T.C.
CUMHURİYET ÜNİVERSİTESİ
MÜHENDİSLİK FAKÜLTESİ
ELEKTRİK – ELEKTRONİK MÜHENDİSLİĞİ
BÖLÜMÜ

TEZİN ADI

PIC MİKRO DENETLEYİCİLERİN BİLGİSAYAR ARAYÜZÜ İLE KONTROLÜ VE VERİ TABANI UYGULAMASI

TEZ SORUMLUSU

YRD. DOÇ. DR. MUSTAFA HOŞTUT

TEZİ HAZIRLAYAN

KENAN ALTUN

2006
SİVAS

PIC MİKRODENETLEYİCİLERİN BİLGİSAYAR ARA YÜZÜ İLE KONTROLÜ VE VERİ TABANI UYGULAMALARI

Kenan ALTUN

YÜKSEK LİSANS TEZİ

ELEKTRONİK ANABİLİM DALI

2006

PIC MİKRODENETLEYİCİLERİN BİLGİSAYAR ARA YÜZÜ İLE KONTROLÜ VE VERİ TABANI UYGULAMALARI

Kenan ALTUN

YÜKSEK LİSANS TEZİ

ELEKTRONİK ANABİLİM DALI

DANIŞMAN : YRD. DOÇ .DR. MUSTAFA HOŞTUT

FEN BİLİMLERİ ENSTİTÜSÜ MÜDÜRLÜĞÜ'NE

Bu çalışma, jürimiz tarafından Elektronik Anabilim Dalı'nda Yüksek Lisans Tezi olarak kabul edilmiştir.

Başkan: Doç. Dr. Rafael HUSEYNOV ……...
Üye : Yrd. Doç. Dr. Cemal KAYA …….…….....
Üye : Yrd. Doç. Dr. Mustafa HOŞTUT ……..

ONAY

Yukarıda imzaların adı geçen öğretim üyelerine ait olduğunu onaylarım.

…../…../ 2006

FEN BİLİMLERİ ENSTİTÜSÜ MÜDÜRÜ

PROF. DR. HALİL GÜRSOY

Bu tez Cumhuriyet Üniversitesi Senatosunun 05.01.1984 tarihli toplantısında kabul edilen ve daha sonra 30.12.1993 tarihinde C.Ü. Fen Bilimleri Enstitüsü Müdürlüğünce hazırlanan ve yayınlanan "Yüksek Lisans ve Doktora Tez Yazım Kılavuzu" adlı yönergeye göre hazırlanmıştır.

ÖZET

Yüksek Lisans Tezi

PİC MİKRO DENETLEYİCİLERİN BİLGİSAYAR ARAYÜZÜ İLE KONTROLÜ VE VERİ TABANI UYGULAMASI

KENAN ALTUN
YÜKSEK LİSANS TEZİ
ELEKTRİK ELEKTRONİK MÜHENDİSLİĞİ
ELEKTRONİK ANABİLİM DALI
EYLÜL 2006

DANIŞMAN : YRD. DOÇ. DR. MUSTAFA HOŞTUT

PIC Mikro denetleyicilerle bilgisayar arasında veri haberleşmesini sağlayarak, elde edilen dataları Visual Basic programlama dilinde yazılan ara yüz programı ile kullanılabilir veriler haline getirildi. Projede kullanılan zamanlama aygıtında gerçek zaman rutini Assembler dilinde yazılarak oluşturuldu. Seri iletişim protokolleri Max-232 entegresiyle oluşturuldu ve mikro denetleyicilerinin bilgisayar ara yüzü ile haberleşmesi sağlandı.

ANAHTAR KELİMELER: Mikro denetleyiciler, PIC Mikro kontrolörler, Seri İletişim, Visual Basic.

TEŞEKKÜRLER

Bu konu üzerinde çalışma yapmama olanak sağlayan ve çalışmalarım süresince hiçbir yardımını benden esirgemeyen;

Yrd. Doç. Dr. Mustafa HOŞTUT'a,
Doç. Dr. Rafael HUSEYNOV'a,

Ve Eşim Didem ALTUN'a Teşekkür ederim.

Kenan ALTUN

ŞEKİLLER DİZİNİ

ÇİZELGELER DİZİNİ

BÖLÜM – 1 : GİRİŞ

Mikrodenetleyicilerin hızla gelişimi mikrodenetleyicilerin kullanım alanını artırmıştır. Hazırlanan tez çalışmasında PIC16F877 mikrodenetleyicisi programlanarak zamanlama devresi tasarlanmıştır. Mikrodenetleyici ile bilgisayar haberleşmesi sağlanarak elde edilen verinin bilgisayar ortamında incelenmesi amaç edilmektedir. Yapılan çalışmada ilk önce mikrodenetleyiciler, assembler programlama dili, seri iletişim protokolü ve Visual Basic(V.B.) programlama dili kullanılacak ve anlatılacaktır.

Yapılan çalışmada PIC 16F877 mikrodenetleyicisi ile seri iletişim haberleşmesi sağlanmıştır. Böylece zamanlama devresinde başlangıç ve bitiş zaman dilimi arasında elde edilen kronometrik zamanın V.B. programlama dilinde yazılan arayüz programıyla puan olarak hesaplanarak bir veritabanı oluşturulmuştur.

PIC(Peripheral Interface Controller) mikrodenetleyicilerin bilgisayarla seri iletişimi ve arayüz programı uygulamasında; Mikrodenetleyicinin tasarlanması, assembler dilinde programlanması ve seri iletişim kurallarının uygulanması kullanılmıştır. Ayrıca elde edilen veri visual basic programlama dilinde veritebanında sıralama yapmak için kullanılmıştır.

Mikrodenetleyici tasarlanmasında osilatör cinsi, PIC'in cinsi ayrıca PIC'in programlanmasında kullanılan programlama dili önemli olmuştur. Çünkü projemiz gerçek zamanlı bir kronometre devresinden oluştuğundan bu gibi materyaller önem arz etmektedir. PIC mikrodenetleyicisinin programlanması assembler dilinde yapılmış olup gecikme rutini döngülerle sağlanmıştır. Ayrıca PIC programında seri iletişim protokolü de yazılmıştır. Yazılan protokole göre PIC çıkış portları Max-232 entegresiyle dönüştürülerek verinin seri porta ulaşması sağlanmıştır. Projede kullanılan 7 segment displayler PIC entegresinin çıkış portları ile sürülmüştür. Seri porttan bilgisayara alınan veri Visual basic progralama dilinde yazılan arayüz programıyla istenilen puan hesaplanmıştır. Bu projede verileri bilgisayara seri port ile taşınarak veri tabanına aktarılması yoluyla bilgisayara elle girilen verilerden kaynaklanan kişisel hataların ortadan kaldırılması planlanmıştır.

BÖLÜM – 2 : MİKRODENETLEYİCİLER

2.1 GİRİŞ

Günümüzde hızla gelişen teknoloji bilgisayarlarla kontrol edilen cihazları bizlere çok yaklaştırdı. Öyle ki, cebimizde taşıdığımız telefon, büromuzdaki faks makinesi, evimizdeki çamaşır makinesi gibi cihazlar artık çok küçük, adına mikrodenetleyici denilen elektronik elemanlar yardımıyla kontrol edilir hale geldi. Aynı şeyler, 1990 öncesinde mikroişlemci (CPU) kontrol elemanlarıyla yapılmaktaydı. CPU ile bir cihazı kontrol etmek için GİRİŞ / ÇIKIŞ (I/O) elemanları, RAM gibi ek devreler gerekmekteydi. Bunlar hem maliyeti artırıyor, hem de programlamayı zorlaştırıyordu. İşte Microchip firmasının PIC adını verdiği mikrodenetleyicilerle bu sorunlar ortadan kaldırıldı.

Fiyatları 5-6 $ olan bir PIC16F84 yongayla, RAM ve GİRİŞ / ÇIKIŞ portu gibi birimlere ihtiyaç duymadan istediğimiz devreyi kurabilir, ücretsiz olarak edinebileceğimiz PIC ASSEMLY adı verilen program sayesinde PIC kolaylıkla programlanabilir.

Bilgisayar denetimi gerektiren bir uygulamayı geliştirirken seçilecek mikrodenetleyicinin ilk olarak tüm isteklerinizi yerine getirip getirmeyeceğine, daha sonra da maliyetinin düşüklüğüne bakmalıyız. Ayrıca yapacağımız uygulamanın devresini kurmadan önce seçtiğimiz mikrodenetleyicinin desteklediği bir yazılım üzerinde simülasyonunu ile, yapıp yapamayacağımızı dikkate almalıyız.

İşte bütün bu özellikleri göz önüne aldığımızda PIC'leri kullanmanın akılcı bir yol olduğu görülmektedir.

2.2 PIC'LERİN ORTAYA ÇIKIŞI

Harvard mimarisindeki ilk mikrodenetleyici ünitesi, General Instruments firması tarafından 1970'lerin ortalarında üretilen Signetics 8X300 modeliydi. Bu 16 bitlik CP1600 MPU için programlanabilen giriş/çıkış portu olmak üzere Peripheral Interface Controller (Çevrebirim arayüz denetleyicisi - PIC) olarak tasarlandı.

General Instruments firması mikroelektronik bölümünü sattı ve bu bölüm 1988 yılında Arizona Microchip Technology adıyla yeni bir firmaya dönüştü. Microchip'in ana ürünü, bugün de hala öyle olan, PIC serisi mikrokontrollörlerdir. 1989'da ilk piyasaya sürülen aile PIC16C5X serisiydi. Bu Harvard mikrontrollörler 33 komutluydu. Bütün komutlar 12-bit word olarak kodlanıyordu.

Azaltılmış Komut Kümesi (Reduced Instruction Set Computer - RISC) temelli olan komut seti hızlı, etkili ve ucuz işlemci üretimini sağladı. PIC16C5XX 12-bit çekirdekli ailede 512 ve 2048 komutluk tek sefer programlanabilen (One Time Programmable (OTP)) EEPROM Program belleği, 25-73 byte veri belleği,18- ve 28-pinli paketlerde 12 veya 20 giriş/çıkış pini ve 8-bit zamanlayıcı gibi özellikler bulunmaktaydı. PIC12CXXX ailesi bunların 8-pinlik eşdeğerleridir.

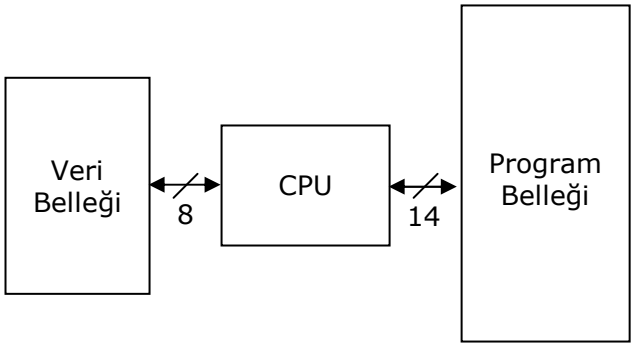
1992 yılında 14-bitlik çekirdeğe sahip PIC16CXXX ailesi daha fazla program alanının ve interrupt(kesme) işlemleri yanında A/D çeviriciler, 16 bit sayıcılar gibi çevre birimlerinin kullanımına olanak sağladı. Bu ailedeki RISC komut seti de 12-bit çekirdektekilerle hemen hemen aynıydı ve 35 komuttan oluşuyordu. 1997'de çarpma yapabilen bir Aritmetik Logic Unit (ALU)'e ve ileri arabirim yeteneklerine sahip 16-bit PIC17CXXX ailesi piyasaya sunuldu. Ardından 1999 yılında da genişletilmiş 16-bit çekirdekli PIC18CXXX ailesi sunuldu. Bu ailedeki işlemcilerde komut sayısı 77 idi ve bu yüksek-seviye dillerin derleyicilerin ihtiyaçlarını daha fazla karşılıyordu.

Bu 3 aile arasında, 14-bit çekirdekli olan aile hem kullanım kolaylığı hem de maliyet olarak en uygunudur. Burada ve birçok kaynakta hakkında bilgiler bulabileceğiniz PIC16F84, orta seviye ailesinin bir üyesidir. Yazılım açısından baktığımızda bütün cihazlar aynı çekirdeğe sahiptirler. Ancak donanım açısından birçok ortak noktaları olmakla birlikte farklı giriş/çıkış birimlerinin karışımıdırlar. Örneğin 16C74'de 8 kanal analog giriş portu, PIC16C66'da senkronize seri port ve PIC16F84'de de kalıcı veri belleği bulunmaktadır. Bu üç cihaz da benzer paralel giriş/çıkış, sayıcı ve kesme idare birimlerine sahiptir.

2.3 PIC 16F877 MİMARİSİ

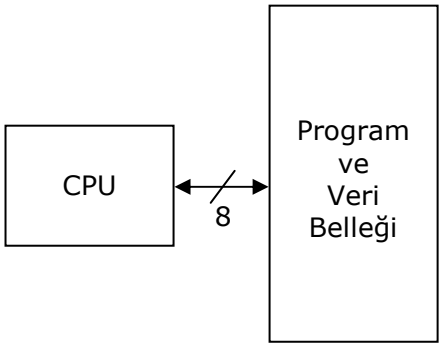
PIC 16F87X ve 16F8X serisi öncelikle, PIC 16CXX ailesinin özelliklerini taşır. PIC 16CXX'de Harvard mimarisi kullanılmıştır. Von Neuman mimarisinde, veri ve program belleğine aynı yoldan erişilebilirken, bu mimaride program belleği ve veri belleğine erişim farklı boylarda yapılır. Veri yolu (databus) 8 bit genişliğindedir. Aynı anda, veri belleğine 8 bit genişliğindeki bu yolla erişilirken; program belleğine program yolu ya da adres yolu (program bus / adress bus) denilen 14 bit genişliğindeki diğer bir yolla erişilir[1]. Bunun için PIC 16F87X ve PIC 16F84'de komut kodları (opcode), 14 bittir. 14 bitlik program belleğinin her bir adresi, bir komut koduna (Instruction Code / Instruction Word) karşılık gelir. Dolayısıyla her komuta bir çevrim süresinde (cycle) erişilir ve komut kaydedicisine yüklenir. Komut kaydedicisi, CPU tarafından kullanılan bir kaydedicidir ve dallanma komutları dışındaki bütün komutlar, aynı çevrim süresinde çalıştırılırlar. Bu sırada program

sayacı, PC (Program Counter) bir artar. Dallanma ya da sapma komutları ise, iki ardışık periyotta çalıştırılır ve program sayacı PC, iki arttırılır.



Harvard

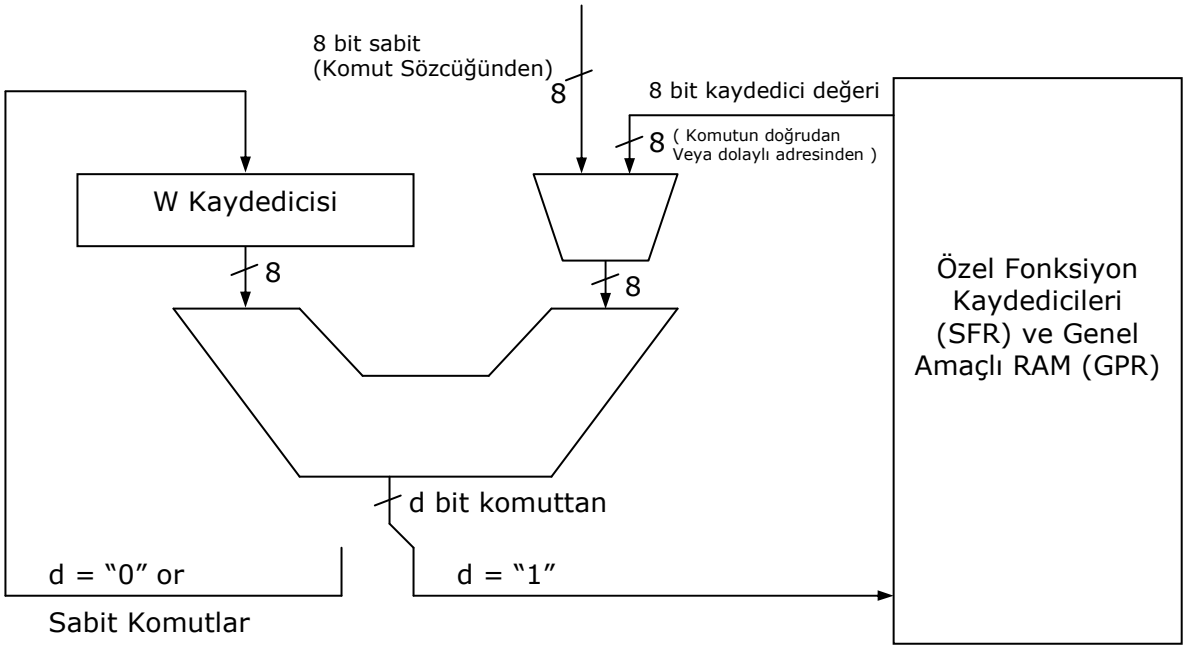
ŞEKİL – 1.1 : Harvard Mimarisi



Von-Neuman

ŞEKİL – 1.2 : Von-Neuman Mimarisi

Merkezi işlem biriminin (CPU) en önemli alt birimlerinden biri, ALU (Aritmetik Logic Unit) olarak adlandırılan aritmetik mantık birimidir. ALU'nun görevi, kendisine yollanan veriler üzerinde, aritmetik ya da mantıksal işlemler yapmaktır. ALU'nun, biri kontrol girişi diğeri W (Working Register) ismi verilen kaydediciden olmak üzere, iki ana girişi vardır. ALU kendisine gelen iki veriyi (işleçler), toplayıp çıkarılabilir. Çeşitli mantık işlemleri yapabilir (and,or, xor gibi).



ŞEKİL – 1.3 : Aritmetik Mantık Ünitesi

Mikroişlemcilerde en çok kullanılan kaydedici, "working register"dır. Bu kısaca W olarak adlandırılır. W, aritmetik ve mantık işlemlerinde, iki işlevi bir arada yürütür. İşlemden önce, işlenenlerden birini barındırır. İşlemden sonra ise işlem sonucunu saklar, PIC 16F8X ve 16F87X serisi mikrodenetleyicilerde, komutun sonuna konan 1 veya 0 sayısıyla (d), sonucun W'de ya da başka bir kaydedicisinde tutulacağı mikroişlemciye bildirilir.

PIC 16F877 ve 16F876, 8 KB word büyüklüğünde belleğe sahiptir. Program belleği yonganın içerisindedir. PIC 16F84'ün belleği ise 1KB word büyüklüğündedir.

PIC 16F84 ve 16F87X serisi mikrodenetleyiciler, kendi kaydedicilerini ve veri belleğini, doğrudan, dolaylı ve göreceli olarak adresleyebilirler.

<u>16F87X Mikrodenetleyici ailesi aşağıdaki temel özellikleri taşır.</u>

- CPU azaltılmış komut setine sahiptir.
- RISC temeline dayanır.
- Öğrenilecek 35 komut vardır ve her biri 14 bit uzunluktadır.
- Dallanma komutları iki çevrim (cycle) sürede, diğerleri ise bir çevrimlik sürede uygulanır.
- İşlem hızı 16F877'de DC-20 MHz'dir. (16F877'de bir komut DC-200 ns hızında çalışır.)
- Veri yolu (databus) 8 bittir.

- 32 adet SFR (Special Function Register) olarak adlandırılan özel işlem kaydedicisi vardır ve bunlar statik RAM üzerindedir.
- 8 Kword'e kadar artan flash belleği 1 milyon kez programlanabilir.
- 368 Byte'a kadar artan veri belleği (RAM),
- 256 Byte'a kadar artan EEPROM veri belleği vardır.
- Pin çıkışları PIC 16C73B/74B/76 ve 77 ile uyumludur.
- 14 kaynaktan kesme yapabilir.
- Yığın derinliği 8'dir.
- Doğrudan, dolaylı ve göreceli adresleme yapabilir.
- Power-on Reset (Enerji verildiğinde sistemi resetleme özelliği)
- Power-up Timer (Power-up zamanlayıcı)
- Osilatör Start-up Timer (Osilatör başlatma zamanlayıcısı)
- Watch-dog Timer (Özel tip zamanlayıcı), devre içi RC osilatör
- Programla kod güvenliğinin sağlanabilmesi özelliği
- Devre içi Debugger (Hata ayıklamakta kullanılabilecek modül)
- Düşük gerilimli programlama
- Flash ROM program belleği (EEPROM özellikli program belleği)
- Enerji tasarrufu sağlayan, uyku –Sleep Modu
- Seçimli osilatör özellikleri
- Düşük güçle, yüksek hızla erişilebilen, CMOS-Flash EEPROM teknoloji
- Tümüyle statik tasarım
- 2 pinle programlanabilme özelliği
- Yalnız 5V girişle, devre içi seri programlanabilme özelliği
- İşlemcinin program belleğine, okuma/yazma özelliği ile erişimi
- 2.0 V – 5.0 V arasında değişen geniş işletim aralığı
- 25 mA'lik kaynak akımı
- Devre içi, iki pin ile hata ayıklama özelliği
- Geniş sıcaklık aralığında çalışabilme özelliği
- Düşük güçle çalışabilme özelliği

<u>Çevresel özellikleri ise şöyle sıralanabilir:</u>

- TMR0: 8 bitlik zamanlayıcı, 8 bit önbölücülü
- TMR1: Önbölücülü, 16 bit zamanlayıcı, uyuma modundayken dış kristal zamanlayıcıdan kontrolü arttırılabilir.
- TMR2: 8 bitlik zamanlayıcı, hem önbölücü hem de sonbölücü sabiti
- İki Capture / Compare / PWM modülü

- 10 bit çok kanallı A/D çevirici
- Senkron seri port (SSP), SPI (Master mod) ve I^2C (Master Slave) ile birlikte çalışır.
- Paralel Slave Port, 8 bit genişlikte ve dış RD, WR, CS kontrolleri
- USART/SCI, 9 bit adres yakalamalı
- BOR Reset (Brown Out Reset) özelliği

BÖLÜM – 3 : SERİ PORT İLETİŞİM

3.1 PC'LERDE SERİ PORT DONANIMI

Seri Port ile haberleşmek Paralel Porta göre daha zordur. Çoğu zaman, seri porta bağlanan bir aygıtın kullanılabilmesi için, yine paralele dönüştürülecek, seri iletime ihtiyaç vardır. Bu UART (Universal Asynvhronous Receive Transmit) kullanılarak yapılabilir. Yazılım açısından bakılınca da Standart Paralel Porta (SPP) ulaşırken kullanılan registerlardan çok daha fazlası kullanılmaktadır.

Seri veri transferinin paralele göre avantajları şunlardır :

1. Seri kablolar paralel kablolardan uzun olabilir. Seri port kullanımında '1' biti -3 'ten -25 volta, '0' biti de +3'ten +25 volta kadar temsil edilir. Paralel port kullanımında ise '0' 0 volt, '1' ise 5 volt olarak temsil edilir. Bundan dolayı, paralel porttaki maksimum salınım 5 volt iken seri portta bu 50 volta kadar çıkmaktadır. Bu nedenle seri portta kablodaki gerilim kaybı paralel porttaki kadar fazla sorun oluşturmaz.
2. Paralel iletimdeki kadar çok kabloya ihtiyaç duyulmaz. Eğer aygıtın bilgisayardan oldukça uzak bir mesafede kurulması gerekirse 3 kablo kullanmak, paralel iletimdeki gibi 19 veya 25 kablo kullanmaktansa daha ucuzdur. Bununla birlikte her uçtaki arabirim maliyeti gözardı edilmemelidir.
3. Infra Red (Kızıl Ötesi) aygıtlar son zamanlarda oldukça yaygın. Kızıl ötesi özelliği olan elektronik günlükler veya cep bilgisayarları görmüş olabilirsiniz. Bununla birlikte, 8 bitlik bir verinin aynı anda bir oda içinde iletilirken, aygıtın hangi bitin ne olduğunu anlamasının imkansız olduğu açıktır. Bundan dolayı bir bit gönderildiğinde seri iletişim kullanılır. IrDA-1 (ilk infra red özelliklerinden) 115.2k baud 'tur ve UART'a bağlıdır. Bununla birlikte, bu aygıtların genellikle elektronik günlüklerde, dizüstü veya avuçiçi bilgisayarlarda kullanıldığını varsayarak, gücü muhafaza etmek için RS232 bit uzunluğunun 3/16'sına kısaltılmıştır.
4. Mikrodenetleyici kullanımı da günümüzde artmıştır. Bunların birçoğu dış dünya ile iletişim kurabilecek SCI (Serial Communication Interface - Seri İletişim Arabirimi) ünitelerine sahiptir. Seri iletişim bu mikrodenetleyicilerdeki pin sayısını azaltır. 8 bitlik paralel iletimde 8 pin kullanılmasına karşın, seri iletimde yalnızca iki pin ortak olarak kullanılır, Transmit Data (TxD) ve Receive Data (RxD) (Strobe ucuna da ihtiyaç duyulabilir).

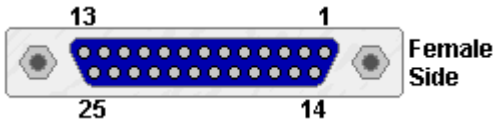
3.1.1 DONANIM ÖZELLİKLERİ

Seri iletişim kullanan aygıtlar iki sınıfa ayrılır. Bunlar DCE (Data Communications Equipment – Veri İletim Cihazları) ve DTE (Data Terminal Equipment – Veri Terminal Cihazları). Veri İletim Cihazları için modem, TA adaptörü, plotter örnek olarak verilebilir. Bilgisayar veya terminal Veri Terminal Cihazları sınıfına girer. Seri portun elektriksel özellikleri EIA (Electronic Industries Association – Elektronik Sanayi Birliği) RS232 standardı tarafından belirlenir. Bu standartın bazı parametreleri aşağıdaki gibidir :

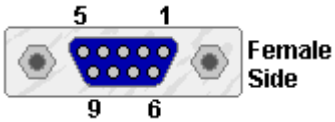
1. "Logic 0 " +3 ile +25 volt arasındadır.
2. "Logic 1" -3 ile -25 volt arasındadır.
3. +3 ile -3 arasındaki bölge tanımsızdır.
4. Açık devre voltajı 25 voltu asla aşmamalıdır (GND referans alınarak).
5. Kısa devre akımı 500mA aşmamalıdır. Aygıt bu akımla, devre zarar görmeden çalışabilmelidir.

Yukarıda sayılan özellikler EIA standardı tam listesinin çok küçük bir kısmıdır. Ayrıca RS232C standardının maksimum baud rate olarak bugünki standartlardan oldukça yavaş olan 20,000 bps'i desteklemesi ilginçtir. Yenilenmiş standartlar, EIA-232D ve EIA-232E sırasıyla 1987 ve 1991 yıllarında çıkmıştır.

Seri portlar iki şekilde olur : D Tipi 25 pin connetcor ve D Tipi 9 pin connector vardır. Bunlar PC'nin arkasında erkek connector olarak yerleştirilirler ve seri porta bir aygıt bağlamak için dişi connector'e ihtiyaç duyulur. Aşağıda 9 pin ve 25 pin D-tipi connector'lerin pin bağlantıları verilmiştir.



ŞEKİL – 2.1: 25 Pinli Seri Port Konnektör



ŞEKİL – 2.2: 9 Pinli Seri Port Konektör

D-Tipi 25 pin No	D-Tipi 9 pin No	Kısaltma	Adı (İşlevi)
Pin 2	Pin 3	TD	Transmit Data (Veri Gönder)
Pin 3	Pin 2	RD	Receive Data (Veri Al)
Pin 4	Pin 7	RTS	Request To Send (Gönderme İsteği)
Pin 5	Pin 8	CTS	Clear To Send (Göndermeye Müsait)
Pin 6	Pin 6	DSR	Data Set Ready (Veri Paketi Hazır)
Pin 7	Pin 5	SG	Signal Ground (Sinyal Topraklama)
Pin 8	Pin 1	CD	Carrier Detect (Taşıyıcı Tanımlandı)
Pin20	Pin 4	DTR	Data Terminal Ready (Veri Terminali Hazır)
Pin 22	Pin 9	RI	Ring Indicator (Çevrim Göstergesi)

ÇİZELGE – 2.1: 25 ve 9 Pinli Seri Port Pin Bağlantıları

3.1.2 PİN FONKSİYONLARI

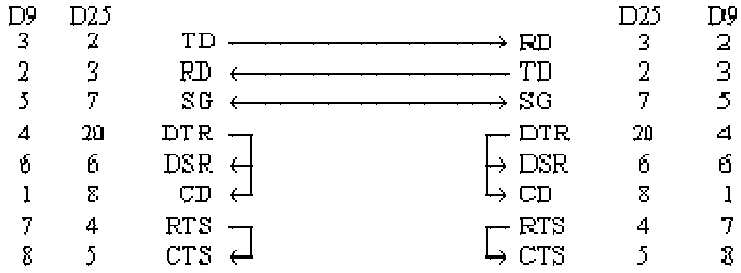
Kısaltma	Adı	Fonksiyonu
TD	Transmit Data	Seri Veri Çıkışı (TxD)
RD	Receive Data	Seri Veri Girişi (RxD)
CTS	Clear To Send	Bu hat seri portun veriyi göndermek için hazır olup olmadığını belirler.
DCD	Data Carrier Detect	Seri port telefon hattının diğer ucundaki portta bir taşıyıcı (Carrier) tespit ettiğinde hat aktif olur.
DSR	Data Set Ready	UART'a modemin bağlantı için hazır olduğunu belirtir.
DTR	Data Terminal Ready	DSR'nin tersini yapar. Modeme UART'ın bağlantı için hazır olduğunu belirtir.
RTS	Request To Send	Modeme, UART'ın veriyi göndermek için hazır olduğunu belirtir.

RI	Ring Indicator	Modem, PSTN'den bir çevrim sinyali tespit ettiğinde aktif duruma geçer.

ÇİZELGE – 2.2: Seri Port Pin Fonksiyonları

3.1.3 NULL MODEMLER

Null Modem iki DTE'yi birbirine bağlamak için kullanılır. Bu yöntem, network oyunları ya da Zmodem Protocol, Xmodem Protocol gibi protokoller kullanarak bilgisayarlar arasında dosya transfer etmek için oldukça yaygın kullanıma sahiptir. Ayrıca birçok Mikroişlemci Geliştirme Sistemleri ile birlikte kullanılabilir [2].



ŞEKİL – 2.3: Null Modem Bağlantısı

Şekil – 2.3'de en çok tercih edilen Null Modem bağlantısı verilmiştir. Yalnızca 3 tel uzatılmış olması uzun bağlantılar için maliyeti düşürür. İşlemin teorisi oldukça basittir. Hedef, bilgisayarın bilgileri bir başka bilgisayara değil de bir modeme ilettiğini düşündürmektir. Birinci bilgisayardan gönderilen herhangi bir verinin ikinciye ulaşması için TD çıkışı diğer bilgisayarın RD girişine bağlanmalıdır. İkinci bilgisayarın da aynı şekilde TD ucu birinci bilgisayarın RD ucuna bağlanır. Son olarak da SG uçları birbirine bağlanır.

Her iki bilgisayarda da Data Terminal Ready (DTR) çıkışı, Data Set Ready (DSR) ve Carrier Detect (CD) ucuna uygulanmıştır. DTR aktif olduğunda DST ve CD hemen aktif duruma geçerler. Bu durumda bilgisayar, bağlı olduğu Sanal Modemin hazır olduğunu, bu modem için taşıyıcı (carier) tespit ettiğini düşünür.

Bu işlemlerden sonra RTS ve CTS çıkışları ayarlanır. İki bilgisayar aynı hız ile haberleştiklerinden akış kontrolüne ihtiyaç duyulmaz ve böylece her iki bilgisayarda da bu iki çıkış birbirine bağlanır. Bilgisayar veri göndermek istediğinde, RTS sinyali üretir. Bu sinyal direkt olarak CTS çıkışına varacağından, verinin gönderilmeye hazır olduğu cevabını hemen alır ve işlem gerçekleştirilir.

3.2 SERİ İLETİŞİM PROTOKOLLERİ

	RS-232	RS-422	RS-485
Operasyon modu	single ended	differential	differential
Satır başına sürücü	1	1	32
Satır başına alıcı	1	10	32
Maximum kablo uzunluğu	50 feet	4000 feet	4000 feet
Maximum veri hızı	20 kbps	10 Mbps	10 Mbps
Maximum sürücü çıkışı voltajı	±25V	-0.25 to 6V	-7 to +12V
Sürücü çıkışı sinyal seviyesi(yüklenen)	±5V	±2V	±1.5V
Sürücü çıkışı sinyal seviyesi (yüklenmeyen)	±15V	±5V	±5V
Sürücü yükleme empedansı	3kΩ to 7kΩ	100kΩ	54kΩ
Max. Sürücü çıkışı akımı(açıkken)	n/a	n/a	±100μA
Max. Sürücü çıkışı akımı(kapalıyken)	VMAX/300Ω	±100μA	±100μA
Düşme hızı	30V/μs max.	n/a	n/a
Alıcı giriş voltaj sahası	±15V	-7V to +7V	-7V to +12V
Alıcı giriş hassasiyeti	±3V	±200mV	±200mV
Alıcı giriş direnci	3kΩ to 7kΩ	4kΩ	12kΩ

ÇİZELGE – 2.3: Seri İletişim Protokolleri

3.3 SERİ PORTUN PROGRAMLANMASI

3.3.1 SORGULAMA VEYA KESME

Bir haberleşme programı yazarken kullanılabilecek iki metod vardır. Yeni bir verinin mevcut olup olmadığını görmek için UART'ı sorgulayabilirsiniz veya UART'taki verinin bir kesme ürettiği zaman bu veriyi silecek bir kesme yöneticisi oluşturabilirsiniz. UART'ı sorgulamak işlemciye çok fazla iş yüklediği için oldukça yavaştır ve veriyi kaybetmeden önce maksimum 34.8 Kbps hızına ulaşmaktadır. Pentium Pro'dan sonra çıkan yeni işlemciler bu işi daha hızlı yapabilmektedirler. Diğer seçenek ise bu çalışmada yapıldığı gibi, kesme yöneticisi kullanmaktır. Bu yöntem düşük hızlı bilgisayarlarda bile 115.2 Kbps hızını desteklemektedir.

3.4 RS-232 PORTU İLE AYGITLARIN HABERLEŞMESİ

3.4.1 SERİ VERİ İLETİŞİMİ

Seri haberleşmede 8 yada daha farklı sayıda veriler 2 tel üzerinden iletilirler. Paralel haberleşmeye göre daha uzun mesafelere veriler iletilebilirler. Paralele göre dezavantajı ise veri iletim hızının yavaş olmasıdır.

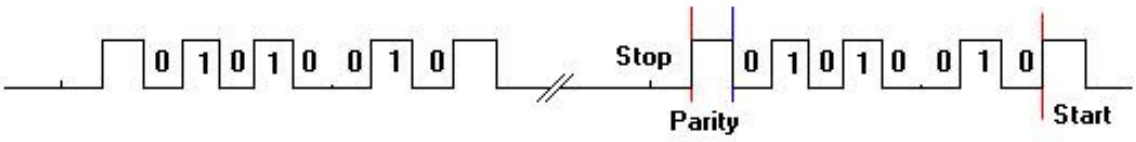
Seri Haberleşmede Önemli Terimler:

Baud Rate: Veri iletim hızıdır. 1 saniyede iletilen veri adedine denir. Standart olarak veri hızları 300,600,1200,2400,4800,9600,19200, ... şeklindedir.

Start Bit: Seri haberleşmede veriler senkron ya da asenkron olarak iletilebilirler. Senkron iletimde belirli bir başlangıçtan sonra veriler sıralı olarak ard arda gelirler. Uzun bir dosyanın iletiminde bu yol kullanılabilir. Asenkron iletimde ise bir veri gönderildikten sonra yeni bir veri belirli olmayan bir zamanda gelir. İşte bu yeni verinin başlangıcı start biti ile bildirilir.

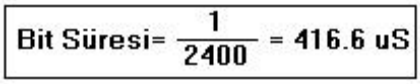
Stop Bit: Gönderilen verinin bittiğini ifade eden bittir. Bu biti alan alıcı yeni bir veri için start bitini gözlemeye başlar. Haberleşmenin tipine göre 1 yada 2 bit uzunluğunda olabilir.

Eşlik Biti (Parity Bit): Bir çeşit hata denetim bitidir. Ya da start ve 8 bitlik bir veri iletildikten sonra stop biti gönderilmeden parity gönderilir. İletilen 8 bitlik veride 1'lerin sayısı çift ise EVEN, tek ise ODD biti gönderilir. Alıcı bu biti kontrol ederek alınan verinin doğru olup olmadığını kontrol eder. 1 parity 1 bit uzunluğundadır.



ŞEKİL – 2.4: Seri İletişim Dalga Şekli

Burada dikkat edilmesi gereken Start ,Stop ve veri bitlerinin süresidir. Buda bize baud rate'i verir. Örnek olarak 2400 Baud'da Start birinin süresi:



ŞEKİL – 2.5: Seri İletişim Bit Süresi

Buna göre 8 bitlik bir veri parity ile birlikte 4.58 mS de iletim hattından iletilir. 9600 baud için bu hız 1.14 mS ye düşer.

3.4.2 SERİ İLETİŞİM KURALLARI

Seri iletişimde senkron ve asenkron veri iletişimi bulunmakta ve asenkron veri iletişimi için özel yazmaçlar bulunmaktadır[3].

USART'ın çalışması için Baud oranı son derece önem taşımaktadır. İstenilen Baud oranı SPBRG yazmacına yüklenmelidir. BRGH yazmacının durumuna göre SPBRG yazmacına İki değişik sayı yüklenebilir. Çizelge – 2.4 4 MHz kristal ile çalışırken ve BRGH lojik 1 olduğunda çeşitli Baud oranları için SPBRG yazmacına yüklenecek olan sayı verilmiştir.

4MHz kristal kullanırken ve BRGH = 1 olduğunda SPBRG değerleri

Baud Oranı	SPBRG değeri
1200	207
2400	103
9600	25
19200	12
28800	8
33600	6
57600	3

ÇİZELGE – 2.4: Seri İletişim Baud Oranları

USART'ı asenkron modda kullanırken gönderilecek olan veri TXREG yazmacına yüklenir. Gelen veri ise RXREG yazmacından okunur. USART'ı asenkron modda kullanmak için aşağıdaki işlemler takip edilir.

- SPBRG yazmacını istenilen BAUD oranı için Çizelge – 2.4 den yükleyiniz.
- Asenkron seri portu aktif yapmak için SYNC bitini sıfırlayınız ve SPEN bitini 1 yapınız.

- Kesme (interrupt) istiyorsanız RCIE bitini aktif yapınız.
- 9 uncu biti istiyorsanız RX9 bitini 1 yapınız.
- CREN bitini 1 yaparak seri almayı aktif yapınız
- Bayrak RCIF seri veri alındığında 1 olacaktır Seri veri beklerken bu biti kontrol ediniz.
- RCSTA yazmacında 9 uncu biti okunur. (aktif oluşsa)
- RCREG yazmacını okuyarak gelen veri alınır.
- Hata varsa CREN bitini temizleyiniz.
- Eğer kesme kullanılırsa GIE ve PEIE bitleri (INTCON) 1 yapılır.

3.4.3 SERİ İLETİŞİM STRANDARTLARI

Mikroişlemci ile çalışan sistemlerde, bilgi seri şekilde gönderilmek istenildiğinde uyulması gereken bazı standartlar vardır. Bunlar senkron seri ve asenkron seri iletişim standartlarıdır. Her iki yöntemde de bilginin seri olarak karşı tarafa ulaşması sağlanır.

3.4.3.1 SENKRON SERİ İLETİŞİM STANDARTI

Senkron seri iletişimde, seri verinin 1 ve 0'larının doğru sıralanıp sıralanmadığını araştırmanın yanı sıra verinin ilk bitini de belirlemeye ihtiyaç vardır. Alıcı ve verici arayüz ünitelerinin başlangıç senkronizasyonu ile bu işlem yapılır. Senkronizasyondan sonra alıcı, n bitlik bir sözcüğü oluşturmak için n tane darbe alır. Güvenirliliği devam ettirmek için alıcı ve verici arayüz ünitelerinin, iletim süresi boyunca senkronizayon içinde olması gerekir. Alıcı saatindeki gürültü ve kaymadan dolayı senkronisazyonun kaybolmasını engellemek için başlangıçtaki senkronizasyon yeterli değildir. Bunun için gönderici ve alıcı aynı saat sinyali ile çalışırlar. Genellikle bu saat sinyali, gönderici uçtaki saat generatöründen alınır. Bu iletişim moduna senkron seri iletişim adı verilir.

Senkron seri iletişimin başlangıcında verici bir seri darbe gönderir. Bu gönderilen darbelerin ilk birkaç biti önceden belirlenmiş olan formattadır. Bunlara match karakter veya sync pattern adı verilir. Bu match karekter, alıcıda belirli bir registerda depolanır. Alınan diğer sinyaller, match karekterle uyuşuyorsa alıcı match karekteri kendi alıcı registerine yerleştirir ve devamına n tane bit sayar. Seri veri transferinin senkron modunda peşpeşe iki sözcüğün arasında bekleme yoktur[4].

Veri genellikle darbe dizisi şeklinde veya sözcük sayısı önceden belirlenmiş darbe blokları halinde gönderilir. Verinin iletilmediği zaman aralığı genellikle null veya fill karekterlerle doldurulur. Bu karekterler, herhangi bir bilgiyi içermezler. Fakat alıcının zamanlama darbelerinin akışını sağlarlar.

3.4.3.2 ASENKRON SERİ İLETİŞİM STANDARTI

Asenkron iletişimde, veri demetinin özel bitlere ayrılmış hali olan word'ün başladığını alıcıya işaret eden bir start bitinin arkasındadır. Diğer bitlerle karışıklığı önlemek için, start biti iletimdeki diğer herhangi bir bitin boyutunu ikiler. Wordün sonu alıcıya wordün sona erdiğini söyleyen bir stop bitiyledir. Bu diğer start biti olabilir. Veri bütünlüğünü sağlamak için parity(eşlik) biti stop biti ile verinin son biti arasına eklenir. Parity biti gönderilen veri ile gelen verinin bit diziminin aynılığını ve bit sayısının doğruluğunu kontrol eder.

Veri iletiminin bu şeklinde, alıcı ve verici arayüz modülleri için iki ayrı saat sinyali kullanılır. Bu iki saat sinyalinin iletiminin yapıldığı süre boyunca birbiri ile çok iyi senkronizasyon içinde olması gerekir. Bu metod, alıcı modülüne bazı karmaşıklıklar getirmesine rağmen avantajı sadece iki hatta ihtiyaç duymasıdır. Eğer ortak topraklama mümkün ise tek hat yeterlidir. Çok sayıda iletişim hattı olduğunda bu metod uygundur. Örneğin telefon,telex gibi. Asenkron modunda start biti alıcı için başlangıç zamanlama sinyali olarak çalışır. Bu bit yardımı ile alıcı saatini senkronize eder. Bu senkronizasyon biti alındıktan sonra önceden belirtilmiş iletilmekte olan verinin karekterleri, veri olarak kaydedilir. Eğer alıcı ve verici saat frekansları tam uyumlu değilse alıcı shift registerına son bit yükleninceye kadar geçen zaman içinde senkronizasyonda küçük bir kayıp olabilir. Buna bağlı olarak hata olasılığını yoketmek için her sözcüğün sonunda stop bitleri bulunur. Bu bitler, doğru okumayı sağlayabilecek ölçüde alıcı saatinin bozulup bozulmadığını anlayabilmek için kontrol amacıyla kullanılırlar. Eğer bozulmuşsa çerçeve hata sinyali üretilir.

3.4.3.2.1 SİSTEM DESTEĞİ

Asenkron formatlarda veri iletimine yönelik programlamanın çok zor olmadığı söylenebilir. Bir çok PC' de ve mikrokontrolörde bulunabilen UART(Universal Asynchorous Receiver/Transmitter) adlı devre elemanı seri-veri gönderim ve alımına ilişkin ayrıntıları halleder.

PC' lerde işletim sistemi ve programlama dilleri UART'ın mimarisini detaylarıyla bilmeyi gerektirmeksizin seri linklerin programlanmasına imkan vermektedir. Bir linki açmak için, uygulama bir veri hızıyla birlikte diğer ayarları belirler ve istenilen portu devreye sokar. Gönderilecek byte uygulama tarafından seçilen portun tamponuna yazılır. Format belirlemesi, Start ve Stop bitinin eklenmesi, parite bitinin gerekip gerekmediğine ilişkin ayrıntılar ve nihayet gönderim, UART tarafından yapılır. Benzer şekilde, gelen veri portun tamponunda tutulur. UART bir kesme (interrupt) tetikleyerek CPU' ya, dolayısıyla uygulamaya, bir veri geldiğini bildirir.

Bazı mikrokontrolörlerde bir UART yoktur. Oysa birden fazla UART da ihtiyaç olunabilir. Böyle durumlarda iki seçenek vardır: İlki UART' ın yerini tutacak bir program yazmaktır. İkincisi ise

harici bir UART eklemektir. İkinciye örnek Parallax' ın Basic Stamp' ıdır. Yongada bir UART programı bulunur.

Hem senkron hem de asenkron iletimleri destekleyen cihazlar da mümkündür. Bunlara bir örnek olarak USART(Universal SynchorousAsynchorous Receiver/Transmitter) gösterilebilir.

3.4.3.2.2 BYTE İLETİMİ

Seri linkin kullanımı, programlaması ya da tasarımı için bir byte'ın nasıl iletildiğini bilmek bir zorunluluk değildir. Ancak, protokol seçimi ve arabirim konusunda doğabilecek sorunlarla başedebilmek için bir parça bilgiden zarar gelmez.

3.4.3.2.3 VERİ KAYBININ ENGELLENMESİ

Seri linklerdeki bilgisayarlar veri almayı beklemek dışında işler de yaparlar. Örneğin, bir veri-kabul (data-acquisition) birimi, bir diğer düğüm isteyene kadar veri toplayıp kaydedebilir. Yahut, bir kontrolör link üzerinden bilgi göndererek ya da talimat alarak kontrol ve takip yapabilir.

Düğümün, alıcı meşgulken iletimde bulunmak istemesi pekala mümkündür. Bir linkte her alıcının gönderilen veriyi görebilmesi ve verinin de hatasız iletilmesi gerekir.

Bunu sağlamanın çeşitli yolları vardır. Elsıkışma, tamponlama, alınan veriyi algılamak üzere kesme ya da yoklamaya (polling) başvurulması, hata kontrolü ve tasdik bu yollar arasındadır. Bir link bunların birini ya da birkaçını kullanabilir.

3.4.3.2.4 EL SIKIŞMA

El sıkışma sinyalleriyle vericiler veri göndermeye, alıcılarsa almaya hazır olduklarını belirtirler. RS-232 ve RS-485 linkler standart protokolleri izlerler[5]. Ancak sinyallerin izlediği protokoller değişebilmektedir.

Yaygın donanım el sıkışma biçimlerinden birinde, alıcı alıma hazır olduğunda hattı yükseğe getirir. Verici ise gönderime başlamadan önce bu sinyalin gelmesini bekler. Alıcı, hattı her hangi bir anda, hatta blok veri alımının tam ortasında, düşüğe getirebilir. Verici bunu algılamalıdır ki, iletimi bitirmek için hattın yeniden yükseğe geçmesini beklemek üzere o anda gönderimi durdurabilsin. Bazı linkler bu işi yazılım el sıkışmasıyla yaparlar. Alıcı veri almaya hazır olduğu yolunda bir kod verici ise gönderimi durduracağı şeklinde başka bir kod yollar.

3.4.3.2.5 TAMPONLAR

Alıcılara yollanan verinin kaybolmasını engellemede kullanılan yöntemlerden biridir. Verici tarafında da işe yarayabilmektedirler. Özellikle uygulamaların link üzerinden yollayacakları bilgileri depolayarak daha verimli çalışmalarını sağlarlar. Tamponlar donanımda, yazılımda ya da her ikisinde de olabilirler. En eskiler hariç bütün PC'lerde UART üzerinde 16-byte donanım tamponları bulunur. Bu, yazılım tarafından okununcaya kadar UART'ın 16-byte veriyi depolayabileceğini anlatır. Gönderimde ise, 16-byte veri tamponda tutulurken UART protokole bağlı olarak, byteların tek tek bitler şeklindeki gönderim ayrıntılarıyla uğraşmasına imkan verir.

Donanım tamponlarının yeterince geniş olmadığı durumlarda, yazılım tamponlarına başvurulabilir. Boyutları programlanabilmektedir. Sistemin bellek kapasitesi kadar geniş tutulabilirler. Donanım ve yazılım tamponları arasındaki transfer portun yazılım sürücüsü tarafından yapılır.

Mikrokontrolörlerdeki tamponlar oldukça küçüktür. Hatta bazı yongalarda donanım tamponu bulunmayabilir. Donanım tamponları küçüldükçe veri kaybının önlenmesi için, diğer tekniklere başvurma gereği artar.

3.4.3.2.6 YOKLAMA VE KESMELER

Veri alımı/gönderimi, el sıkışma sinyallerindeki değişmeler, hata mesajlarının iletimi bir seri portta meydana gelen olaylardandır. Bunlara yol açan veya bunları algılayan bir uygulamanın önünde iki yöntem vardır:

Birincisi, olayın gerçekleşmesi halinde programın bir rutine dallanmasını sağlamaktır. Uygulama, porttaki oluşuma derhal ve otomatik olarak tepki verir. Kontrolle vakit kaybı söz konusu olmaz. Yani, tek kontrol olayın olmadığına ilişkin bilgiyi almaya yöneliktir.

Bu tip bir programlama olay-uyarımlıdır. Program dışsal olaya bağlı olarak herhangi bir anda kesilebilir. Bu durumda rutin devreye girer.

İkinci yöntemde, bir olay meydana gelip gelmediğini öğrenmek için port periyodik yoklanır. Buna, prosedür-programlama adı verilir. Uygulama, bir kayba yol açmamak için portu uygun aralıklarla yoklamak durumundadır. Yoklama sıklığı tamponun büyüklüğüne, beklenen veri miktarına ve verilecek tepkinin aciliyetine bağlıdır. 16-byte tamponlu bir cihaz, portu saniyede bir yokluyorsa,

saniyede 16-byte'ten veri alamayacak demektir. Yoksa, taşma meydana gelecek ve veriler kaybolacaktır.

Yoklama (polling) genellikle, bir anda yapılması gereken yoğun veri transferlerinde ya da, bilgisayarın gönderdiği veriye acil cevap gerektiği durumlarda uygundur. Yoklamalı bir arabirimde donanım kesmelerine gerek olmaz. Dolayısıyla bu tür bir program, kesme hattı olmayan bir programda çalıştırılabilir. Çoğu yoklamalı programlar porttan okuma aralıklarını belirlemede system zamanlayıcısına başvururlar.

3.4.4 RS-232 DALGA FORMATI

RS232 iletişimi asenkron özelliğindedir. Yani veri ile birlikte bir saat sinyali gönderilmez.

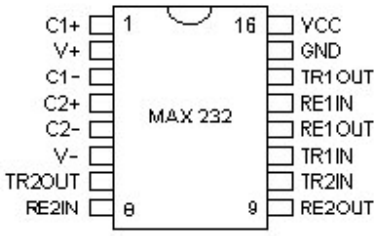
8N1 8 veri biti, No Parity ve 1 Stop bitini ifade etmektedir. RS232 hattı boşta iken Logic 1 durumundadır. İletim Logic 0 olan bir Start biti ile başlar ve her bit hattan gönderilir. İlk önce LSB (Least Significant Bit) biti gönderilir. Son olarak Stop Biti (Logic 1) gönderilerek iletim tamamlanır. Şekil – 2.4'de Stop Bitinden sonraki bit de Logic 0 olarak gösterilmiştir. Bu, başka bir kelimenin devam ettiği anlamına gelir ve bu bir Start Bitidir. Eğer başka gelen veri yoksa hat Logic 1 seviyesinde kalır. Veri hattı tüm bir kelimeyi gönderecek kadar uzun bir süre Logic 0 seviyesinde tutulursa bu kesme sinyali anlamına gelir. Bundan dolayı hat boş duruma (Logic 1) getirilmez ise alıcı bunu kesme sinyali olarak algılar. Bu şekilde gönderilen veri *"çerçevelenmiş"* (framed) özelliğine sahiptir. Yani veri Start Biti ile Stop Biti arasında *çerçevelenmiştir.*

RS232 standardında +3 volt ile +25 volt arası Boşluk (Space – Logic 0) belirtir; -3 volt ile -25 volt arası İşaret (Mark–Logic 1) belirtir. Bu voltlar arasındaki bir değer (-3 ile +3 arası) tanımsızdır. Bundan dolayı sinyal RS232 Seviye Dönüştürücü'ye uygulanmaktadır. Şekil 2.4' deki dalga şekli RS-232 port üzerindeki Alıcı ve Verici hatlarına uygulanmaktadır. Bu hatlar seri veri taşırlar. RS-232 port üzerinde bazı paralel hatlar da vardır. Bu hatlar (RTS, CTS, DCD, DSR, DTR, RTS ve RI) da RS-232 Logic Seviyesindedirler.

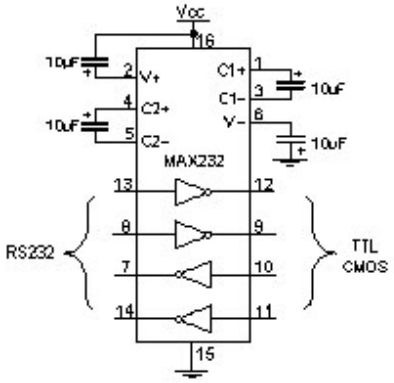
3.4.5 RS-232 SEVİYE DÖNÜŞTÜRÜCÜLERİ

Kullandığımız dijital aygıtların neredeyse tümü TTL veya CMOS logic seviyelerine ihtiyaç duyarlar. Bundan dolayı bir aygıtı RS232 portuna bağlamadan önce yapılacak ilk iş RS-232 seviyelerini 0 ve 5 volta dönüştürmektir. Bölüm 3.4.2'de de bahsedildiği gibi bu işlem RS-232 Seviye Dönüştürücüleri tarafından yapılmaktadır[6].

1488 RS-232 Driver ve 1489 RS-232 Receiver en çok kullanılan RS-232 Seviye Dönüştürücüleridir. Her paket tek tip 4 çevirici içermektedir, Sürücü (Driver) veya Alıcı (Receiver). Sürücü iki besleme hattını gerektirir, +7.5'ten +15'e ve –7.5'ten –15'e. Tahmin edilebileceği gibi yalnızca +5V bir kaynağın olduğu durumlarda birçok sorun ortaya çıkar. Ancak bu devrelerin avantajı ucuz olmalarıdır.



ŞEKİL – 2.6: Max 232 Pin Özellikleri



ŞEKİL – 2.7: Max 232 İç Yapısı

Diğer bir aygıt da 5V tek bir kaynaktan +10V ve –10V üreten MAX-232'dir[7]. Bu entegre aynı zamanda Şekil – 2.7'de görüldüğü gibi iki alıcı ve iki verici içermektedir. Yalnızca alıcı ve verici veri hatlarının kullanılacağı durumlar için oldukça kullanışlıdır. Biri alıcı hattı için biri de verici hattı için olmak üzere iki ayrı chip kullanmak zorunda kalınmaz.

BÖLÜM – 4 : ASSEMBLER PROGRAMLAMA DİLİ

4.1 ASSEMBLER DİLİNDE PIC PROGRAMLAMA

Bir mikrokontroller olan PIC serisi farklı çeşitlerle programlanabilir. Bunlar; Assembler dili, PIC C, Mikrobasic gibi farklı programlama dilleridir. Uygulamamızda Assembler programlama dili kullanılmıştır.

Assembler programa dilinde yalnız 35 komut bulunmakta ve yapılan programımız da bu komutlar ile yapılmıştır. Programımızda genel hatlarıyla sorgulama komutları, yani start, stop ve reset butonlarının sorgulanması ve bu algılamayla ana rutini devam etmesi sağlandı. Ayrıca uygulama programımızda esas önemli olan program döngümüz gecikme rutinimizdir. Çünkü yapılan uygulama kronometre temelli bir uygulama olduğundan gerçek zamanda çalışması önemlidir[8].

4.2 PROGRAMDA GECİKME RUTİNİ

Mikrokontrolör sistemi çok hızlı çalışmaktadır. Bir çok dış dünya ile olan bağlantılarda, örneğin bir LED'i yakıp söndürürken yazılımımıza gecikme rutinleri koymamız gerekir. Bu rutinler genellikle bir kaç yüz milisaniye ve bir kaç saniye arasında gecikme verebilirler. Yazılımda gecikme elde etmenin en kolay yolu bir döngü kurmak ve bu döngü içerisinde beklemektir. Bu metod çok hassas bir gecikme vermese de bir çok uygulamalar için yeterli olabilir.

Assemble programa dili ile gecikme rutinini döngü ile veya delay komutları ile yapılabilir. Uygulama devremizde delay komutlar ile gecikme rutini sağlanmıştır. Bu rutini sağlayan *delay.h* ve *delay.c* isimli iki tane dosya bulunmaktadır. Bu dosyaları kullanarak programımızda istediğimiz gecikmeyi yapabiliriz. Programın başına *#include <delay.c>* komutu ilave edilir. Milisaniye gecikme için *DelayMs(x)* fonksiyonu, ve mikrosaniye gecikme için ise *DelayUs(x)* fonksiyonu kullanılır. Burada x tanımlama cinsinden olup 255' e kadar değer alabilir. Dolayısıyla, 255 milisaniye ye kadar gecikme hassas ve kolaylıkla yapılabilir. Daha büyük gecikmeler için bu fonksiyonları bir döngü içerisinde kullanmak gerekecektir.

Aşağıdaki örnekte programda 200 milisaniyelik bir gecikme olacaktır:

```
DelayMs{200);
```

Yukarıda açıklanan her iki fonksiyonun da yapmış oldukları gecikme kullanmış olduğumuz saat frekansına bağlıdır. Normal olarak saat frekansımız 4 MHz olarak kabul edilmiş ve fonksiyonlar ona göre ayarlanmışlardır. Eğer mikrokontrolörümüz başka bir frekansta

çalışıyorsa, örneğin 2MHz de çalışıyorsa, yeni frekansımızı programımızın başında şu şekilde belirtebiliriz (Aşağıdaki komutların *#include <delay.c>* komutundan sonra olmaları gerekmektedir):

```
#undef XTAL_FREQ

#define XTAL_FREQ 2MHZ
```

Burada birinci komut kabul edilmiş olan 4MHz frekans tanımını kaldırır ve ikinci komut yeni frekansımız olan 2MHz i tanımlar.

Değişik frekans belirtmenin başka bir metodu daha vardır. Ve bu metod bir öncekine tercih edilebilir. *Make* menüsü ve oradan da *CPP pre-defined symbols* menusu seçilir. *Insert (F10)* tuşuna basarak yeni sembol ilave etmek için yer açılır. Açılan yere mikrokontrolörün saat frekansı yazılır. Örneğin, frekans 2MHz ise aşağıda verileni olduğu gibi yazılması gerekir:

```
-DXTAL FREQ=2MHZ
```

Daha sonra *DONE* tuşuna basılarak bu menüden çıkılır. Bu durumda saat frekansı 2MHz olarak tanımlanmıştır ve gecikme rutinleri yeni frekansı için kullanabilir.

BÖLÜM – 5 : VİSUAL BASİC PROGRAMLAMA DİLİ

1963 yılında Darmouth College'de John G. Kemeny ve Thomas E. Kurtz tarafından Basic dili geliştirilmiştir. Daha sonralarda Microsoft tarafından PC'lerde kullanılmak üzere uyarlanmıştır. Microsoft Qbasic ve Microsoft-Dos Qbasic de dahil olmak üzere çeşitli sürümleri bulunmaktadır. Microsoft ileriki yıllarda Basic dilini geliştirerek Windows ortamına uyarlamış ve geliştirilen bu yeni dile Visual Basic adını vermiştir. Visual Basic, devamlı geliştiği bu süre sonunda; yüksek hızlı uygulamalar, OLE Serverlar, ActiveX kontrolleri ve daha bir çok projeyi geliştirebilecek hale gelmiştir. Visual Basic yapısal bir programlama dili olan Basic dilinden türetilmiş olmasına rağmen olaya bağlı bir programlama dilidir.

Yapısal ya da yordamsal uygulamalar, uygulama kodun hangi kısımlarının çalışacağını ve hangi sırada çalışacağını denetler. Uygulama, kodun ilk satırı ile başlar ve gerektiğinde yordamları çağırarak uygulama boyunca önceden tanımlanmış bir yolu izler.

Olaya bağlı bir uygulamanın çalışması, önceden belirlenmiş bir yolu izlemez. Farklı kod bölümleri olaylara bağlı olarak çalışır. Olaylar, kullanıcının eylemlerinden, sistem yada diğer uygulamalardan gelen iletilerden tetiklenir. Olaya bağlı programlamanın en gerekli bölümü bir uygulamada oluşabilecek olası tüm olaylara yanıt veren kodlar yazmaktır[9].

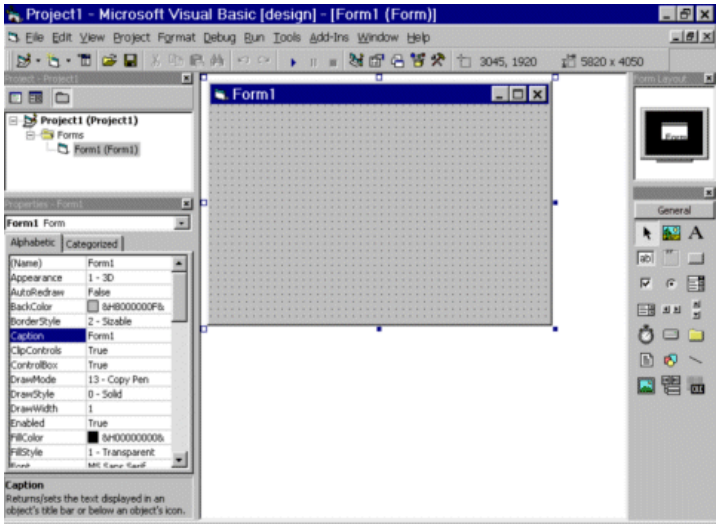
ŞEKİL – 5.1: Visual Basic Program Startı

Visual Basic'i çalıştırdığımızda karşımıza Şekil – 5.2'deki dialog penceresi gelir. Bu dialog penceresinde 3 adet sekme bulunur. **New** sekmesinde oluşturulmak istenilen yeni proje için alternatifler bulunmaktadır. Genellikle **Standart.EXE** seçeneği seçilerek yeni bir projeye başlanır.

Eğer istenirse diğer seçeneklerde kullanılarak ActiveX denetimleri, Dll dosyaları, DHTML sayfalar oluşturulabilir. **Existing** sekmesi ile daha önceden oluşturulmuş projeler sürücü ve klasör seçimi yapılarak açılabilir. **Recent** sekmesi ise üzerinde çalışmış olduğumuz projelerin bir listesini verir ve bunlar arasından istediğimizi seçerek çalıştırabiliriz.

Visual Basic Çalışma Ortamı

Visual Basic'de bir proje başlattığımızda aşağıdaki gibi bir görüntü ile karşılaşırız. Bu görüntüyü elde edebilmeniz için açılışta "Standart EXE" seçeneğini kullanmalıyız.



ŞEKİL – 5.2: Visual Basic Çalışma Ortamı

Proje geliştirme ekranında aşağıdaki araçlar bulunur.

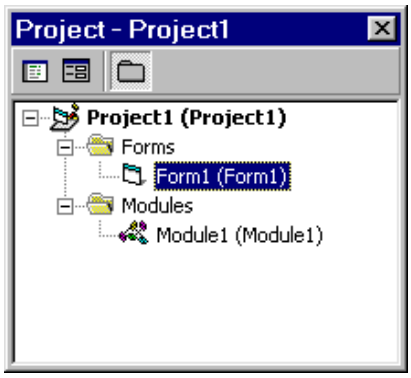
Menü Çubuğu
Araç Çubuğu
Project Explorer
Properties penceresi
Form Layout penceresi
Araç Kutusu
Form Designer

Menü çubuğu Visual Basic penceresinin üst tarafında duran metin satırıdır. Diğer Windows uygulamalarında bulunan menü çubukları ile hemen hemen aynıdır. **File** Menüsünde projeyi açma kaydetme gibi işlemler, **Edit** Menüsünde standart edit işlemleri, **View** Menüsünde programın mevcut olan fakat ekranda açık olmayan pencerelerini açma işlemleri, **Project** Menüsünde form ekleme-

kaldırma gibi proje ile ilgili işlemler, **Format** Menüsünde forma eklenen nesnelerin düzenleme işlemleri, **Debug** Menüsünde program çalışırken programı kontrol etmeye yarayan işlemler bulunur. **Run** Menüsü aracılığı ile programı çalıştırabilir veya durdurabiliriz. **Tolls** Menüsünde Visual Basic'i özelleştirebileceğimiz ve Projeye menü ekleyebileceğimiz seçenekler bulunur. **Add-Ins** Menüsü ise raporlar ve database oluşturma seçeneklerini bulundurur.

ŞEKİL – 5.3: Menü Çubuğu

Menü çubuğunun hemen altında Şekil – 5.3'de görülen araç çubuğu bulunur.

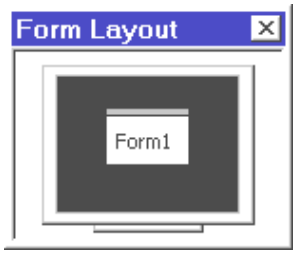


ŞEKİL – 5.4: Project Explorer Penceresi

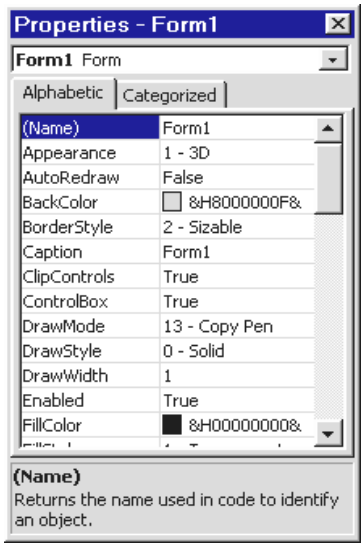
Project Explorer penceresi projenizde bulunan elemanlara erişmenizi sağlar. Bu pencerede formlar, sınıflar ve modüller listelenir. Bu pencerenin araç çubuğunda 3 adet buton bulunur. Project Explorer penceresi içerisinden bir form seçip View Object butonuna tıklayarak formu görüntüleyebilirsiniz. View Code butonu Code Editöre ulaşmanızı sağlar. Toggle Folders butonu ise tüm form ve modülleri kategoriler halinde görebilmemizi sağlar. Project Explorer penceresinde bir öğeye sağ düğme ile tıkladığımızda bir çok işlev sunan bir menü açılır.

Visual Basic'de bütün nesnelerin kendilerine has özellikleri bulunur. Properties penceresi kullanılarak nesnelere ait özellikler değiştirilebilir. Bir nesne seçildikten sonra Properties penceresinde seçili olan nesneye ait özellikler yer alır. Visual Basic'de formlarda birer nesnedir. Şekil 5.7'de görülen pencerede Form1'e ait özellikler listelenmektedir. Properties penceresinin altında aktif olan özelliğe ait bir açıklama görülebilir.

ŞEKİL – 5.5: Kısayol Çubuğu



ŞEKİL – 5.6: Form Layer

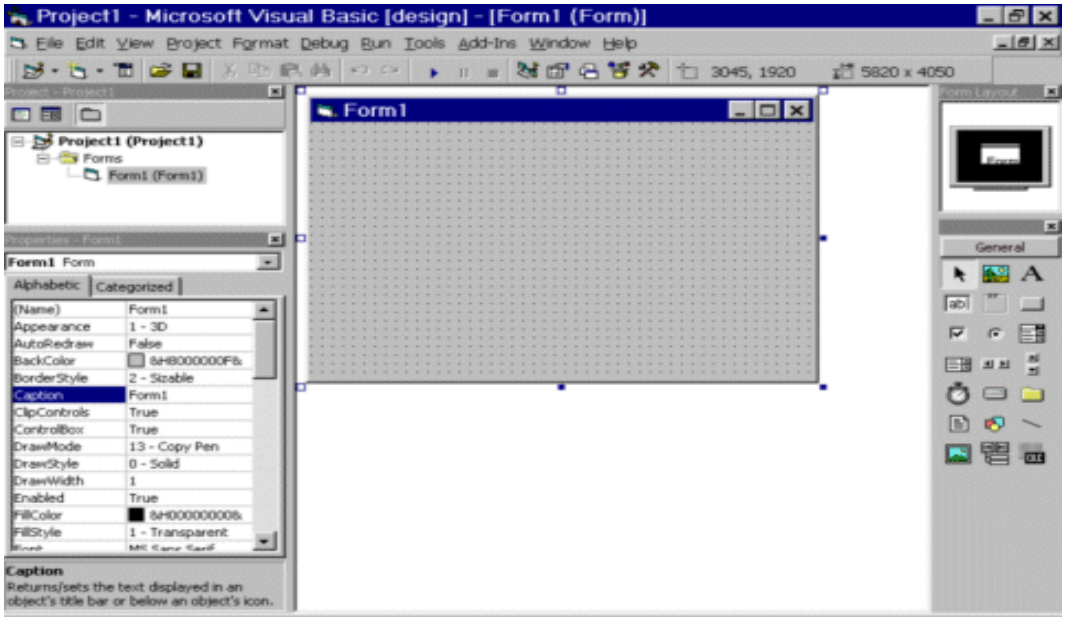


ŞEKİL – 5.7: Properties Penceresi

Form Layout penceresi ile formun çalışma esnasında ekranda nasıl görüleceğini belirlenir (Şekil – 5.6).

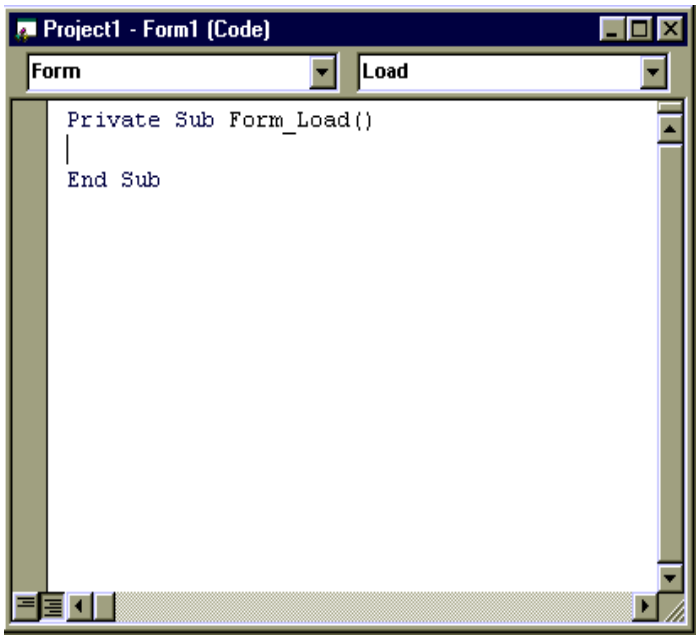
Toolbox uygulama arabirimimizi oluşturmak için gereksinim duyacağımız denetimleri içerir. Toolbox'da bulunan bütün simgeler birer denetimi temsil ederler. Visual Basic, mouse ikonunu bir denetimin üzerine getirdiğimizde bu denetimlerin adını bize verir. Toolbox'a yeni denetimler ekleyebilir ya da varolan denetimleri çıkabiliriz.

Toolbox'da bulunan bir denetimi kullanmak istersek mouse ikonu ile denetim üzerine bir kez tıklarız ve ardından formumuza bu denetimi çizeriz. Bu şekilde istediğimiz denetimi formumuza eklemiş oluruz.



ŞEKİL – 5.8: Form Tasarımcısı

Şekil – 5.8 'deki resimde ekranının ortasında Form Tasarımcısı görülür. Form görünümü, form üzerindeki denetimlerin düzenlendiği yerdir.



ŞEKİL – 5.9: Form Code Penceresi

İyi bir programcının, kullandığı derleyicinin editörünü çok iyi tanıması gerekir. Visual Basic'de bu editöre Code Editör adı verilir. Code Editör'ü bir form üzerine veya nesneye çift tıklayarak veya sağ click yaptıktan sonra View Code seçeneğini kullanarak açabiliriz. Code Editor, Project Explorer penceresinden bir öge seçimi yapıldıktan sonra View Code butonuna tıklayarak da açılabilir.

Bu pencerenin üst tarafında iki adet açılır liste kutusu bulunmaktadır. Sol taraftaki açılan metin kutusunda (Object), form içerisinde bulunan nesnelerin bir listesi bulunur, sağ taraftakinde (Procedure) ise seçili nesneye ait olaylar bulunur, aşağıdaki bölüm ise kodları yazacağımız kısımdır.

Programlamaya Giriş

Program, bilgisayar üzerinde bir takım yararlı işlerin gerçekleşmesini sağlayan bir grup yönergedir. Program, kişilerin adres, telefon gibi bilgilerini tutmamızı sağlayan küçük bir uygulama olabileceği gibi Microsoft Excel, Microsoft Word gibi çok gelişmiş bir uygulamada olabilir[10].

Programlamanın ilk adımı programımızın tam ve kesin olarak neler yapması gerektiğini belirlemektir. Bu işlem oldukça basit gibi görünmektedir. Herşeyin yolunda gitmesi için en başta herşeyin iyice düşünülüp tasarlanması gereklidir. Burada yapılan planlama olayına **algoritma** adı verilir. Önceden yapılan bu çalışma ileride programı geliştirirken kaybedilecek zamanı azaltır.

Program için bir algoritma oluşturulduktan sonra kullanıcı arabirimini tasarlama işi gerçekleştirilir. Kullanıcı arabirimi Visual Basic denetimleri kullanılarak hazırlanır. Kullanıcı arabirimi, uygulamayı kullananların gördüğü bütün menüleri, iletişim kutularını, nesneleri ve resimleri kapsar. Kullanıcı arabirimi ne kadar anlaşılır, sade, görsel ve Windows standartlarına uygun olursa programın kullanımı da bir o kadar kolay ve esnek olur. Kullanıcı arabirimi tasarlandıktan sonra denetimlerin özelliklerini belirler ve verilerin nasıl işleneceği ile ilgili kodları yazılır.

Visual Basic'de proje geliştirmeye başlarken yukarıda bahsedilen işlemleri gerçekleştiririz. Bu işlemlerin neler olduğuna karar verebilmemiz için kendimize bir takım sorular sormalıyız. Bu sorular programın nasıl çalışacağına karar vermemize ve kullanıcı arabirimini tasarlamamıza yardımcı olur.

Visual Basic'de bu programı yazabilmek için ilk önce yeni bir proje başlatmamız gerekir. File-New Project komutunu verdikten sonra gelen pencereden "Standart.EXE" seçeneğini seçip "OK" butonuna basarız. Karşımıza Visual Basic proje geliştirme ortamı gelir. Visual Basic'de her projede en az bir adet Form olmalıdır. Bunu yapabilmek için Code Editör'e bir satırlık bir kod yazmamız yeterli olacaktır. Code Editöre ulaşabilmek için Form üzerine çift tıklanır ve Code Editör karşımıza gelir.

Visual Basic Code Editörde bizim için otamatik olarak yandaki gibi Form'a ait Load prosedürünü açar. Code Editörde **Private Sub Form_Load()** ve **End Sub** bildirileri arasına **Print** yazıp araç çubuğunda start düğmesine bastığımız anda programımız çalışacaktır.

Projeleri Çalıştırmak

Visual Basic'de, hazırladığımız projeleri çalıştırmak için Run-Start komutunu veririz veya Araç çubuğunda bulunan Start butonuna tıklarız. Projeleri çalıştırmak için klavyeden F5 tuşuna basabiliriz. Çalışmakta olan projeyi durdurup tekrar tasarım moduna dönmek istersek Run-End komutunu veririz veya Araç Çubuğunda bulunan End butonuna tıklanır.

Projeleri Kaydetmek

Visual Basic'de hazırladığımız projeleri bilgisayara kaydetmek için File-Save Project komutunu veririz. Gelen dialog penceresinden projemizi kaydetmek istediğimiz klasörü seçip bir isim belirleriz. Visual Basic, forma ait kod ve denetimleri bir dosyaya, proje bileşenlerini ise farklı bir dosyaya kaydeder. Visual Basic form dosyalarına **"frm"** uzantısı, proje dosyalarına ise **"vbp"** uzantısını verir.

Projeleri Derlemek

Hazırladığımız projeleri Visual Basic olmadan çalışabilir hale getirebilmek için derlememiz gerekir. Derleme işlemini başlatabilmek için File-Make Project.EXE komutunu vermemiz gerekir. Bu komutu verdikten sonra karşımıza gelen dialog penceresine oluşturmak istediğimiz Exe dosyasının adını ve oluşturulmasını istediğimiz klasörü belirtmemiz gerekir ve OK butonuna tıklanır.

Visual Basic Denetimlerini Kullanmak

Visual Basic'de kullanıcı arabirimini oluştururken Toolbox'dan faydalanırız. Form üzerinde bulunmasını istediğimiz denetimleri Toolbox üzerinden seçerek form üzerine çizebiliriz. Bu şekilde kullanıcı arabirimini dizayn etmiş oluruz. Toolbox üzerinde bulunan simgeler üzerine imleci getirdiğimizde bize o simgenin ne olduğunu açıklayan mesaj gelecektir. Toolbox üzerinde istediğimiz denetime ait simge üzerine tıkladıktan sonra form üzerinde mouse'un sol tuşunu basılı tutarak denetimimizi çizebiliriz[11].

Visual Basic'de form üzerinde bulunan bir denetimin boyutlarını değiştirmek için, denetim üzerine tıkladıktan sonra etrafında oluşan noktacıklar üzerine mouseumuzu getirir ve sol tuşu basılı tutarak yeni boyutları belirleyebiliriz. Bir nesnenin üzerine mouse imlecini getirip sol tuş basılı iken hareket ettirerek taşıyabiliriz. Herhangi bir nesneyi seçili hale getirip klavyeden **"Delete"** tuşuna basarak form üzerinden silebiliriz.

Not : Birden fazla denetimi seçmek için, **"Shift"** tuşu basılı iken sırası ile nesnelere tıklayarak çok sayıda denetimi seçili hale getirebiliriz.

Özellikler (Properties)

Visual Basic'de bütün nesnelerin karakteristiklerini belirleyen özellikleri bulunur. Özellikler nesnenin tanımını, görünümünü ve hatta tutumunu değiştirebilmemizi sağlar. Bu özellikler tasarım aşamasında değiştirilebileceği gibi çalışma esnasında da değiştirilebilir. Bir nesneye ait özelliği değiştirmek için **Propeties Penceresinin** ekranda olması gerekir. Eğer Propeties Penceresi ekranda değilse View menüsü aracılığı ile ekrana getirilebilir.

Form üzerinde bir nesne seçildikten sonra **Propeties Penceresine** o nesneye ait özelliklerin listesi gelir. Buradan istenilen özellik seçilerek istediğimiz değeri atayabiliriz. Bir başka nesnenin özelliklerini değiştirmek için ise form üzerinde diğer nesneyi seçili hale getirmemiz gerekir. Bu işlemi yaptığımız anda Propeties Penceresinde listelenen özellikler değişir. Bir başka yol ise Propeties Penceresinin üst tarafında bulunan açılan liste kutusundan farklı bir nesne adı seçmektir. Birden fazla nesne seçilerek, o nesnelere ait ortak özellikler değiştirilebilir.

Çalışma anında bir nesnenin özelliklerini değiştirmek için ise Code Editörde kod yazmamız gerekir. Bir nesneye ait kod yazarken önce nesne adı yazılır ardından bir "." işareti konulur ve özellik adı yazılır.

Visual Basic olay güdümlü bir programlama dilidir. Visual Basic'de her nesne için önceden belirlenmiş bir takım olaylar vardır. Windows'a yeni geçiş yapan bir DOS programcısıysanız veya Visual Basic'e henüz yeni başlamış bir programcıysanız öğrenmeniz gereken en önemli konulardan biri olay güdümlü programlardır. DOS ortamında yazılan projeler, uygulama başladıktan sonra bütün olaylardan sorumludur. Visual Basic bu detayları bizim için düşünmüştür. Örneğin bir butona tıkladığınızda mesaj yazdıran bir program yazmak istiyorsunuz; DOS tabanlı bir programlama dilinde, mouse'un konumunu, mouse'un tıklanıp tıklanmadığını takip etmeniz gerekir. Visual Basic'de ise aynı işi yapmak için form üzerinde bulunan Command Button nesnesini **click** olayına bir satır kod yazmamız yeterlidir. Visual Basic bizim için mouse'u izler ve kullanıcı butona tıkladığı anda yazdığınız kodu devreye sokar.

Bir olay prosedürü yazmak için Code Editörde nesneyi ve olayı seçmemiz yeterlidir. Visual Basic bizim için Code Editörde bir prosedür yaratacaktır. Bizim ise tek yapmamız gereken bu olay gerçekleştiğinde meydana gelmesini istediğimiz işlemin kodunu yazmaktır.

Veri Tipleri

Visual Basic'de verilerin geçici olarak depolandıkları yere **değişken** adı verilir. Değişkenlerde program çalışırken kullanıcının girdiği bir bilgi, bir hesaplamanın sonucu, sözcük, sayı, tarih veya özellikleri saklayabiliriz. Değişkenlerin kolay hatırlanır isimlere sahip olmaları gerekir. Visual Basic'de bir değişken kullanılmadan önce tanıtılması gerekir. Değişken tanımlanırken **Dim** bildirisi kullanılır ve değişkenin tipi belirtilir.

Değişkenlerle çalışmayı düşünüyorsak projemizin **Declarations** bölümüne **Option Exeplicit** söz dizimini girmemiz ileride karşılaşabileceğimiz sorunlara bir çözüm oluşturur. Bu söz dizimini kullandığımızda kod içerisinde tanıtılmamış bir değişken kullanırsak Visual Basic bizi uyaracaktır. Tools-Options komutunu vererek gelen pencerede Edit sekmesinden Require Variable Declaration onay kutusunu işaretlersek, Visual Basic başladığımız her projeye Option Explicit ifadesini otomatik olarak yerleştirecektir.

Değişken Tanımlama

Visual Basic'de degişken tanımlanırken dikkat edilmesi gereken kurallar vardır. Tanımlanacak degişkenlerin ilk karakteri mutlaka bir harf ile başlamalıdır. Geri kalan karakterler; harflerden, rakamlardan, alt çizgi karakterinden oluşabilir. Degişken isimlerinde noktalama işaretlerini, matematiksel ve mantıksal ve karşılaştırma operatörlerini kullanamayız. Visual Basic'de

kullanılan anahtar sözcükler, nesne adları, özellikler değişken adı olarak kullanılamaz. Degişken isimleri 255 karakter uzunluğunda olabilir. Degişken tanımlarken Visual Basic'te Dim bildiri deyimini kullanabiliriz. Değişkenin tanomlanırken saklayabileceği veri türünün belirtilmesi hafızada ayrılacak miktarının belirli olmasını sağlar. Eger değişkenlerin tipini belirtmeden bir kullanım yaparsak bu değişkenlerin Variant tipinde olduğu kabul edilir. Bu da hafızada gereksiz yer kaybına sebep olur.

Frame, OptionBox, CheckBox

Frameler çerçeve olarak da geçerler ve formların biraz daha profosyonel görülmelerini sağlarlar. Genellikle seçim kutuları ve onay kutuları ile birlikte kullanılırlar. Seçim kutuları ise kullanıcıya birkaç seçenek sunar. Kullanıcı da bu seçeneklerden sadece bir tanesini seçebilir. Onay kutuları da tıpkı seçim kutularına benzer fakat aynı anda birden fazla onay kutusu işaretlenebilir. Seçim ve onay kutularının en önemli özelliği Value'dur. Bu özellik True ve False değeri atayarak işaret koydurabilir veya işareti kaldırabiliriz.

Visual Basic ile yukarıdaki özellikleri ve EK-1'de verilen program kodlarını kullanarak istenilen özellik ve yapılarda arayüz hazırlanabilir. Tezde uygulamasında seri port haberleşmesine yönelik bir arayüz programı hazırlandı, ayrıca tezde temel olarak seri porttan alınmış veriler veri tabanı oluşturarak belli kriterlere göre sıralama yapılmıştır.

BÖLÜM – 6 : KULLANILAN MICRO DENETLEYİCİ PROGRAMI, BİLGİSAYAR SERİ İLETİŞİM PROGRAMI ve BASKI DEVRE ŞEKİLLERİ

6.1 PIC 16F877 MİKRO DENETLEYİ PROGRAMI

6.1.1 PIC 16F877'ye YÜKLENEN PROGRAMIN ASSEMBLE KAYNAK KODU

```
;Windows hyper Terminali 19200 bps, N,8,1 ile bilgi gönderme bu göndermede el sıkışması YOK olarak
; Program adı : Seri port üzerinden birden çok PIC ile haberleşme.

        LIST    P=16F877,C=140
        #INCLUDE        "P16F877.INC"
        __CONFIG  _PWRTE_ON&_XT_OSC&_LVP_OFF&_WDT_OFF

        TEMP            EQU     7FH
        SAYAC1          EQU     8FH


        ORG             0                       ;program başlangıç satırı
        GOTO            MSTART

MSTART

        BCF             STATUS,RP1
        BCF             STATUS,RP0              ;BANK0 geç.

        MOVLW           0X00
        MOVWF           PORTA
        MOVWF           PORTB
        MOVWF           PORTC
        MOVWF           PORTD
        MOVWF           PORTE

        BCF             STATUS,RP1
        BSF             STATUS,RP0              ;BANK1 bank geç.

        CLRF            TRISA
        CLRF            TRISD
        CLRF            TRISE
        MOVLW           0XFF
        MOVWF           TRISB
        MOVLW           0XFF
        MOVWF           TRISC


;  Baud Values with BRGH = 0
;  ((20000000/9600)/64)-1 = 32
;    movlw     d'6'                          ;  9600 baud @ 4 Mhz Fosc brgh=0
;    movlw     d'25'                         ;  9600 baud @ 4 Mhz Fosc brgh=1
;    movlw     d'207'                        ;  1200 baud @ 4 Mhz Fosc brgh=1
;    movlw     d'2272'                       ;  110 baud @ 4 Mhz Fosc brgh=1
;    movlw     d'103'                        ;  2400 baud @ 4 Mhz Fosc brgh=1
;    movlw     d'52'                         ;  4800 baud @ 4 Mhz Fosc brgh=1
```

```
;    movlw    d'12'                    ;  19200 baud @ 4 Mhz Fosc brgh=1


        MOVLW       D'12'       ;19200 BAUD AT 4 MHZ FOSC
        MOVWF       SPBRG
        MOVLW       B'00100100'
        MOVWF       TXSTA       ;Enable Async Transmıssıon,Brgh=1 Hıgh Speed
        BCF         STATUS,RP1
        BCF         STATUS,RP0  ;BANK0
        MOVLW       B'10010000'
        MOVWF       RCSTA       ;ENABLE ASYNC RECEPTION
        CLRF        TEMP

SETTLE

        DECFSZ      TEMP
        GOTO        SETTLE

MAIN

        CLRF        TEMP
        CALL        RXLOOP
        SUBLW       D'49'       ;klavyeden gönderilen 1'in ascii karşılığı 49'dur.
        BTFSC       STATUS,Z
        CALL        PRN
        GOTO        MAIN

PRN

        MOVF        TEMP,W
        CALL        TABLE
        CALL        TXLOOP
        INCF        TEMP,F
        SUBLWD'62'              ; '>' GREATER SYMB
        BTFSS       STATUS,Z
        GOTO        PRN
        CALL        TAB
        MOVF        PORTB,w
        CALL        TXLOOP
        CALL        ENTER
        CALL        LF
        RETURN

ENTER

        MOVLW       D'13'       ;code for return
        CALL        TXLOOP
        RETURN


TAB

        MOVLW       D'9'        ;code for tab
        CALL    TXLOOP
        RETURN

LF
```

```
        MOVLW   D'10'       ;code for linefeed
        CALL    TXLOOP
        RETURN

TXLOOP

        NOP
        BTFSS   PIR1,TXIF   ; Verinin hazır olup olmadığı kotrol edilir.
        GOTO    TXLOOP
        MOVWF   TXREG       ; giriş verisi W registerine aktarılır.
        RETURN

RXLOOP

        NOP
        BTFSS   PIR1,RCIF
        GOTO    RXLOOP
        MOVF    RCREG,W
        RETURN

TABLE

        ADDWF   PCL,F

        DT

        RETURN

        END
```

6.1.2 PIC 16F877'ye YÜKLENEN PROGRAMIN ÇALIŞMA MANTIĞI

Programın ilk başında fonksiyonlar tanımlanır. Tanımladığımız fonksiyonlar sırası ile Pic entegresinin dosyalarını tanımlamak, programda kullanacağımız atamalara adres tanımlamak, ana program ve alt programları'dır. Fonksiyon tanımından sonra ana programda ilk olarak sayma işlemini ve stop butonuna göre seri iletişim prtotokolünü (USART) hazırlamak gerekir.

Çıkış için kullanılacak port çıkış portu yapılır. Diğer portları kullanılmayacağı için bu portlara bir atama yapılmaz.

Daha sonrasında da sonsuz döngü içerisinde seri iletişimden USART_al(); fonksiyonu ile alınan bilgi (start-stop) komutu ile karşılaştırmalı bir şekilde değerlendirilir.

Burada seri iletişimden aldığımız bilgi "1" ise PIC'in B portunun 0. bitini logic 1 yapar, "3" ise PIC in B portunun 1. bitini logic 1 yapar, "A" ise PIC'in B portunun 2. bitini logic 1 yapar ve "E" ise PIC in B portunun 4. bitini logic 1 yapar.

6.2 BİLGİSAYAR SERİ İLETİŞİM PROGRAMI

Bilgisayar seri iletişim uygulamamızı kullanmak için Visual Basic arayüz programı kullanılmıştır. Uygulamamızda kullanacağımız arayüz programımız, PIC programlanarak tasarlanan elektronik zamanlama devresinden elde edilen sürenin seri porttan Visual Basic'te tasarlanan veri tabanına aktarılarak uygulamamızda kullanılan bilgilere ek olarak bir veri elde etmektir.

Visual Basic programlama dilinde yazılan arayüz programı aşağıda gösterilmektedir.

```
Private Sub al_Click()

    MSComm1.Settings = "9600,N,8,1"
    'rs232 ile ilgili ayarlar
    MSComm1.PortOpen = True
    'port açiliyor
    deger = Asc(MSComm1.Input)
    'pic'ten okunan 8 bitin ascii karsiligi aliniyor
    Text1.Text = deger
    'okunan bilgi text penceresinde görünüyor
    MSComm1.PortOpen = False
    'port kapatiliyor

End Sub
```

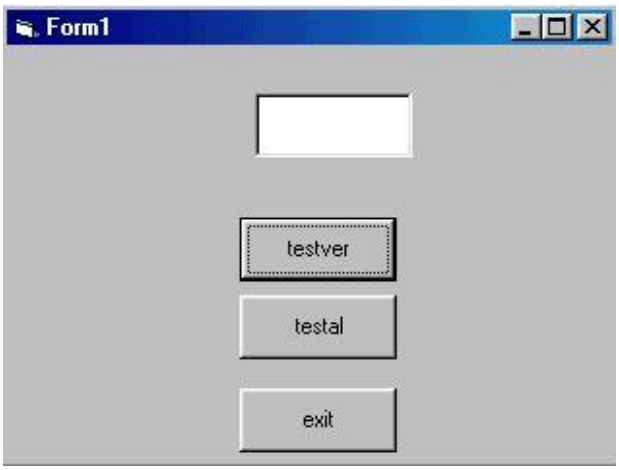
```
Private Sub exit_Click()
End
End Sub
```

```
Private Sub ver_Click()

    MSComm1.Settings = "9600,N,8,1"
    'rs232 ile ilgili ayarlar
    MSComm1.PortOpen = True
    'port açiliyor
    MSComm1.Output = Text1.Text
    'text penceresine yazilan deger pic'e gonderiliyor
    MSComm1.PortOpen = False
    'port kapatiliyor

End Sub
```

6.2.2 SERİ İLETİŞİM PROGRAMININ GENEL GÖRÜNÜMÜ VE ÇALIŞMA MANTIĞI

Bilgisayar ile PIC haberleşmesinde arayüz Visual Basic ile yapıldı. Arayüzün görünümü aşağıda (Şekil – 5.1) görülmektedir. Testver butonu text penceresine yazılan sayıyı veya karekteri

PIC'e göndermek için kullanılır. testalal butonu PIC'ten alınan 8 bitlik bilgiyi text penceresine yazdırır. Exit butonu ise programdan çıkmamızı sağlar.



ŞEKİL – 6.1: Text Penceresi

Arayüzde görünmeyen ve comm port ile ilgili işlemler yapmamızı sağlayan Mscomm kontrolörü çalışma alanında görülür. Bu kontrolörü yüklemek için sırasıyla projects > componenets > microsoft comm control seçilir.

Program ile ilgili kodlar sırasıyla şöyledir . Testver butonu için :

```
Private Sub ver_Click()
   MSComm1.Settings = "9600,N,8,1"
   'rs232 ile ilgili ayarlar
   MSComm1.PortOpen = True
   'port açiliyor
   MSComm1.Output = Text1.Text
   'text penceresine yazilan deger pic'e gonderiliyor
   MSComm1.PortOpen = False
   'port kapatiliyor
   End Sub
```

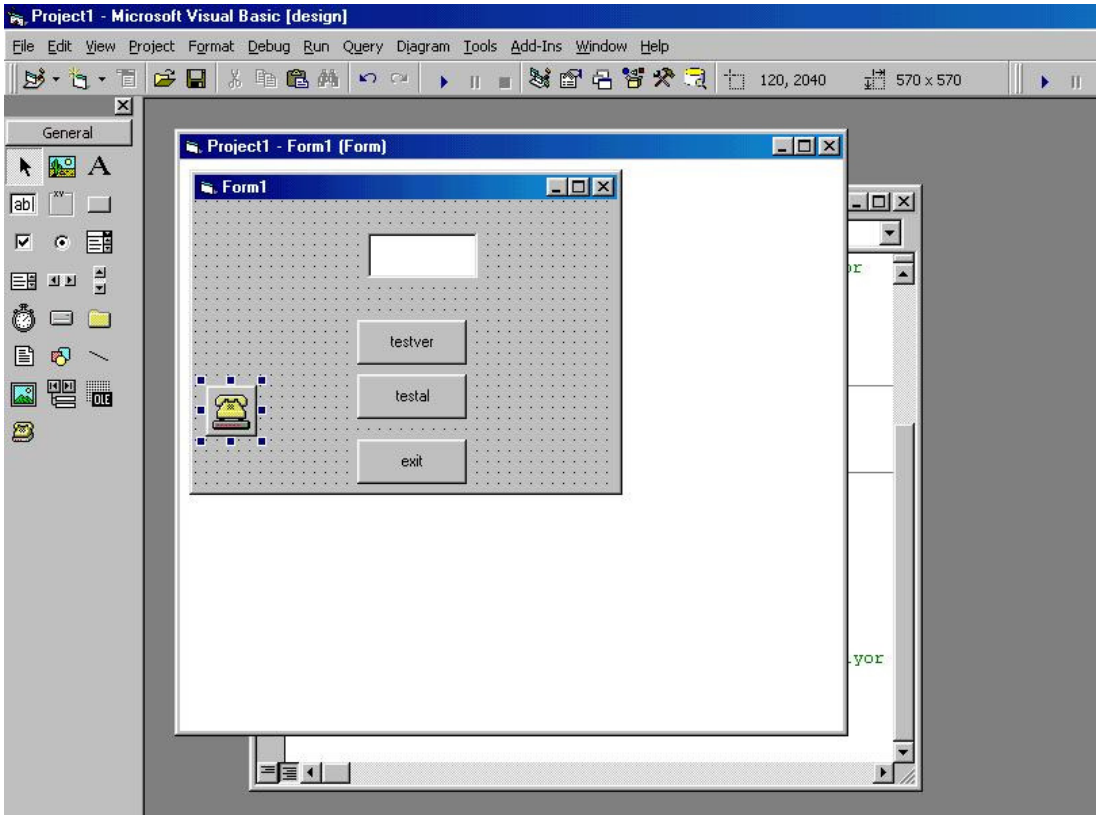
Yorumlardan da anlaşılacağı gibi önce comm port ile ilgili ayarlar yapılmaktadır. Bu ayarlar bilginin 9600 baud (bit per second), parity bitsiz (no parity), 8 bitlik data ve bir stop biti ile gönderildiğini veya alındığını gösteriyor. MSComm1.PortOpen=True deyimi portun açıldığını

MSComm1.Output porta bilgi gönderildiğini (Gönderilen bilgi text1'e yazılan sayı veya karekterdir) ve MSComm1.PortOpen=False portun kapandığını belirtiyor.

Buton testal için yazılan kodlar şunlardır.

```
Private Sub al_Click()

   MSComm1.Settings = "9600,N,8,1"
   'rs232 ile ilgili ayarlar
   MSComm1.PortOpen = True
   'port açiliyor
   deger = Asc(MSComm1.Input)
   'pic'ten okunan 8 bitin ascii karsiligi aliniyor
   Text1.Text = deger
   'okunan bilgi text penceresinde görünüyor
   MSComm1.PortOpen = False
   'port kapatiliyor

End Sub
```

Kaynak kodunda testver butonunda kullanılan kodlara benzer kodlar vardır. Dışardan gelen bilgiyi almak için MSComm1.Input değişkeni kullanılmıştır. Asc fonksiyonu alınan karekterin ascii değerini almak için kullanılmıştır. Değişkenin "dim deger as integer" olarak tanımlanması daha uygun olabilir. Portun MSComm1.PortOpen = False satırı ile kapatılması programın "port zaten açık" hatası vermemesi açısından önemlidir.

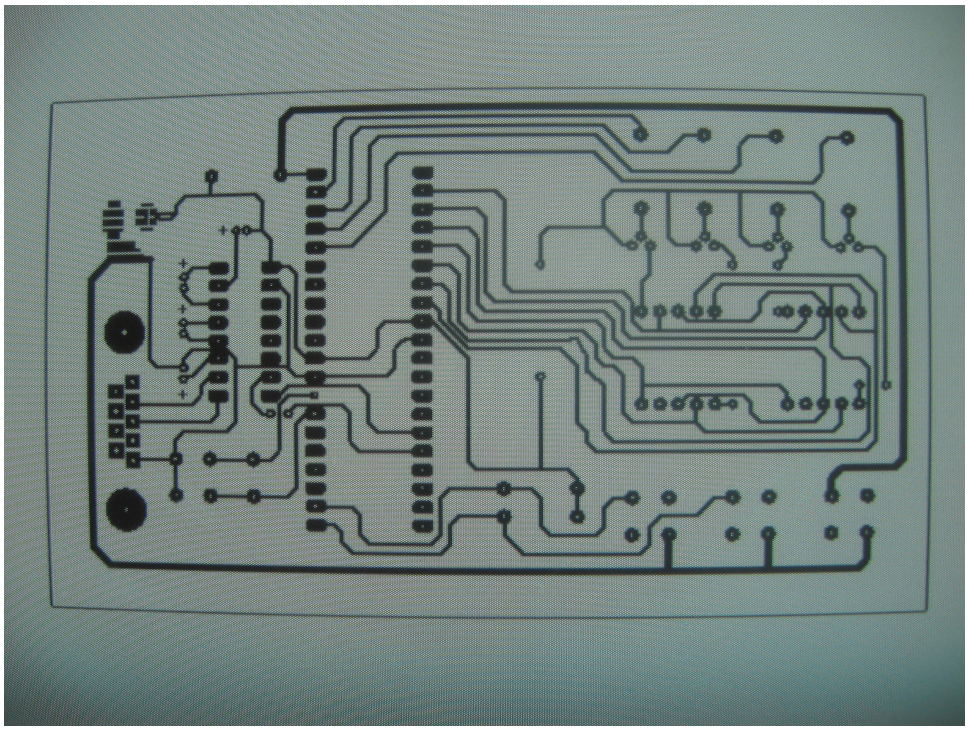


ŞEKİL – 6.2: Seri iletişim Penceresi

Şekil – 5.6'da kurulacak devre görülmektedir. Devrede kullanılan PIC'e daha önce programlama devresi ve programlama yazılımıyla seri porttan gelecek bilgiyi alması için programlanması gerekmektedir

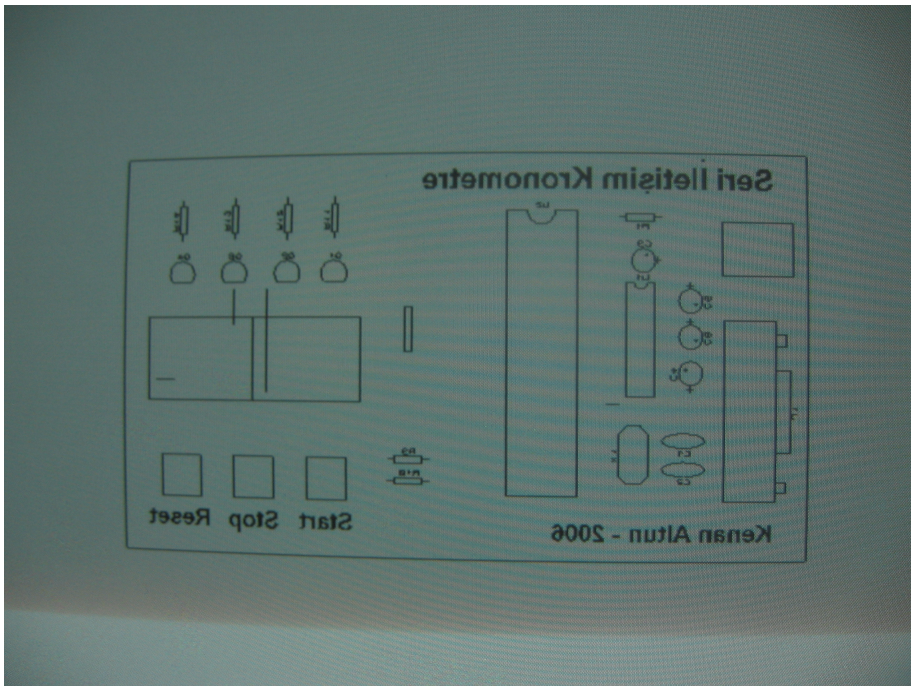
6.3 DEVRENİN BASKI DEVRESİ ve GENEL GÖRÜNÜMÜ

6.3.1 DEVRENİN PCB'si



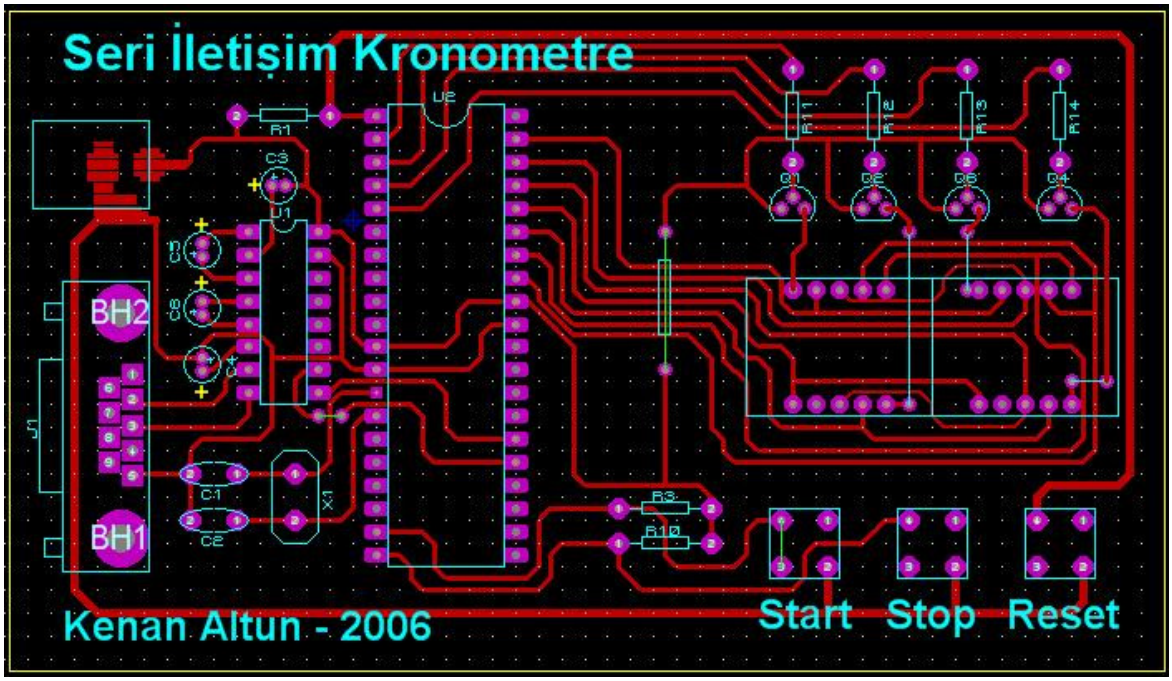
ŞEKİL – 6.3: Devrenin Alt PCB'si

6.3.2 DEVRENİN ÜST YÜZÜ



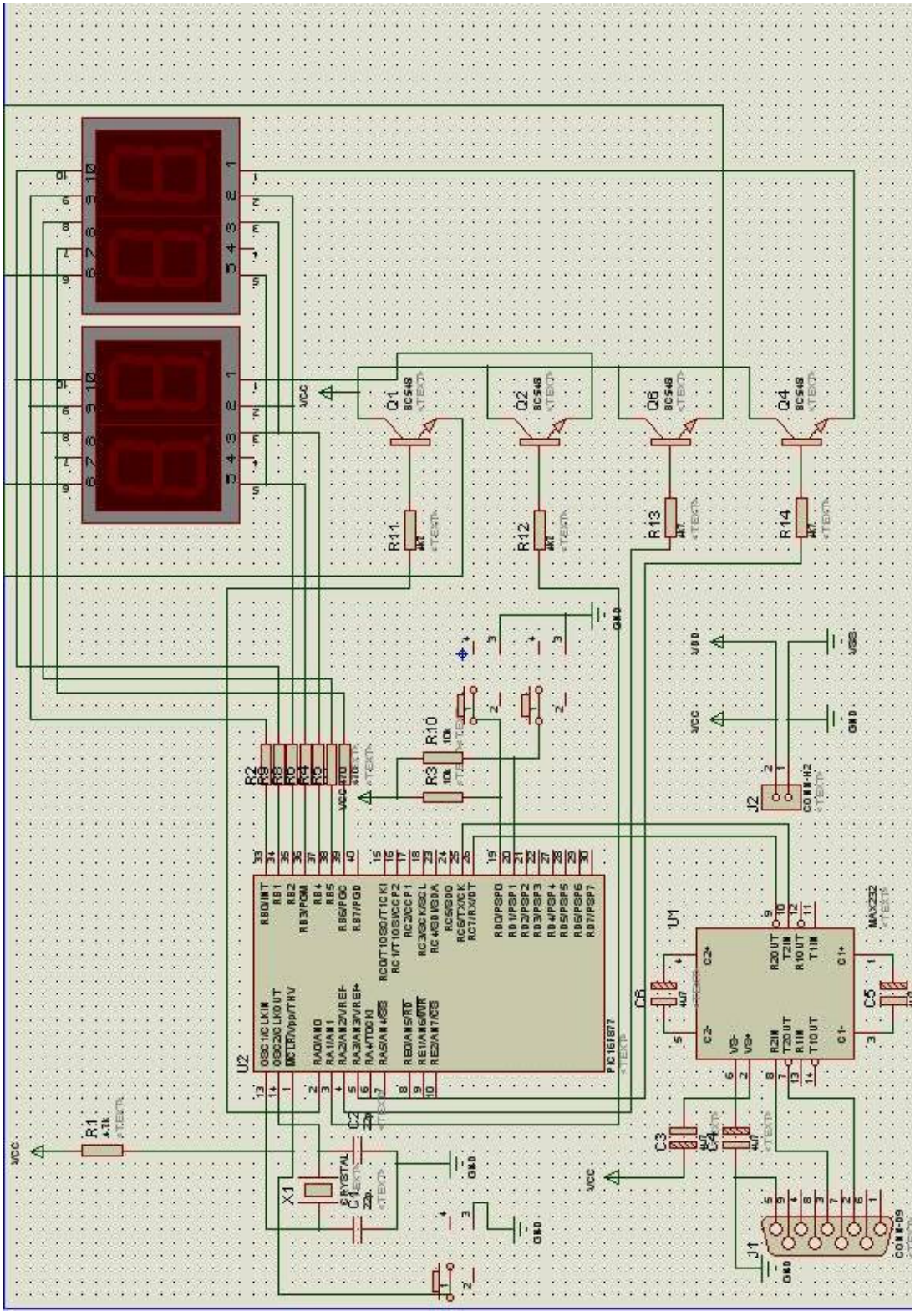
ŞEKİL – 6.4: Devrenin Üst PCB'si

6.3.3 DEVRENİN PROTEUS ARES GÖRÜNTÜSÜ



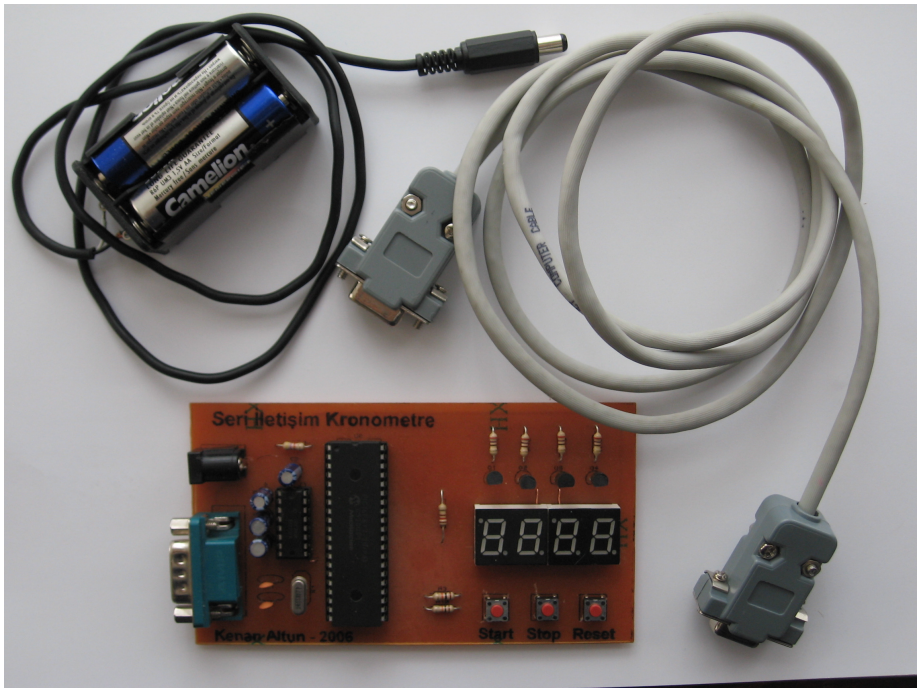
ŞEKİL – 6.5: Devrenin Ares Görüntüsü

6.3.4 DEVRENİN PROTEUS ISIS GÖRÜNTÜSÜ



ŞEKİL – 6.6: Devrenin İsis Görüntüsü

6.3.5 DEVRENİN GENEL GÖRÜNÜMÜ



ŞEKİL – 6.7: Devrenin Genel Görünümü - 1

ŞEKİL – 6.8: Devrenin Genel Görünümü - 2

BÖLÜM – 7 SONUÇLAR

Bu projede PIC16F877 mikrodenetleyicisi programlanarak tasarlanan elektronik zamanlama devresi ile alınan veriler gerçek zamanda LCD displaylere aktarılarak zaman olarak gözlenmlenmiştir. Eş zamanlı olarak aynı veriler seri port protokolü ile ve V.B.'de oluşturulan arayüz yolu ile bilgisayarda gözlemlenmiştir.

Ayrıca V.B.'de kullanılan hesaplama yolu ile elde edilen ham veriler işlenerek veri tabanı oluşturulmuştur.

Sonuç olarak elektronik devre tarafından alınan verilerin elle bilgisayara girilmesi sonucunda oluşacak kişisel hatalar, elektronik devreyle bilgisayarın haberleşmesi sonucunda ortadan kaldırılmıştır.

KAYNAKLAR

[1] Microchips Ships 2 Billionth PicMicro Microcontroller, Boston, Jul1, 2002, N.A.

[2] Recent Progress in Space Based Micro controller Design, Avery. K,Alexander D.

[3] PIC PC iletişim Projeleri *Doç. Dr. Doğan İbrahim* Elektronik Hobi *Güçlü Tugay*

[4] Cyber Elektronik Dergisi Mart – 2005

[5] http://www.beyondlogic.org

[6] http://www.tetraedre.com

[7] http://www.techcentrium.com

[8] http://www.osborne.com

[9] http://www.microchip.com

[10] http://www.maxim-ic.com

[11] http://www.htsoft.com

ÖZGEÇMİŞ

1981 yılında Sivas'ın Şarkışla ilçesinde doğdu. İlk, orta ve Lise öğrenimini 1998 yılında tamamladı. 1999 yılında Cumhuriyet Üniversitesi Elektrik – Elektronik Mühendisliği bölümünü kazandı ve aynı bölümden 2003 yılında mezun oldu. 2004 yılında TEDAŞ Sivas İl Müdürlüğünde Elektrik – Elektronik Mühendisi olarak göreve başladı ve halen bu görevi sürdürmektedir. 2006 yılında Didem Kalender ile evlendi.