C# ile SerialPort işlemleri

C# serial port nesnesi ile usb/seri ile bilgisayarımıza bağladığımız ve veri aktarımı sağlayan cihazlarla iletişim kurabiliriz. seri port nesnesini kullanabilmek için, System IO Ports kütüphanemizi projemize dahil ediyoruz. Seri port nesnesini kullanabilmek için öncelikle bir seri port oluşturuyoruz;

- **PortName** : Seri port adını belirtir. String türü değer alır. Bilgisayarınızda bağlı olan seri portları aygıt yöneticisinden görebilirsiniz. Port adı kesinlikle belirtilmeli.
- **BaudRate** : Seri haberleşme hızını belirtir. int32 türü değer alır. Bir saniyede gönderilebilecek maksimum bit sayısıdır. Alıcı ile aynı değer almak zorundadır. Standartlaşmış olarak 300, 600, 1200, 2400, 4800, 9600, 19200, 38400, 57600, 115200 değerlerini alabilir. Varsayılan değeri 9600 Baud'dur.
- **DataBits** : Gönderilecek her bilgini/karakterin kaç bitten oluşacağını belirtir. int32 türü değer alır. 5 ile 8 arasında değer alabilir. Genelde 7 ve 8 değerini alır. Varsayılan değeri 8'dir.
- **Parity** : Eşlik bitinin sayısını belirtir. Alınan verinin doğruluğunu kontrol etmek için kullanılır. None (0), Odd (1), Even (2), Mark (3), Space (4) değerlerini alabilir. Varsayılan değeri None 'dur.
- **StopBits** : Durdurma bitinin kaç bit olacağını belirtir. One (1), Two(2), OnePointFive(3) değerlerini alabilir. Varsayılan değeri One 'dır.

C# ile SerialPort işlemleri-Open()

```
public partial class main : Form
{
    SerialPort sp = new SerialPort(); // Seri port nesnesi oluşturuyoruz
    sp.PortName = "COM7"; // Kullanacağımız seri port adını seçiyoruz (String)
        sp.BaudRate = 9600;  // Seri haberleşme hızını seçiyoruz (int32)
        sp.DataBits = 8; // göndereceğimiz bilginin kaç bitten oluşacağını
bildiriyoruz (int32).
        sp.Parity = Parity.None; // Eşlik bitidir. Gönderilen verinin
doğruluğunu kontrol etmek için kullanılır.
        sp.StopBits = StopBits.One; // Stop bitinin kaç bit olacağını
belirtir.
        sp.Open();
    }
```

Seri port nesnesini tanımladıktan sonra veri gönderip alabilmek için seri portumuzu açmamız gerekiyor. Bunun için Open() fonksiyonunu çağırıyoruz.

C# ile SerialPort işlemleri-Write()

Seri Port Veri Gönderme –WRITE():
Seri porta veri göndermek için Write() fonksiyonu kullanılır. Gönderilen veri String, Byte dizisi ya da Char dizisi olabilir. String türü veri göndermek için parantezler içerisinde Çift tırnak içerisinde gönderilecek veri yazılır.

```
sp.Write("kocaeli"); // Burada String türü veri
```

Byte ya da Char dizisi göndermek için ise parantezler içerisinde dizi yazılıp Write ile porta gönderilir.

```
byte[] veri = new byte[] { 12, 15, 22, 25 };
/*
* Şimdi veri dizimizin sıfırıncı indisinden son indisine kadar
* olan tüm verileri seri porta yazdıralım.
*/
sp.Write(veri, 0, veri.Length);

char[] karakter = new char[] { 'a', '1', (char)15 };
/*
* Şimdi dizinin tüm elemanlarını seri porta yazalım
*/
sp.Write(karakter, 0, karakter.Length);
```

C# ile SerialPort işlemleri- ReadExisting(), ReadLine(),Read()

Seri Port Veri Okuma :

Seri port sadece açık olduğu zaman veri alabilir. Bir timer yardımıyla sürekli seri port kontrol edilebilir yada seri portun **DataReceived** eventi ile seri port dinlenebilir. Timer kullanarak seri porttan alınan veride kayıp yaşanma ihtimali yüksektir. Bu sebeple **DataReceived** eventi kullanılır. Bunun için öncelikle DataReceived eventini tanımlamamız gerekiyor.

```
sp.DataReceived += new SerialDataReceivedEventHandler(sp_DataReceived);

private void sp_DataReceived(object sender, SerialDataReceivedEventArgs e)
{
string veri = sp.ReadExisting(); yada string veri = sp.ReadLine();
}
```

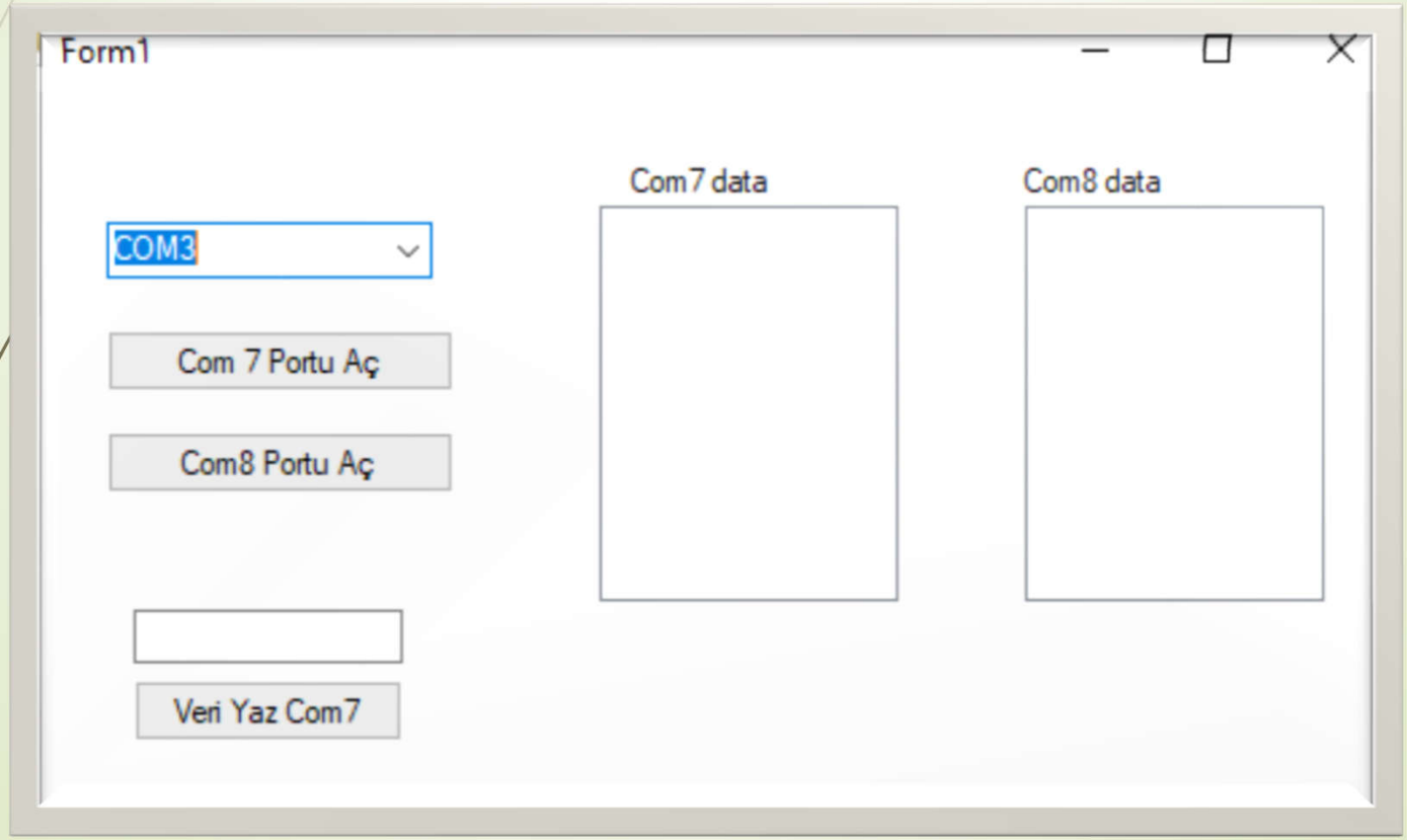
Bu event içerisinde seri porttan gelen bilgileri otomatik olarak okuyup bir değişkene aktarabiliriz ya da textBox, listBox gibi araçlara yazdırabiliriz.

Read() fonksiyonu ile Byte dizisi olarak okuma yapılabilir.

ReadByte() ve ReadChar() fonsiyonları ile bir byte ve bir char veri okuma:

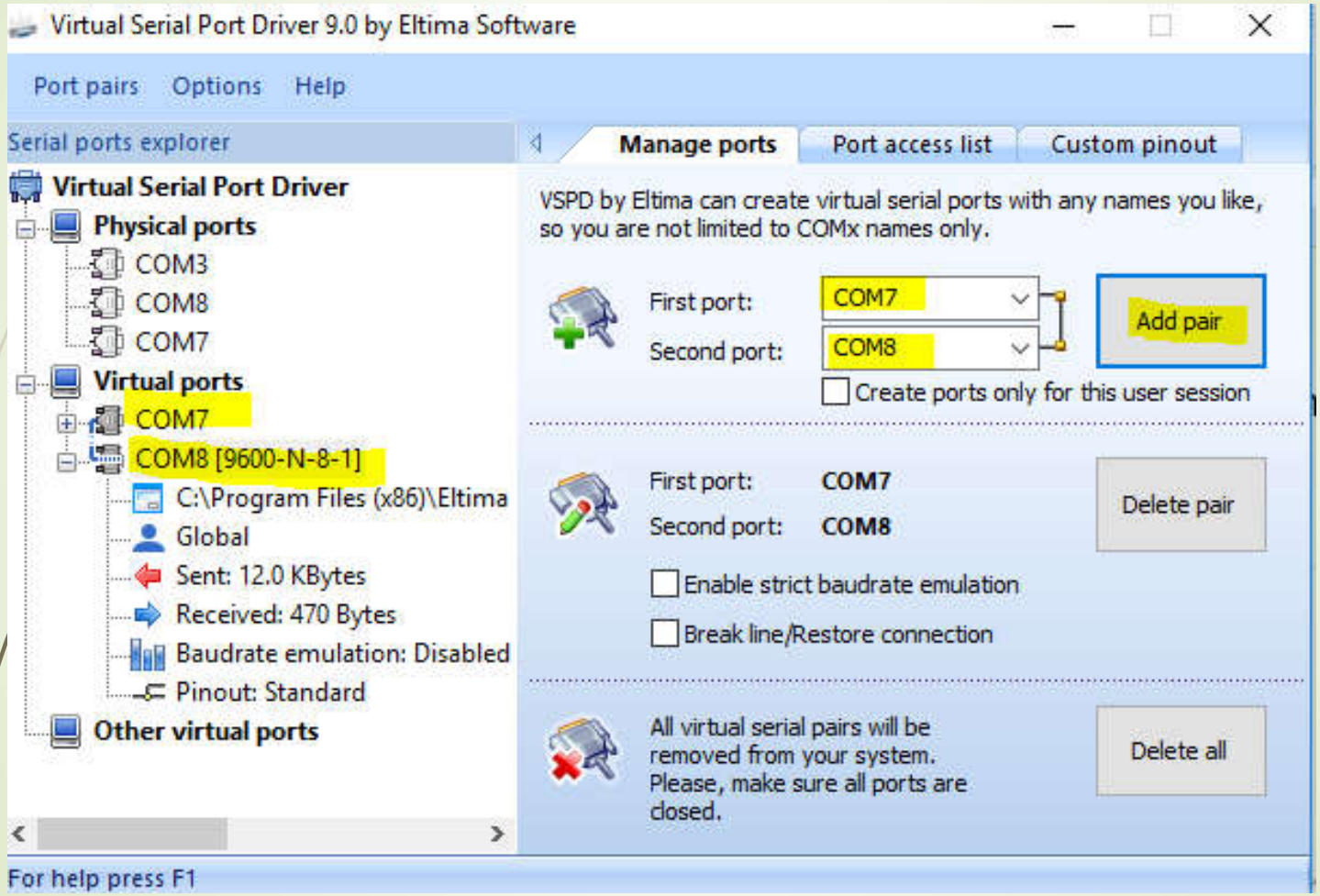
C# ile SerialPort işlemleri-Örnek

Tanımlanacak 2 sanal portu (Virtual Serial Port Driver ve) tanımlayıp ve pair ederek uygulamamızdan her iki porta veri okuyup yazarak gelen yada gönderilen bilgileri listboxlarda göreceğimiz bir uygulama geliştireceğiz.



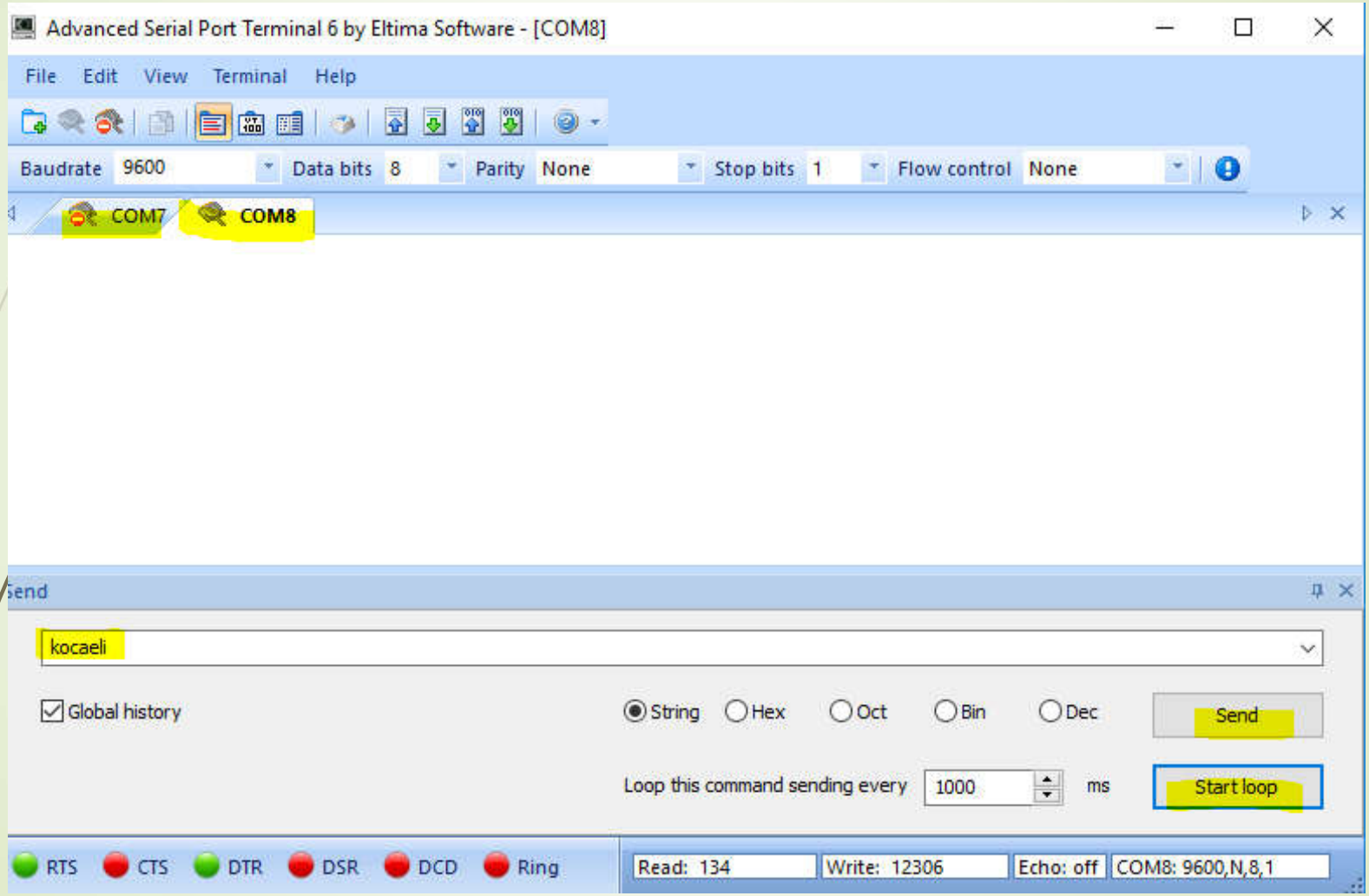
C# ile SerialPort işlemleri-Örnek

Sanal port tanımlama ve monitör…Sanal port tanımlama



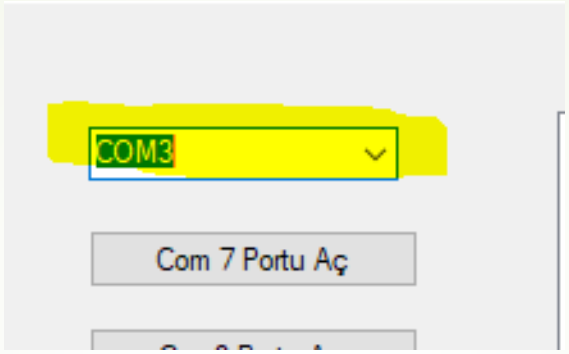
C# ile SerialPort işlemleri-Örnek

Portları görüntülemek, test veri göndermek için Serial Port Terminal uygulaması.



C# ile SerialPort işlemleri-Örnek

Port listesini *GetPortNames* ile çekip kullanıcıya seçme imkanı verilebilir.




```
private void Form1_Load(object sender, EventArgs e)
        {
            //Listbox'a yazarken thread hatası vermemesi için
            Control.CheckForIllegalCrossThreadCalls = false;
             comboBox1.DataSource = SerialPort.GetPortNames();
        }
```

C# ile SerialPort işlemleri-Örnek

COM7 Port kodu

```
private void button1_Click(object sender, EventArgs e)
        {
            if (acikKapali == false)
            {
                try
                {
                    //serialPort1.PortName = comboBox1.SelectedItem.ToString();
                    serialPort1.PortName = "COM7";
                    serialPort1.Open();
                    button1.BackColor = Color.LightGreen;
                    button1.Text = "Com 7 Port'u kapat";
                    acikKapali = true;
                }
                catch (Exception ee)
                {
                    MessageBox.Show(ee.Message);
                }
            }
            else
            {
                serialPort1.Close();
                button1.Text = "Com7 Port'u aç";
                button1.BackColor = Color.LightPink;
                acikKapali = false;
            }

        }
```

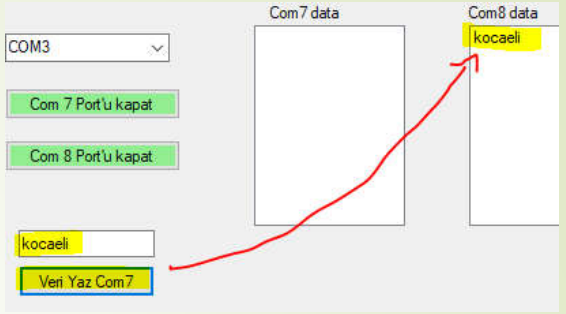
C# ile SerialPort işlemleri-Örnek

COM8 Port kodu

```
private void button4_Click(object sender, EventArgs e)
        {
            if (acikKapali2 == false)
            {
                try
                {
                    // serialPort2.PortName = comboBox1.SelectedItem.ToString();
                    serialPort2.PortName = "COM8";
                    serialPort2.Open();
                    button4.BackColor = Color.LightGreen;
                    button4.Text = "Com 8 Port'u kapat";
                    acikKapali2 = true;
                }
                catch (Exception ee)
                {
                    MessageBox.Show(ee.Message);
                }
            }
            else
            {
                serialPort2.Close();
                button4.Text = "Com 8 Port'u aç";
                button4.BackColor = Color.LightPink;
                acikKapali2 = false;
            }
        }
```

C# ile SerialPort işlemleri-Örnek

COM7 portuna veri göndermek. COM7den COM8'e gideceği için çıktı COM8 listbox'ta gözükecektir.

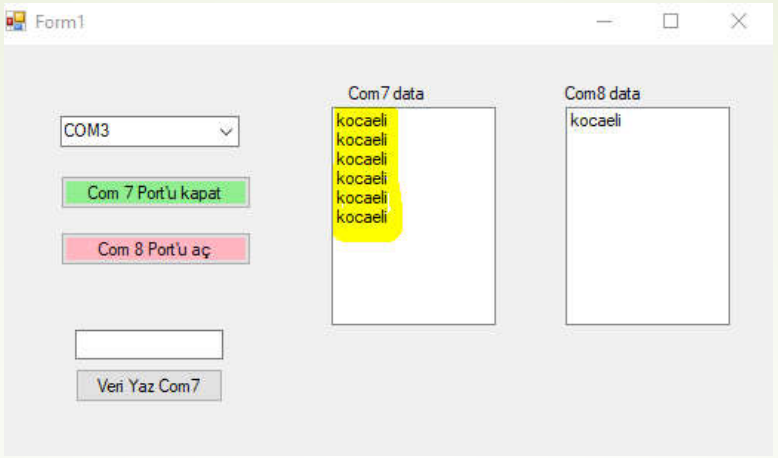



```
private void button3_Click(object sender, EventArgs e)
        {
            try
            {
               serialPort1.Write(textBox1.Text+"\r\n");
            }
            catch(Exception ee)
            { MessageBox.Show(ee.Message); }
        }


private void serialPort1_DataReceived(object sender, SerialDataReceivedEventArgs e)
        {
            Debug.Print("Com7'ye veri geliyor");
            gelenVeri = serialPort1.ReadExisting();
            listBox1.Items.Add(gelenVeri);
        }
```

C# ile SerialPort işlemleri-Örnek

Portlara veri geldikçe DataReceived metodu çalışır




```
private void serialPort2_DataReceived(object sender, SerialDataReceivedEventArgs e)
        {
            Debug.Print("Com8'e veri geliyor");
            gelenVeri2 = serialPort2.ReadExisting();
            listBox2.Items.Add(gelenVeri2);
        }


    private void serialPort1_DataReceived(object sender, SerialDataReceivedEventArgs
e)
        {
            Debug.Print("Com7'ye veri geliyor");
            gelenVeri = serialPort1.ReadExisting();
            listBox1.Items.Add(gelenVeri);
        }
```