ay çalıştıktan sonra OYAK-RENAULT Otomobil Fabrikaları'nda metot mühendisi olarak çalışmaya başladı. Bu görevini halen sürdürmektedir. Fransızca, İngilizce ve İtalyanca biliyor. Uzakdoğu kökenli strateji oyunu Go'ya meraklı. Futbol oynamayı çok seviyor. Mühendis olmasaydı Galatasaray'ın forveti olacaktı.

EKLER

EK-1 PIC 16F877 Timer0 Birimi Veri Kitapçığı

EK-2 PIC 16F877 Timer1 Birimi Veri Kitapçığı

EK-3 PIC 16F877 Timer Birimi Veri Kitapçığı

EK-4 PIC 16F877 PWM Birimleri Veri Kitapçığı

EK-5 PIC 16F877 USART Birimi Veri Kitapçığı

EK-6 PITTMAN GM9236S019 DC Motor Veri Kitapçığı

EK-7 RF Modem Veri Kitapçığı

EK-8 CD



5.0 TIMER0 MODULE

The Timer0 module timer/counter has the following features:

- 8-bit timer/counter
- Readable and writable
- 8-bit software programmable prescaler
- Internal or external clock select
- Interrupt on overflow from FFh to 00h
- Edge select for external clock

Figure 5-1 is a block diagram of the Timer0 module and the prescaler shared with the WDT.

Additional information on the Timer0 module is available in the PICmicro™ Mid-Range MCU Family Reference Manual (DS33023).

Timer mode is selected by clearing bit T0CS (OPTION_REG<5>). In timer mode, the Timer0 module will increment every instruction cycle (without prescaler). If the TMR0 register is written, the increment is inhibited for the following two instruction cycles. The user can work around this by writing an adjusted value to the TMR0 register.

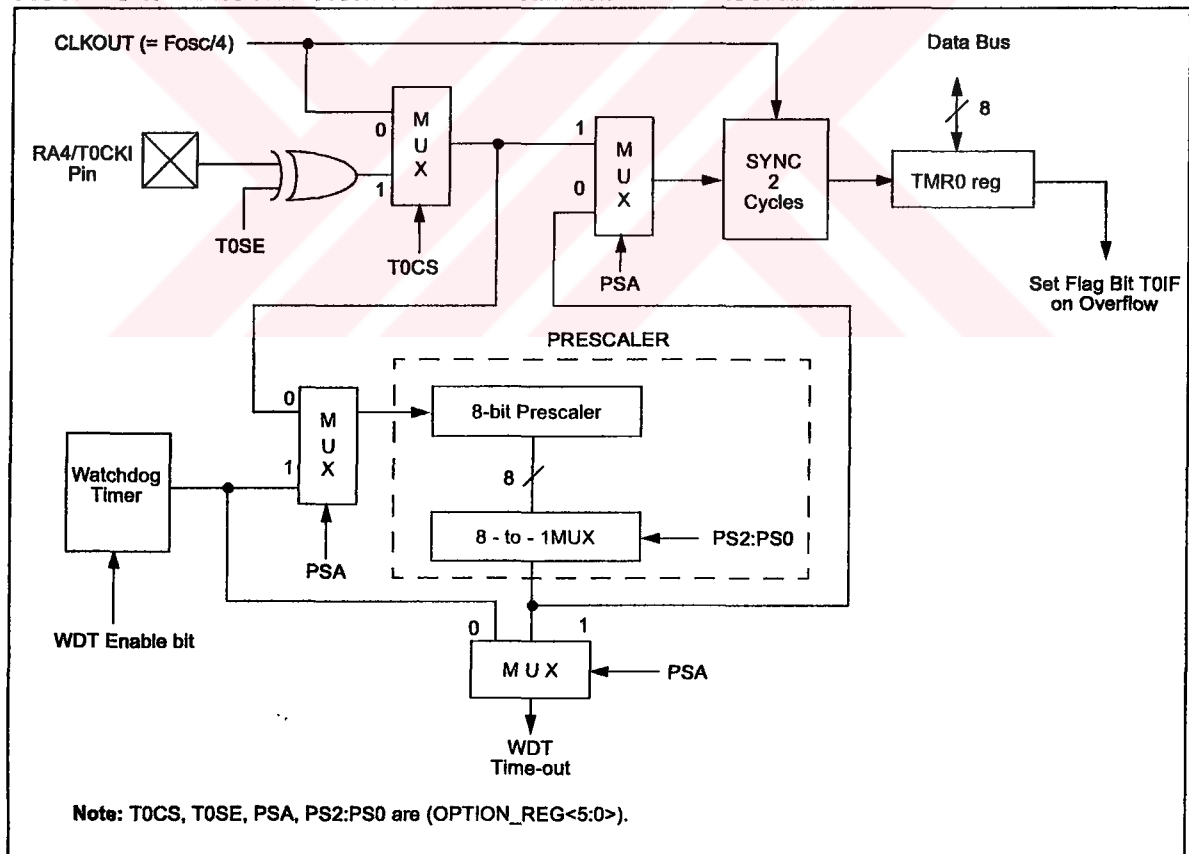
Counter mode is selected by setting bit T0CS (OPTION_REG<5>). In counter mode, Timer0 will increment either on every rising or falling edge of pin RA4/T0CKI. The incrementing edge is determined by the Timer0 Source Edge Select bit T0SE (OPTION_REG<4>). Clearing bit T0SE selects the rising edge. Restrictions on the external clock input are discussed in detail in Section 5.2.

The prescaler is mutually exclusively shared between the Timer0 module and the watchdog timer. The prescaler is not readable or writable. Section 5.3 details the operation of the prescaler.

5.1 Timer0 Interrupt

The TMR0 interrupt is generated when the TMR0 register overflows from FFh to 00h. This overflow sets bit T0IF (INTCON<2>). The interrupt can be masked by clearing bit T0IE (INTCON<5>). Bit T0IF must be cleared in software by the Timer0 module interrupt service routine before re-enabling this interrupt. The TMR0 interrupt cannot awaken the processor from SLEEP since the timer is shut off during SLEEP.

FIGURE 5-1: BLOCK DIAGRAM OF THE TIMER0/WDT PRESCALER



5.2 Using Timer0 with an External Clock

When no prescaler is used, the external clock input is the same as the prescaler output. The synchronization of T0CKI with the internal phase clocks is accomplished by sampling the prescaler output on the Q2 and Q4 cycles of the internal phase clocks. Therefore, it is necessary for T0CKI to be high for at least 2Tosc (and a small RC delay of 20 ns) and low for at least 2Tosc (and a small RC delay of 20 ns). Refer to the electrical specification of the desired device.

5.3 Prescaler

There is only one prescaler available, which is mutually exclusively shared between the Timer0 module and the watchdog timer. A prescaler assignment for the Timer0

module means that there is no prescaler for the watchdog timer, and vice-versa. This prescaler is not readable or writable (see Figure 5-1).

The PSA and PS2:PS0 bits (OPTION_REG<3:0>) determine the prescaler assignment and prescale ratio.

When assigned to the Timer0 module, all instructions writing to the TMR0 register (e.g. CLRF 1, MOVWF 1, BSF 1, x...etc.) will clear the prescaler. When assigned to WDT, a CLRWDI instruction will clear the prescaler along with the Watchdog Timer. The prescaler is not readable or writable.

Note: Writing to TMR0, when the prescaler is assigned to Timer0, will clear the prescaler count, but will not change the prescaler assignment.

REGISTER 5-1: OPTION_REG REGISTER

R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
RBPU	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0
bit 7							bit 0
bit 7:	RBPU						
bit 6:	INTEDG						
bit 5:	T0CS: TMR0 Clock Source Select bit						
	1 = Transition on T0CKI pin						
	0 = Internal instruction cycle clock (CLKOUT)						
bit 4:	T0SE: TMR0 Source Edge Select bit						
	1 = Increment on high-to-low transition on T0CKI pin						
	0 = Increment on low-to-high transition on T0CKI pin						
bit 3:	PSA: Prescaler Assignment bit						
	1 = Prescaler is assigned to the WDT						
	0 = Prescaler is assigned to the Timer0 module						
bit 2-0:	PS2:PS0: Prescaler Rate Select bits						
	Bit Value	TMR0 Rate	WDT Rate				
	000	1:2	1:1				
	001	1:4	1:2				
	010	1:8	1:4				
	011	1:16	1:8				
	100	1:32	1:16				
	101	1:64	1:32				
	110	1:128	1:64				
	111	1:256	1:128				

R = Readable bit
W = Writable bit
U = Unimplemented bit, read as '0'
- n = Value at POR reset

Note: To avoid an unintended device RESET, the instruction sequence shown in the PICmicro™ Mid-Range MCU Family Reference Manual (DS33023) must be executed when changing the prescaler assignment from Timer0 to the WDT. This sequence must be followed even if the WDT is disabled.

TABLE 5-1: REGISTERS ASSOCIATED WITH TIMER0

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR, BOR	Value on all other resets
01h,101h	TMR0	Timer0 module's register								xxxx xxxx	uuuu uuuu
0Bh,8Bh, 10Bh,18Bh	INTCON	GIE	PEIE	TOIE	INTE	RBIE	TOIF	INTF	RBIF	0000 000x	0000 000u
81h,181h	OPTION_REG	RBP $\bar{U}$	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0	1111 1111	1111 1111

Legend: x = unknown, u = unchanged, - = unimplemented locations read as '0'. Shaded cells are not used by Timer0.

6.0 TIMER1 MODULE

The Timer1 module is a 16-bit timer/counter consisting of two 8-bit registers (TMR1H and TMR1L), which are readable and writable. The TMR1 Register pair (TMR1H:TMR1L) increments from 0000h to FFFFh and rolls over to 0000h. The TMR1 Interrupt, if enabled, is generated on overflow, which is latched in interrupt flag bit TMR1IF (PIR1<0>). This interrupt can be enabled/disabled by setting/clearing TMR1 interrupt enable bit TMR1IE (PIE1<0>).

Timer1 can operate in one of two modes:

- As a timer
- As a counter

The operating mode is determined by the clock select bit, TMR1CS (T1CON<1>).

In timer mode, Timer1 increments every instruction cycle. In counter mode, it increments on every rising edge of the external clock input.

Timer1 can be enabled/disabled by setting/clearing control bit TMR1ON (T1CON<0>).

Timer1 also has an internal "reset input". This reset can be generated by either of the two CCP modules (Section 8.0). Register 6-1 shows the Timer1 control register.

When the Timer1 oscillator is enabled (T1OSCEN is set), the RC1/T1OSI/CCP2 and RC0/T1OSO/T1CKI pins become inputs. That is, the TRISC<1:0> value is ignored.

Additional information on timer modules is available in the PICmicro™ Mid-range MCU Family Reference Manual (DS33023).

REGISTER 6-1: T1CON: TIMER1 CONTROL REGISTER (ADDRESS 10h)

U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	T1CKPS1	T1CKPS0	T1OSCEN	T1SYNC	TMR1CS	TMR1ON
bit7							bit0

R = Readable bit
W = Writable bit
U = Unimplemented bit, read as '0'
-n = Value at POR reset

bit 7-6: **Unimplemented:** Read as '0'

bit 5-4: **T1CKPS1:T1CKPS0:** Timer1 Input Clock Prescale Select bits
11 = 1:8 Prescale value
10 = 1:4 Prescale value
01 = 1:2 Prescale value
00 = 1:1 Prescale value

bit 3: **T1OSCEN:** Timer1 Oscillator Enable Control bit
1 = Oscillator is enabled
0 = Oscillator is shut off (The oscillator inverter is turned off to eliminate power drain)

bit 2: **T1SYNC:** Timer1 External Clock Input Synchronization Control bit

TMR1CS = 1
1 = Do not synchronize external clock input
0 = Synchronize external clock input

TMR1CS = 0
This bit is ignored. Timer1 uses the internal clock when TMR1CS = 0.

bit 1: **TMR1CS:** Timer1 Clock Source Select bit
1 = External clock from pin RC0/T1OSO/T1CKI (on the rising edge)
0 = Internal clock (Fosc/4)

bit 0: **TMR1ON:** Timer1 On bit
1 = Enables Timer1
0 = Stops Timer1

6.1 Timer1 Operation in Timer Mode

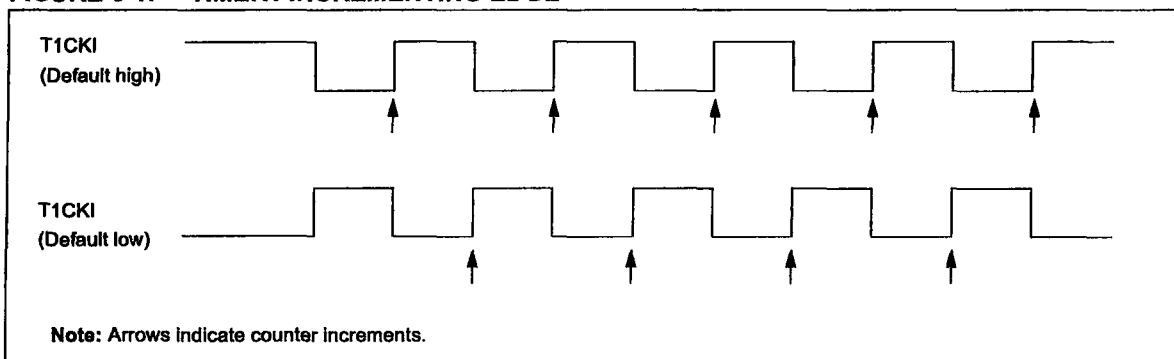
Timer mode is selected by clearing the TMR1CS (T1CON<1>) bit. In this mode, the input clock to the timer is Fosc/4. The synchronize control bit T1SYNC (T1CON<2>) has no effect since the internal clock is always in sync.

6.2 Timer1 Counter Operation

Timer1 may operate in asynchronous or usynchronous mode depending on the setting of the TMR1CS bit.

When Timer1 is being incremented via an external source, increments occur on a rising edge. After Timer1 is enabled in counter mode, the module must first have a falling edge before the counter begins to increment.

FIGURE 6-1: TIMER1 INCREMENTING EDGE



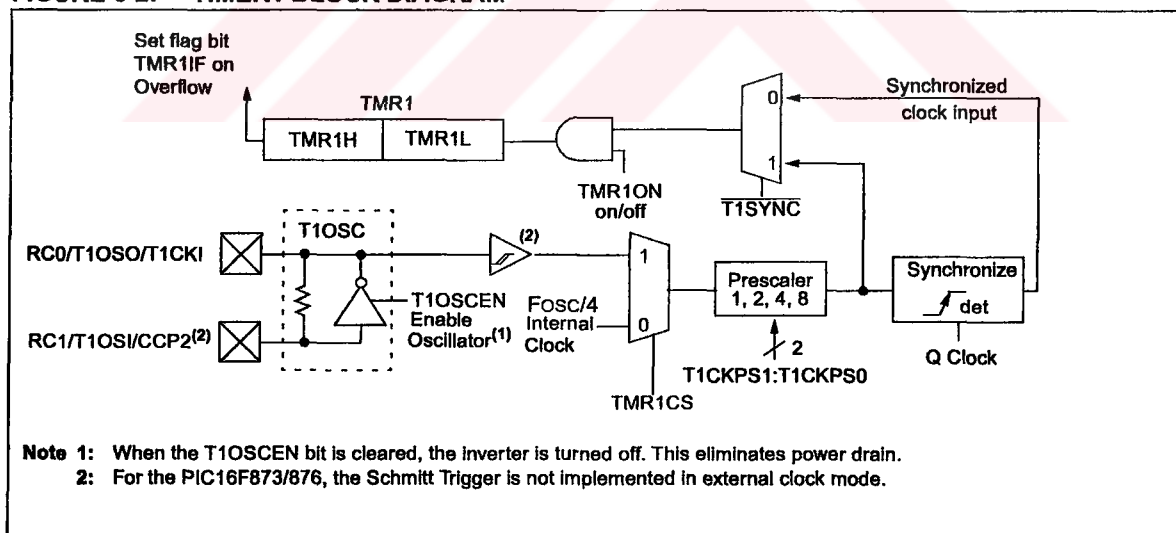
6.3 Timer1 Operation in Synchronized Counter Mode

Counter mode is selected by setting bit TMR1CS. In this mode, the timer increments on every rising edge of clock input on pin RC1/T1OSI/CCP2, when bit T1OSCEN is set, or on pin RC0/T1OSO/T1CKI, when bit T1OSCEN is cleared.

If T1SYNC is cleared, then the external clock input is synchronized with internal phase clocks. The synchronization is done after the prescaler stage. The prescaler stage is an asynchronous ripple-counter.

In this configuration, during SLEEP mode, Timer1 will not increment even if the external clock is present, since the synchronization circuit is shut off. The prescaler however will continue to increment.

FIGURE 6-2: TIMER1 BLOCK DIAGRAM



6.4 Timer1 Operation in Asynchronous Counter Mode

If control bit T1SYNC (T1CON<2>) is set, the external clock input is not synchronized. The timer continues to increment asynchronous to the internal phase clocks. The timer will continue to run during SLEEP and can generate an interrupt on overflow, which will wake-up the processor. However, special precautions in software are needed to read/write the timer (Section 6.4.1).

In asynchronous counter mode, Timer1 can not be used as a time-base for capture or compare operations.

6.4.1 READING AND WRITING TIMER1 IN ASYNCHRONOUS COUNTER MODE

Reading TMR1H or TMR1L while the timer is running from an external asynchronous clock will guarantee a valid read (taken care of in hardware). However, the user should keep in mind that reading the 16-bit timer in two 8-bit values itself poses certain problems, since the timer may overflow between the reads.

For writes, it is recommended that the user simply stop the timer and write the desired values. A write contention may occur by writing to the timer registers, while the register is incrementing. This may produce an unpredictable value in the timer register.

Reading the 16-bit value requires some care. Examples 12-2 and 12-3 in the PICmicro™ Mid-Range MCU Family Reference Manual (DS33023) show how to read and write Timer1 when it is running in asynchronous mode.

6.5 Timer1 Oscillator

A crystal oscillator circuit is built-in between pins T1OSI (input) and T1OSO (amplifier output). It is enabled by setting control bit T1OSCEN (T1CON<3>). The oscillator is a low power oscillator rated up to 200 kHz. It will continue to run during SLEEP. It is primarily intended for use with a 32 kHz crystal. Table 6-1 shows the capacitor selection for the Timer1 oscillator.

The Timer1 oscillator is identical to the LP oscillator. The user must provide a software time delay to ensure proper oscillator start-up.

TABLE 6-1: CAPACITOR SELECTION FOR THE TIMER1 OSCILLATOR

Osc Type	Freq	C1	C2
LP	32 kHz	33 pF	33 pF
	100 kHz	15 pF	15 pF
	200 kHz	15 pF	15 pF
These values are for design guidance only.			
Crystals Tested:			
32.768 kHz	Epson C-001R32.768K-A	± 20 PPM	
100 kHz	Epson C-2 100.00 KC-P	± 20 PPM	
200 kHz	STD XTL 200.000 kHz	± 20 PPM	
Note 1: Higher capacitance increases the stability of oscillator but also increases the start-up time.			
2: Since each resonator/crystal has its own characteristics, the user should consult the resonator/crystal manufacturer for appropriate values of external components.			

6.6 Resetting Timer1 using a CCP Trigger Output

If the CCP1 or CCP2 module is configured in compare mode to generate a "special event trigger" (CCP1M3:CCP1M0 = 1011), this signal will reset Timer1.

Note: The special event triggers from the CCP1 and CCP2 modules will not set Interrupt flag bit TMR1IF (PIR1<0>).

Timer1 must be configured for either timer or synchronized counter mode to take advantage of this feature. If Timer1 is running in asynchronous counter mode, this reset operation may not work.

In the event that a write to Timer1 coincides with a special event trigger from CCP1 or CCP2, the write will take precedence.

In this mode of operation, the CCPRxH:CCPRxL register pair effectively becomes the period register for Timer1.

6.7 Resetting of Timer1 Register Pair (TMR1H, TMR1L)

TMR1H and TMR1L registers are not reset to 00h on a POR or any other reset except by the CCP1 and CCP2 special event triggers.

T1CON register is reset to 00h on a Power-on Reset or a Brown-out Reset, which shuts off the timer and leaves a 1:1 prescaler. In all other resets, the register is unaffected.

6.8 Timer1 Prescaler

The prescaler counter is cleared on writes to the TMR1H or TMR1L registers.

PIC16F87X

TABLE 6-2: REGISTERS ASSOCIATED WITH TIMER1 AS A TIMER/COUNTER

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR, BOR	Value on all other resets
0Bh,8Bh, 10Bh, 18Bh	INTCON	GIE	PEIE	TOIE	INTE	RBIE	TOIF	INTF	RBIF	0000 000x	0000 000u
0Ch	PIR1	PSPIF ⁽¹⁾	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	0000 0000	0000 0000
8Ch	PIE1	PSPIE ⁽¹⁾	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	0000 0000	0000 0000
0Eh	TMR1L	Holding register for the Least Significant Byte of the 16-bit TMR1 register								xxxx xxxx	uuuu uuuu
0Fh	TMR1H	Holding register for the Most Significant Byte of the 16-bit TMR1 register								xxxx xxxx	uuuu uuuu
10h	T1CON	—	—	T1CKPS1	T1CKPS0	T1OSCEN	T1SYNC	TMR1CS	TMR1ON	--00 0000	--uu uuuu

Legend: x = unknown, u = unchanged, - = unimplemented read as '0'. Shaded cells are not used by the Timer1 module.

Note 1: Bits PSPIE and PSPIF are reserved on the PIC16F873/874; always maintain these bits clear.

7.0 TIMER2 MODULE

Timer2 is an 8-bit timer with a prescaler and a postscaler. It can be used as the PWM time-base for the PWM mode of the CCP module(s). The TMR2 register is readable and writable, and is cleared on any device reset.

The input clock ($F_{osc}/4$) has a prescale option of 1:1, 1:4 or 1:16, selected by control bits T2CKPS1:T2CKPS0 (T2CON<1:0>).

The Timer2 module has an 8-bit period register PR2. Timer2 increments from 00h until it matches PR2 and then resets to 00h on the next increment cycle. PR2 is a readable and writable register. The PR2 register is initialized to FFh upon reset.

The match output of TMR2 goes through a 4-bit postscaler (which gives a 1:1 to 1:16 scaling inclusive) to generate a TMR2 interrupt (latched in flag bit TMR2IF, (PIR1<1>)).

Timer2 can be shut off by clearing control bit TMR2ON (T2CON<2>) to minimize power consumption.

Register 7-1 shows the Timer2 control register.

Additional information on timer modules is available in the PICmicro™ Mid-Range MCU Family Reference Manual (DS33023).

7.1 Timer2 Prescaler and Postscaler

The prescaler and postscaler counters are cleared when any of the following occurs:

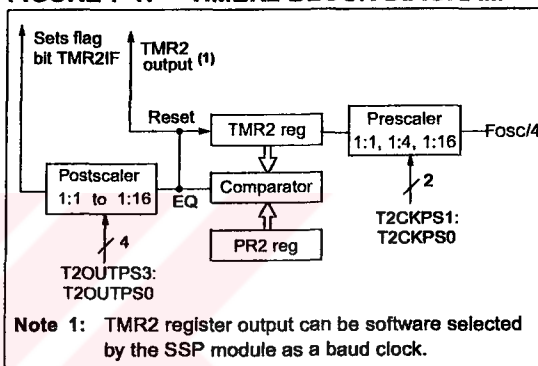
- a write to the TMR2 register
- a write to the T2CON register
- any device reset (POR, MCLR reset, WDT reset or BOR)

TMR2 is not cleared when T2CON is written.

7.2 Output of TMR2

The output of TMR2 (before the postscaler) is fed to the SSPort module, which optionally uses it to generate shift clock.

FIGURE 7-1: TIMER2 BLOCK DIAGRAM



REGISTER 7-1: T2CON: TIMER2 CONTROL REGISTER (ADDRESS 12h)

U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	TOUTPS3	TOUTPS2	TOUTPS1	TOUTPS0	TMR2ON	T2CKPS1	T2CKPS0
bit7							bit0

bit 7: **Unimplemented:** Read as '0'

bit 6-3: **TOUTPS3:TOUTPS0:** Timer2 Output Postscale Select bits

0000 = 1:1 Postscale

0001 = 1:2 Postscale

0010 = 1:3 Postscale

•

•

•

1111 = 1:16 Postscale

bit 2: **TMR2ON:** Timer2 On bit

1 = Timer2 is on

0 = Timer2 is off

bit 1-0: **T2CKPS1:T2CKPS0:** Timer2 Clock Prescale Select bits

00 = Prescaler is 1

01 = Prescaler is 4

1x = Prescaler is 16

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

- n = Value at POR reset

TABLE 7-1: REGISTERS ASSOCIATED WITH TIMER2 AS A TIMER/COUNTER

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR, BOR	Value on all other resets
0Bh,8Bh, 10Bh,18Bh	INTCON	GIE	PEIE	TOIE	INTE	RBIE	TOIF	INTF	RBIF	0000 000x	0000 000u
0Ch	PIR1	PSPIF ⁽¹⁾	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	0000 0000	0000 0000
8Ch	PIE1	PSPIE ⁽¹⁾	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	0000 0000	0000 0000
11h	TMR2	Timer2 module's register								0000 0000	0000 0000
12h	T2CON	—	TOUTPS3	TOUTPS2	TOUTPS1	TOUTPS0	TMR2ON	T2CKPS1	T2CKPS0	-000 0000	-000 0000
92h	PR2	Timer2 Period Register								1111 1111	1111 1111

Legend: x = unknown, u = unchanged, - = unimplemented read as '0'. Shaded cells are not used by the Timer2 module.

Note 1: Bits PSPIE and PSPIF are reserved on the PIC16F873/874; always maintain these bits clear.

8.0 CAPTURE/COMPARE/PWM MODULES

Each Capture/Compare/PWM (CCP) module contains a 16-bit register which can operate as a:

- 16-bit Capture register
- 16-bit Compare register
- PWM master/slave Duty Cycle register

Both the CCP1 and CCP2 modules are identical in operation, with the exception being the operation of the special event trigger. Table 8-1 and Table 8-2 show the resources and interactions of the CCP module(s). In the following sections, the operation of a CCP module is described with respect to CCP1. CCP2 operates the same as CCP1, except where noted.

CCP1 Module:

Capture/Compare/PWM Register1 (CCPR1) is comprised of two 8-bit registers: CCPR1L (low byte) and CCPR1H (high byte). The CCP1CON register controls the operation of CCP1. The special event trigger is generated by a compare match and will reset Timer1.

CCP2 Module:

Capture/Compare/PWM Register1 (CCPR2) is comprised of two 8-bit registers: CCPR2L (low byte) and CCPR2H (high byte). The CCP2CON register controls the operation of CCP2. The special event trigger is generated by a compare match and will reset Timer1 and start an A/D conversion (if the A/D module is enabled).

Additional information on CCP modules is available in the PICmicro™ Mid-Range MCU Family Reference Manual (DS33023) and in Application Note 594, "Using the CCP Modules" (DS00594).

TABLE 8-1: CCP MODE - TIMER RESOURCES REQUIRED

CCP Mode	Timer Resource
Capture	Timer1
Compare	Timer1
PWM	Timer2

TABLE 8-2: INTERACTION OF TWO CCP MODULES

CCPx Mode	CCPy Mode	Interaction
Capture	Capture	Same TMR1 time-base.
Capture	Compare	The compare should be configured for the special event trigger, which clears TMR1.
Compare	Compare	The compare(s) should be configured for the special event trigger, which clears TMR1.
PWM	PWM	The PWMs will have the same frequency and update rate (TMR2 interrupt).
PWM	Capture	None.
PWM	Compare	None.

REGISTER 8-1: CCP1CON REGISTER/CCP2CON REGISTER (ADDRESS: 17h/1dh)

U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	CCPxX	CCPxY	CCPxM3	CCPxM2	CCPxM1	CCPxM0
bit7							bit0

R = Readable bit
W = Writable bit
U = Unimplemented bit, read as '0'
- n = Value at POR reset

bit 7-6: **Unimplemented:** Read as '0'

bit 5-4: **CCPxX:CCPxY:** PWM Least Significant bits
Capture Mode: Unused
Compare Mode: Unused
PWM Mode: These bits are the two LSbs of the PWM duty cycle. The eight MSbs are found in CCPRxL.

bit 3-0: **CCPxM3:CCPxM0:** CCPx Mode Select bits
0000 = Capture/Compare/PWM off (resets CCPx module)
0100 = Capture mode, every falling edge
0101 = Capture mode, every rising edge
0110 = Capture mode, every 4th rising edge
0111 = Capture mode, every 16th rising edge
1000 = Compare mode, set output on match (CCPxIF bit is set)
1001 = Compare mode, clear output on match (CCPxIF bit is set)
1010 = Compare mode, generate software interrupt on match (CCPxIF bit is set, CCPx pin is unaffected)
1011 = Compare mode, trigger special event (CCPxIF bit is set, CCPx pin is unaffected); CCP1 resets TMR1; CCP2 resets TMR1 and starts an A/D conversion (if A/D module is enabled)
11xx = PWM mode

8.1 Capture Mode

In Capture mode, CCP1H:CCP1L captures the 16-bit value of the TMR1 register when an event occurs on pin RC2/CCP1. An event is defined as:

- Every falling edge
- Every rising edge
- Every 4th rising edge
- Every 16th rising edge

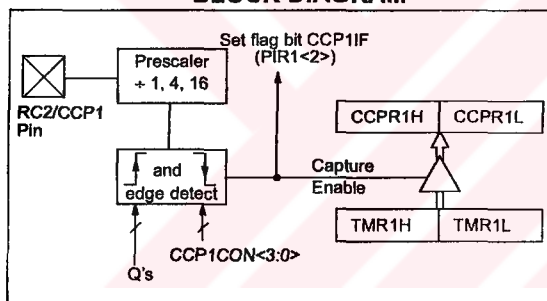
An event is selected by control bits CCP1M3:CCP1M0 (CCP1CON<3:0>). When a capture is made, the interrupt request flag bit CCP1IF (PIR1<2>) is set. The interrupt flag must be cleared in software. If another capture occurs before the value in register CCP1 is read, the old captured value will be lost.

8.1.1 CCP PIN CONFIGURATION

In Capture mode, the RC2/CCP1 pin should be configured as an input by setting the TRISC<2> bit.

Note: If the RC2/CCP1 pin is configured as an output, a write to the port can cause a capture condition.

FIGURE 8-1: CAPTURE MODE OPERATION BLOCK DIAGRAM



8.1.2 TIMER1 MODE SELECTION

Timer1 must be running in timer mode or synchronized counter mode for the CCP module to use the capture feature. In asynchronous counter mode, the capture operation may not work.

8.1.3 SOFTWARE INTERRUPT

When the capture mode is changed, a false capture interrupt may be generated. The user should keep bit CCP1IE (PIE1<2>) clear to avoid false interrupts and should clear the flag bit CCP1IF following any such change in operating mode.

8.1.4 CCP PRESCALER

There are four prescaler settings, specified by bits CCP1M3:CCP1M0. Whenever the CCP module is turned off, or the CCP module is not in capture mode, the prescaler counter is cleared. Any reset will clear the prescaler counter.

Switching from one capture prescaler to another may generate an interrupt. Also, the prescaler counter will not be cleared, therefore, the first capture may be from a non-zero prescaler. Example 8-1 shows the recommended method for switching between capture prescalers. This example also clears the prescaler counter and will not generate the "false" interrupt.

EXAMPLE 8-1: CHANGING BETWEEN CAPTURE PRESCALERS

```

CLRf    CCP1CON    ;Turn CCP module off
MOVLW   NEW_CAPT_PS ;Load the W reg with
                    ; the new prescaler
MOVWF   CCP1CON    ; move value and CCP ON
                    ;Load CCP1CON with this
                    ; value
  
```

PIC16F87X

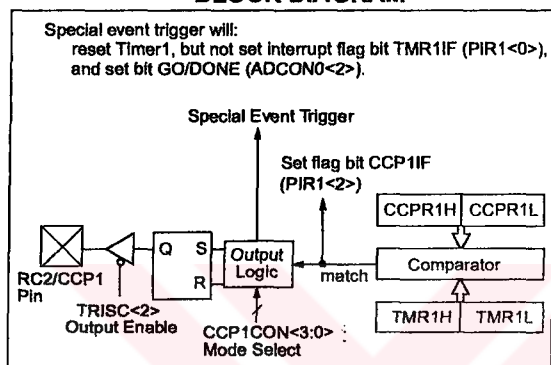
8.2 Compare Mode

In Compare mode, the 16-bit CCPR1 register value is constantly compared against the TMR1 register pair value. When a match occurs, the RC2/CCP1 pin is:

- Driven high
- Driven low
- Remains unchanged

The action on the pin is based on the value of control bits CCP1M3:CCP1M0 (CCP1CON<3:0>). At the same time, interrupt flag bit CCP1IF is set.

FIGURE 8-2: COMPARE MODE OPERATION BLOCK DIAGRAM



8.2.1 CCP PIN CONFIGURATION

The user must configure the RC2/CCP1 pin as an output by clearing the TRISC<2> bit.

Note: Clearing the CCP1CON register will force the RC2/CCP1 compare output latch to the default low level. This is not the data latch.

8.2.2 TIMER1 MODE SELECTION

Timer1 must be running in Timer mode or Synchronized Counter mode if the CCP module is using the compare feature. In Asynchronous Counter mode, the compare operation may not work.

8.2.3 SOFTWARE INTERRUPT MODE

When Generate Software Interrupt mode is chosen, the CCP1 pin is not affected. The CCP1IF bit is set causing a CCP interrupt (if enabled).

8.2.4 SPECIAL EVENT TRIGGER

In this mode, an internal hardware trigger is generated, which may be used to initiate an action.

The special event trigger output of CCP1 resets the TMR1 register pair. This allows the CCPR1 register to effectively be a 16-bit programmable period register for Timer1.

The special event trigger output of CCP2 resets the TMR1 register pair and starts an A/D conversion (if the A/D module is enabled).

Note: The special event trigger from the CCP1 and CCP2 modules will not set interrupt flag bit TMR1IF (PIR1<0>).

8.3 PWM Mode (PWM)

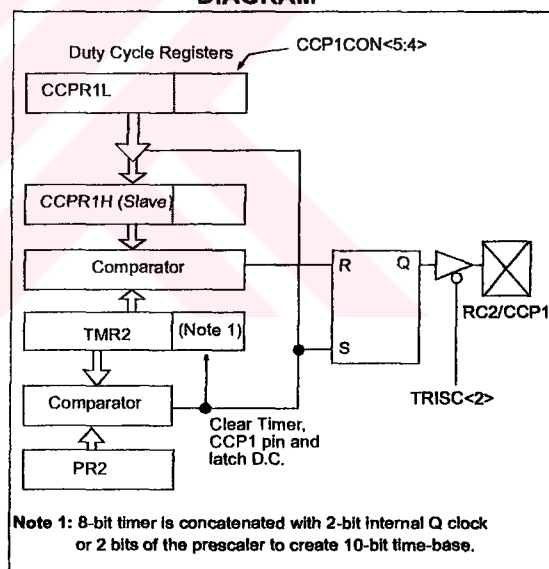
In pulse width modulation mode, the CCPx pin produces up to a 10-bit resolution PWM output. Since the CCP1 pin is multiplexed with the PORTC data latch, the TRISC<2> bit must be cleared to make the CCP1 pin an output.

Note: Clearing the CCP1CON register will force the CCP1 PWM output latch to the default low level. This is not the PORTC I/O data latch.

Figure 8-3 shows a simplified block diagram of the CCP module in PWM mode.

For a step-by-step procedure on how to set up the CCP module for PWM operation, see Section 8.3.3.

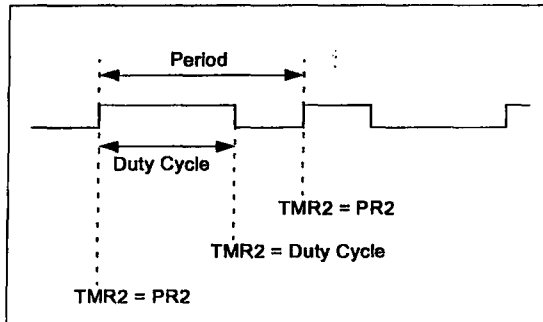
FIGURE 8-3: SIMPLIFIED PWM BLOCK DIAGRAM



Note 1: 8-bit timer is concatenated with 2-bit internal Q clock or 2 bits of the prescaler to create 10-bit time-base.

A PWM output (Figure 8-4) has a time-base (period) and a time that the output stays high (duty cycle). The frequency of the PWM is the inverse of the period (1/period).

FIGURE 8-4: PWM OUTPUT



8.3.1 PWM PERIOD

The PWM period is specified by writing to the PR2 register. The PWM period can be calculated using the following formula:

$$\text{PWM period} = [(PR2) + 1] \cdot 4 \cdot T_{osc} \cdot (\text{TMR2 prescale value})$$

PWM frequency is defined as $1 / [\text{PWM period}]$.

When TMR2 is equal to PR2, the following three events occur on the next increment cycle:

- TMR2 is cleared
- The CCP1 pin is set (exception: if PWM duty cycle = 0%, the CCP1 pin will not be set)
- The PWM duty cycle is latched from CCPR1L into CCPR1H

Note: The Timer2 postscaler (see Section 8.1) is not used in the determination of the PWM frequency. The postscaler could be used to have a servo update rate at a different frequency than the PWM output.

8.3.2 PWM DUTY CYCLE

The PWM duty cycle is specified by writing to the CCPR1L register and to the CCP1CON<5:4> bits. Up to 10-bit resolution is available. The CCPR1L contains the eight MSbs and the CCP1CON<5:4> contains the two LSbs. This 10-bit value is represented by CCPR1L:CCP1CON<5:4>. The following equation is used to calculate the PWM duty cycle in time:

$$\text{PWM duty cycle} = (\text{CCPR1L:CCP1CON<5:4>}) \cdot T_{osc} \cdot (\text{TMR2 prescale value})$$

CCPR1L and CCP1CON<5:4> can be written to at any time, but the duty cycle value is not latched into CCPR1H until after a match between PR2 and TMR2 occurs (i.e., the period is complete). In PWM mode, CCPR1H is a read-only register.

The CCPR1H register and a 2-bit internal latch are used to double buffer the PWM duty cycle. This double buffering is essential for glitchless PWM operation.

When the CCPR1H and 2-bit latch match TMR2 concatenated with an internal 2-bit Q clock or 2 bits of the TMR2 prescaler, the CCP1 pin is cleared.

Maximum PWM resolution (bits) for a given PWM frequency:

$$\text{Resolution} = \frac{\log\left(\frac{F_{osc}}{F_{PWM}}\right)}{\log(2)} \text{ bits}$$

Note: If the PWM duty cycle value is longer than the PWM period, the CCP1 pin will not be cleared.

8.3.3 SET-UP FOR PWM OPERATION

The following steps should be taken when configuring the CCP module for PWM operation:

1. Set the PWM period by writing to the PR2 register.
2. Set the PWM duty cycle by writing to the CCPR1L register and CCP1CON<5:4> bits.
3. Make the CCP1 pin an output by clearing the TRISC<2> bit.
4. Set the TMR2 prescale value and enable Timer2 by writing to T2CON.
5. Configure the CCP1 module for PWM operation.

PIC16F87X

TABLE 8-3: REGISTERS ASSOCIATED WITH CAPTURE, COMPARE, AND TIMER1

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR, BOR	Value on all other resets
0Bh,8Bh, 10Bh,18Bh	INTCON	GIE	PEIE	TOIE	INTE	RBIE	TOIF	INTF	RBIF	0000 000x	0000 000u
0Ch	PIR1	PSPIF ⁽¹⁾	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	0000 0000	0000 0000
0Dh	PIR2	—	—	—	—	—	—	—	CCP2IF	---- --0	---- --0
8Ch	PIE1	PSPIE ⁽¹⁾	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	0000 0000	0000 0000
8Dh	PIE2	—	—	—	—	—	—	—	CCP2IE	---- --0	---- --0
87h	TRISC	PORTC Data Direction Register								1111 1111	1111 1111
0Eh	TMR1L	Holding register for the Least Significant Byte of the 16-bit TMR1 register								xxxx xxxx	uuuu uuuu
0Fh	TMR1H	Holding register for the Most Significant Byte of the 16-bit TMR1 register								xxxx xxxx	uuuu uuuu
10h	T1CON	—	—	T1CKPS1	T1CKPS0	T1OSCEN	T1SYNC	TMR1CS	TMR1ON	--00 0000	--uu uuuu
15h	CCPR1L	Capture/Compare/PWM register1 (LSB)								xxxx xxxx	uuuu uuuu
16h	CCPR1H	Capture/Compare/PWM register1 (MSB)								xxxx xxxx	uuuu uuuu
17h	CCP1CON	—	—	CCP1X	CCP1Y	CCP1M3	CCP1M2	CCP1M1	CCP1M0	--00 0000	--00 0000
1Bh	CCPR2L	Capture/Compare/PWM register2 (LSB)								xxxx xxxx	uuuu uuuu
1Ch	CCPR2H	Capture/Compare/PWM register2 (MSB)								xxxx xxxx	uuuu uuuu
1Dh	CCP2CON	—	—	CCP2X	CCP2Y	CCP2M3	CCP2M2	CCP2M1	CCP2M0	--00 0000	--00 0000

Legend: x = unknown, u = unchanged, - = unimplemented read as '0'. Shaded cells are not used by Capture and Timer1.

Note 1: The PSP is not implemented on the PIC16F873/876; always maintain these bits clear.

TABLE 8-4: REGISTERS ASSOCIATED WITH PWM AND TIMER2

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR, BOR	Value on all other resets
0Bh,8Bh, 10Bh,18Bh	INTCON	GIE	PEIE	TOIE	INTE	RBIE	TOIF	INTF	RBIF	0000 000x	0000 000u
0Ch	PIR1	PSPIF ⁽¹⁾	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	0000 0000	0000 0000
0Dh	PIR2	—	—	—	—	—	—	—	CCP2IF	---- --0	---- --0
8Ch	PIE1	PSPIE ⁽¹⁾	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	0000 0000	0000 0000
8Dh	PIE2	—	—	—	—	—	—	—	CCP2IE	---- --0	---- --0
87h	TRISC	PORTC Data Direction Register								1111 1111	1111 1111
11h	TMR2	Timer2 module's register								0000 0000	0000 0000
92h	PR2	Timer2 module's period register								1111 1111	1111 1111
12h	T2CON	—	TOUTPS3	TOUTPS2	TOUTPS1	TOUTPS0	TMR2ON	T2CKPS1	T2CKPS0	-000 0000	-000 0000
15h	CCPR1L	Capture/Compare/PWM register1 (LSB)								xxxx xxxx	uuuu uuuu
16h	CCPR1H	Capture/Compare/PWM register1 (MSB)								xxxx xxxx	uuuu uuuu
17h	CCP1CON	—	—	CCP1X	CCP1Y	CCP1M3	CCP1M2	CCP1M1	CCP1M0	--00 0000	--00 0000
1Bh	CCPR2L	Capture/Compare/PWM register2 (LSB)								xxxx xxxx	uuuu uuuu
1Ch	CCPR2H	Capture/Compare/PWM register2 (MSB)								xxxx xxxx	uuuu uuuu
1Dh	CCP2CON	—	—	CCP2X	CCP2Y	CCP2M3	CCP2M2	CCP2M1	CCP2M0	--00 0000	--00 0000

Legend: x = unknown, u = unchanged, - = unimplemented read as '0'. Shaded cells are not used by PWM and Timer2.

Note 1: Bits PSPIE and PSPIF are reserved on the PIC16F873/876; always maintain these bits clear.

10.0 ADDRESSABLE UNIVERSAL SYNCHRONOUS ASYNCHRONOUS RECEIVER TRANSMITTER (USART)

The Universal Synchronous Asynchronous Receiver Transmitter (USART) module is one of the two serial I/O modules. (USART is also known as a Serial Communications Interface or SCI). The USART can be configured as a full duplex asynchronous system that can communicate with peripheral devices such as CRT, terminals and personal computers, or it can be configured as a half duplex synchronous system that can communicate with peripheral devices such as A/D or D/A integrated circuits, serial EEPROMs etc.

The USART can be configured in the following modes:

- Asynchronous (full duplex)
- Synchronous - Master (half duplex)
- Synchronous - Slave (half duplex)

Bit SPEN (RCSTA<7>) and bits TRISC<7:6> have to be set in order to configure pins RC6/TX/CK and RC7/RX/DT as the Universal Synchronous Asynchronous Receiver Transmitter.

The USART module also has a multi-processor communication capability using 9-bit address detection.

REGISTER 10-1: TXSTA: TRANSMIT STATUS AND CONTROL REGISTER (ADDRESS 98h)

R/W-0	R/W-0	R/W-0	R/W-0	U-0	R/W-0	R-1	R/W-0
CSRC	TX9	TXEN	SYNC	—	BRGH	TRMT	TX9D
bit7							bit0

R = Readable bit
W = Writable bit
U = Unimplemented bit, read as '0'
- n = Value at POR reset

bit 7: **CSRC:** Clock Source Select bit
Asynchronous mode
Don't care
Synchronous mode
1 = Master mode (Clock generated internally from BRG)
0 = Slave mode (Clock from external source)

bit 6: **TX9:** 9-bit Transmit Enable bit
1 = Selects 9-bit transmission
0 = Selects 8-bit transmission

bit 5: **TXEN:** Transmit Enable bit
1 = Transmit enabled
0 = Transmit disabled
Note: SREN/CREN overrides TXEN in SYNC mode.

bit 4: **SYNC:** USART Mode Select bit
1 = Synchronous mode
0 = Asynchronous mode

bit 3: **Unimplemented:** Read as '0'

bit 2: **BRGH:** High Baud Rate Select bit
Asynchronous mode
1 = High speed
0 = Low speed
Synchronous mode
Unused in this mode

bit 1: **TRMT:** Transmit Shift Register Status bit
1 = TSR empty
0 = TSR full

bit 0: **TX9D:** 9th bit of transmit data. Can be parity bit.

REGISTER 10-2: RCSTA: RECEIVE STATUS AND CONTROL REGISTER (ADDRESS 18h)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R-0	R-0	R-x
SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D
bit7							bit0

R = Readable bit
W = Writable bit
U = Unimplemented bit, read as '0'
- n = Value at POR reset

bit 7: **SPEN:** Serial Port Enable bit
1 = Serial port enabled (Configures RC7/RX/DT and RC6/TX/CK pins as serial port pins)
0 = Serial port disabled

bit 6: **RX9:** 9-bit Receive Enable bit
1 = Selects 9-bit reception
0 = Selects 8-bit reception

bit 5: **SREN:** Single Receive Enable bit
Asynchronous mode
Don't care
Synchronous mode - master
1 = Enables single receive
0 = Disables single receive
This bit is cleared after reception is complete.
Synchronous mode - slave
Unused in this mode

bit 4: **CREN:** Continuous Receive Enable bit
Asynchronous mode
1 = Enables continuous receive
0 = Disables continuous receive
Synchronous mode
1 = Enables continuous receive until enable bit CREN is cleared (CREN overrides SREN)
0 = Disables continuous receive

bit 3: **ADDEN:** Address Detect Enable bit
Asynchronous mode 9-bit (RX9 = 1)
1 = Enables address detection, enable interrupt and load of the receive burffer when RSR<8> is set
0 = Disables address detection, all bytes are received, and ninth bit can be used as parity bit

bit 2: **FERR:** Framing Error bit
1 = Framing error (Can be updated by reading RCREG register and receive next valid byte)
0 = No framing error

bit 1: **OERR:** Overrun Error bit
1 = Overrun error (Can be cleared by clearing bit CREN)
0 = No overrun error

bit 0: **RX9D:** 9th bit of received data (Can be parity bit)

10.1 USART Baud Rate Generator (BRG)

The BRG supports both the asynchronous and synchronous modes of the USART. It is a dedicated 8-bit baud rate generator. The SPBRG register controls the period of a free running 8-bit timer. In asynchronous mode, bit BRGH (TXSTA<2>) also controls the baud rate. In synchronous mode, bit BRGH is ignored. Table 10-1 shows the formula for computation of the baud rate for different USART modes which only apply in master mode (internal clock).

Given the desired baud rate and Fosc, the nearest integer value for the SPBRG register can be calculated using the formula in Table 10-1. From this, the error in baud rate can be determined.

It may be advantageous to use the high baud rate (BRGH = 1) even for slower baud clocks. This is because the $F_{osc}/(16(X + 1))$ equation can reduce the baud rate error in some cases.

Writing a new value to the SPBRG register causes the BRG timer to be reset (or cleared). This ensures the BRG does not wait for a timer overflow before outputting the new baud rate.

10.1.1 SAMPLING

The data on the RC7/RX/DT pin is sampled three times by a majority detect circuit to determine if a high or a low level is present at the RX pin.

TABLE 10-1: BAUD RATE FORMULA

SYNC	BRGH = 0 (Low Speed)	BRGH = 1 (High Speed)
0	(Asynchronous) Baud Rate = $F_{osc}/(64(X+1))$	Baud Rate = $F_{osc}/(16(X+1))$
1	(Synchronous) Baud Rate = $F_{osc}/(4(X+1))$	NA

X = value in SPBRG (0 to 255)

TABLE 10-2: REGISTERS ASSOCIATED WITH BAUD RATE GENERATOR

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR, BOR	Value on all other resets
98h	TXSTA	CSRC	TX9	TXEN	SYNC	—	BRGH	TRMT	TX9D	0000 -010	0000 -010
18h	RCSTA	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D	0000 000x	0000 000x
99h	SPBRG	Baud Rate Generator Register								0000 0000	0000 0000

Legend: x = unknown, - = unimplemented read as '0'. Shaded cells are not used by the BRG.

TABLE 10-3: BAUD RATES FOR ASYNCHRONOUS MODE (BRGH = 0)

BAUD RATE (K)	Fosc = 20 MHz			Fosc = 16 MHz			Fosc = 10 MHz		
	KBAUD	% ERROR	SPBRG value (decimal)	KBAUD	% ERROR	SPBRG value (decimal)	KBAUD	% ERROR	SPBRG value (decimal)
0.3	-	-	-	-	-	-	-	-	-
1.2	1.221	1.75	255	1.202	0.17	207	1.202	0.17	129
2.4	2.404	0.17	129	2.404	0.17	103	2.404	0.17	64
9.6	9.766	1.73	31	9.615	0.16	25	9.766	1.73	15
19.2	19.531	1.72	15	19.231	0.16	12	19.531	1.72	7
28.8	31.250	8.51	9	27.778	3.55	8	31.250	8.51	4
33.6	34.722	3.34	8	35.714	6.29	6	31.250	6.99	4
57.6	62.500	8.51	4	62.500	8.51	3	52.083	9.58	2
HIGH	1.221	-	255	0.977	-	255	0.610	-	255
LOW	312.500	-	0	250.000	-	0	156.250	-	0

BAUD RATE (K)	Fosc = 4 MHz			Fosc = 3.6864 MHz		
	KBAUD	% ERROR	SPBRG value (decimal)	KBAUD	% ERROR	SPBRG value (decimal)
0.3	0.300	0	207	0.301	0.33	185
1.2	1.202	0.17	51	1.216	1.33	46
2.4	2.404	0.17	25	2.432	1.33	22
9.6	8.929	6.99	6	9.322	2.90	5
19.2	20.833	8.51	2	18.643	2.90	2
28.8	31.250	8.51	1	-	-	-
33.6	-	-	-	-	-	-
57.6	62.500	8.51	0	55.930	2.90	0
HIGH	0.244	-	255	0.218	-	255
LOW	62.500	-	0	55.930	-	0

TABLE 10-4: BAUD RATES FOR ASYNCHRONOUS MODE (BRGH = 1)

BAUD RATE (K)	Fosc = 20 MHz			Fosc = 16 MHz			Fosc = 10 MHz		
	KBAUD	% ERROR	SPBRG value (decimal)	KBAUD	% ERROR	SPBRG value (decimal)	KBAUD	% ERROR	SPBRG value (decimal)
0.3	-	-	-	-	-	-	-	-	-
1.2	-	-	-	-	-	-	-	-	-
2.4	-	-	-	-	-	-	2.441	1.71	255
9.6	9.615	0.16	129	9.615	0.16	103	9.615	0.16	64
19.2	19.231	0.16	64	19.231	0.16	51	19.531	1.72	31
28.8	29.070	0.94	42	29.412	2.13	33	28.409	1.36	21
33.6	33.784	0.55	36	33.333	0.79	29	32.895	2.10	18
57.6	59.524	3.34	20	58.824	2.13	16	56.818	1.36	10
HIGH	4.883	-	255	3.906	-	255	2.441	-	255
LOW	1250.000	-	0	1000.000	-	0	625.000	-	0

BAUD RATE (K)	Fosc = 4 MHz			Fosc = 3.6864 MHz		
	KBAUD	% ERROR	SPBRG value (decimal)	KBAUD	% ERROR	SPBRG value (decimal)
0.3	-	-	-	-	-	-
1.2	1.202	0.17	207	1.203	0.25	185
2.4	2.404	0.17	103	2.406	0.25	92
9.6	9.615	0.16	25	9.727	1.32	22
19.2	19.231	0.16	12	18.643	2.90	11
28.8	27.798	3.55	8	27.965	2.90	7
33.6	35.714	6.29	6	31.960	4.88	6
57.6	62.500	8.51	3	55.930	2.90	3
HIGH	0.977	-	255	0.874	-	255
LOW	250.000	-	0	273.722	-	0

10.2 USART Asynchronous Mode

In this mode, the USART uses standard non-return-to-zero (NRZ) format (one start bit, eight or nine data bits, and one stop bit). The most common data format is 8 bits. An on-chip, dedicated, 8-bit baud rate generator can be used to derive standard baud rate frequencies from the oscillator. The USART transmits and receives the LSb first. The USART's transmitter and receiver are functionally independent, but use the same data format and baud rate. The baud rate generator produces a clock either x16 or x64 of the bit shift rate, depending on bit BRGH (TXSTA<2>). Parity is not supported by the hardware, but can be implemented in software (and stored as the ninth data bit). Asynchronous mode is stopped during SLEEP.

Asynchronous mode is selected by clearing bit SYNC (TXSTA<4>).

The USART Asynchronous module consists of the following important elements:

- Baud Rate Generator
- Sampling Circuit
- Asynchronous Transmitter
- Asynchronous Receiver

10.2.1 USART ASYNCHRONOUS TRANSMITTER

The USART transmitter block diagram is shown in Figure 10-1. The heart of the transmitter is the transmit (serial) shift register (TSR). The shift register obtains its data from the read/write transmit buffer, TXREG. The TXREG register is loaded with data in software. The TSR register is not loaded until the STOP bit has been transmitted from the previous load. As soon as the STOP bit is transmitted, the TSR is loaded with new data from the TXREG register (if available). Once the TXREG register transfers the data to the TSR register (occurs in one Tcy), the TXREG register is empty and flag bit TXIF (PIR1<4>) is set. This interrupt can be enabled/disabled by setting/clearing enable bit TXIE

(PIE1<4>). Flag bit TXIF will be set, regardless of the state of enable bit TXIE and cannot be cleared in software. It will reset only when new data is loaded into the TXREG register. While flag bit TXIF indicates the status of the TXREG register, another bit TRMT (TXSTA<1>) shows the status of the TSR register. Status bit TRMT is a read only bit, which is set when the TSR register is empty. No interrupt logic is tied to this bit, so the user has to poll this bit in order to determine if the TSR register is empty.

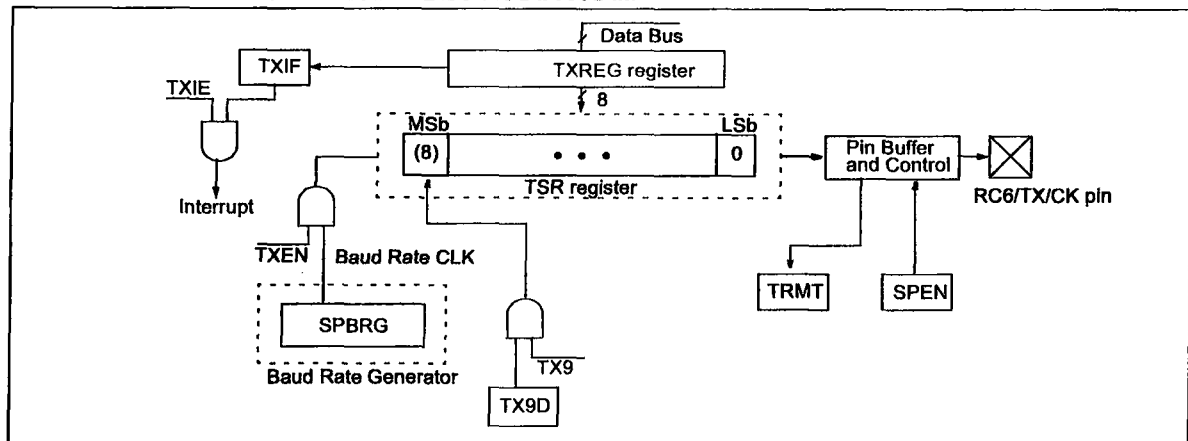
Note 1: The TSR register is not mapped in data memory, so it is not available to the user.

2: Flag bit TXIF is set when enable bit TXEN is set. TXIF is cleared by loading TXREG.

Transmission is enabled by setting enable bit TXEN (TXSTA<5>). The actual transmission will not occur until the TXREG register has been loaded with data and the baud rate generator (BRG) has produced a shift clock (Figure 10-2). The transmission can also be started by first loading the TXREG register and then setting enable bit TXEN. Normally, when transmission is first started, the TSR register is empty. At that point, transfer to the TXREG register will result in an immediate transfer to TSR, resulting in an empty TXREG. A back-to-back transfer is thus possible (Figure 10-3). Clearing enable bit TXEN during a transmission will cause the transmission to be aborted and will reset the transmitter. As a result, the RC6/TX/CK pin will revert to hi-impedance.

In order to select 9-bit transmission, transmit bit TX9 (TXSTA<6>) should be set and the ninth bit should be written to TX9D (TXSTA<0>). The ninth bit must be written before writing the 8-bit data to the TXREG register. This is because a data write to the TXREG register can result in an immediate transfer of the data to the TSR register (if the TSR is empty). In such a case, an incorrect ninth data bit may be loaded in the TSR register.

FIGURE 10-1: USART TRANSMIT BLOCK DIAGRAM



Steps to follow when setting up an Asynchronous Transmission:

1. Initialize the SPBRG register for the appropriate baud rate. If a high speed baud rate is desired, set bit BRGH. (Section 10.1)
2. Enable the asynchronous serial port by clearing bit SYNC and setting bit SPEN.
3. If interrupts are desired, then set enable bit TXIE.
4. If 9-bit transmission is desired, then set transmit bit TX9.
5. Enable the transmission by setting bit TXEN, which will also set bit TXIF.
6. If 9-bit transmission is selected, the ninth bit should be loaded in bit TX9D.
7. Load data to the TXREG register (starts transmission).

FIGURE 10-2: ASYNCHRONOUS MASTER TRANSMISSION

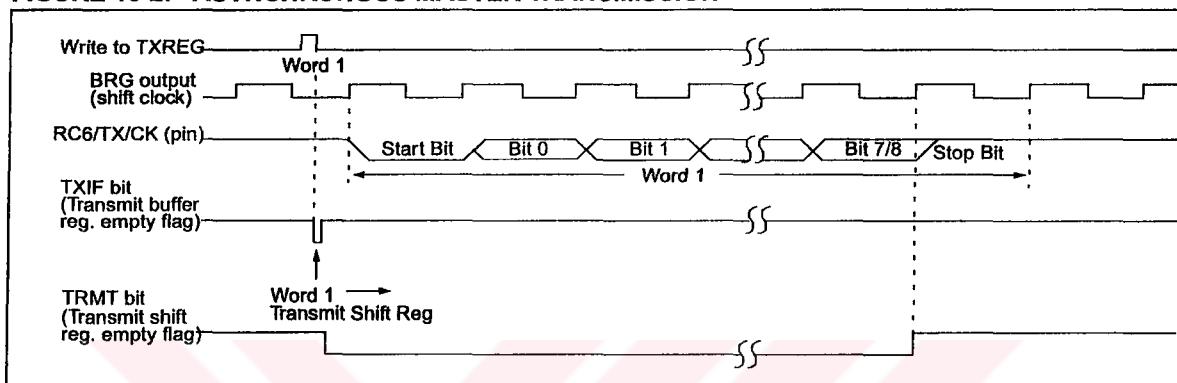


FIGURE 10-3: ASYNCHRONOUS MASTER TRANSMISSION (BACK TO BACK)

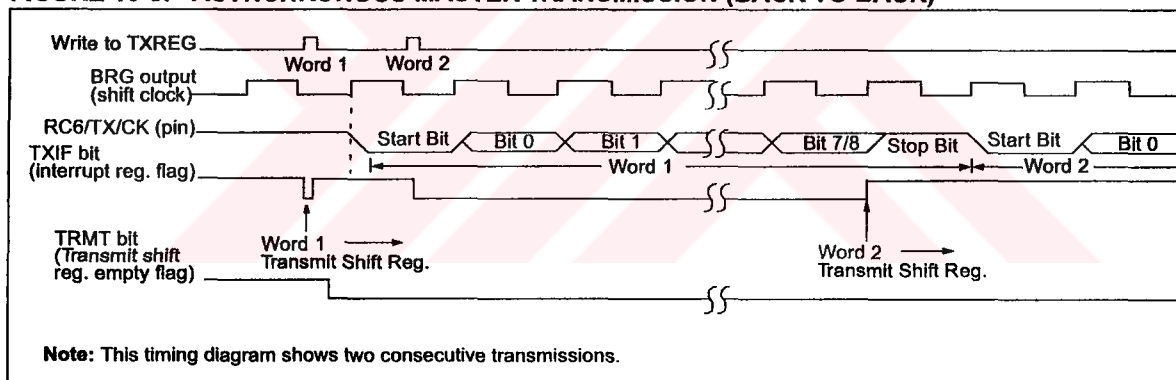


TABLE 10-5: REGISTERS ASSOCIATED WITH ASYNCHRONOUS TRANSMISSION

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR, BOR	Value on all other Resets
0Ch	PIR1	PSPIF ⁽¹⁾	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	0000 0000	0000 0000
18h	RCSTA	SPEN	RX9	SREN	CREN	—	FERR	OERR	RX9D	0000 -00x	0000 -00x
19h	TXREG	USART Transmit Register								0000 0000	0000 0000
8Ch	PIE1	PSPIE ⁽¹⁾	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	0000 0000	0000 0000
98h	TXSTA	CSRC	TX9	TXEN	SYNC	—	BRGH	TRMT	TX9D	0000 -010	0000 -010
99h	SPBRG	Baud Rate Generator Register								0000 0000	0000 0000

Legend: x = unknown, - = unimplemented locations read as '0'. Shaded cells are not used for asynchronous transmission.

Note 1: Bits PSPIE and PSPIF are reserved on the PIC16F873/876; always maintain these bits clear.

10.2.2 USART ASYNCHRONOUS RECEIVER

The receiver block diagram is shown in Figure 10-4. The data is received on the RC7/RX/DT pin and drives the data recovery block. The data recovery block is actually a high speed shifter operating at x16 times the baud rate, whereas the main receive serial shifter operates at the bit rate or at Fosc.

Once asynchronous mode is selected, reception is enabled by setting bit CREN (RCSTA<4>).

The heart of the receiver is the receive (serial) shift register (RSR). After sampling the STOP bit, the received data in the RSR is transferred to the RCREG register (if it is empty). If the transfer is complete, flag bit RCIF (PIR1<5>) is set. The actual interrupt can be enabled/disabled by setting/clearing enable bit RCIE (PIE1<5>). Flag bit RCIF is a read only bit which is cleared by the hardware. It is cleared when the RCREG register has been read and is empty. The RCREG is a double buffered register (i.e. it is a two deep FIFO). It is possible

for two bytes of data to be received and transferred to the RCREG FIFO and a third byte to begin shifting to the RSR register. On the detection of the STOP bit of the third byte, if the RCREG register is still full, the overrun error bit OERR (RCSTA<1>) will be set. The word in the RSR will be lost. The RCREG register can be read twice to retrieve the two bytes in the FIFO. Overrun bit OERR has to be cleared in software. This is done by resetting the receive logic (CREN is cleared and then set). If bit OERR is set, transfers from the RSR register to the RCREG register are inhibited, so it is essential to clear error bit OERR if it is set. Framing error bit FERR (RCSTA<2>) is set if a stop bit is detected as clear. Bit FERR and the 9th receive bit are buffered the same way as the receive data. Reading the RCREG will load bits RX9D and FERR with new values, therefore it is essential for the user to read the RCSTA register before reading RCREG register in order not to lose the old FERR and RX9D information.

FIGURE 10-4: USART RECEVE BLOCK DIAGRAM

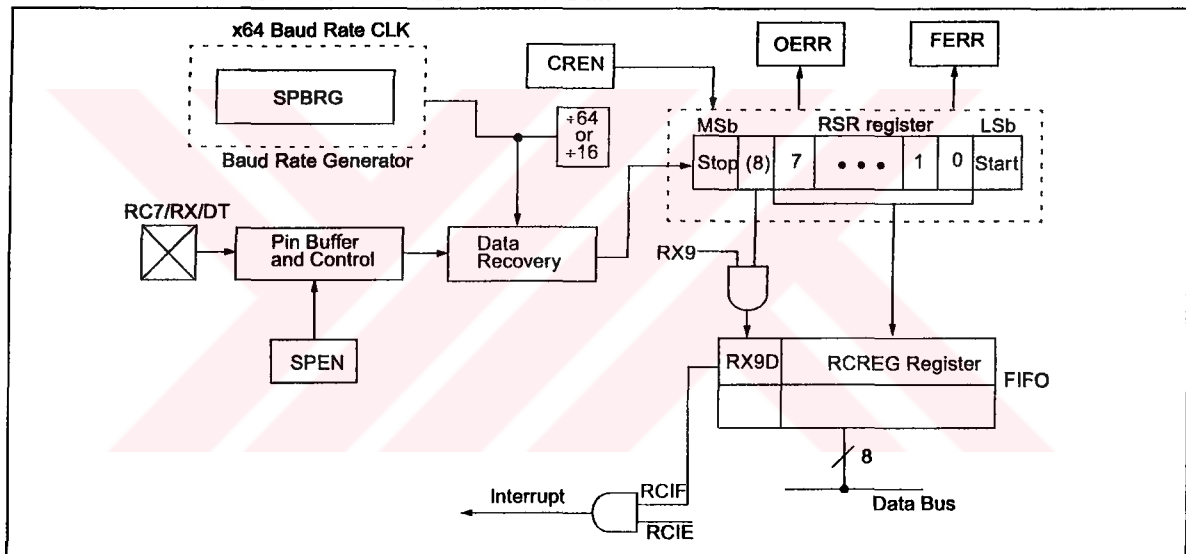
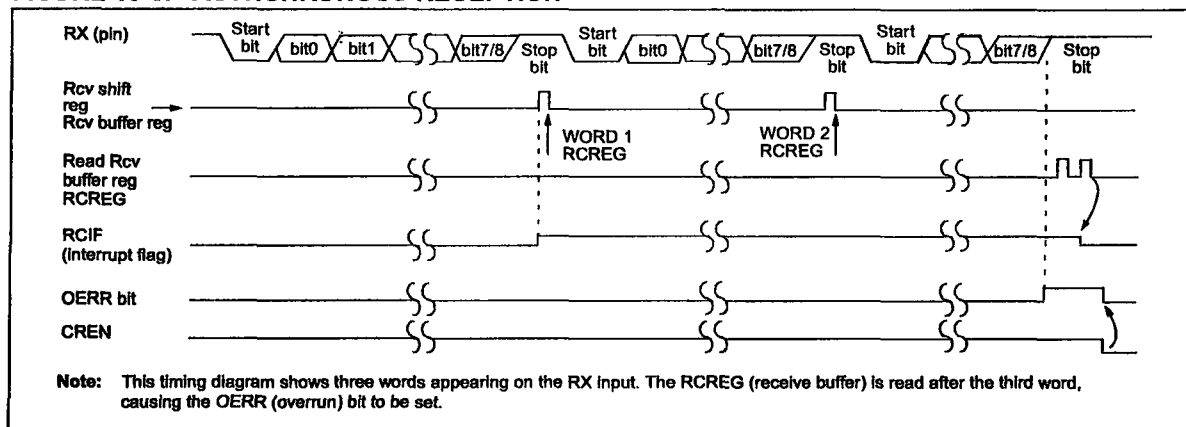


FIGURE 10-5: ASYNCHRONOUS RECEPTION



Steps to follow when setting up an Asynchronous Reception:

1. Initialize the SPBRG register for the appropriate baud rate. If a high speed baud rate is desired, set bit BRGH. (Section 10.1).
2. Enable the asynchronous serial port by clearing bit SYNC and setting bit SPEN.
3. If interrupts are desired, then set enable bit RCIE.
4. If 9-bit reception is desired, then set bit RX9.
5. Enable the reception by setting bit CREN.
6. Flag bit RCIF will be set when reception is complete and an interrupt will be generated if enable bit RCIE is set.
7. Read the RCSTA register to get the ninth bit (if enabled) and determine if any error occurred during reception.
8. Read the 8-bit received data by reading the RCREG register.
9. If any error occurred, clear the error by clearing enable bit CREN.

TABLE 10-6: REGISTERS ASSOCIATED WITH ASYNCHRONOUS RECEPTION

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR, BOR	Value on all other Resets
0Ch	PIR1	PSPIF ⁽¹⁾	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	0000 0000	0000 0000
18h	RCSTA	SPEN	RX9	SREN	CREN	—	FERR	OERR	RX9D	0000 -00x	0000 -00x
1Ah	RCREG	USART Receive Register								0000 0000	0000 0000
8Ch	PIE1	PSPIE ⁽¹⁾	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	0000 0000	0000 0000
98h	TXSTA	CSRC	TX9	TXEN	SYNC	—	BRGH	TRMT	TX9D	0000 -010	0000 -010
99h	SPBRG	Baud Rate Generator Register								0000 0000	0000 0000

Legend: x = unknown, - = unimplemented locations read as '0'. Shaded cells are not used for asynchronous reception.

Note 1: Bits PSPIE and PSPIF are reserved on the 28-pin devices; always maintain these bits clear.

10.2.3 SETTING UP 9-BIT MODE WITH ADDRESS DETECT

Steps to follow when setting up an Asynchronous Reception with Address Detect Enabled:

- Initialize the SPBRG register for the appropriate baud rate. If a high speed baud rate is desired, set bit BRGH.
- Enable the asynchronous serial port by clearing bit SYNC and setting bit SPEN.
- If interrupts are desired, then set enable bit RCIE.
- Set bit RX9 to enable 9-bit reception.
- Set ADDEN to enable address detect.
- Enable the reception by setting enable bit CREN.
- Flag bit RCIF will be set when reception is complete, and an interrupt will be generated if enable bit RCIE was set.
- Read the RCSTA register to get the ninth bit and determine if any error occurred during reception.
- Read the 8-bit received data by reading the RCREG register, to determine if the device is being addressed.
- If any error occurred, clear the error by clearing enable bit CREN.
- If the device has been addressed, clear the ADDEN bit to allow data bytes and address bytes to be read into the receive buffer, and interrupt the CPU.

FIGURE 10-6: USART RECEIVE BLOCK DIAGRAM

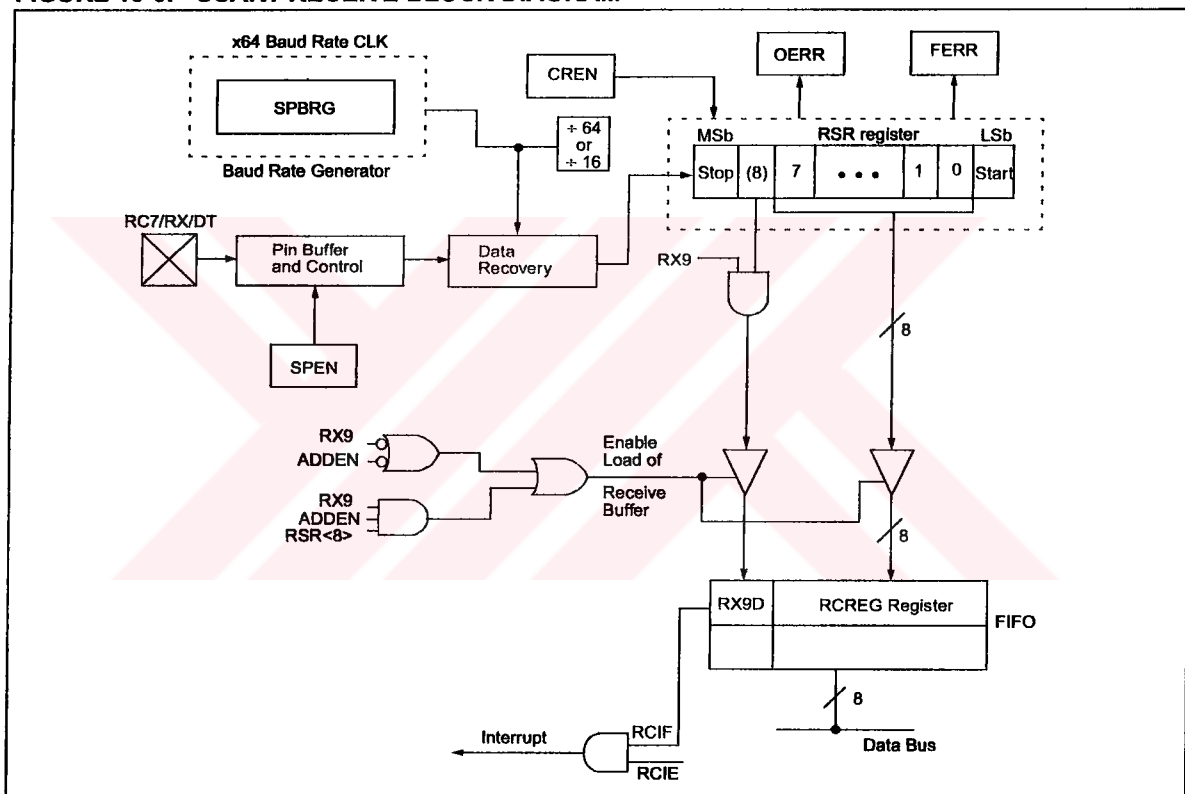
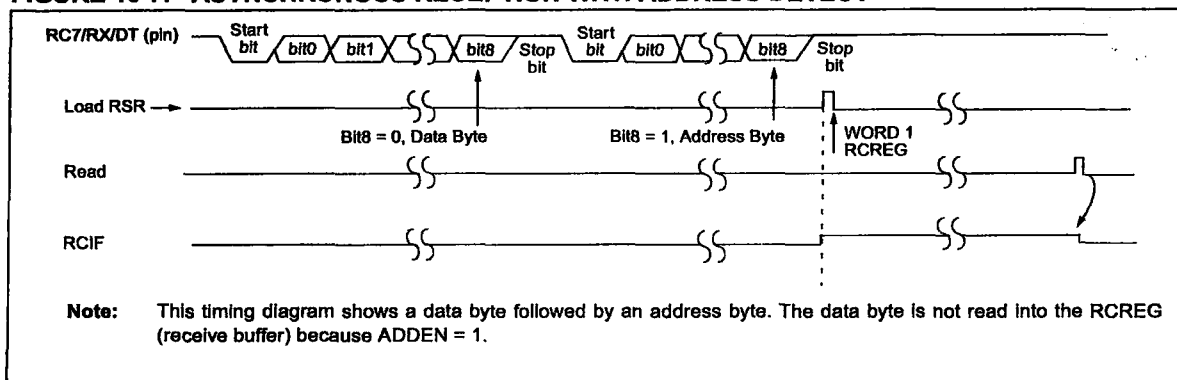
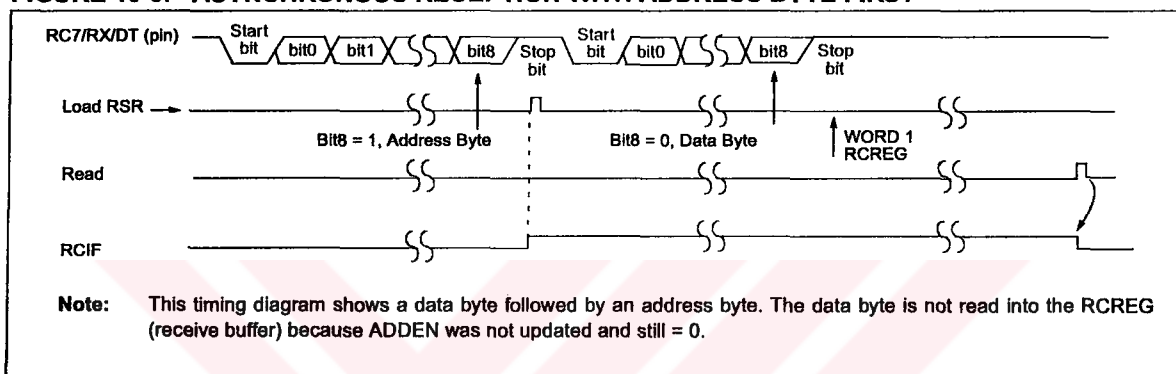


FIGURE 10-7: ASYNCHRONOUS RECEPTION WITH ADDRESS DETECT

FIGURE 10-8: ASYNCHRONOUS RECEPTION WITH ADDRESS BYTE FIRST

TABLE 10-7: REGISTERS ASSOCIATED WITH ASYNCHRONOUS RECEPTION

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR, BOR	Value on all other Resets
0Ch	PIR1	PSPIF ⁽¹⁾	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	0000 0000	0000 0000
18h	RCSTA	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D	0000 000x	0000 000x
1Ah	RCREG	USART Receive Register								0000 0000	0000 0000
8Ch	PIE1	PSPIE ⁽¹⁾	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	0000 0000	0000 0000
98h	TXSTA	CSRC	TX9	TXEN	SYNC	—	BRGH	TRMT	TX9D	0000 -010	0000 -010
99h	SPBRG	Baud Rate Generator Register								0000 0000	0000 0000

Legend: x = unknown, - = unimplemented locations read as '0'. Shaded cells are not used for Asynchronous Reception.

Note 1: Bits PSPIE and PSPIF are reserved on the 28-pin devices; always maintain these bits clear.

10.3 USART Synchronous Master Mode

In Synchronous Master mode, the data is transmitted in a half-duplex manner (i.e., transmission and reception do not occur at the same time). When transmitting data, the reception is inhibited and vice versa. Synchronous mode is entered by setting bit SYNC (TXSTA<4>). In addition, enable bit SPEN (RCSTA<7>) is set in order to configure the RC6/TX/CK and RC7/RX/DT I/O pins to CK (clock) and DT (data) lines respectively. The Master mode indicates that the processor transmits the master clock on the CK line. The Master mode is entered by setting bit CSRC (TXSTA<7>).

10.3.1 USART SYNCHRONOUS MASTER TRANSMISSION

The USART transmitter block diagram is shown in Figure 10-6. The heart of the transmitter is the transmit (serial) shift register (TSR). The shift register obtains its data from the read/write transmit buffer register TXREG. The TXREG register is loaded with data in software. The TSR register is not loaded until the last bit has been transmitted from the previous load. As soon as the last bit is transmitted, the TSR is loaded with new data from the TXREG (if available). Once the TXREG register transfers the data to the TSR register (occurs in one Tcycle), the TXREG is empty and interrupt bit TXIF (PIR1<4>) is set. The interrupt can be enabled/disabled by setting/clearing enable bit TXIE (PIE1<4>). Flag bit TXIF will be set regardless of the state of enable bit TXIE and cannot be cleared in software. It will reset only when new data is loaded into the TXREG register. While flag bit TXIF indicates the status of the TXREG register, another bit TRMT (TXSTA<1>) shows the status of the TSR register. TRMT is a read only bit which is set when the TSR is empty. No interrupt logic is tied to this bit, so the user has to poll this bit in order to determine if the TSR register is empty. The TSR is not mapped in data memory, so it is not available to the user.

Transmission is enabled by setting enable bit TXEN (TXSTA<5>). The actual transmission will not occur until the TXREG register has been loaded with data. The first data bit will be shifted out on the next available rising edge of the clock on the CK line. Data out is stable around the falling edge of the synchronous clock (Figure 10-9). The transmission can also be started by first loading the TXREG register and then setting bit TXEN (Figure 10-10). This is advantageous when slow baud rates are selected, since the BRG is kept in reset when bits TXEN, CREN and SREN are clear. Setting enable bit TXEN will start the BRG, creating a shift clock immediately. Normally, when transmission is first started, the TSR register is empty, so a transfer to the TXREG register will result in an immediate transfer to TSR resulting in an empty TXREG. Back-to-back transfers are possible.

Clearing enable bit TXEN during a transmission will cause the transmission to be aborted and will reset the transmitter. The DT and CK pins will revert to hi-impedance. If either bit CREN or bit SREN is set during a transmission, the transmission is aborted and the DT

pin reverts to a hi-impedance state (for a reception). The CK pin will remain an output if bit CSRC is set (internal clock). The transmitter logic, however, is not reset, although it is disconnected from the pins. In order to reset the transmitter, the user has to clear bit TXEN. If bit SREN is set (to interrupt an on-going transmission and receive a single word), then after the single word is received, bit SREN will be cleared and the serial port will revert back to transmitting, since bit TXEN is still set. The DT line will immediately switch from hi-impedance receive mode to transmit and start driving. To avoid this, bit TXEN should be cleared.

In order to select 9-bit transmission, the TX9 (TXSTA<6>) bit should be set and the ninth bit should be written to bit TX9D (TXSTA<0>). The ninth bit must be written before writing the 8-bit data to the TXREG register. This is because a data write to the TXREG can result in an immediate transfer of the data to the TSR register (if the TSR is empty). If the TSR was empty and the TXREG was written before writing the "new" TX9D, the "present" value of bit TX9D is loaded.

Steps to follow when setting up a Synchronous Master Transmission:

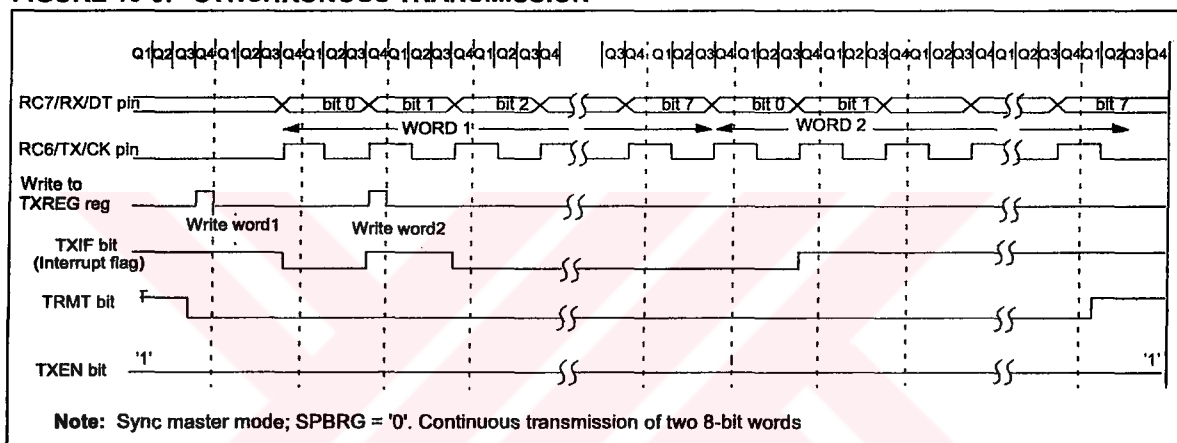
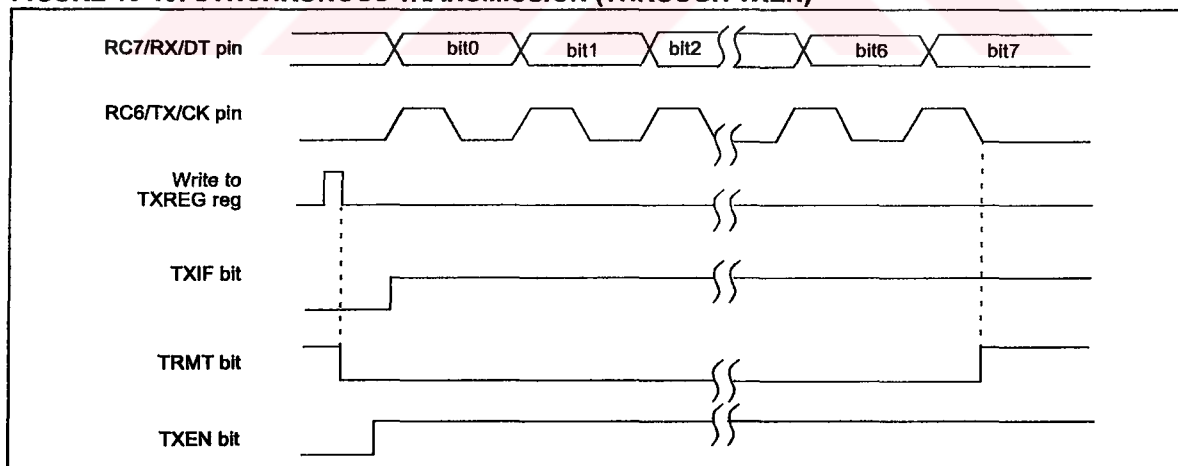
1. Initialize the SPBRG register for the appropriate baud rate (Section 10.1).
2. Enable the synchronous master serial port by setting bits SYNC, SPEN and CSRC.
3. If interrupts are desired, set enable bit TXIE.
4. If 9-bit transmission is desired, set bit TX9.
5. Enable the transmission by setting bit TXEN.
6. If 9-bit transmission is selected, the ninth bit should be loaded in bit TX9D.
7. Start transmission by loading data to the TXREG register.

TABLE 10-8: REGISTERS ASSOCIATED WITH SYNCHRONOUS MASTER TRANSMISSION

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR, BOR	Value on all other Resets
0Ch	PIR1	PSPIF ⁽¹⁾	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	0000 0000	0000 0000
18h	RCSTA	SPEN	RX9	SREN	CREN	—	FERR	OERR	RX9D	0000 -00x	0000 -00x
19h	TXREG	USART Transmit Register								0000 0000	0000 0000
8Ch	PIE1	PSPIE ⁽¹⁾	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	0000 0000	0000 0000
98h	TXSTA	CSRC	TX9	TXEN	SYNC	—	BRGH	TRMT	TX9D	0000 -010	0000 -010
99h	SPBRG	Baud Rate Generator Register								0000 0000	0000 0000

Legend: x = unknown, - = unimplemented, read as '0'. Shaded cells are not used for synchronous master transmission.

Note 1: Bits PSPIE and PSPIF are reserved on the 28-pin devices; always maintain these bits clear.

FIGURE 10-9: SYNCHRONOUS TRANSMISSION

FIGURE 10-10: SYNCHRONOUS TRANSMISSION (THROUGH TXEN)


1. Initialize the SPBRG register for the appropriate baud rate. (Section 10.1)
2. Enable the synchronous master serial port by setting bits SYNC, SPEN and CSRC.
3. Ensure bits CREN and SREN are clear.
4. If interrupts are desired, then set enable bit RCIE.
5. If 9-bit reception is desired, then set bit RX9.
6. If a single reception is required, set bit SREN. For continuous reception set bit CREN.
7. Interrupt flag bit RCIF will be set when reception is complete and an interrupt will be generated if enable bit RCIE was set.
8. Read the RCSTA register to get the ninth bit (if enabled) and determine if any error occurred during reception.
9. Read the 8-bit received data by reading the RCREG register.
10. If any error occurred, clear the error by clearing bit CREN.

PIC16F87X

10.4 USART Synchronous Slave Mode

Synchronous slave mode differs from the Master mode in the fact that the shift clock is supplied externally at the RC6/TX/CK pin (instead of being supplied internally in master mode). This allows the device to transfer or receive data while in SLEEP mode. Slave mode is entered by clearing bit CSRC (TXSTA<7>).

10.4.1 USART SYNCHRONOUS SLAVE TRANSMIT

The operation of the synchronous master and slave modes are identical except in the case of the SLEEP mode.

If two words are written to the TXREG and then the SLEEP instruction is executed, the following will occur:

- a) The first word will immediately transfer to the TSR register and transmit.
- b) The second word will remain in TXREG register.
- c) Flag bit TXIF will not be set.
- d) When the first word has been shifted out of TSR, the TXREG register will transfer the second word to the TSR and flag bit TXIF will now be set.
- e) If enable bit TXIE is set, the interrupt will wake the chip from SLEEP and if the global interrupt is enabled, the program will branch to the interrupt vector (0004h).

Steps to follow when setting up a Synchronous Slave Transmission:

1. Enable the synchronous slave serial port by setting bits SYNC and SPEN and clearing bit CSRC.
2. Clear bits CREN and SREN.
3. If interrupts are desired, then set enable bit TXIE.
4. If 9-bit transmission is desired, then set bit TX9.
5. Enable the transmission by setting enable bit TXEN.
6. If 9-bit transmission is selected, the ninth bit should be loaded in bit TX9D.
7. Start transmission by loading data to the TXREG register.

10.4.2 USART SYNCHRONOUS SLAVE RECEPTION

The operation of the synchronous master and slave modes is identical, except in the case of the SLEEP mode. Bit SREN is a "don't care" in slave mode.

If receive is enabled by setting bit CREN prior to the SLEEP instruction, then a word may be received during SLEEP. On completely receiving the word, the RSR register will transfer the data to the RCREG register and if enable bit RCIE bit is set, the interrupt generated will wake the chip from SLEEP. If the global interrupt is enabled, the program will branch to the interrupt vector (0004h).

Steps to follow when setting up a Synchronous Slave Reception:

1. Enable the synchronous master serial port by setting bits SYNC and SPEN and clearing bit CSRC.
2. If interrupts are desired, set enable bit RCIE.
3. If 9-bit reception is desired, set bit RX9.
4. To enable reception, set enable bit CREN.
5. Flag bit RCIF will be set when reception is complete and an interrupt will be generated, if enable bit RCIE was set.
6. Read the RCSTA register to get the ninth bit (if enabled) and determine if any error occurred during reception.
7. Read the 8-bit received data by reading the RCREG register.
8. If any error occurred, clear the error by clearing bit CREN.

TABLE 10-10: REGISTERS ASSOCIATED WITH SYNCHRONOUS SLAVE TRANSMISSION

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR, BOR	Value on all other Resets
0Ch	PIR1	PSPIF ⁽¹⁾	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	0000 0000	0000 0000
18h	RCSTA	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D	0000 000x	0000 000x
19h	TXREG	USART Transmit Register								0000 0000	0000 0000
8Ch	PIE1	PSPIE ⁽¹⁾	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	0000 0000	0000 0000
98h	TXSTA	CSRC	TX9	TXEN	SYNC	—	BRGH	TRMT	TX9D	0000 -010	0000 -010
99h	SPBRG	Baud Rate Generator Register								0000 0000	0000 0000

Legend: x = unknown, - = unimplemented read as '0'. Shaded cells are not used for Synchronous Slave Transmission.

Note 1: Bits PSPIE and PSPIF are reserved on the 28-pin devices; always maintain these bits clear.

TABLE 10-11: REGISTERS ASSOCIATED WITH SYNCHRONOUS SLAVE RECEPTION

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR, BOR	Value on all other Resets
0Ch	PIR1	PSPIF ⁽¹⁾	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	0000 0000	0000 0000
18h	RCSTA	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D	0000 000x	0000 000x
1Ah	RCREG	USART Receive Register								0000 0000	0000 0000
8Ch	PIE1	PSPIE ⁽¹⁾	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	0000 0000	0000 0000
98h	TXSTA	CSRC	TX9	TXEN	SYNC	—	BRGH	TRMT	TX9D	0000 -010	0000 -010
99h	SPBRG	Baud Rate Generator Register								0000 0000	0000 0000

Legend: x = unknown, - = unimplemented read as '0'. Shaded cells are not used for Synchronous Slave Reception.

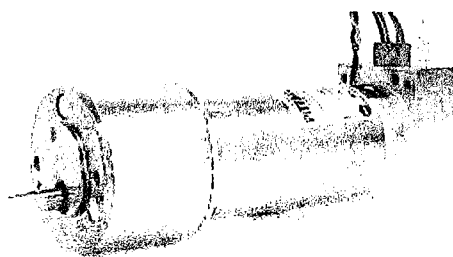
Note 1: Bits PSPIE and PSPIF are reserved on the 28-pin devices; always maintain these bits clear.

EK-6 PITTMAN GM9236S019 DC Motor Veri Kitapçığı



GM9236S019

Lo-Cog® DC Servo Gearmotor



Assembly Data	Symbol	Units	Value
Reference Voltage	E	V	12
No-Load Speed	S_{NL}	rpm (rad/s)	236 (24.7)
Continuous Torque (Max.) ¹	T_C	oz-in (N-m)	153 (1.1E+00)
Peak Torque (Stall) ²	T_{PK}	oz-in (N-m)	860 (6.1E+00)
Weight	W_M	oz (g)	23.3 (661)
Motor Data			
Torque Constant	K_T	oz-in/A (N-m/A)	3.25 (2.29E-02)
Back-EMF Constant	K_E	V/krpm (V/rad/s)	2.40 (2.29E-02)
Resistance	R_T	Ω	0.71
Inductance	L	mH	0.66
No-Load Current	I_{NL}	A	0.33
Peak Current (Stall) ²	I_P	A	16.9
Motor Constant	K_M	oz-in/ $\sqrt{W}$ (N-m/ $\sqrt{W}$)	4.11 (2.90E-02)
Friction Torque	T_F	oz-in (N-m)	0.80 (5.6E-03)
Rotor Inertia	J_M	oz-in-s ² (kg-m ²)	1.0E-03 (7.1E-06)
Electrical Time Constant	τ_E	ms	1.06
Mechanical Time Constant	τ_M	ms	8.5
Viscous Damping	D	oz-in/krpm (N-m-s)	0.053 (3.5E-06)
Damping Constant	K_D	oz-in/krpm (N-m-s)	12.5 (8.5E-04)
Maximum Winding Temperature	θ_{MAX}	°F (°C)	311 (155)
Thermal Impedance	R_{TH}	°F/watt (°C/watt)	56.3 (13.5)
Thermal Time Constant	τ_{TH}	min	13.5
Gearbox Data			
Reduction Ratio			19.7
Efficiency ³			0.84
Maximum Allowable Torque		oz-in (N-m)	500 (3.53)
Encoder Data			
Channels			3
Resolution		CPR	500

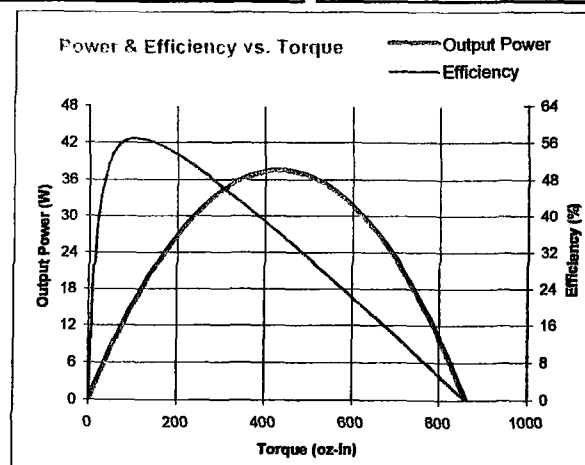
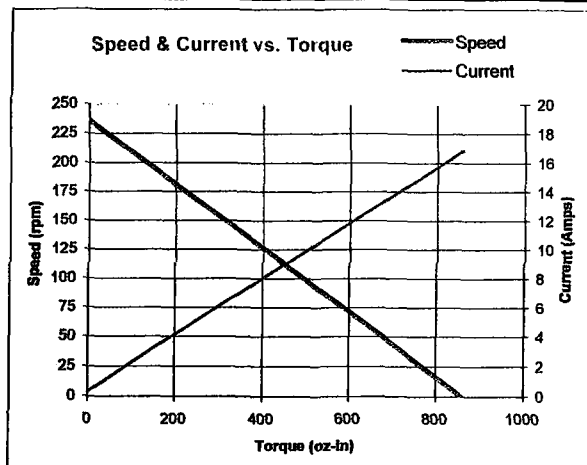
1 - Specified at max. winding temperature at 25°C ambient without heat sink. 2 - Theoretical values supplied for reference only.
3 - Effective gearbox efficiency for this unit improved by use of ball bearings.

Included Features

2-Pole Stator
Ceramic Magnets
Heavy-Gauge Steel Housing
7-Slot Armature
Silicon Steel Laminations
Stainless Steel Shaft
Copper-Graphite Brushes
Diamond Turned Commutator
Motor Ball Bearings
Output Ball Bearing
Wide Face Gears

Customization Options

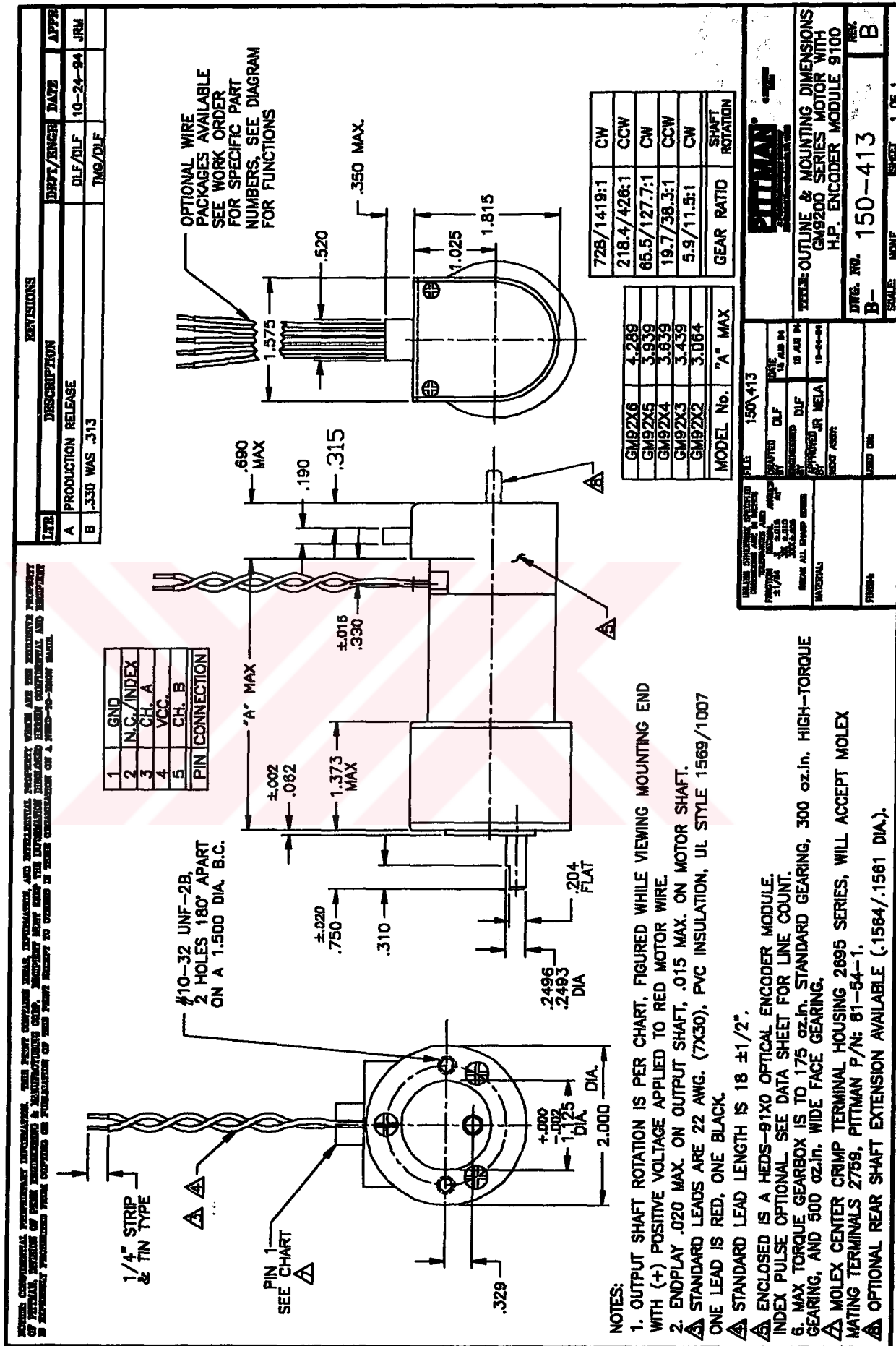
Alternate Winding
Sleeve or Ball Bearings
Modified Output Shaft
Custom Cable Assembly
Special Brushes
EMI/RFI Suppression
Alternate Gear Material
Special Lubricant
Optional Encoder
Fail-Safe Brake

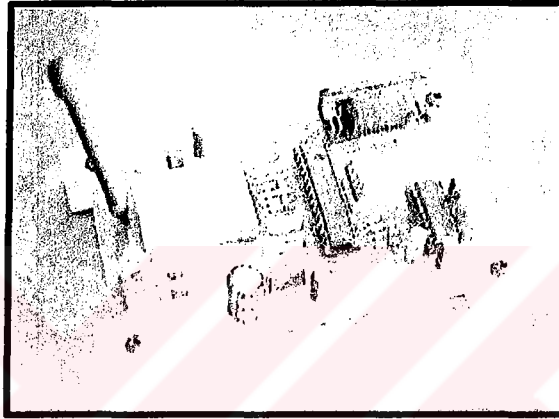


All values are nominal. Specifications subject to change without notice. Graphs are shown for reference only.

© 2001 Pittman.

UNCONTROLLED



**UMD-B12
RF MODEM BOARD**

ÜRÜN KILAVUZU
Version 1.2 (ARALIK 2004)

GENEL ÖZELLİKLER

Özellikler :

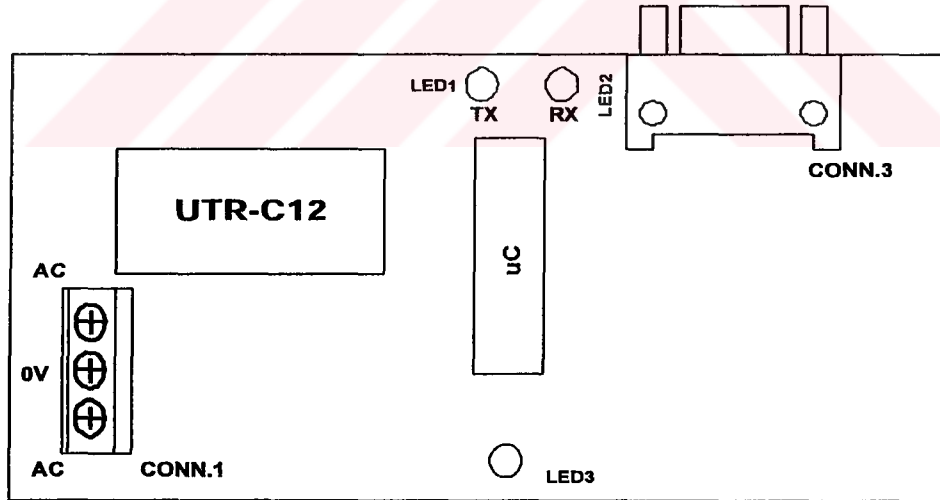
- * 433-434 MHz. UHF bandında EN 300 220 uyumlu. UTR-C12 modül
- * RS232 D-sub 9 pin konektör.
- * Frekans seçebilme özelliği.

Uygulama Alanları :

- * Kablosuz seri data haberleşme
- * Telemetry sistemleri
- * PC ve RS232 haberleşmeye uygun ekipmanlar arasında kurulacak RF link

Genel Açıklamalar :

UMD-B12 RF Modem RS232 data iletimine uygun ekipmanların RF iletim kurabilmeleri için tasarlanmıştır. UMD-B12 RF Modem, Kısa Mesafe Erisimli Telsiz Cihazlarının Temel Standartları ile Kurma ve Kullanma Esasları Hakkında Yönetmelik (TGM-STK-001) 'in 434MHz. ISM bandı ile ilgili bölümünü kapsıyacak şekilde tasarlanmıştır. 6 farklı kanal seçimi yapılabilecek şekilde üretilmektedir.



TEKNİK ÖZELLİKLER

MODEL			RF Modem UMD-B12
Besleme Voltaji			2 x 7,5 AC
			2 x 12V AC
			1 x 7,5 DC
			1 x 12 DC
Akim	RX ON	Min	TBD
		Max	TBD
	TX ON	Min	TBD
		Max	TBD
Power		RX ON	TBD
		TX ON	TBD
RF	Power Output		10 mW
	Receiver Sensitivity		-120dBm(UTR-C12)
	Data Rate		2400 Kbit/s
RS link	Cable		RS-232 TTL (PC Standard)
	Data Rate		9600 Kbit/s
	Data Format		8 bit, No parity, 1 stop, No Flow control
Ebat (mm)			
Agirlik (gr)			
Frekans Kanallari	CH1		433.056 MHz
	CH2		
	CH3		
	CH4		
	CH5		
	CH6		
Anten Girişi			50 Ohm
Anten Tipi			Helical / BNC

DATA COMMUNICATION

1.1. Fiziksel Karakteristik

Konnektör: Erkek DB-9 (Cannon sub-D)
 Transmission: Serial asynchronous
 Baud Rate: 19200 Bauds
 Link: RS 232

1.2. Data Format

8 data bits, no parity, 1 stop bit
 CTS ve RTS kullanılmıyor.

1.3. General Data Format

STX	SRC	DST	DATA	ETX
3Ch	Xh	Xh	MAX (75 BYTE)	3Eh

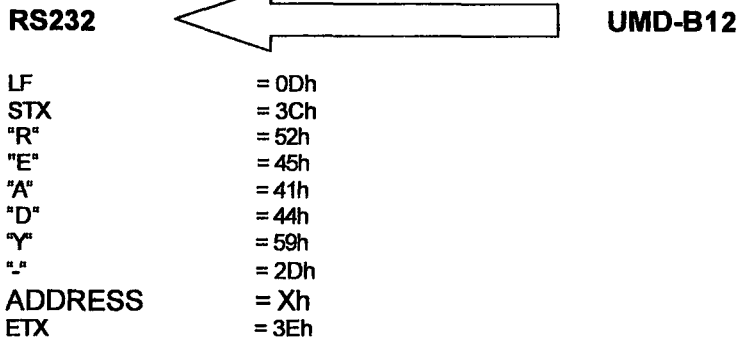
<SRC+DST+D1+D2+.....+Dx>

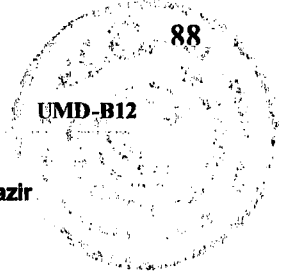
< = FRAME BASLANGICI 0x3C
 SRC = GÖNDEREN MODEM ADRESİ TX MODEM ADRES
 DST = GÖNDERİLEN MODEM ADRESİ RX MODEM ADRES
 D1 = DATA1
 D2 = DATA2
 .
 Dx = DATAx
 > = FRAME SONU 0x3E

1.4. UMD-B12 'den RS232 'ye Bilgi Akisi

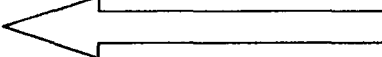
1.4.1. Start up

UMD-B12 enerjilendirildiği zaman RS232 'ye adresi ile birlikte hazır bilgisini gönderir "<READY-X>".



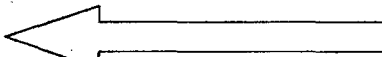


1.4.2. UMD-B12 RF'ten aldığı bilgiyi RS232 'ye aşağıdaki gibi verir. Ardından hazır bilgisini gönderir.

RS232  **UMD-B12**

STX = 3Ch
SRC = GÖNDEREN MODEM ADRESİ
DST = GÖNDERİLEN MODEM ADRESİ
INFO1 = FIRST BYTE

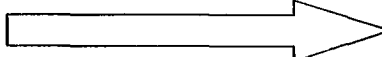
INFOX = LAST BYTE
ETX = 3Eh

RS232  **UMD-B12**

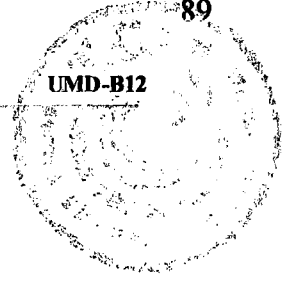
LF = 0Dh
STX = 3Ch
"R" = 52h
"E" = 45h
"A" = 41h
"D" = 44h
"Y" = 59h
" " = 2Dh
ADDRESS = Xh
ETX = 3Eh

1.5. RS232 'den UMD-B12 'ye Bilgi Akisi

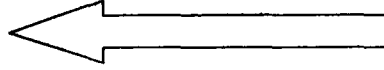
RF'e gönderilmek istenen data RS232'den UMD-B12'ye aşağıdaki formatta verilir. UMD-B12 verilmiş datayı RF ile gönderdikten sonra, RS232 'ye hazır bilgisini verir.

RS232  **UMD-B12**

< = 3Ch
SRC = TXMODEM ADRES
DST = RXMODEM ADRES
D1 = DATA1
D2 = DATA2
"
"
"
DX = DATA_X (max X = 75)
> = 0X3E



RS232



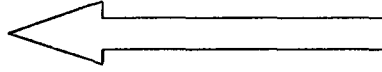
UMD-B12

LF	= 0Dh
STX	= 3Ch
"R"	= 52h
"E"	= 45h
"A"	= 41h
"D"	= 44h
"Y"	= 59h
"_"	= 2Dh
ADDRESS	= Xh
ETX	= 3Eh

1.6. HATA KODLARI

1.6.1. GENEL FORMAT

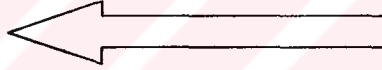
RS232



UMD-B12

STX	= 02h
ERRCODE0	= BYTE
ERRCODE1	= BYTE
STX	= 03h

RS232



UMD-B12

LF	= 0Dh
STX	= 3Ch
"R"	= 52h
"E"	= 45h
"A"	= 41h
"D"	= 44h
"Y"	= 59h
"_"	= 2Dh
ADDRESS	= Xh
ETX	= 3Eh

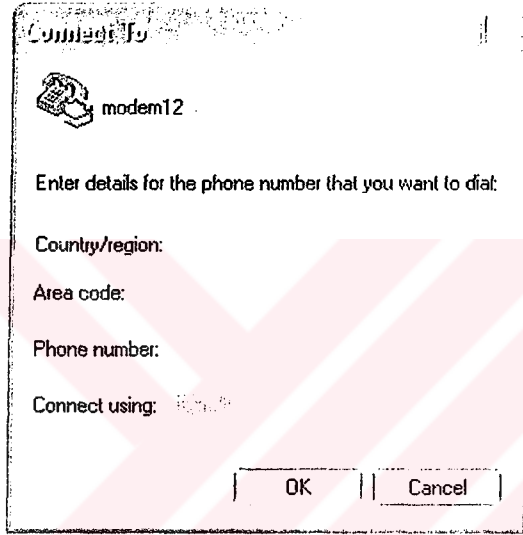
1.6.2. HATA KODLARININ ANLAMLARI

4Fh → FAZLA BILGI GIRILDI
54h → ZAMAN ASIMI

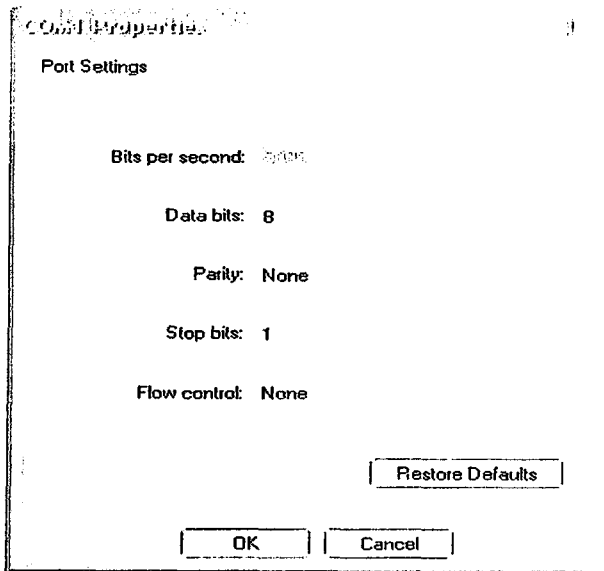
HABERLESME TESTİ

UMD-B12 Modem iletişim algoritması ve mesafesinin test edilmesi için Windows Hyper Terminal yazılımı kullanılabilir. Aşağıdaki işlemler testin yapılabilmesi için PC'de yapılacak işlemleri tarif etmektedir.

1. Click Start > Programs > Accessories > Communication > Hyper Terminal.
2. Açılan "Connection Description" penceresine isim ("name") girilmesi gerekir. İsim girildikten sonra (ör: MODEM12) OK butonuna basın.
3. Açılan "Connect To" penceresinden "Connect using" kutusunun içinden modemın bağlı olduğu COM portu seçin ve OK butonuna basın.



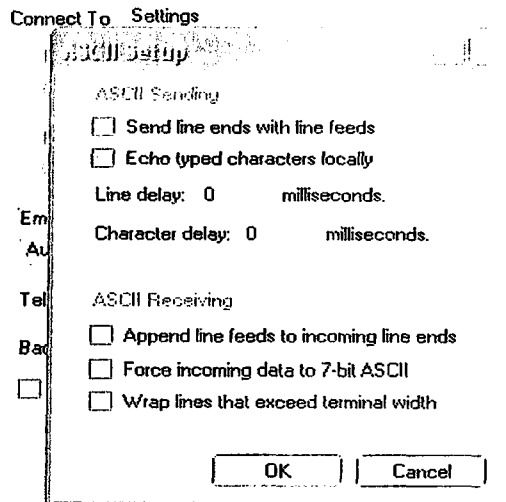
4. Açılan "COM Properties" penceresinde aşağıdaki seçenekler seçilmelidir. İşlem OK butonuna basılarak tamamlanmalıdır.

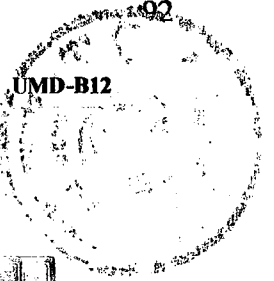


5. Aynı işlem diğer bir modem için farklı bir PC de gerçekleştirilmelidir.
6. UMD-B12 modemleri, beraberinde gelen 232C kabloları ile PC lerin tanımlanmış COM portlarına takılır ve enerjileri bağlanır. Bu aşamada her iki Hyper Terminal ekranında <READY> görünmelidir.



7. Hyper Terminale karakter yazıldığı zaman bunlar ekranınızda görünmeye bilir. Yazılanların ekranda görünmesi için Hayper Terminal 'de "File" > "Properties" seçilmeli, ardından "Settings" içinden sağ alt köşede olan "ASCII Setup" butonuna basılmalıdır. Açılan "ASCII Setup" penceresinde aşağıdaki seçenekler seçilmelidir.





8. 1 nolu modem'den 2 nolu modem'e bilgi göndermek için ekrana asagidaki gibi data girilebilir. Data gönderildikten sonra ekrana <READY> gelir.

