آموزش پورت

سریال

در #C

از ۹۲/۶/۵

تا ۹۲/۷/۷

به کار بردن هر قسمتی از این نوشته در هر جای دیگر بدون شرط آزاد است.

ناشران اینترنتی می توانند بدون هماهنگی، تنها در قسمت مشخص شده پایین همین صفحه، لوگو یا آدرس سایت خود را درج کنند.

تغییر صفحات یا قرار دادن لوگو یا هر متن دیگری در سایر صفحات مورد پذیرش نگارنده نیست.

استفاده از متن صفحات این نوشته، خوانا و بدون هیچ شکل یا نوشته اضافی حق خواننده است. خواهشمندم از خراب کردن این نوشته با هرگونه تغییر یا افزودن لوگو یا آدرسی خودداری فرمایید.

دانلود از تکنوالکترو و دنیاها

پشتیبانی فنی در ایران میکرو

نگارنده

نعمان عزیزی گردکشانه

امیر اسعد

Shahrivar88@gmail.com

فهرست

فهرست ---------- ۳
پیشگفتار ---------- ۵

بخش یکم: آغاز به کار در #C ---------- ۷

ایجاد یک پروژه در Visual Studio 2012 ---------- ۸
استفاده از ابزار ---------- ۱٤
کدنویسی ---------- ۱٦
اجرای برنامه ---------- ۱۸
حل یک مشکل ---------- ۲۰
تولید فایل exe. برنامه ---------- ۲۲

بخش دوم: کار با پورت سریال ---------- ۲۳

وارد کردن کامپونت پورت سریال ---------- ۲٤
باز کردن پورت ---------- ۲۵
بستن پورت ---------- ۲۷
ارسال داده ---------- ۲۸
دریافت داده ---------- ۳۱
سورس پایانی برنامه. ---------- ۳٤

بخش سوم: پروژه های پورت سریال ---------- ۳۷

پروژه یکم: ساخت یک ترمینال ---------- ۳۷
پروژه دوم: ساخت یک برنامه کنترلی ---------- ٥٤
کاربرد برنامه ---------- ٥٤
کارکرد برنامه ---------- ٥٥
قرار دادن ابزار مورد نیاز روی فرم ---------- ٥٦
کد برنامه کامپیوتر ---------- ٦۰
برنامه میکرو کنترلر ---------- ۷٥
توابع تعریف شده در Function.c ---------- ۸٦

بخش چهارم: متد و ویژگی های کامپونت پورت سریال ---------- ۹٦

متدها ---------- ۹۹
متد کاربردی ---------- ۹۹

متد های دریافت داده ---------- *۱۰۱*
متد های ارسال داده ---------- *۱۰۷*
ویژگی های پورت ---------- ۱۱۰

بخش پنجم: پیوست ---------- ۱۱۶

پیوست ۱ ---------- *۱۱۶*
پیوست ۲ ---------- *۱۱۷*
منابع ---------- *۱۲۷*

پیشگفتار

سلام بر همه انسانها[1]! و درود بر همه دوستداران و دست اندرکاران[2،3] در زمینه الکترونیک[4] به ویژه کسانی که داشته ها و دانسته های خود را با دیگران در میان میگذارند[5].

کسانی که در زمینه الکترونیک و میکرو کنترلر ها فعالیت دارند، با اهمیت پورت سریال آشنا هستند. پورت سریال آسان ترین و ارزانترین راه ارتباط، بین میکروکنترلر ها، یا به طور کلی، هر ابزار که با میکرو کنترلر ساخته شده و کامپیوتر شخصی می باشد. با توجه به کمبودی که در زمینه منابع برنامه نویسی، به زبان فارسی، برای پورت های کامپیوتر احساس کردم، برآن شدم تا، آموزش این گونه مباحث را، با زبانی در حد امکان ساده و نسبتا کامل بنویسم، به گونه ای که خواننده با کلیات کار آشنا شود و جزئیات را بتواند بسته به نیاز برنامه خود تغییر دهد.

در این آموزش به راه اندازی پورت سریال می پردازیم[6]. اکنون هر آنچه را که برای ایجاد این ارتباط نیاز دارید، در این نوشته گردآوری کرده ام به این امید که لحظات زندگی شما در پی یافتن این مهم، هدر نروند. در این نوشته شما با آموزش گام به گام تصویری، برنامه نویسی برای پورت سریال در C# را یاد خواهید گرفت تا در پروژه های مورد نیاز با ایجاد یک برنامه[7]، بین PC و میکرو کنترل ارتباط برقرار کنید. در پایان خواهید توانست که برنامه مورد نیاز، برای ارتباط یا کنترل ابزاری که می سازید و نیاز دارید تا به PC متصل شود، یا به وسیله PC کنترل شود، خودتان در C# ایجاد کنید.

من مخاطب اصلی این نوشته را کسانی قرار داده ام که در زمینه الکترونیک فعالیت دارند، و با توجه به اینکه آشنایی کمی با برنامه نویسی در محیط ویژول استدیو دارند، تلاش کرده ام با مطالب را با زبانی ساده بیان کنم.

برای مطالعه این نوشته نیازی به هیچ دانش قبلی از C# نیست. اما باید خوانند، با برنامه نویسی به زبان C و میکرو کنترل ها و همچنین مفاهیم پورت سریال آشنایی داشته باشد، این نوشته برای همه افرادی که میخواهند برای این پورت برنامه بنویسند قابل استفاده است.

این آموزش دارای ٤ بخش است.

در **بخش نخست**، ایجاد پروژه در ویژوال استدیو، اضافه کردن یک Button[8] به فرم، کد نوبسی برای آن و سرانجام اجرا و Debug برنامه ایجاد شده را بسیار ساده و کوتاه یاد خواهید گرفت.

در **بخش دوم** یک برنامه بسیار ساده را برای ارتباط با پورت سریال خواهید ساخت.

در **بخش سوم** با دو تمرین، نوشتن برنامه های کاربردی و مورد نیاز خود را فرا میگیرید.

---

[1] برای اینکه تبعیض نباشه زبان و کشور و گروه و دسته! ولی کاش می نوشتم سلام به همه خوانندگان!

[2] اینم نمیدونم املاش درسته یا نه. اول فعالان نوشته بودم بعد عوضش کردم تا میهنی باشه.

[3] میگم یک حرکتی هم انجام بدیم، این (ریال) رو از بغل (،) بردارند. تا این لحظه که در مرحله ویرایش پایانی این نوشته هستم بیشتر از ۱۰۰ بار به جای (،) دستم خورده،(ریال) نوشته شده. تا پاکشم میکنی کلی طول میکشه. کیبورد من برچسب فارسی نداره.

[4] من از سایر اقشار جامعه پوزش میخوام هدف ترویج تبعیض نیست، فقط خواستم یک جوری بگم من از این صنف جامعه هستم.

[5] منظورم خودم نیست هاآآ !.

[6] البته به نظر نگارنده فعل پرداخته می شویم بهتر است به دو دلیل. نخست زمان این فعل به آینده اشاره دارد. و حتما شما در زمان آینده نسبت با زمان نگارش به آن می پردازید. دوم که پرداخته شدن یک فعل کاربردی است مثل ساخته و پرداخته کردن.!

[7] Application

[8] Button

در **بخش چهارم** همه متدی[1] که برای این کنترل ارائه شده بررسی می شود.

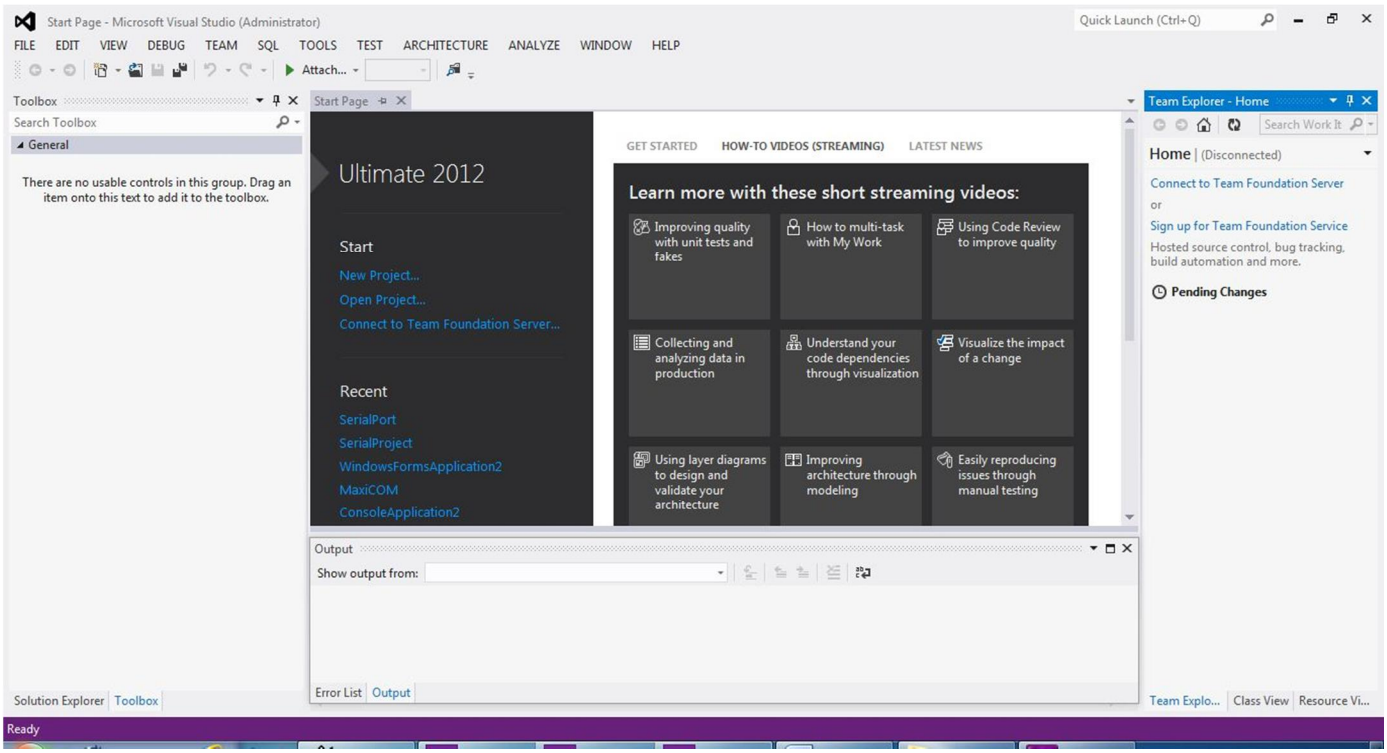
من برای برنامه نویسی از Visual Studio 2012 استفاده کرده ام .Net Framework4.5[2] ، که برنامه نویسی برای پورت سریال در ورژن های ۲۰۰۵ و ۲۰۰۸ و ۲۰۱۰ کاملا یکسان است. کلیه متد ارائه شده برای کار با پورت سریال، در کلیه .Net Framework ها، از ورژن ۲ به بالا یکسان است. یعنی شما دقیقا از همین متد ها برای راهندازی پورت سریال در سایر نسخه ها میتوانید استفاده کنید.

من به خوانندگان پیشنهاد میکنم در خواندن این کتاب شکیبایی داشته باشند. با توجه به حجم و تخصصی بودن مطلب، شاید کمی گنگ بنمایاند. ولی به شما این اطمینان را میدهم که همه مطالب در ادامه برای شما قابل فهم خواهد شد، حتی اگر بسیار تازه کار باشید.

[1] منظور همون چیزیه که تو C بهش تابع می گیم. البته منم اول همشونو تابع نوشته بودم اما بعد اصلاح کردم. اینجا هم به جاست که از سازنده Find & Replace تشکر کنم.

[2] این نقطه به معنای پایان خط نیست همراه خود اسمشه. پس بخونید دات نت فِریم وُرک نه نت فریم ورک نقطه! در ضمن به کسره فرم هم به اندازه کافی توجه کنید که خیلی زحمت کشیدم تو صفحه کلید کامپیوتر پیداش کردم. بزرگ هم نوشتم که قشنگ نکته اش دستتون بیاد. اگر هم خودتون تلفظ اش رو بلد بودید پس این پاورقی رو نخونید.

بخش یکم: آغاز به کار[1] در #C

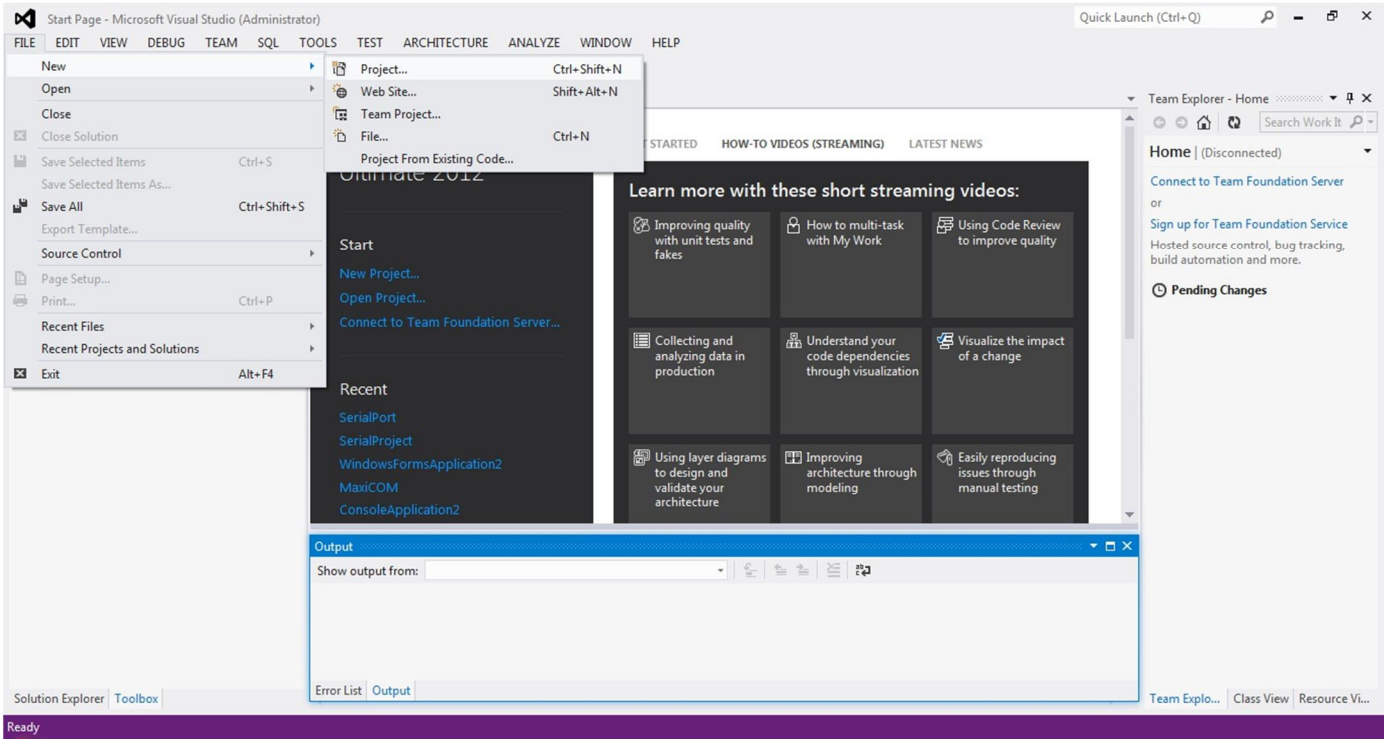


تصویر _ ۱ محیط برنامه Visual Studio 2012

[1] خدا وکیلی اینو نوشتم که روی خارجی ها رو کم کنم یه جا هایی همچین مینویسن **Getting Started** که انگار همینکه باز کنی همه چیو می فهمی ولی وقتی رفتی توش هر چیزی رو گفته الا اون که باید بگه. انگار تو دنده داری استارت میزنی هی میخوره اینور اونور ولی راه نمیافته. خلاصه اینم نمونه بومی **Getting Started** یک جور پریدم توش که #C رو ۱۰ دقیقه ای ضربه فنی کنید.

ایجاد یک پروژه در Visual Studio 2012

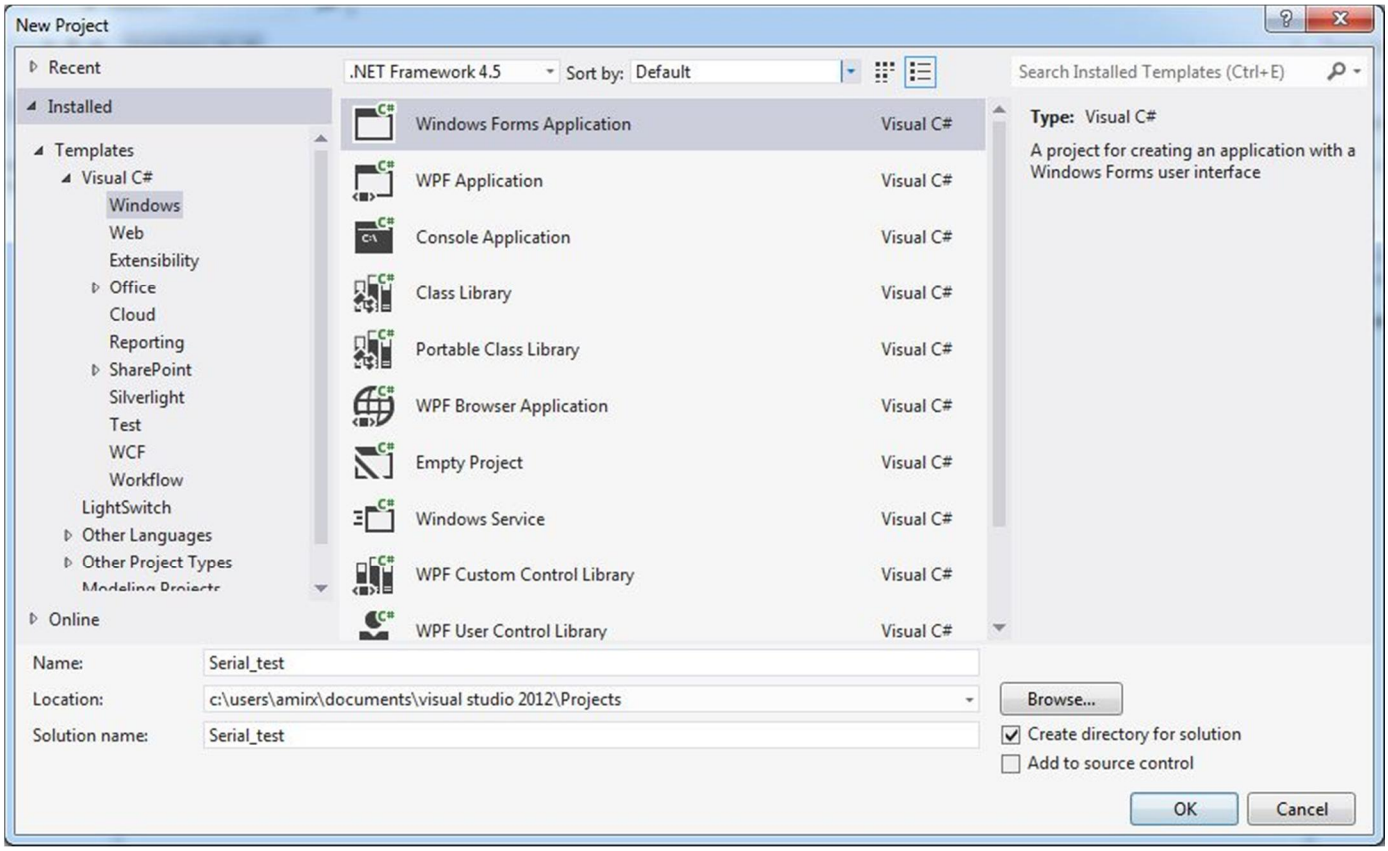
از مسیر FILE\New\Project یک پروژه جدید ایجاد کنید[1].



تصویر _ ۲ ایجاد یک پروژه جدید

[1] اینم بگم به نظر من رنگ بندی این ورژن اصلا خوب نیست زیادی یکنواخته. به نظر من **میکروسُفت** اینو یک نواخت داده میخاد برای ورژن بعدی تلافی کنه. میخواد اون یکی اگه اومد به چشم بیاد.

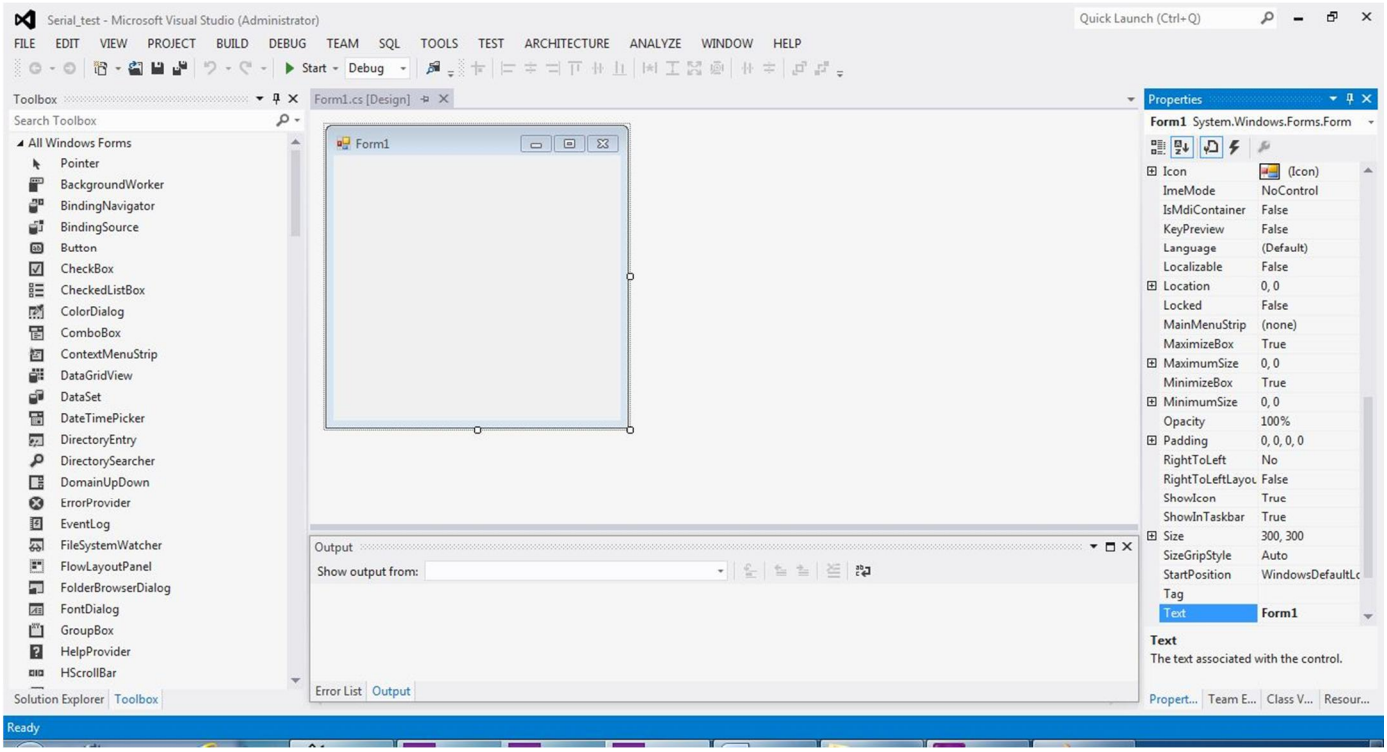
از قسمت سمت چپ پنجره باز شده، نوع پروژه را Visual C# سپس Window و سرانجام، Windows Form Application انتخاب کنید. در پایین پنجره نام دلخواه خود را برای پروژه در قسمت Name وارد کنید و روی دکمه[1] OK کلیک کنید. پیشنهاد می کنم برای هماهنگی با ادامه آموزش، نام پروژه تان را Serial_test بگذارید.



تصویر _ ۳ انتخاب نوع پروژه و وارد کردن نام پروژه

[1] یک چیزی میخواستم بگم ولی یادم رفت اگه یادم اومد مینویسم؛ دیگه با چشم خواهر برادری به پاورقی ها نگاه کنید.

پروژه ما برای کد نویسی و استفاده از ابزار و کنترل ها آماده است. مشاهده می فرمایید که یک صفحه یا فرم، در ویژوال استدیو برای شما ایجاد شده است. این فرم، بعدا تبدیل به صفحه ی اصلی برنامه ای که خواهد ساخت، می شود. فقط توجه داشته باشید که پنجره های Toolbox و Properties مانند آنچه در دو سمت تصویر زیر هست، باز باشند. البته ممکن است مکان این پنجره ها در کامپیوتر شما با تصویر یکسان نباشد. صفحه ای که فرم ایجاد شده در آن نمایش داده میشود، صفحه Design[1]، است. برای تغییر اندازه این فرم میتوان با کشیدن گوشه های فرم، یا وارد کردن اندازه مورد نظر در ویژگی Size[2] از پنجره Properties که در سمت راست تصویر زیر قرار دارد، اندازه فرم را تغییر دهید.

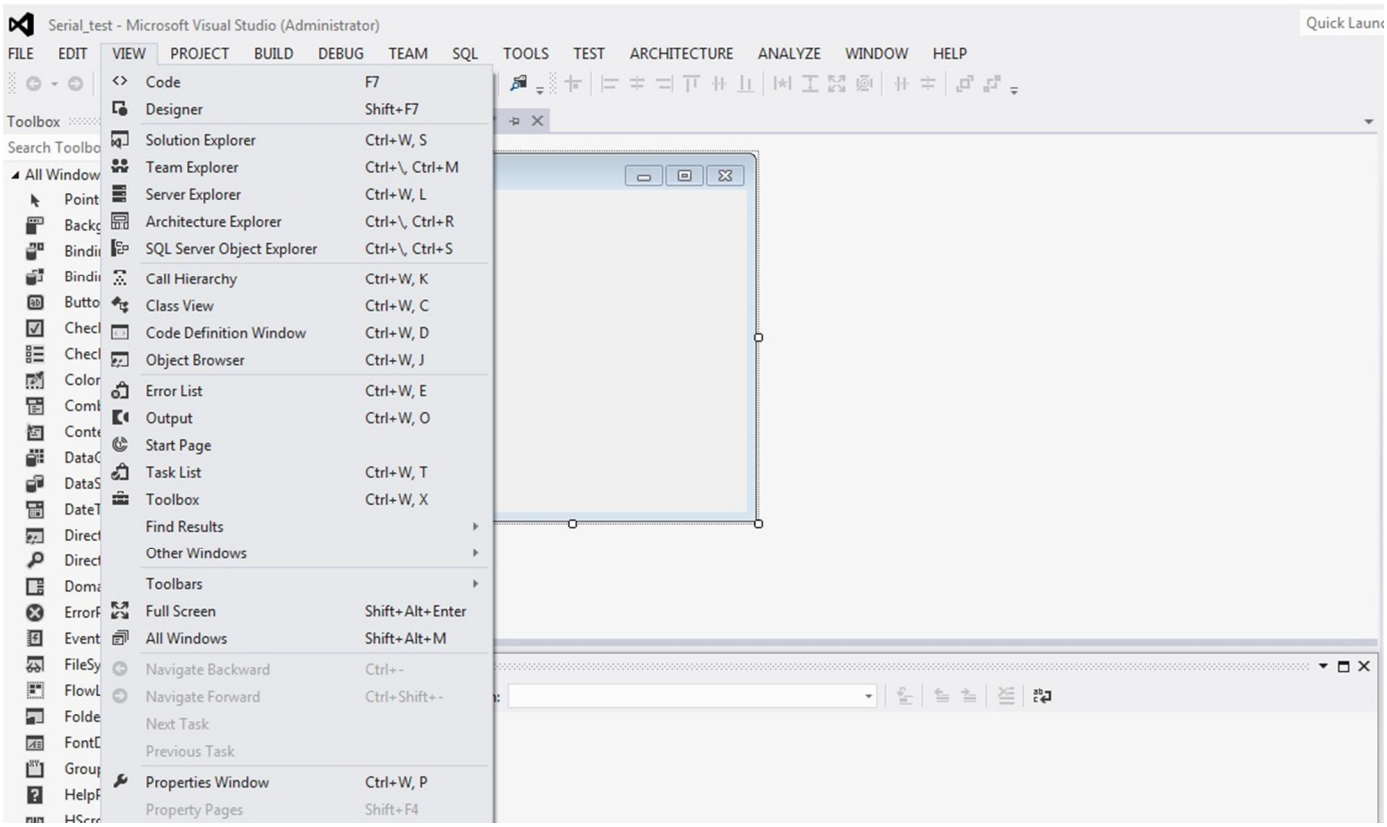


تصویر _ ٤ یک پروژه ایجاد شده در محیط Visual Studio 2012

[1] تب Form1.cs [Design]

[2] جاهای دیگه بهش خاصیت هم میگن، که اصالتا کلمه غیر بومیه. به مشتقاتش توجه کنید: خاص خصیصه خواص خصوصیات خصایص مخصوص اختصاص تخصیص مختص مختصات تخصص و شاید هم خسیس! اما من ویژگی را برگزیده ام چون این واژه کاملا بومیه اگه شما یک مشتق ازش پیدا کردی من کل این آموزش رو حرف به حرف از آخر به اول مینویسم حتی عکساشم برمیدارم سرو ته میذارم سرجاشون. الان برنامه رو بیخیال، بیافتید دنیال این که منو ضایه کنید اینم بگم کل سر و ته نویسی را تو ۱۰ دقیقه میتونم انجام بدم نگران من نباشید براش یک برنامه مینویسم متن میدم سروته تحویلش میگیرم فقط کپی پیست لازم داره. پس بچسپسد به آموزش.

اگر به هر دلیلی این دو پنجره را مشاهده نمی کنید، برای فعال کردن نمایش آنها مانند تصویر زیر از منوی VIEW نمایش آنها فعال کنید، یا در بین سایر پنجره ها در دو سمت صفحه آنها را بیابید، و با جابجایی پنجره ها، آنها در محل مناسب قرار دهید. با راست کلیک روی فرم و انتخاب Properties نیز می توانید پنجره Properties را باز کنید. شما میتوانید با کشیدن پنجره ها آن ها را جابجا کنید و در مکان مورد نظر قرار دهید. بهتر است این هر یک از این دو پنجره در یک سمت صفحه باشند چون همزمان با هر دوی آنها کار میکنیم.



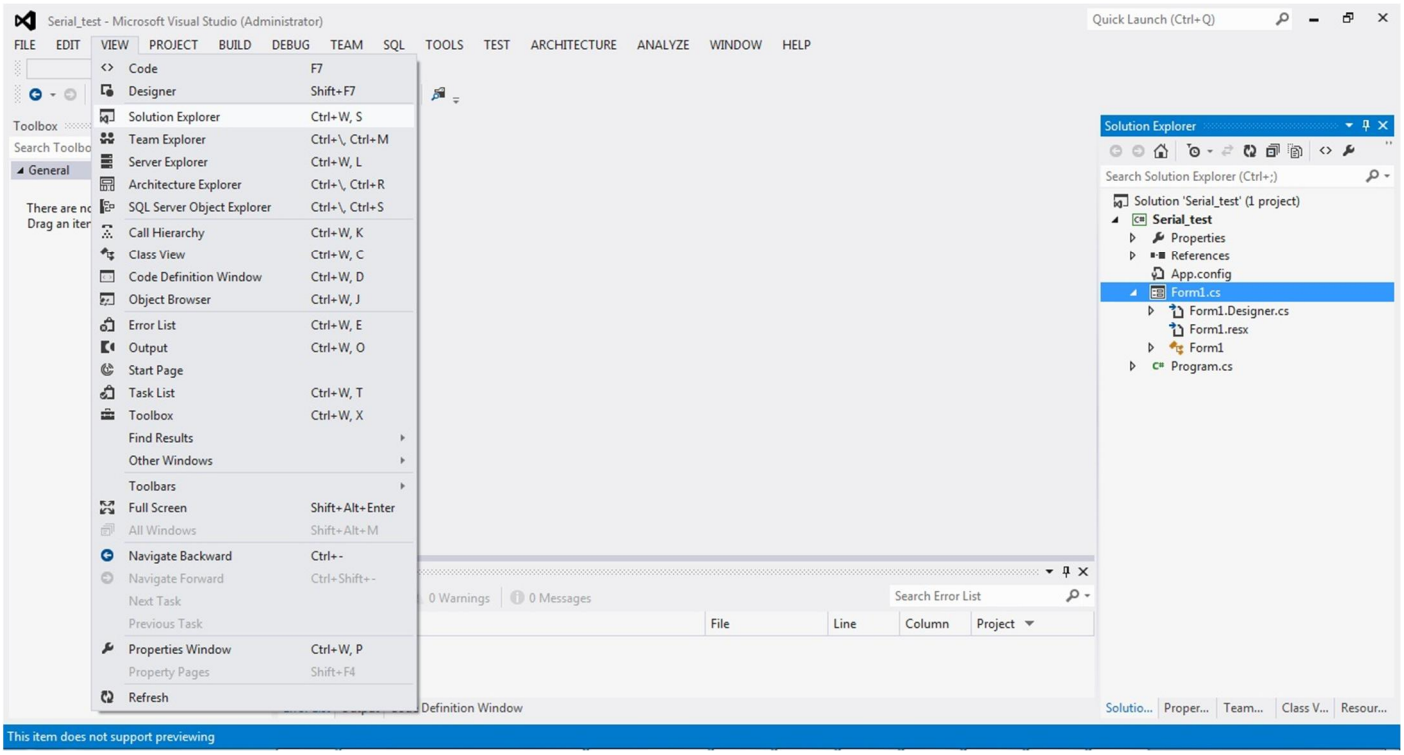
تصویر _ ۵ فعال کردن نمایش پنجره های مورد نیاز در Visual Studio 2012

در صفحه Design شما می توانید ویژگی های دیداری[1] برنامه خود شامل دکمه ها منو ها و کلیه ابزاری که در برنامه خود قرار می دهید را ببینید و یا تغییر کنید.

احتمالا در آینده، هنگامه که پروژه تان را باز می کنید، صفحه Design و یا حتی صفحه Code پروژه شما به هنگام باز شدن پروژه، به صورت پیش فرض برایتان نمایش داده نشود، یا ممکن است به صورت اتفاقی آن ها را ببندید. اصلا نگران نباشید، حتی نیازی به زنگ زدن به دوستان تان ندارید تا در این زمینه به کمکتان بشتابند. کافی است پنجره Solution Explorer را پیدا کنید. می دانید که نخستین مکانی که برای نمایش دادن هر پنجره ای باید بروید، منوی VIEW است. با کلیک روی نام هر پنجره ای، آن را در هر کنجی که پنهان شده باشد بیرون می کشید. کم کم مکان پنجره ها را یاد می گیرید و بعد خیلی راحت پنجره های دیگر را پیدا می کنید. البته هیچگاه به هیچ پنجره ی دیگری نیاز نخواهیم داشت. در Solution Explorer روی Form1.cs دابل کلیک کنید.

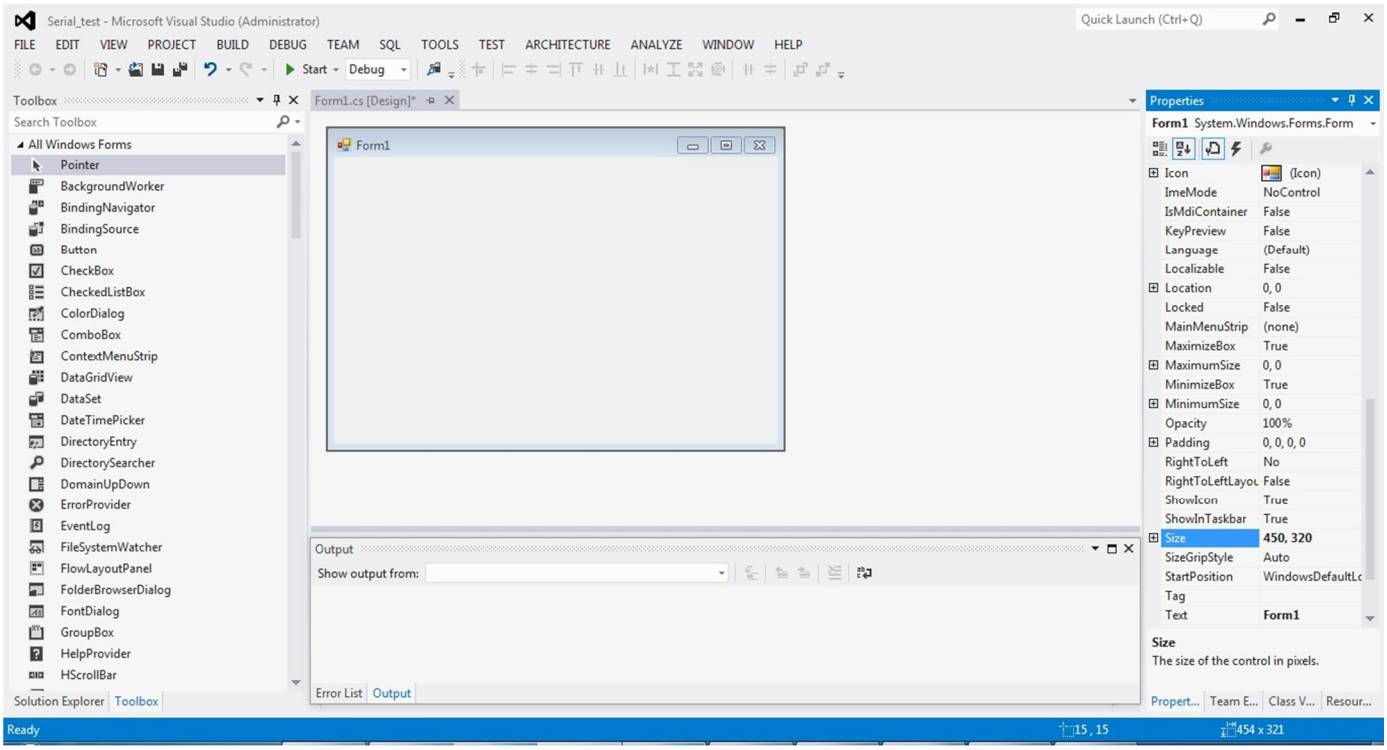
[1] بصری (عربی)

پنجره Solution Explorer نقشه پیوند اجزای برنامه ی ما است. اما چون ما یک برنامه ساده می سازیم که تنها شامل ۱ فرم است، فقط این فرم و کد های آن در Solution Explorer وجود دارد. اگر به فرض این که برنامه ای بسازید که شامل فرم ها یا صفحات مختلف باشد همه آن ها را میتوانید در این پنجره پیدا کنید.



تصویر _ ٦ باز کردن پنجره Solution Explorer از منوی VIEW

اگر اندازه فرم برنامه تان را تنظیم نکرده اید، روی فرم کلیک کنید و در پنجره Properties ویژگی Size آن را برابر **450,320** قرار دهید.

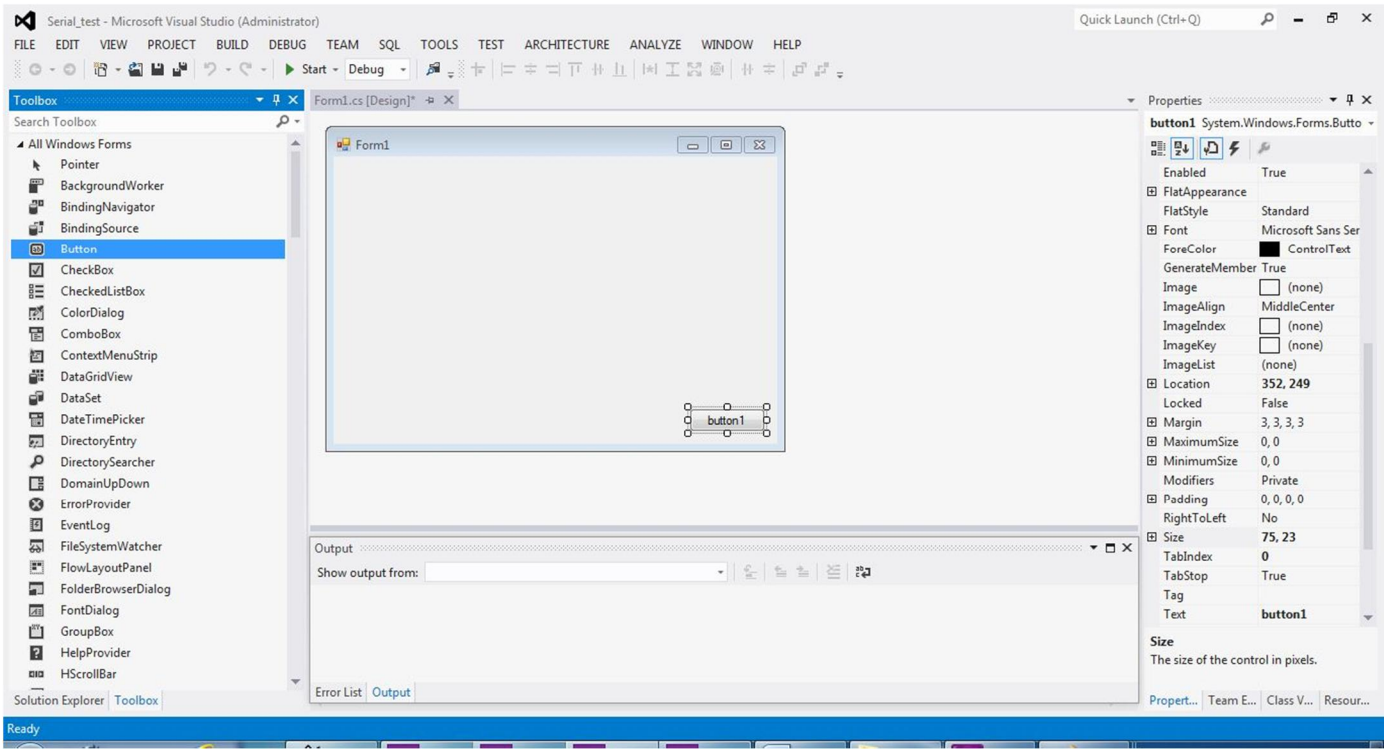


تصویر _ ۷ فرم ایجاد شده در پنجره Design نمایش داده می شود.

استفاده از ابزار

از پنجره Toolbox یک Button را با درگ کردن روی فرم، یا دابل کلیک روی Button در پنجره Toolbox ، روی فرم قرار دهید و با جابجا کردن Button آن را در محل مناسب در فرم قرار دهید.

برای وارد کردن هر ابزاری به فرم شما باید در صفحه Design باشید. وقتی شما یک پروژه ایجاد میکنید، صفحه Design پروژه تان را می بینید.



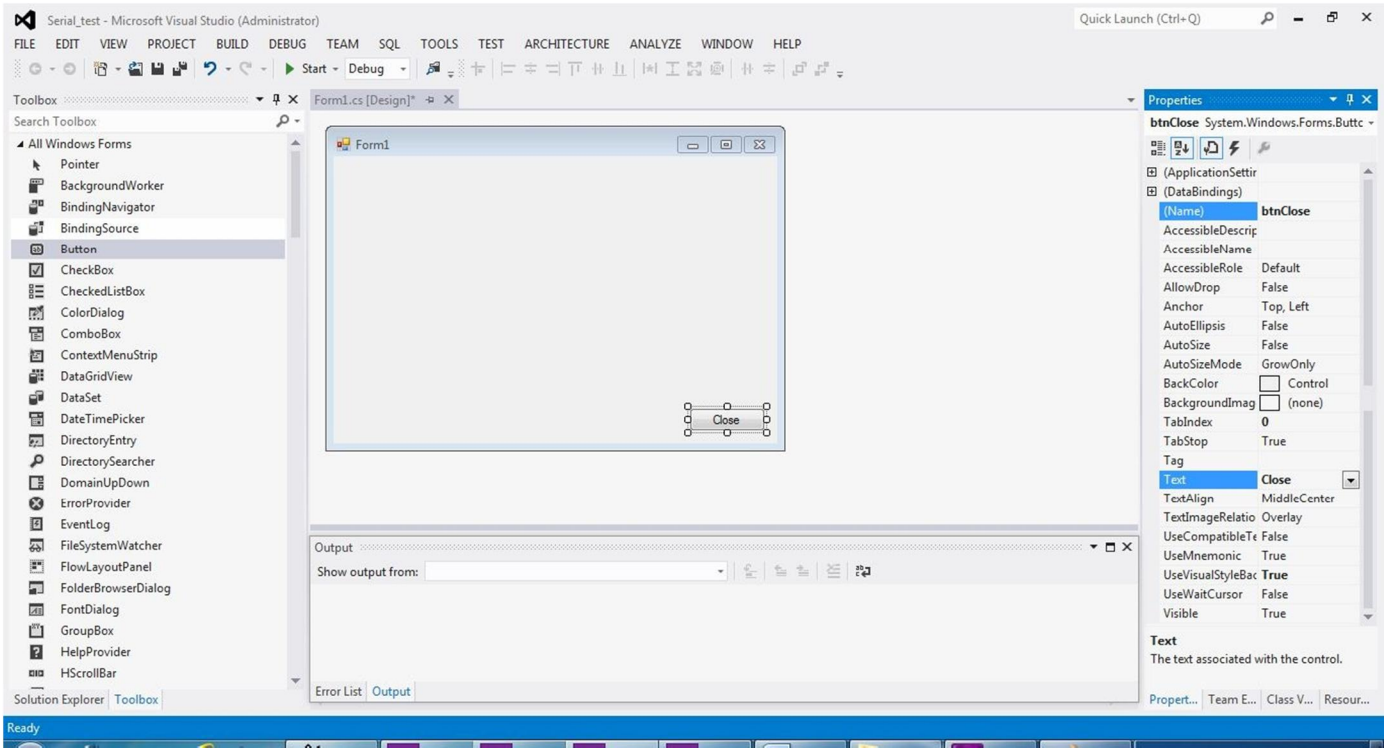
تصویر _ ۸ قرار دادن یک Button روی فرم

.

می خواهیم از این Button جهت خروج از برنامه استفاده کنیم. یعنی کاربر میتواند هنگام اجرای برنامه با کلیک روی این Button برنامه را ببندد.

روی Button که در فرم قرار داده اید کلیک کنید[1]، و از پنجره Properties ویژگی Name و Text مربوط به این Button را مانند تصویر زیر تغییر دهید. اگر پنجره Properties را نمی یبینید روی Button راست کلیک کنید و از منوی راست کلیک روی Properties کلیک کنید.

(Name) btnClose
(Text) Close



تصویر _ ۹ تنظیم ویژگی های Button در پنجره Properties

ویژگی Text مربوط به نمایش متنی است که روی Button یا هر کنترل دیگری نمایش داده میشود اما Name برای کدنویسی اهمیت دارد، چون در کد نویسی برای هر ابزار، از Name آن ابزار استفاده می شود. پس باید نام یک ابزار، همیشه منحصر به فرد انتخاب شود، و به گونه ای باشد که به هنگام کد نویسی با نام سایر کنترل ها اشتباه گرفته نشود.

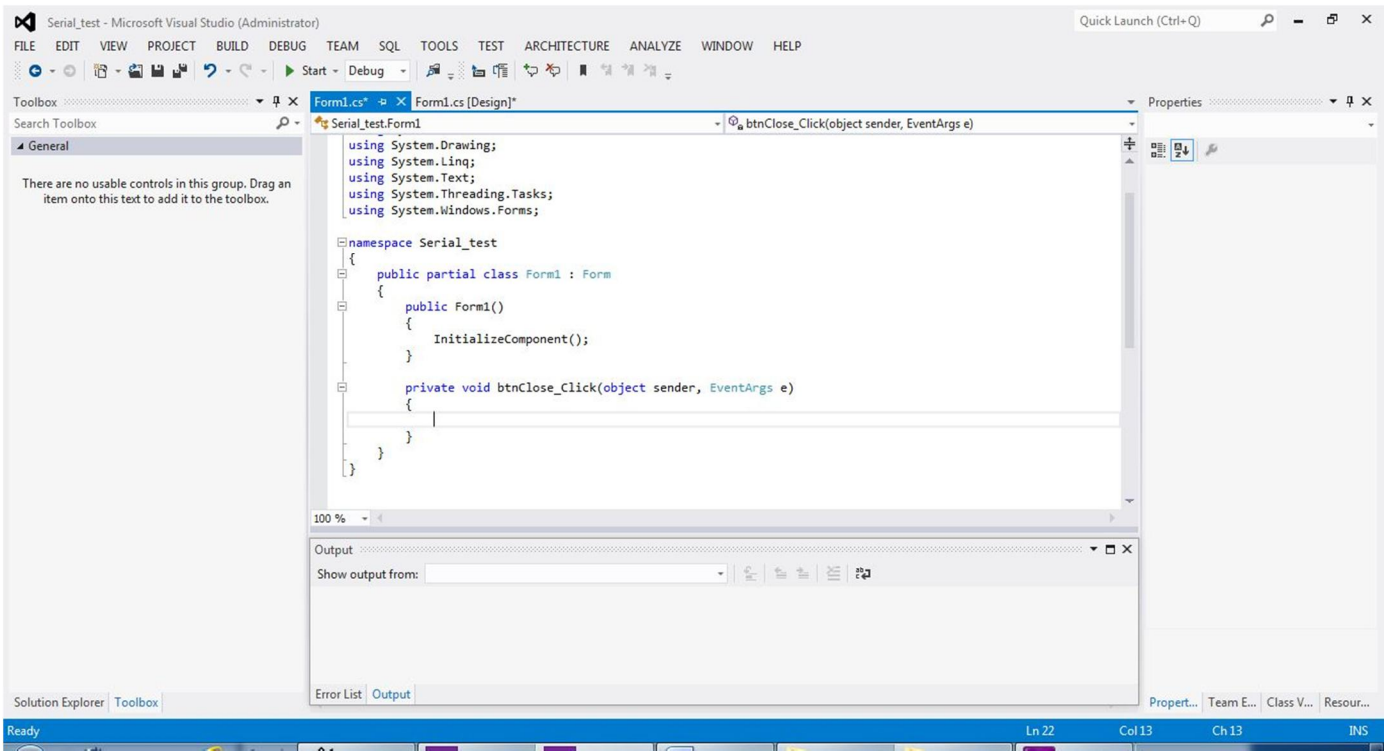
[1] دابل کلیک نکنید ها! مسولیتش پای خودتونه!

کدنویسی

حالا روی دکمه Close که ویژگی نام و متن آن را تنظیم کرده ایم دابل کلیک کنید[1]. مشاهده میکنید که به صفحه Code[2] پروژه می روید. توجه داشته باشید که با دابل کلیک روی هر Button کد های رویداد مربوط به **کلیک** آن Button به صورت اتوماتیک در پنجره Code وارد می شود. این کد ها به این معنی است که هنگام اجرای این برنامه، اگر روی این Button کلیک شود، برنامه چه کاری را باید انجام دهد. یعنی هنگام اجرای برنامه، هنگام روی دادن هر رویدادی متد آن رویداد اجرا میشود. مثلا برای یک Button متد رویداد کلیک زمانی اجرا می شود که در زمان اجرا روی آن کلیک شود. پس، اگر میخواهید با کلیک روی Button کار خاصی انجام شود. باید کد های مربوط به آن کار را در متد رویداد کلیک Button وارد کنید.

با دابل کلیک روی هر ابزاری به طور پیش فرض متد فرایند خاصی که به نظر سازنده پر کاربرد تر است ساخته می شود. یعنی ابزار ها فرایند های دیگری هم دارند که شما میتوانید بسته به نیازتان آنها را فعال و استفاده کنید. در ادامه با چگونگی ایجاد یک فرایند آشنا خواهید شد.

همیشه یادتان باشد که همه کد ها یا دستورات برنامه را باید در صفحه Code پروژه بنویسید. پس هرجایی سخن از کد نویسی به میان آمد، به این صفحه بروید. و هر جا که در مورد گذاشتن ابزار روی فرم گفته شده با چابکی، به آن صفحه کوچ کنید.



تصویر _ ۱۰ صفحه Code پروژه

[1] حالا همگی با هم دابل کلیک میکنیم. حرکت انفرادی مورد قبول نگارنده نمی باشد!

[2] Form1.cs

ما میخواهیم از این Button برای بستن صفحه برنامه استفاده کنیم، پس برای رویداد کلیک آن، کد زیر را وارد کنید[1].

```
private void btnClose_Click(object sender, EventArgs e)
{
  this.Close();
}
```

متد this.Close() باعث بسته شدن پنجره جاری میشود. که در رویداد کلیک دکمه[2] btnClose تعریف شده است. یعنی اگر برنامه اجرا شود، با کلیک روی این دکمه پنجره یا صفحه ای که این دکمه روی آن قرار دارد بسته می شود.

```
namespace Serial_test
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void btnClose_Click(object sender, EventArgs e)
        {
            this.Close();
        }
    }
}
```

تصویر11 _ نوشتن دستورات در متد فرابند کلیک Button

مشاهده میکنید که در رویداد کلیک این Button، از نامی که برای Button انتخاب کردید،[3] استفاده شده است.

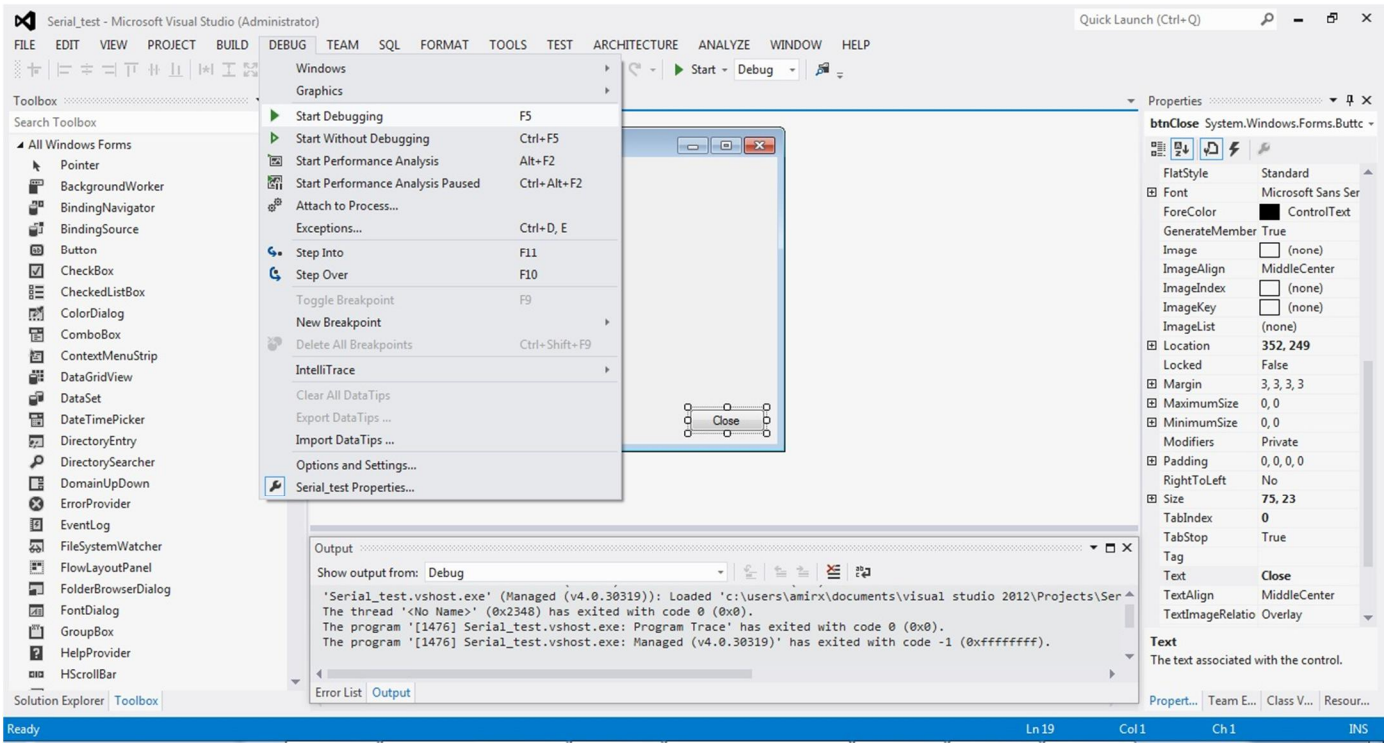
[1] بعد از دابل کلیک روی Button در صفحه Design به صفحه Code رفته اید و متد فرایند کلیک دکمه Close ایجاد شده است. پس تنها باید کد های درون متد را بنویسید. یعنی تنها باید this.Close(); را درون متد ایجاد شده بنویسید.

[2] Button

[3] btnClose

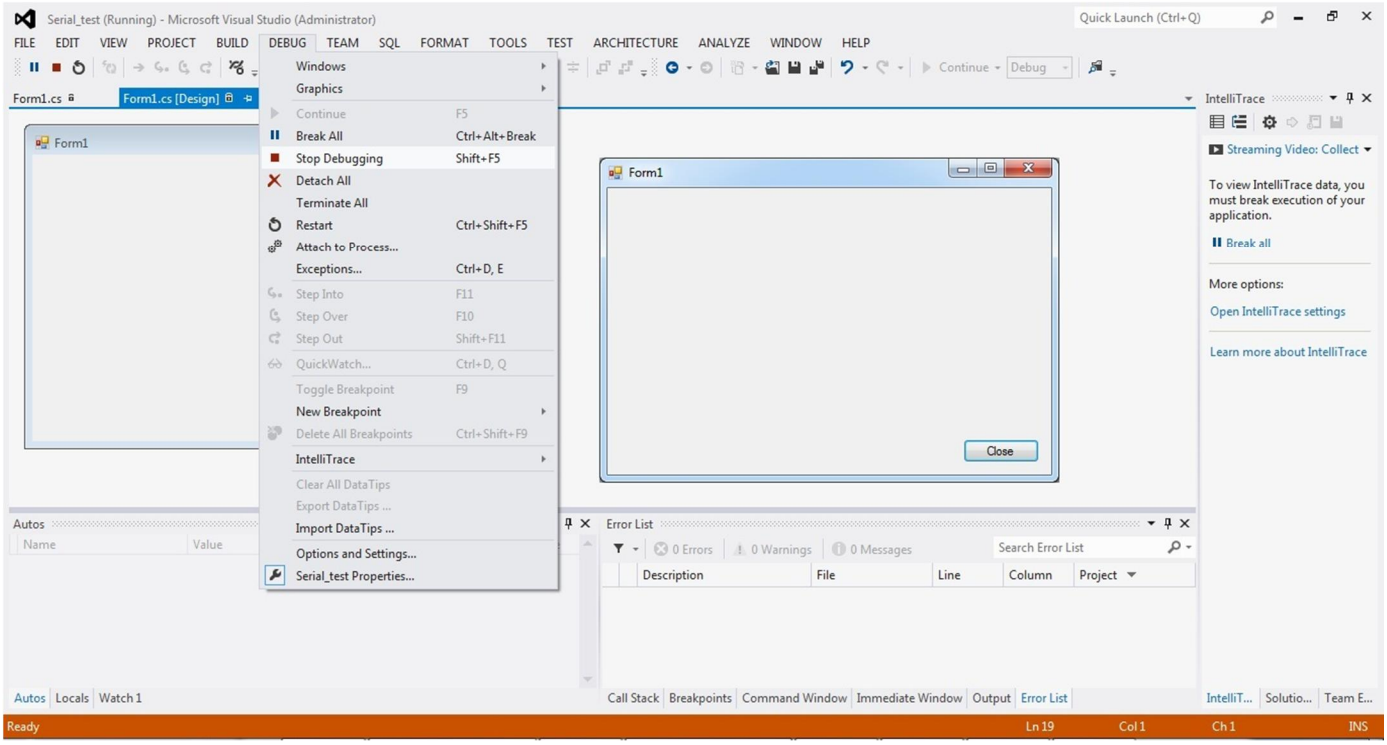
اجرای برنامه

پس از انجام همه مراحل بالا، باید برنامه را آزمایش، و از اجرای درست برنامه آسودگی خاطر پیدا کنیم. در منوی DEBUG روی Start Debugging کلیک کنید، و یا کلید F5 را روی صفحه کلید کامپیوترتان فشار دهید و یا روی آیکن Start Debugging در پنجره اصلی ویژوال استدیو کلیک کنید. اگر شما در کدنویسی اشتباهی نداشته باشید برنامه ساخته شده اجرا میشود و اگر برنامه شما ایرادی داشته باشد شما میتوانید مکان و نوع ایراد کار خود را در پنجره Error List ببینید. حتی اگر این پنجره را بسته باشید، هنگام اجرای برنامه به صورت خود کار در پایین پنجره اصلی ویژوال استدیو باز می شود.



تصویر _ ۱۲ اجرای برنامه ساخته شده

مشاهده می فرمایید فرم شما مانند هر برنامه دیگری، همه ویژگی های یک برنامه ویندوزی را دارد. برای خارج شدن از حالت Debug و بازگشت به Visual Studio روی X [1] پنجره یا دکمه Close که خودتان ساخته اید، کلیک کنید. از مسیر DEBUG\Stop Debugging (Shift+F5) نیز میتوان پنجره برنامه را ببندیم و به Visual Studio برگردیم. توجه داشته باشید که زمانی که در حالت Start Debugging هستید، نمی توانید هیچ تغییری در کد و یا فرم برنامه تان ایجاد کنید.

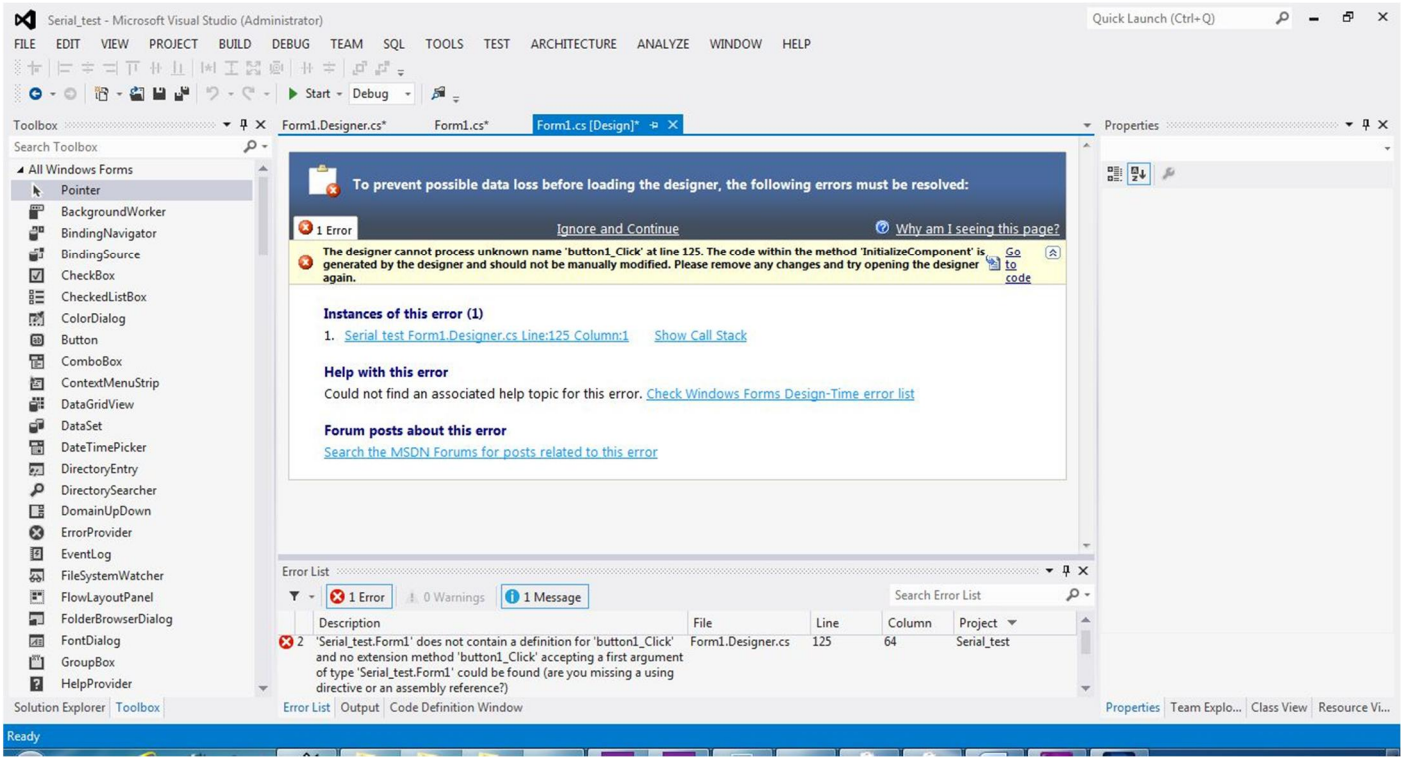


تصویر _ ۱۳ پایان Debug و پایان دادن به اجرای برنامه ساخته شده

[1] این که من ایکس نوشتم، منظور همون ضربدر خودمونه اونجا یک مقدار جدی بودم اینجوری نوشتم. اگر هنوز متوجه این ایکسه نشدید بگم همون گوشه بالایی سمت راست پنجره های ویندوز وقتی که روش کلیک کنی پنجره بسته میشه. خارجی ها ایکس میگن، ما ایرانی ها ضربدر. فک کنم متوجه شدید چی میگم. خدا وکیلی حوصله ندارم ازش عکس بگیرم بذارم اینجا. حتی حوصله انگلیسی کردن صفحه کلید رو هم ندارم دیدید دیگه ایکس هم فارسی نوشتم. (پیری و هزار عیب!)

حل یک مشکل[1]

اگر به هردلیلی ابزاری روی فرم قرار دهید و پس از ایجاد شدن متد رویداد آن، اقدام به پاک کردن آن رویداد کنید، مثلا شما یک Button روی فرم قرار میدهید و پس از دابل کلیک روی آن، وقتی متد رویداد کلیک ساخته شد متوجه میشوید نام Button را تنظیم نکرده اید! وقتی متد فرایند کلیک آن را پاک می کنید، با مشکل زیر روبرو می شوبد.



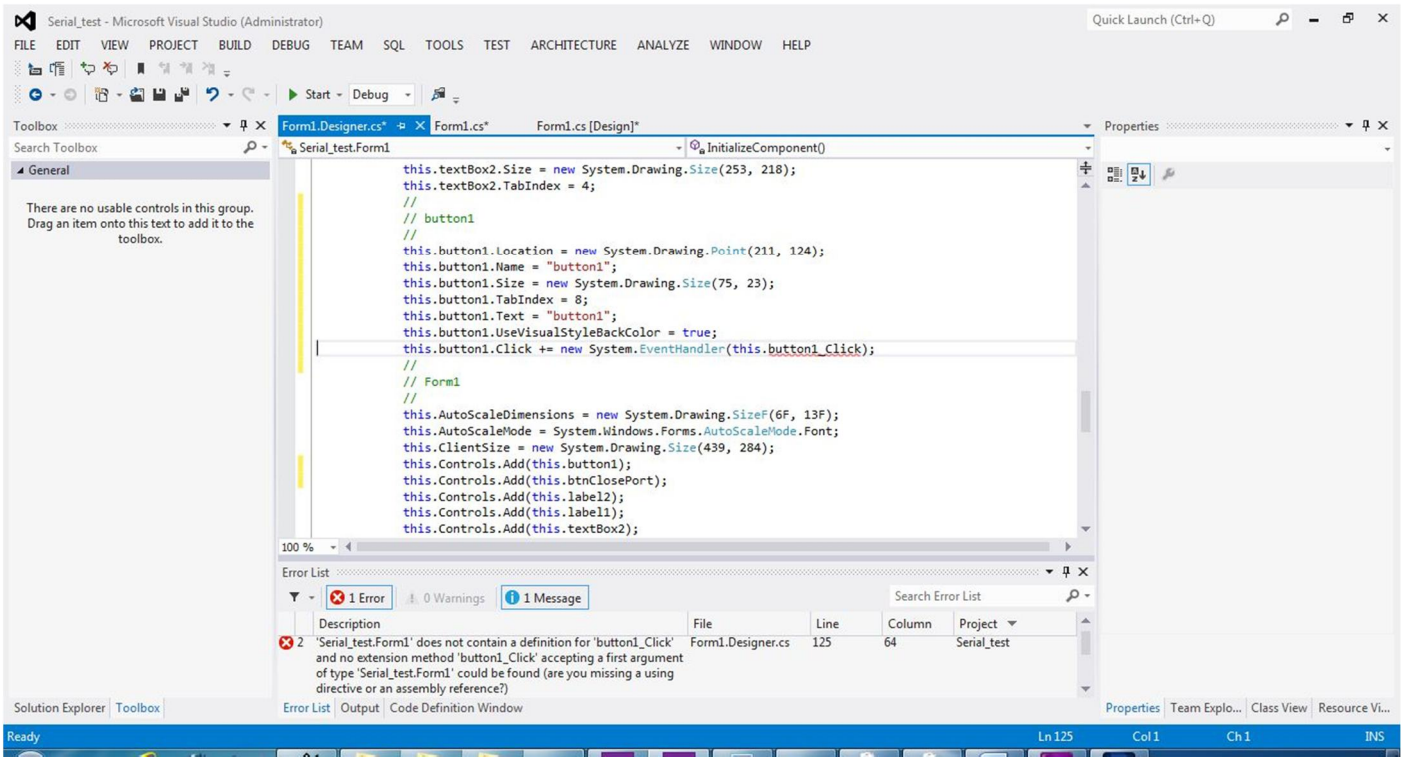
تصویر _ ١٤مشکل ناشی از پاک کردن متد رویداد یک ابزار

برای پیش گیری از این مشکل باید نخست، Button را از فرم حذف کنید، سپس متد فرایند آن را پاک کنید. اما اگر با صفحه بالا روبرو شدید، خونسردی خود را حفظ کنید و در همان صفحه، روی لینک Serial_test Form1.Designer.cs Line:125Column:1 که زیر نوشته، **Instances of this error (1)** قرار دارد، کلیک کنید[2]. ویژوال استدیو، به صورت خودکار شما را به مکان ایراد،

[1] اونجا که گفتم کلیک کنید دابل کلیک نکنید، به همین خاطر بود. فرض کنید شما در اون مرحله دابل کلیک میکردید و وقتی متد فرایند کلیک در صفحه Code ساخته میشد، متوجه میشدید نام دکمه را تنظیم نکرده اید. وقتی متد را پاک میکردید تا مثلا بعد از عوض کردن نام، دوباره این متد را ایجاد کنید.خرسند خرسند به صفحه Design برمی گشتید و با صحنه بالا مواجه می شدید. انگار رسیدید دم در خونه می بینید کلید همراتون نیست. خب نمیتونید برید تو دیگه. پشت در خونه می موندید. شاید شما هم مجبور میشدید پروژه را از آغاز ایجاد کنید. مثل اینکه برید یک خونه نو بخرید و اون رو ول کنید.

[2] شاید بعضی ها روی این نوشته کلیک کنند. خطاب به این دسته از دوستان بگم توی صفحه ویژوال استدیو روی لینک کلیک کنید. اینم بگم لینک همون نوشته های آبی رنگی هست که زیر خط کشیده شده و وقتی روش کلبک میکنید برنامه شما رو به یه جای دیگه میبره. مثلا تو ابترنت از اینها زیاده هی رو اینو اون کلیک میکنی از این صفحه به اون صفحه میرید. کنند خطاب به اون دسته از دوستان هم که دارن به این دسته از دوستان میخندند، بگم، که ایشون بلد نیست لینک چیه. خب همه ما یه روزی بلد نبودیم. اصلا خیلی خوبه یعنی ایشون تازه با کامپیوتر آشنا شده اومده سراغ برنامه نویسی من که اراده اش راتحسین میکنم به احترام این عده از دوستان تصمیم گرفتم بخش ششم هم به این نوشته اضافه کنم. نمیدونم چی توش بنویسم! فکر کنم اگه به نتیجه رسیدم میگم.

راهنمایی می کند. خطی که کورسر[1] در ابتدای آن قرار دارد، مربوط به ایجاد متد کلیک برای دکمه ای است که وجود ندارد، و همین مسئله باعث پیش آمدن این مشکل گشته است. برای حل مشکل، آن خط را کامل پاک کنید. بقیه کد های مربوط به این ابزار بعدا به صورت خودکار پاک می شوند پس اصلا نگران نباشید. اما مواظب خط های دیگر که مربوط به سایر ابزار است، باشید که دستکاری نشوند. این صفحه که عنوان آن Form1.Designer.cs است مربوط به کدهای، ابزار هایی است که ویژگی های آن ها را در پنجره Properties تنظیم کرده اید. در این جا کد های مربوط به تنظیمات ابزار هایی که شما استفاده کرده اید وجود دارد.



تصویر _ ۱۵ برطرف کردن ایراد، با پاک کردن خط مربوط به ابزاری که متد رویدادش پاک شده است.

[1] مکان نما Cursor به نظر من خط عمودی چشمک زن بهتره. البته به جای عمودی هم بگیم ایستاده کاملا میهنی میشه. خط ایستاده چشمک زن. مخففش رو هم خواستیم بکار ببریم خاچز میشه هم از نمونه مشابه خارجی کوتاه تره هم ادای اون شوقی در دل آدم ایجاد میکنه.

تولید فایل exe. برنامه

برای تولید فایل exe. برنامه ای که ساخته اید، نخست پروژه را به حالت Release ببرید. وقتی یک پروژه ایجاد میکنید، برنامه به صورت پیش فرض در حالت Debug قرار دارد. اگر در صفحه ویژوال استودیو **زیرِ** منوی TEAM را نگاه کنید، منوی کشویی که Debug در آن نوشته شده را میبینید. به همین سادگی؛ روی آن کلیک کنید و آن به Release تغییر دهید. سپس، از منوی، BUILD[1] روی **Build Solution** (F6) کلیک کنید. سپس دوباره از منوی، BUILD این بار روی **Build <*Your Project Name*>**[2] کلیک کنید.[3] گزارش انجام عملیات بالا، در پنجره Output قابل مشاهده است. در صورتی که عملیات با موفقیت به پایان برسد، شما پیام زیر را در پنجره Output مشاهده خواهید کرد.

```
========== Rebuild All: 1 succeeded, 0 failed, 0 skipped ==========
```

فایل exe. برنامه شما از مسیر زیر قابل دستیابی است.

Documents\Visual Studio 2012\Projects\Serial_test\Serial_test\bin**Release**\Serial_test.exe

میتوانید فایل exe. را بردارید و بدون نیاز به ویژوال استدیو، حتی در کامپیوتر های دیگر نیز اجرا کنید. البته برنامه شما برای اجرا شدن روی هر کامپیوتری نیاز به نصب بودن، ورژن Net Framework. که برنامه را با آن نوشتید دارد. البته نگران این موضوع نباشید چون بیشتر برنامه ها هنگام نصب شدن، Net Framework. ها را نصب می کنند. به احتمال زیاد همراه نصب سایر برنامه ها مانند Microsoft Office یا از طریق Update ویندوز قبلا نصب شده است. اگر هم خواستید آنرا نصب کنید در مجموعه برنامه های کامپیوتری مانند LORD و King موجود است.

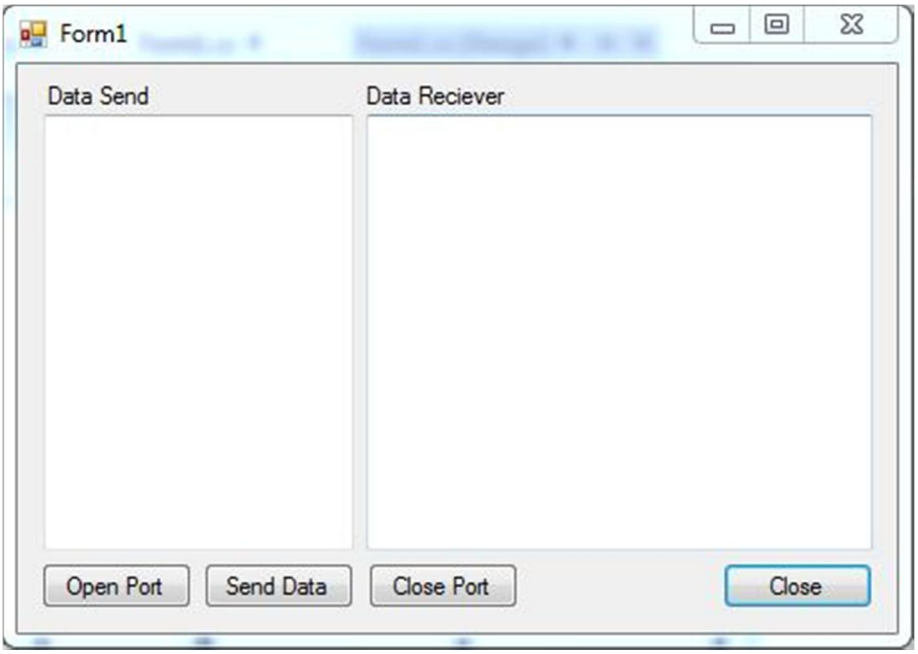
[1] بهتر است برای دفعات بعدی، از گزینه **Rebuild** استفاده کنید، چون این گزینه فایل های ایجاد شده دفعه قبل را پاک میکند وهمه را از نو می سازد.

[2] پروژتان را با هر نامی ذخیره کرده باشید در آنجا نوشته میشود. در پروژه من Serial_test **Build** می باشد.

[3] این گزینه هم بهتر است برای دفعات بعدی از **Reuild <*Your Project Name*>** استفاده شود.

بخش دوم: کار با پورت سریال

در این بخش با ساخت یک برنامه[1] ساده، با مفاهیم اولیه برنامه نویسی پورت سریال آشنا می شوید. کلیه مراحل کار و دستورات، خط به خط توضیح داده شده است. کار با این پورت بسیار ساده تر از آنچه که فکر می کنید می باشد. از قوی ترین ابزار برنامه نویسی غافل نشوید. **COPY+PASTE**[2]!



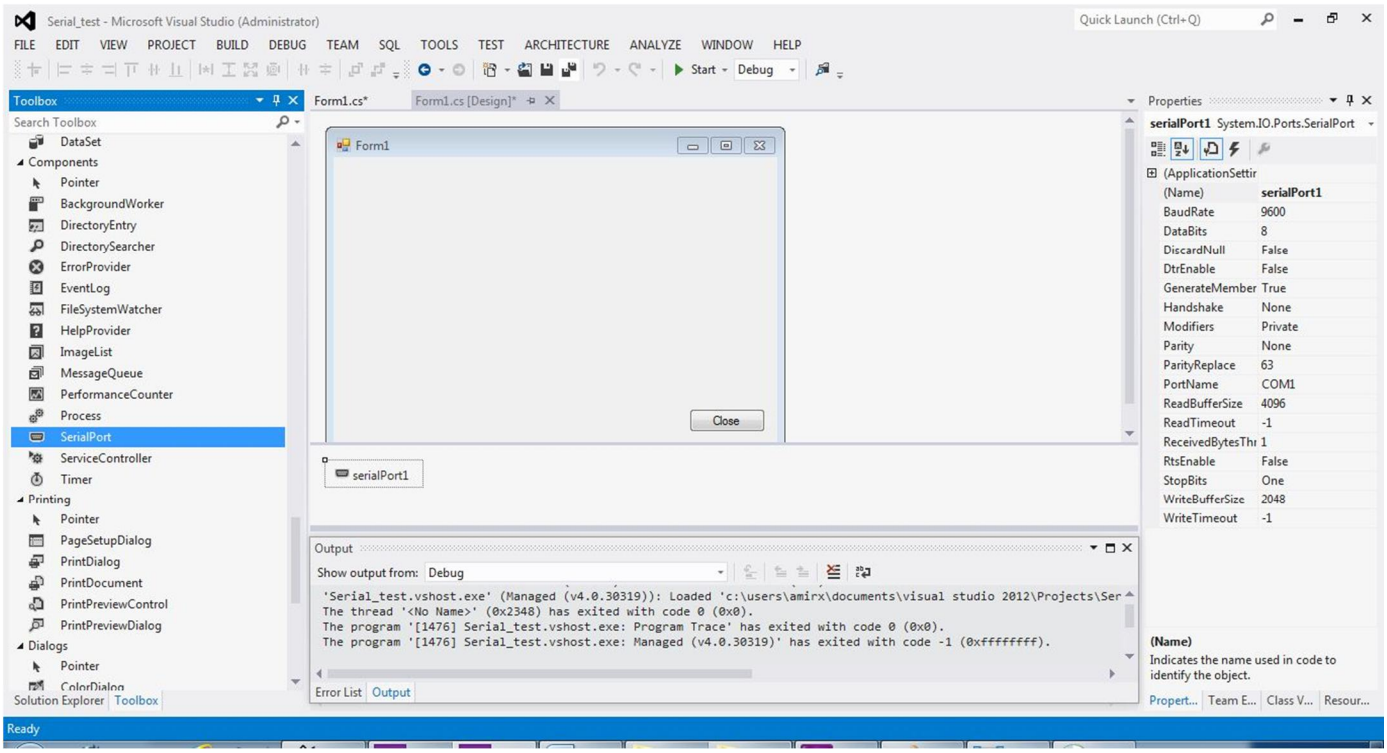
تصویر _ ۱۶ نمای پایانی برنامه

[1] سورس پروژه های همراه این نوشته را دانلود بفرمائید.

[2] خدا وکیلی باید به کاشفش اسکار فیزیک بدهند.

وارد کردن کامپونت پورت سریال

برای کار با پورت سریال باید کامپونت مربوط به این پورت را به برنامه اضافه کنید. برای وارد کردن کامپونت مربوط به این پورت، از پنجره Toolbox و از بخش Componets کنترل SerialPort را روی فرمی که در بخش پیش ایجاد کردید، درگ کنید[1]. در پنجره Properties ویژگی های این کامپونت را می بینید. که نیازی به تغییر هیچکدام از آنها در این صفحه نمی باشد چون در قسمت کد نویسی تنظیمات لازم را انجام می دهیم. فقط به نام کامپونت، serialPort1 توجه کنید که کد نویسی را بر پایه نام آن انجام میدهیم.

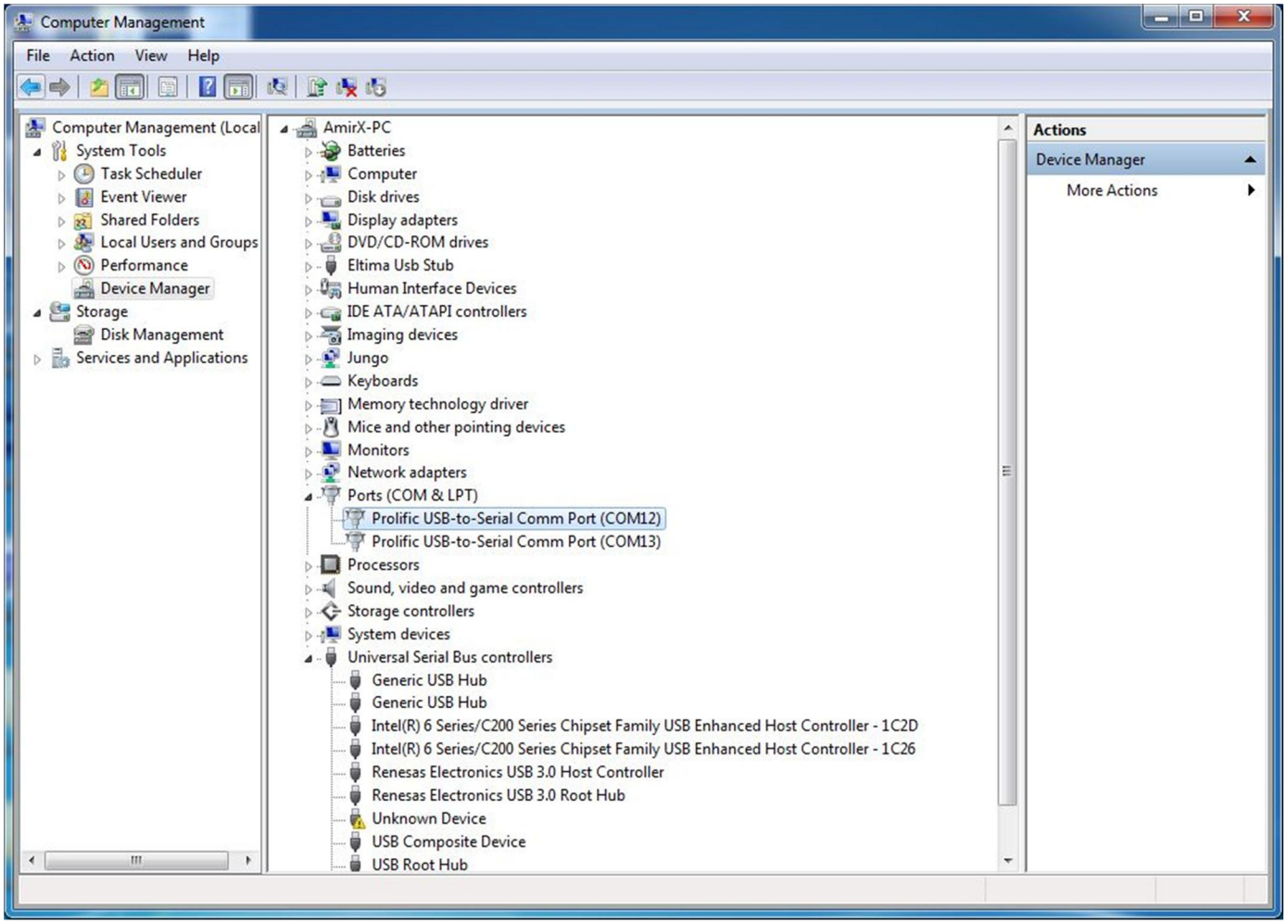


تصویر _ ۱۷ اضافه کردن کامپونت پورت سریال به فرم

[1] در ویژوال استدیو ۲۰۱۲ در بالای Toolbox قسمت جستجو وجود دارد، که برای پیدا کردن راحت تر کنترل ها می توان از آن استفاده کرد.

باز کردن پورت

کامپیوتر خانگی شما احتمالا دو پورت سریال به نام های COM1 و COM2 دارد و اگر از مبدل های USB TO COM استفاده می کنید، می دانید که با وصل کرده این مبدل ها به کامپیوتر، یک پورت سریال مجازی در سیستم شما ایجاد می شود که در Device Manager ویندوز نیز قابل مشاهده است. برای تبادل داده با هر پورت سریالی ها باید آن را در برنامه باز کنید. باز کردن یک پورت یعنی برقرار کردن ارتباط بین برنامه و پورت سریال مورد نظر. پس پورتی که باز شده است، می توانید داده آن را دریافت و یا از طریق آن داده ارسال کنید. برای باز کردن پورت باید متد مربوطه را در C# فراخوانی کنید.



تصویر _ ۱۸ پورت های سریال در Device Manager ویندوز (COM12 , COM13)

برای باز کردن پورت از یک Button استفاده میکنیم که کاربر با کلیک روی آن، پورت را باز کند. از پنجره Toolbox یک Button به فرم خود درگ کنید تا به فرم شما اضافه شود و ویژگی های آن را در پنجره Properties به مانند زیر تغییر دهید.

(Name) btnOpen

(Text) Open Port

سپس با دابل کلیک روی دکمه Open Port به پنجره Code بروید و در متد رویداد کلیک این Button کد های زیر را وارد کنید.

```
private void btnOpen_Click(object sender, EventArgs e)
{
 serialPort1.DataBits = 8;
 serialPort1.Parity = Parity.None;
 serialPort1.StopBits = StopBits.One;
 serialPort1.BaudRate = 9600;
 serialPort1.PortName = "COM1";
 serialPort1.Open();
}
```

این کدها به این مفهوم است که هنگام اجرای این برنامه با کلیک روی دکمه Open Port ، این پورت پس از تنظیم ویژگی های زیر:

۸ بیت داده[1]

بدون بیت پریتی[2]

۱ بیت پایان[3]

باود ریت[4] ۹۶۰۰

نام پورت COM1

باز می شود و آماده دریافت و ارسال داده است.

serialPort1 نام کامپونت پورت سریال است که روی فرم قرار داده اید، و ویژگی های آن با کلیک روی دکمهOpen Port، در فرم برنامه تان هنگام اجرای آن، همانطور که در بالا گفته شد تنظیم می شود.

توجه داشته باشبد اگر شما می خواهید با هر پورتی، و یا هر تنظیماتی ارتباط برقرار کنید تنظیمات خود را وارد کنید و به نام پورتی که باز می کنید به اندازه کافی توجه کنید شاید نام پورت سریال شما در سیستمتان COM5 باشد!

برای استفاده از متد های کامپونت سریال باید فضای نام[5] آن را به برنامه معرفی کنیم. در اول سورس برنامه خط زبر را اضافه کنید[6].

```
using System.IO.Ports;
```

[1] Data

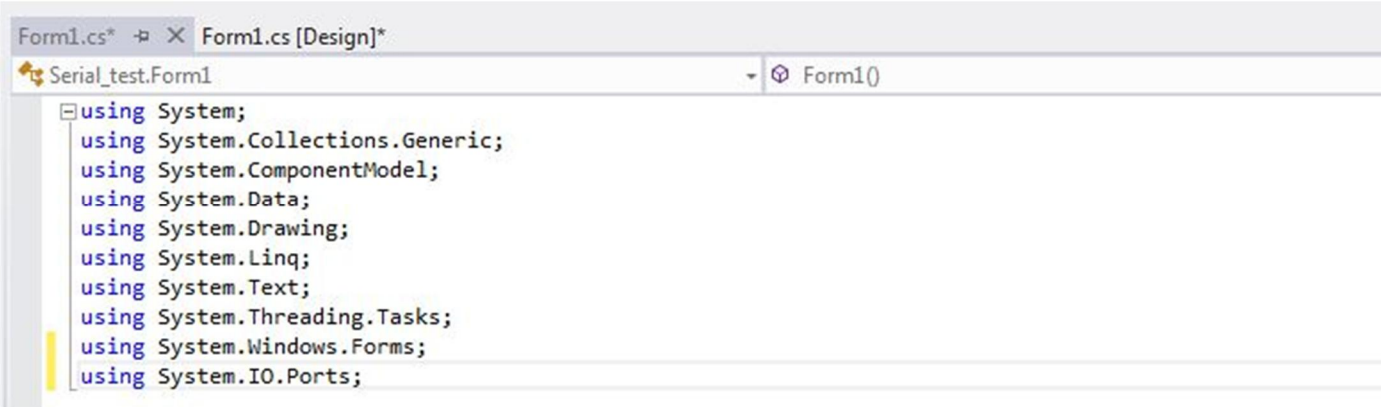
[2] Parity

[3] Stop Bit

[4] Baud Rate

[5] Name Space

[6] مانند #include <name.h> در زبان C



تصویر _ ۱۹ برای کار با کامپونت پورت سریال باید فضای سیستمی مورد نظر در برنامه تعریف شود.

بستن پورت

پورتی که باز است در هیچ برنامه دیگری قابل دستیابی نیست. پس باید پس از پایان کار با هر پورتی، پورت مورد نظر را بست و دسترسی به آن را آزاد کرد. برای بستن پورت از یک Button دیگر استفاده میکنیم که کاربر با کلیک روی آن، پورت را ببندد پس یک Button از Toolbox به فرم خود درگ کنید تا به فرم شما اضافه شود و ویژگی های آن را در پنجره Properties به مانند زیر تغییر دهید.

(Name) btnClosePort

(Text) Close Port

کد های زیر را در متد رویداد کلیک دکمه Close Port بنویسید.

```
private void btnClosePort_Click(object sender, EventArgs e)
{
  serialPort1.Close();
  MessageBox.Show("PORT Closed", "OK", MessageBoxButtons.OK);

}
```

با اجرای این متد، اول متد serialPort1.Close() اجرا می شود و پورت باز شده را، می بندد. سپس پنجره پیام[1]، با عنوان[2] OK و متن PORT Closed نشان داده می شود. توجه کنید بسیاری از مواردی که استفاده میکنیم، مانند نمایش پنجره های پیام تنها برای راهنمایی کاربر است و حتی بدون این موارد هم برنامه شما برای ارتباط با پورت سریال کامل است.

[1] Message Show

[2] Title

ارسال داده

بعد از باز شدن پورت می توان ارسال یا دریافت داده را انجام داد. چندین متد برای ارسال داده از طریق پورت سریال در C# وجود دارد که در بخش چهارم کاملا توضیح داده شده اند. در این قسمت از یک متد ساده برای ارسال داده از طریق پورت سریال استفاده میکنیم.

برای ارسال داده، یک Button دیگر به فرم خود اضافه کنید و ویژگی های آن را مانند زیر، وارد کنید.

(Name) btnSend
(Text) Send Data

سپس در قسمت رویداد کلیک این کلید کد های زیر را وارد کنید.

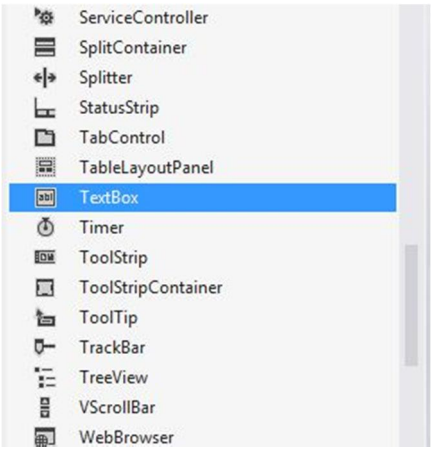
```
private void btnSend_Click(object sender, EventArgs e)
{
  serialPort1.WriteLine("TEST SERIAL");
}
```

این متد، متن TEST SERIAL از طریق پورت مورد نظر ارسال می کند. فقط توجه داشته باشید قبل از ارسال باید پورت مورد نظر باز شده باشد. یعنی باید در زمان اجرای برنامه، اول روی دکمه Open Port کلیک کنید تا پورت مورد نظر باز شود. سپس با کلیک بر روی دکمه Send Data، داده را از طریق پورت باز شده ارسال کنید.

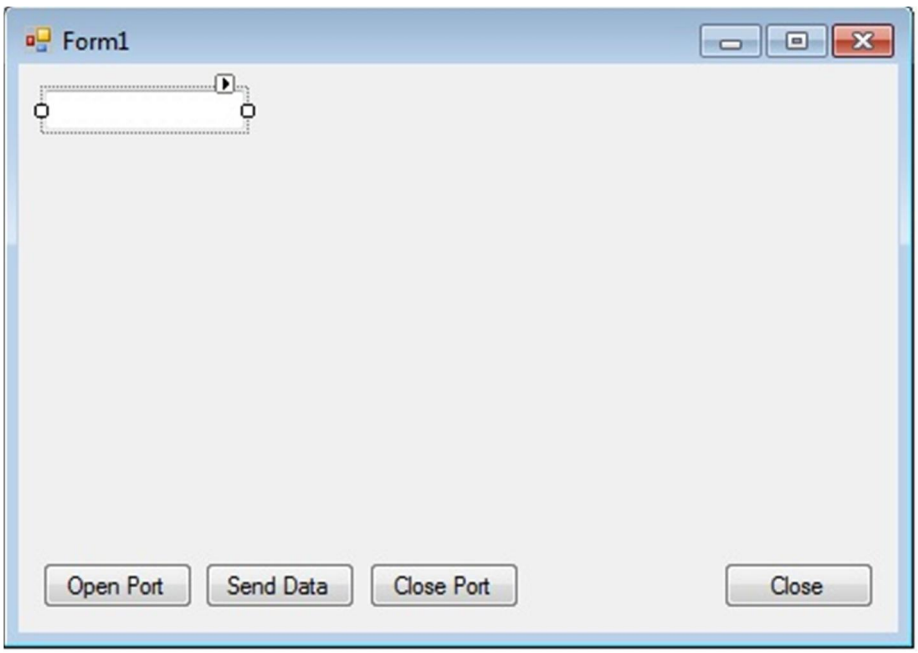
می خواهیم به جای ارسال این متن ثابت، متن وارد شده در یک Textbox را ارسال کنیم. پس از پنجره Toolbox یک Textbox را روی فرم درگ می کنیم و از پنجره Properties ویژگی های آن را به صورت زیر تغییر می دهیم.

(Name) textBox1
(Multiline) True

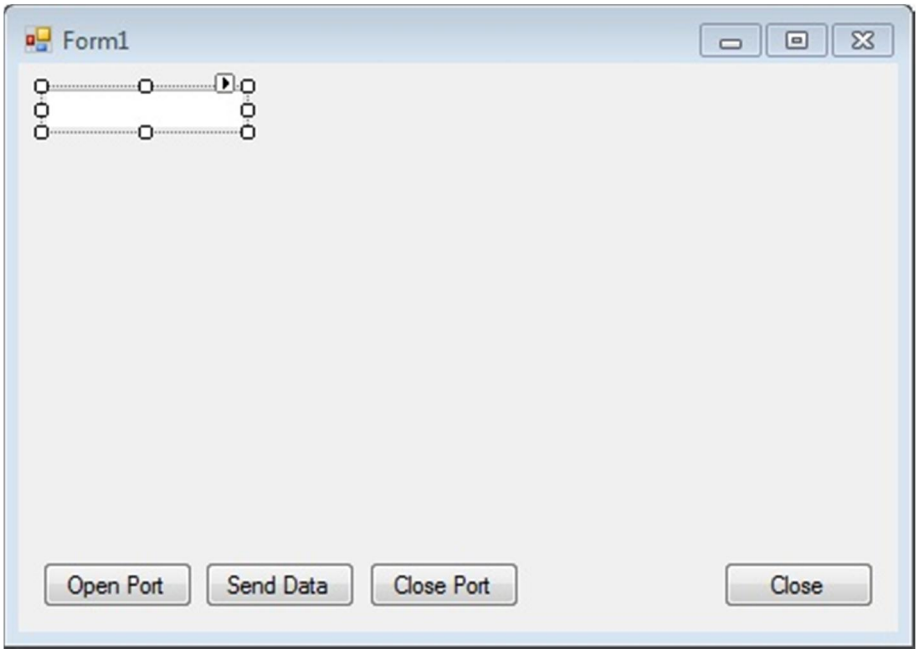
ویژگی نام که لازم به توضیح نیست و هرجایی که خواستیم در برنامه از این Textbox استفاده کنیم از نام آن استفاده می کنیم. و ویژگی Multiline به ما این توانایی را میدهد که از چند خطی بودن آن استفاده کنیم چون درحالت اولیه فقط یک خط از متنی که درون آن است نمایش داده می شود. اما پس از این تنظیم می توانید با درگ کردن گوشه های آن را به اندازه دلخواه تنظیم و چند خطی شدن آن را ببینید.



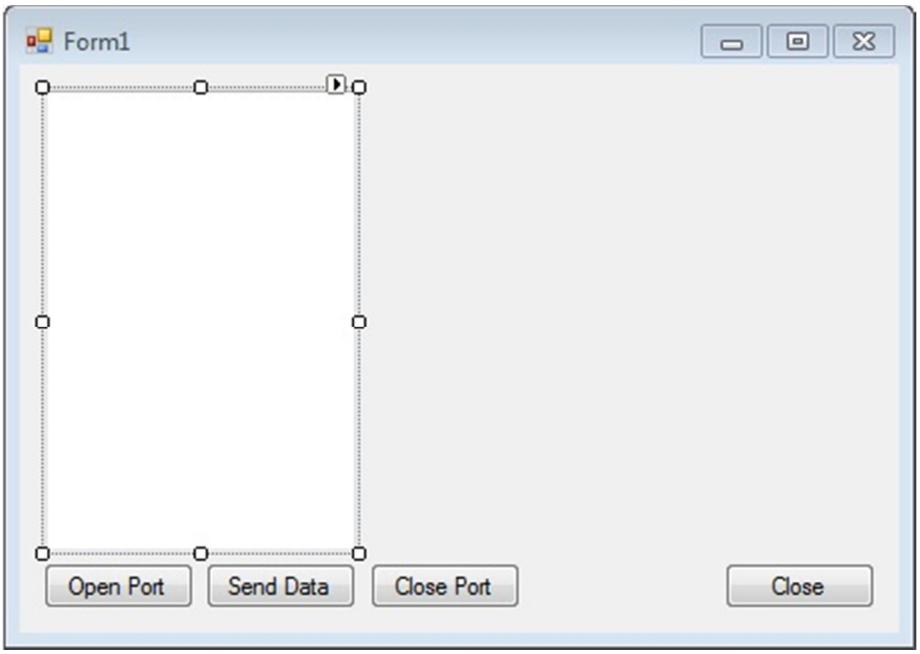
تصویر _ ۲۰ **Textbox** در پنجره **Toolbox**



تصویر _ ۲۱ یک **Textbox** قبل از چند خطی شدن



تصویر _ 22 یک Textbox از چند خطی



تصویر _ ۲۳ تنظیم اندازه Textbox چند خطی

روی دکمه Send Data دابل کلیک کرده و در متد فرایند کلیک این Button به جای کد قبلی که یک متن ثابت را ارسال می کرد، کد زیر را وارد کنید.

```
private void btnSend_Click(object sender, EventArgs e)
{
  serialPort1.WriteLine(textBox1.Text);
```

```
}
```

این کد، یعنی با کلیک روی این Button متنی که در textBox1 وارد شده از طریق پورت سریالی که قبلا باز شده، ارسال شود. متد ()WriteLine یک رشته را از طریق پورت مورد نظر ارسال می کند و در پایان ارسال، کاراکتر NewLine[1] را ارسال میکند.

پس در هنگام اجرای برنامه متن مورد نظر را در textBox1 وارد کنید و پس از باز کردن پورت آن را ارسال کنید.

دریافت داده

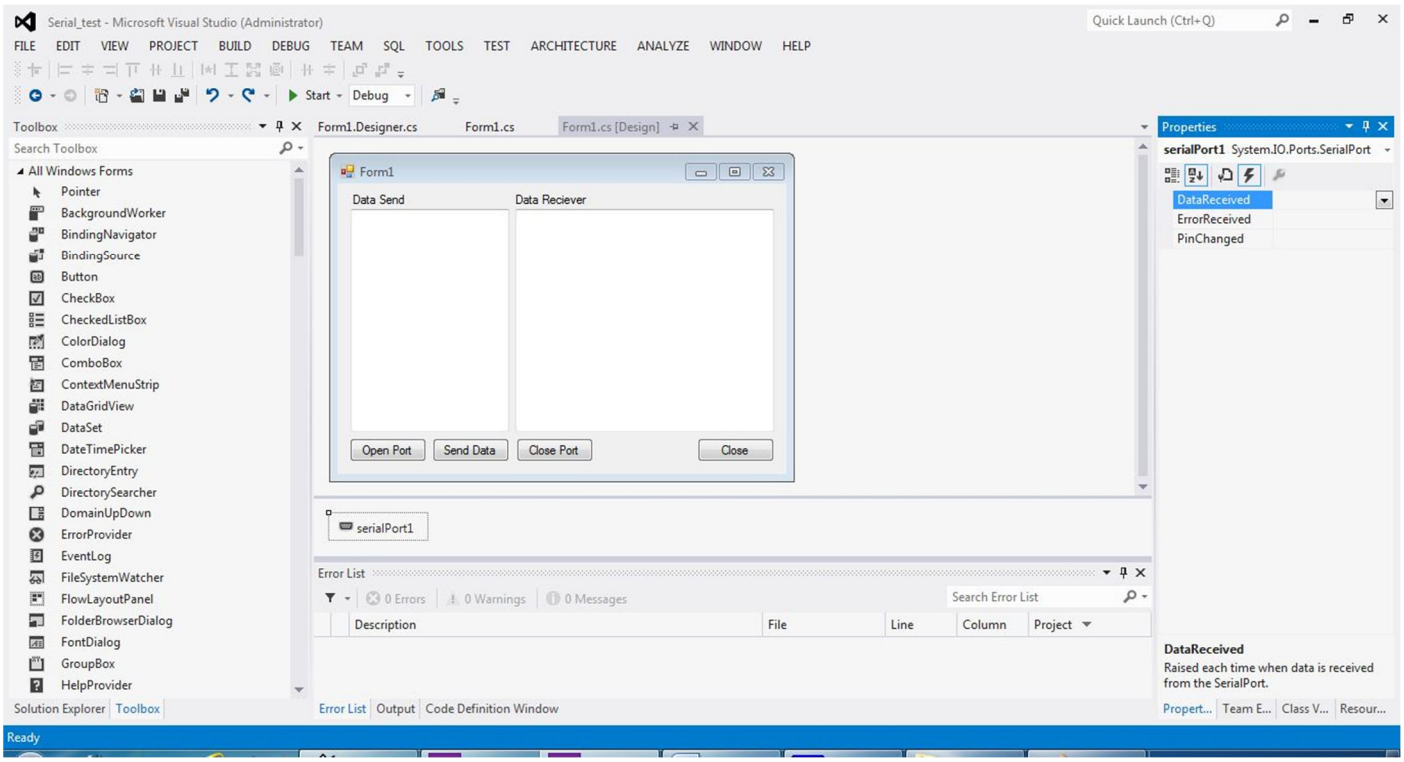
کد نویسی برای دریافت داده کمی با ارسال آن تفاوت دارد، چون شما هر زمان که مایل بودید می توانستید با کلیک روی دکمه Send Data متن وارد شده در textBox1 را ارسال کنید. اما برای دریافت برنامه باید به گونه ای باشد که هنگامی که فرستنده آن را ارسال میکند بدون دخالت کاربر داده را دربافت و اگر لازم شد نشان دهد.

برای نمایش داده دریافت شده نیاز به یک Textbox داریم پس یک Textbox دیگر به فرم خود اضافه کنید وبژگیهای آن را مانند زیر تنظیم کنید.

(Name) textBox2
(Multiline) True

[1] کد اسکی این کاراکتر '\n' برابر 0x0A است. کد اسکی دکمه Enter صفحه کلید کامپیوتر 0x0D و کاراکتر آن '\r' می باشد.

سپس در صفحه Design برنامه روی SerialPort کلیک کنید و در بالای پنجره Properties، روی آیکن رویدادها کلیک کنید[1]. و از رویداد های پیش رو، روی رویداد DataRecieve دابل کلیک کنید.



تصویر _ ۲۴ افزودن متد serialPort1_DataReceived به برنامه

می بینید که متد این رویداد نیز به قسمت کد های ما اضافه می شود.

```
private void btnSend_Click(object sender, EventArgs e)
{
    serialPort1.WriteLine("TEST SERIAL");
}

private void serialPort1_DataReceived(object sender, SerialDataReceivedEventArgs e)
{

}
```

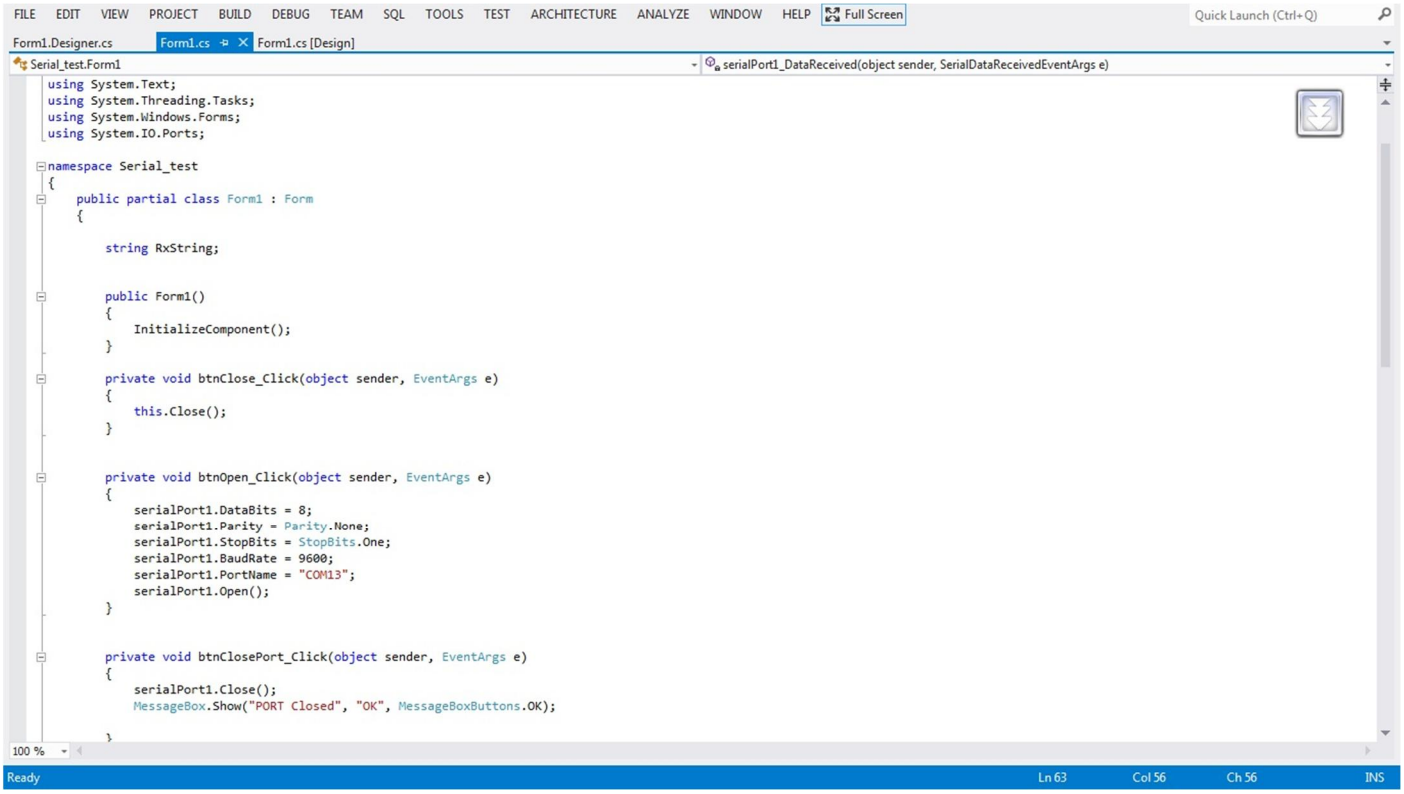
تصویر _ ۲۵ اضافه شدن رویداد دریافت داده به کد های برنامه

نخست، یک متغیر از نوع رشته تعریف کنید تا داده دریافت شده در آن قرار بگیرد. این متغیر باید در بدنه کلاس فرم تعریف شود چون هم در متد رویداد DataReceived و هم در متد DisplayText از آن استفاده می شود.

[1] آیکن شبیه صاعقه!

```
public partial class Form1 : Form
{
  string StrRecieve;
```

مشاهده می کنید که همه متد ها در این بدنه تعریف شده اند.



تصویر _ ۲۶ مشاهده میکنید که همه متد ها در این بدنه تعریف شده اند.

برای نمایش متن به وسیله این رویداد باید یک متد تعریف کنیم و آن را قبل از متد رویداد DataReceived قرار دهید.

```
private void DisplayText(object sender, EventArgs e)
{
  textBox2.AppendText(StrRecieve);
}
```

کد های زیر را در متد رویداد serialPort1_DataReceived مینویسیم.

```
private void serialPort1_DataReceived(object sender, SerialDataReceivedEventArgs e)
{
  string StrRecieve;
  StrRecieve = serialPort1.ReadExisting();
  this.Invoke(new EventHandler(DisplayText));
}
```

متد این رویداد زمانی که داده ی جدید برای دریافت آماده است اجرا می شود و داده را دریافت و در رشته StrRecieve قرار می دهد. سپس متد DisplayText را افراخوانی می کند و این متد رشته دریافت شده را در textBox2 قرار می دهد. توجه داشته باشید که، اگر در خود رویداد serialPort1_DataReceived اقدام به نمایش متن کنید برنامه شما با ایراد مواجه می شود، چون این رویداد از Thread با اولویت بالا استفاده می کند، نمیتوانید در این متد به کنترل ها و یا ابزار دسترسی داشته باشید. پس باید در این متد داده دریافت شده را در رشته StrRecieve قرار دهید و برای نمایش داده، متد DisplayText را فراخوانی کنید. متد DisplayText داده دریافت شده را آن را به متن textBox2 می افزاید.

به جای دستور AppendText شما می توانید از دو نمونه زیر استفاده کنید. هر یک از دو دستور زیر، رشته StrRecieve را به ادامه متن textBox2 می افزایند. یعنی رشته StrRecieve به ادامه متن textBox2 اضافه می شود بدون اینکه آنچه که از قبل در آن نوشته شده پاک شود.

```
textBox2.AppendText(StrRecieve)
textBox2.text = textBox2.text+StrRecieve
textBox2.text += StrRecieve
```

سورس پایانی برنامه.

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.IO.Ports;

namespace Serial_test
{
  public partial class Form1 : Form
  {

    string StrRecieve;


    public Form1()
    {
      InitializeComponent();
```

```
    }


    private void btnClose_Click(object sender, EventArgs e)
    {
      this.Close();
    }


    private void btnOpen_Click(object sender, EventArgs e)
    {
      serialPort1.DataBits = 8;
      serialPort1.Parity = Parity.None;
      serialPort1.StopBits = StopBits.One;
      serialPort1.BaudRate = 9600;
      serialPort1.PortName = "COM13";
      serialPort1.Open();
    }


    private void btnClosePort_Click(object sender, EventArgs e)
    {
      serialPort1.Close();
      MessageBox.Show("PORT Closed", "OK", MessageBoxButtons.OK);
     }


     private void btnSend_Click(object sender, EventArgs e)
     {
       serialPort1.WriteLine(textBox1.Text);
     }

     private void DisplayText(object sender, EventArgs e)
     {
     textBox2.AppendText(StrRecieve);
     }

     private void serialPort1_DataReceived(object sender, SerialDataReceivedEventArgs
e)
     {
     StrRecieve = serialPort1.ReadExisting();
     this.Invoke(new EventHandler(DisplayText));
```

```
    }

  }
}
```

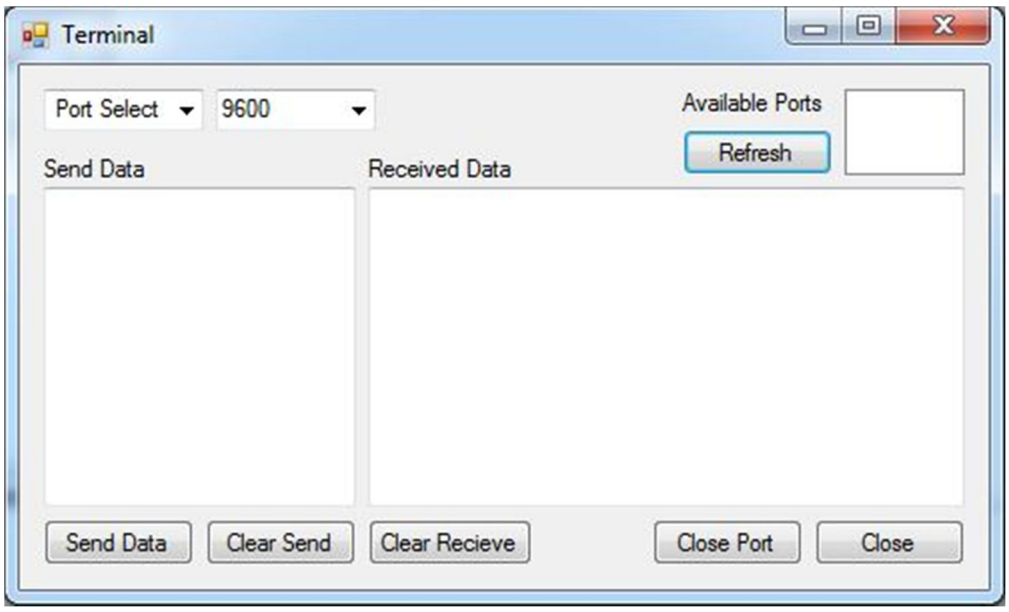
اگر نام پروژه و ابزار استفاده شده شما با پروژه من یکسان باشد، می توانید همه کد بالا را یکجا به صفحه Code پروژه خود COPY+PASTE کنید. اما توصیه می کنم هر بار تنها دستورات درون هر متد را کپی کنید و به عملکرد آنها توجه کنید. فهمیدن این کد ها بسیار ساده است. برای یاد گرفتن برنامه نویسی بهترین کار بدست آوردن کد های آماده و فهمیدن کار آنها و چگونگی استفاده از آن ها است و تلاش کنید با تغییر کد ها به عملکرد مورد نظرتان برسید[1].

[1] این جملات را در کتاب AVR آقای امیر ره افروز خوندم که واقعا برام مفید بود. البته متن من شاید کمی با آقای ره افروز فرق داشته باشه چون دقیقا جملات یادم نیست.

بخش سوم: پروژه های پورت سریال

در این بخش دو پروژه را مرحله به مرحله می سازیم. در پروژه یکم ساخت یک ترمینال سریال[1] را یاد خواهید گرفت. یادآوری میکنم پروژه هایی که در این کتاب ساخته میشوند را میتوانید همراه این نوشته دانلود کنید.

پروژه یکم: ساخت یک ترمینال

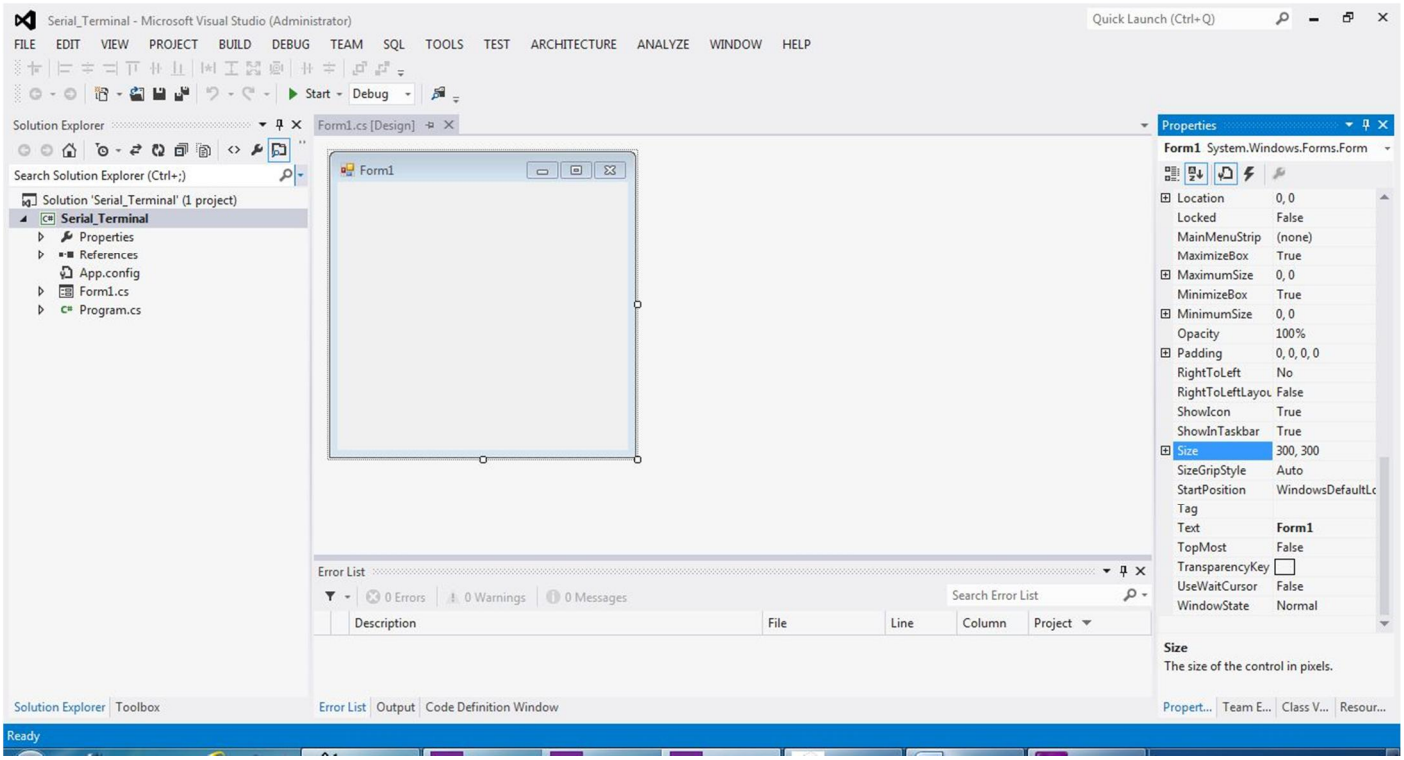


تصویر _ ۲۷ نمای پایانی برنامه ترمینال پورت سریال

[1] ترمینال سریال برنامه هایی هستند که از با نوشتن متن در محیط آنها کد های اسکی هرکلید از طریق پورت سریال ارسال می شود و داده دریافتی از پورت سریال نیز به صورت متن در همان صفحه نمایش داده می شود.

در این قسمت مرحله به مرحله یک ترمینال برای تبادل داده از طریق پورت سریال می سازیم.

از مسیر FILE\New\Project یک پروژه جدید ایجاد کنید و نام پروژه را Serial_Terminl بگذارید.



تصویر _ ۲۸ ایجاد یک پروژه جدید

در پنجره Properties در صفحه Design پروژه، ویژگی Size فرم را برابر 500,300 قرار دهید. و ویژگی Text را Terminal بنویسید. همچنین ویژگی MinimunSize آن را 450,250 قرار دهید تا در هنگام اجرای برنامه، اجازه داده نشود که کاربر صفحه برنامه را از این مقدار کوچکتر کند، چون چینش کنترل ها در صفحه به هم می خورد.

(Size) 500,300

(Text) Terminal

ساخت دکمه خروج

یک Button را در محل مناسب[1] روی فرم، برای خروج از برنامه قرار دهید و ویژگیهای آن را به صورت زیر تغییر دهید.

(Name) btnClose

(Text) Close

در رویداد کلیک[1] دکمه Close دستور زیر را بنویسید.

[1] محل مناسب برای دکمه خروج گوشه پایین سمت چپ فرم است!

```
Private void btnClose_Click(object sender, EventArgs e)
{
  this.Close();
}
```

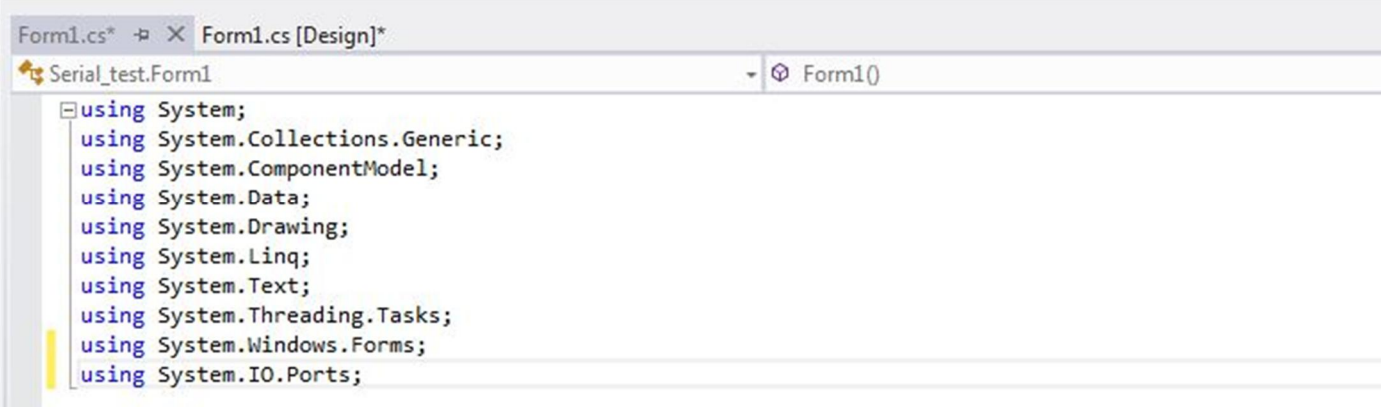
همانطور که در بخش قبلی گفته شد کد this.Close() پنجره جاری را در هنگام اجرای برنامه می بندد.

وارد کردن کامپونت پورت سریال

همانگونه که در بخش قبلی گفتیم برای کار با پورت سریال باید کامپونت مربوط به این پورت به برنامه اضافه شود. برای وارد کردن کامپونت مربوط به این پورت، از پنجره Toolbox و از بخش Componets کنترل SerialPort را روی فرم، درگ کنید[۲].

فضای نام[۳] مورد نیاز برای کار با این کامپونت را به سورس اضافه کنیم پس در اول سورس برنامه خط زبر را اضافه کنید.

```
using System.IO.Ports;
```



تصویر _ ۲۹ اضافه کردنق فضای مورد استفاده کامپونت پورت سریال

سپس دو متغیر زیر را در بدنه کلاس فرم مانند زیر تعریف کنید. که بعدا از آنها استفاده میکنیم.

```
public partial class Form1 : Form
{
  string StrRecieve;
  string strbaud;
```

[۱] با دابل کلیک روی Button، به صفحه Code وارد میشوید.

[۲] در ویژوال استدیو ۲۰۱۲ در بالای Toolbox قسمت جستجو وجود دارد که برای پیدا کردن راحت تر کنترل ها می توان از آن استفاده کرد.

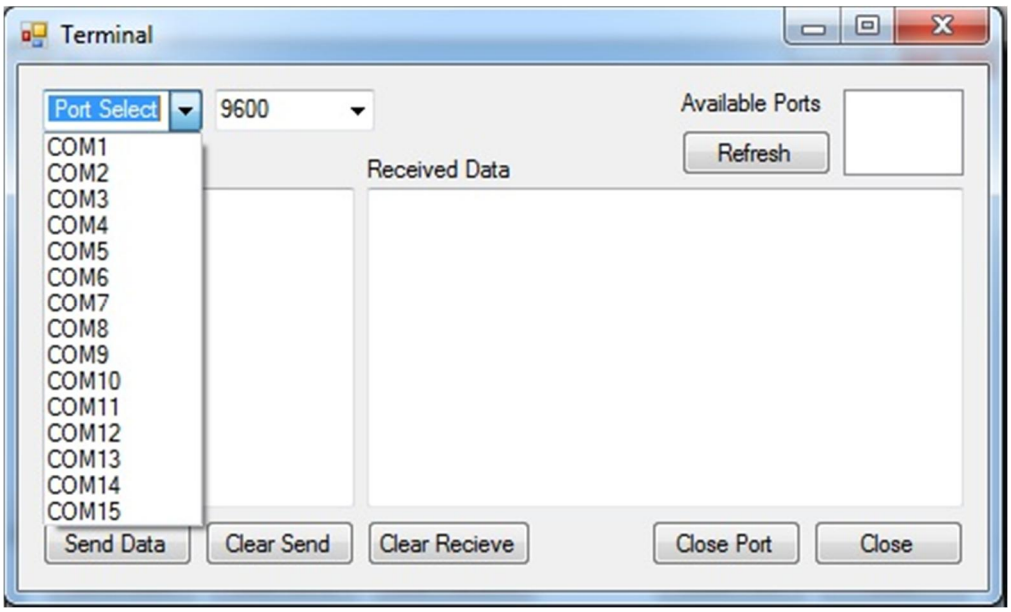
[3] Name Space

باز کردن پورت

در مثال پیشین امکان باز کردن تنها یک پورت وجود داشت که نام مستقیما در سورس برنامه وارد شده بود.

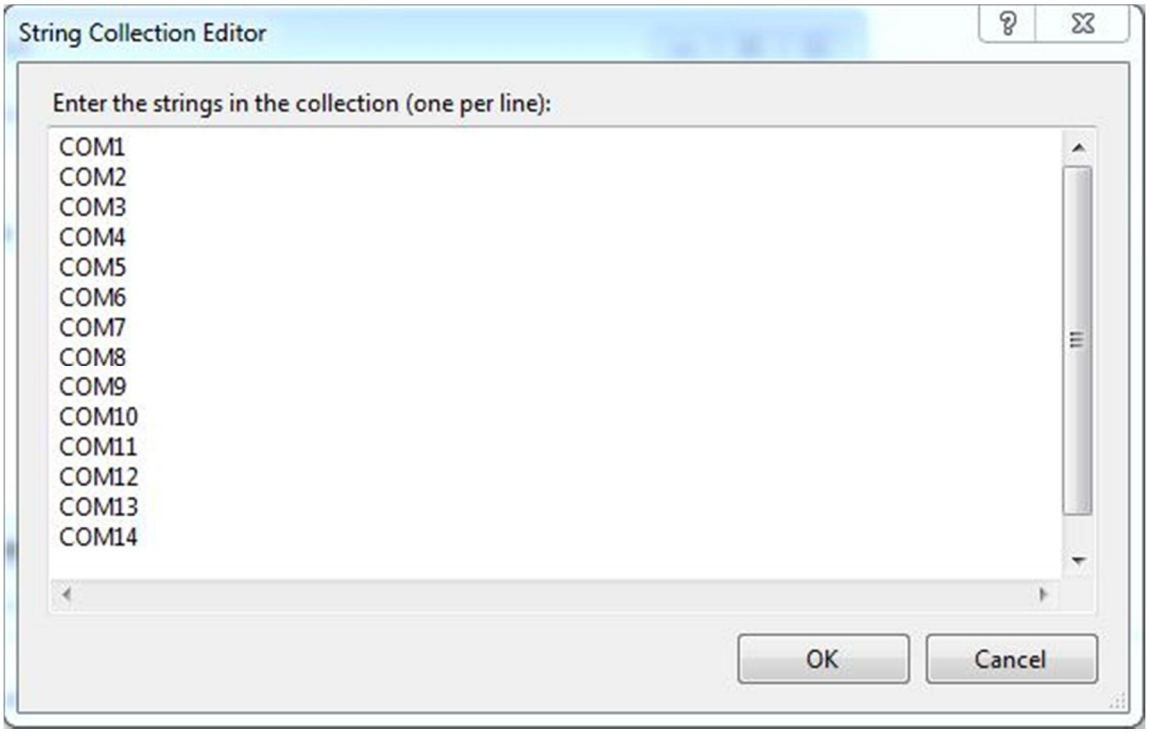
```
serialPort1.PortName = "COM1";
```

اما در این پروژه میخواهیم امکان باز کردن هر پورتی در برنامه وجود داشته باشد. پس برای باز کردن پورت از یک ComboBox استفاده میکنیم. تا با باز کردن ComboBox و کلیک روی نام پورت بتوانیم پورت مورد نظر را در برنامه باز کنیم.



تصویر _ ۳۰ منو آبشاری نام پورت ها

از پنجره Toolbox یک comboBox به فرم بکشید. اندازه و محل قرارگیری آن را تنظیم و ویژگی های آن را در پنجره Properties تنظیم کنید. ویژگی Text را Port Select و سپس روی ویژگی Items کلیک کنید و مانند تصویر نام پورتهای مورد استفاده را وارد کنید. سپس روی OK کلیک کنید.



تصویر _ ۳۱ اضافه کردن نام پورت ها به ویژگی Items

نام همه پورتهای مورد استفاده در این منوی آبشاری وارد شده تا کاربر با باز کردن آن و کلیک روی نام هر پورت، پورت مورد نظرش را باز کند.

در پنجره Design روی comboBox دابل کلیک کنید تا وارد پنجره Code شوید و متد فرایند مربوط به آن به صورت خودکار ایجاد شود. در متد فرایند comboBox1_SelectedIndexChanged کد زیر را وارد کنید.

```
private void comboBox1_SelectedIndexChanged(object sender, EventArgs e)
{

  strbaud = comboBox2.Text;
  serialPort1.Close();

  try
  {
    serialPort1.DataBits = 8;
    serialPort1.Parity = Parity.None;
    serialPort1.StopBits = StopBits.One;
    serialPort1.BaudRate = int.Parse(strbaud);
    serialPort1.PortName = comboBox1.Text;
    serialPort1.Open();
    serialPort1.DiscardInBuffer();
    label1.BackColor = System.Drawing.Color.Green;
```

```
    label1.ForeColor = System.Drawing.Color.White;
    label1.Text = "( " + comboBox1.Text + " )" + "   CONNECTED ";
    listBox1.Text = comboBox1.Text;

  }

  catch
  {
    label1.Text = "Disconnected";
    label1.BackColor = System.Drawing.Color.Red;
    label1.ForeColor = System.Drawing.Color.White;
    MessageBox.Show("Can't Access "+ "(" + comboBox1.Text + ")", "ERROR",
MessageBoxButtons.OK, MessageBoxIcon.Error);

  }
}
```

توضیح خط به خط برنامه

```
private void comboBox1_SelectedIndexChanged(object sender, EventArgs e)
```

این متد همانطور که از نام آن پیدا است زمانی اجرا می شود که تغییری در ایندکس مورد نظر روی دهد. یعنی هربار که کاربر منو آبشاری را باز کند و روی یکی از نام های موجود کلیک کند، این متد اجرا می شود.

```
  strbaud = comboBox2.Text;
```

اگر یادتان باشد متغیر strbaud را در اول برنامه تعریف کردید. این متغیر از نوع رشته است و متن مورد انتخاب شده، در comboBox2 را که بعدا به فرم اضافه میکنیم به عنوان باودریت در خود نگه می دارد.

```
  serialPort1.Close();
```

شما یک کامپونت با نام serialPort1 ایجاد کردید که برای تنظیمات از نام آن استفاده می کنیم. این دستور پورت را می بندد و در ادامه پس از اعمال تنظیمات جدید آن را با تنظیمات جدید باز می کنیم.

```
  try
  {
```

try به زبان ساده؛ اگر در زمان اجرای دستورات نوشته شده درون بلاکش[1] مشکلی به وجود آید، ادامه دستورات متوقف و دستورات درون بلاک catch اجرا می شود. برای مثال اگر در زمان اجرای برنامه شما پورتی را باز کنید که وجود ندارد، چون برنامه نمی تواند

[1] به دستورات بین دو آکولاد، یک بلاک می گویند {} (البته ما هم می گوییم)

این پورت را باز کند، با ایراد مواجه می شود، پس ادامه دستورات بلاک try متوقف و دستورات نوشته شده در بلاک catch اجرا میشوند. با اجرای دستورات بلاک catch یک پنجره پیام خطا نمایش داده می شود و از این راه ایراد را به کاربر گزارش میدهد.

```
serialPort1.DataBits = 8;
```

انتخاب تعداد برابر بیت داده برابر ۸ عدد. این عدد میتواند ۵، ۶، ۷، ۸، و یا ۹ باشد وشما میتوانید این مقدار را مثلا از یک comboBox وارد کنید. فقط توجه کنید که این عدد، یک مقدار Int32 یا عدد صحیح باشد نه رشته و اگر مقدار آن را از comboBox وارد میکنید، حتما قبل از قرار دادن، آن را به int تبدیل کنید.

```
serialPort1.Parity = Parity.None;
```

بدون بیت پریتی[1]. بیت پریتی می تواند به صورت Even, Mark, None, Odd, Spase طبق الگوی بالا تنظیم شود. اگر شما کدهای بالا را خودتان در ویژوال استدیو بنویسید میبینید که با نوشتن هر کاراکتر، AutoComplete برنامه، انتخاب های ممکن را در اختیار شما قرا میدهد. کافی است شما در نوشتن کد بالا Parity را بنویسید. به محض نوشتن نقطه بعد از نوشتن آن، همه گزینه های ممکن شامل ۵ انتخاب بالا برای شما به نمایش در میاید[2].

```
try
{
    serialPort1.DataBits = 8;
    serialPort1.Parity = Parity.
    serialPort1.StopBits = StopB
    serialPort1.BaudRate = int.P                    );
    serialPort1.PortName = combo
    serialPort1.Open();
    serialPort1.DiscardInBuffer(
    label1.Text = "( " + comboBo                    )" + "    CONNECTED ";
    label1.BackColor = System.Drawing.Color.Green;
    label1.ForeColor = System.Drawing.Color.White;
    listBox1.Text = comboBox1.Text;
}
```

Even
Mark
None
Odd
Space

Parity Parity.None
No parity check occurs.

تصویر _ ۳۲ AutoComplete در ویژوال استدیو و توضیحات مربوط به هر انتخاب، در جلوی آن

[1] پریتی زوج؛ تعداد ۱ های داده شمرده میشه و اگر تعداد آنها زوج بود، مقدار ۰ و اگر تعداد فرد بود مقدار ۱ به عنوان پریتی، در نظر گرفته می شود. شما این را به این صورت در نظر بگیرید که کلا تعداد بیت های داده و پریتی زوج باشه. پس زمانی که تعداد بیت های داده فرد بود پریتی ۱ میشه تا قاعده برقرار بشه. قاعده پریتی فرد هم مثل پریتی زوج است با این تفاوت که تعداد کل باید فرد باشه. پریتی ۱ یا Mark این بیت همیشه ۱ است و کاری به تعداد یا هیچ پارامتر دیگه ای نداره. پریتی صفر یا Space هم که همیشه یک بیت ۰ ارسال میشه. بدون پریتی هم یعنی در قالب ارسال هیچ بیت پریتی ارسال نمی شود. البته بیت پریتی فقط از سمت فرستنده ارسال نمی شود؛ بلکه در سمت گیرنده بعد از دریافت داده با مقداری که انتظار آن میرود مقایسه می شود و اگر نا هماهنگ بود منجر به خطای دریافت داده می شود.

[2] به هر حال پیشرفت تکنولوژی داره کارها را بر انسان ساده میکند. البته فعلا به وجود آدمی زاد نیاز داره خدا اون روز رو نیاره واسه خودش کسی بشه. البته توصیه من اینه که همیشه از این ابزار و امکانات استفاده کنید. Help هر برنامه ای و Sample های همراه هر برنامه ای میتواند بسیار سودمند و کمک کننده باشد. به ویژه ما که همیشه از نبود یک منبع خوب رنج میبریم این ها را غنیمت بشمارید و از وقت با ارزشتون برای کار های مفید تر استفاده کنید. در کل سنتی فکر نکنید باید قبول کنیم نقش انسان در دنیای امروز کمی تغییر کرده در پی آنچه به شما نیاز است، بهترین باشید!.(نگارنده پیر دانا می شود)

```
serialPort1.StopBits = StopBits.One;
```

یک بیت پایان. بیت پایان میتواند صورت None, One, OnePointFive, Two تنظیم شود.

```
serialPort1.BaudRate = int.Parse(strbaud);
```

این دستور باودریت را تنظیم می کند. رشته strbaud با استفاده از int.Parse(strbaud) به int تبدیل می شود. باید برای تنظیم باودریت از یک مقدار عددی صحیح استفاده کنیم.

```
serialPort1.PortName = comboBox1.Text;
```

نام پورت نیز از مورد انتخاب شده در comboBox1 گرفته میشود و این ویژگی کامپونت نیز تنظیم میشود.

```
serialPort1.Open();
```

این دستور پورت را باز می کند.

```
serialPort1.DiscardInBuffer();
```

این دستور بافر ورودی پورت را خالی می کند. یعنی اگر از ارتباط قبلی، داده ای در آن مانده باشد، پاک می شود.

```
label1.Text = "( " + comboBox1.Text + " )" + "   CONNECTED ";
```

تا این مرحله اگر همه دستورات بالا بدون ایراد اجرا شده باشند پورت، به درستی باز شده است و شما در label1 یک متن مشاهده میکنید. که نام پورت و عبارت CONNECTED در جلوی آن نمایش داده میشود برای مثال:

(COM3) CONNECTED

در C# کار با رشته ها بسیار ساده است کافی است برای اتصال دو رشته از + استفاده کنیم. دستور بالا ویژگی label1.Text که واضح است باید از جنس رشته باشد را برابر عبارت:

```
"( " + comboBox1.Text + " )" + "   CONNECTED
```

قرار میدهد. هر کنترل یا ابزاری دارای ویژگی های گوناگونی است که در دستور بالا ویژگی متن لیبل را تغییر دادیم. در دستورات زیر میبینید که چگونه دو ویژگی دیگر label1 را تغییر میدهیم.

```
label1.BackColor = System.Drawing.Color.Green;
```

این دستور ویژگی رنگ پس زمینه متن نوشته شده در لیبل را، سبز میکند.

```
label1.ForeColor = System.Drawing.Color.White;
```

این دستور ویژگی رنگ خود متن نوشته شده در لیبل را، سفید میکند.

```
listBox1.Text = comboBox1.Text;
```

این دستور نام انتخاب شده در listBox1 را برابر نام انتخاب شده در comboBox1 میکند. چون می خواهیم اگر با استفاده از comboBox1 پورتی برای باز شدن انتخاب شد نام همین پورت درlistBox1 نیز در وضعیت انتخاب شده[1] قرار بگیرد.

```
}

catch
{
```

همانطور که اشاره شد اگر در اجرای دستورات نوشته شده در بلاک try مشکلی پیش آمد ادامه برنامه در آن بلوک قطع و دستورات این بلوک اجرا میشوند.

```
label1.Text = "Disconnected";
label1.BackColor = System.Drawing.Color.Red;
label1.ForeColor = System.Drawing.Color.White;
```

سه دستور بالا عبارت Disconnected را با رنگ متن سفید و پس زمینه قرمز در لیبل۱ نمایش میدهد. پس اگر باز کردن پورت موفقیت آمیز نبود شما این عبارت را درون لیبل مشاهده میکنید.

```
MessageBox.Show("Can't Access "+ "(" + comboBox1.Text + ")", "ERROR",
MessageBoxButtons.OK, MessageBoxIcon.Error);
```

و سرانجام دستور بالا یک پنجره پیام خطا باز می کند و کاربر را از باز نشدن پورت مورد نظر آگاه می کند.

```
  }
}
```

اگر هنگام اجرای برنامه comboBox1 باز شود و روی نام یکی از آیتم های comboBox1 که نام پورت های سریال هستند کلیک شود، متد بالا اجرا می شود. در این مرحله از اجرای برنامه خودداری کنید چون در کد های بالا از کنترل هایی استفاده شده که در ادامه آنها را به فرم اضافه خواهیم کرد.

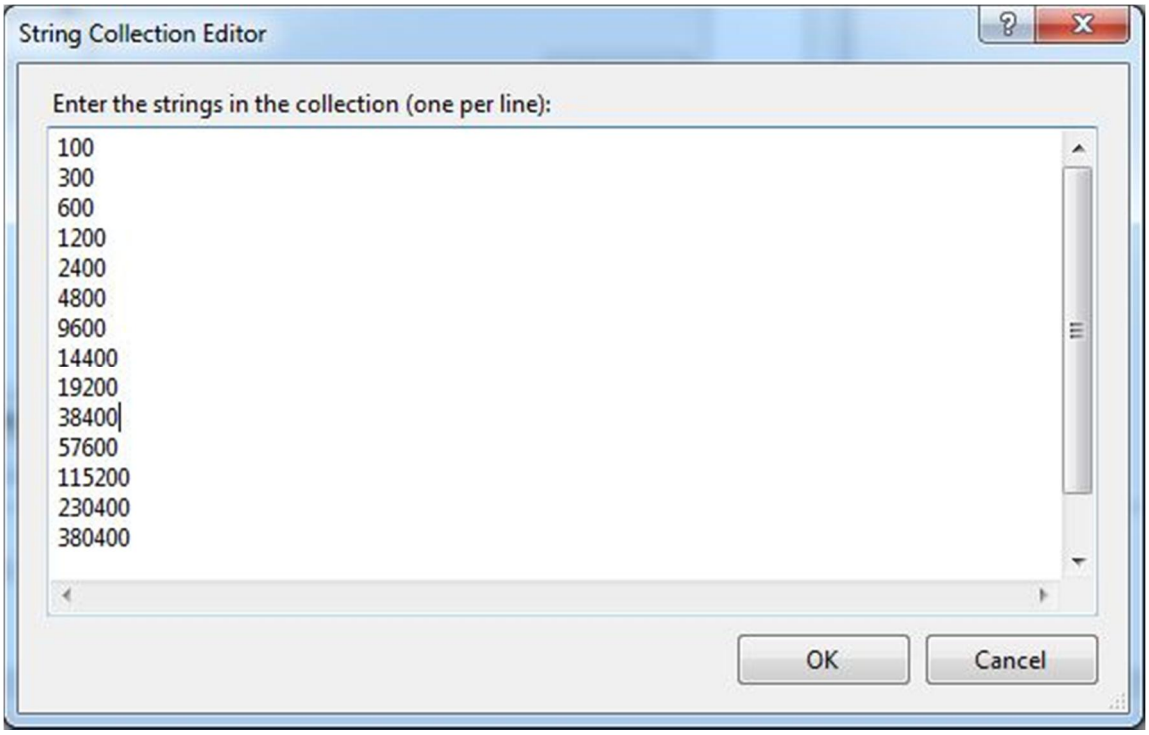
.

انتخاب باودریت

برای انتخاب باودریت مورد نظر مانند انتخاب نام پورت از comboBox استفاده می کنیم. از پنجره Toolbox یک comboBox دیگر به فرم بکشید. اندازه و محل قرارگیری آن را تنظیم و ویژگی Text را **9600** و سپس روی ویژگی Items کلیک کنیدو مانند تصویر باودریت های موجود را وارد کنید و روی OK کلیک کنید. در این comboBox ویژگی Text به عنوان باودریت پیش

[1] Selected: همان است که وقتی روی آیکنی کلیک می کنید پس زمینه آن آبی رنگ می شود.

فرض عمل میکند یعنی اگر کاربر پورتی را باز کند، ولی یکی از باودریت را انتخاب نکند، پورت مورد نظر با باودریت 9600 باز می شود.



تصویر _ ۳۳ اضافه کردن باودریت ها به ویژگی **Items**

روی آن دابل کلیک کنید و در متد فرایند comboBox2_SelectedIndexChanged کد زیر را وارد کنید.

```
private void comboBox2_SelectedIndexChanged(object sender, EventArgs e)
{

  strbaud = comboBox2.Text;

  try
  {
    serialPort1.BaudRate = int.Parse(strbaud);
    serialPort1.DiscardInBuffer();
    label1.BackColor = System.Drawing.Color.Green;
    label1.ForeColor = System.Drawing.Color.White;
    label1.Text = "( " + comboBox1.Text + " )" + "   CONNECTED ";

  }

  catch
```

```
  {
    Label1.Text = "Disconnected";
    label1.BackColor = System.Drawing.Color.Red;
    label1.ForeColor = System.Drawing.Color.White;
MessageBox.Show("Can't Access "+ "(" + comboBox1.Text + ")", "ERROR",
MessageBoxButtons.OK, MessageBoxIcon.Error);
  }
}
```

دستورات درون این متد نیز مشابه متد comboBox**1**_SelectedIndexChanged هستند که کامل توضیح داده شد.

بستن پورت

برای بستن پورت باز شده یک Button دیگر به فرم اضافه کنید و ویژگیهای آن را به صورت زیر تغییر دهید.

(Name) btnClosePort

(Text) Close Port

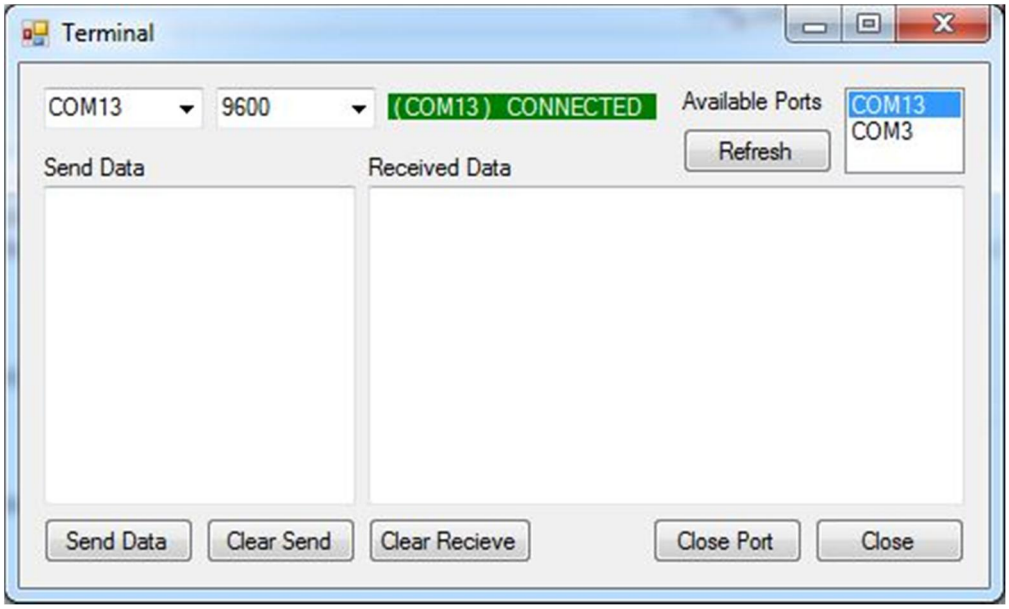
در متد فرایند کلیک این Button کد زیر را وارد کنید.

```
private void btnClosePort_Click_1(object sender, EventArgs e)
{
  serialPort1.Close();
  label1.Text = "Disconnected";
  label1.BackColor = System.Drawing.Color.Red;
  label1.ForeColor = System.Drawing.Color.White;
}
```

نشان دادن وضعیت پورت

در کد های بالا از نام label3 را مشاهده می کنید. از این لیبل برای نشان دادن وضعیت پورت استفاده میکنیم. اگر کاربر پورتی را به صورت موفقیت آمیز باز کند، در این برچسپ[1] عبارت CONNECTED (COM1) به رنگ سفید و در پس زمینه سبز نشان داده خواهد شد. و اگر برنامه نتواند پورتی را باز کند عبارت DISCONNECTED در پس زمینه قرمز در آن نمایش داده می شود. لیبل را از Toolbox روی فرم بکشیدو روبروی comboBox1 قرار دهید. ویژگی Text آن را پاک کنید تا خالی باشد. چون کد های برنامه را بر اساس نام آن مینویسیم حتما توجه کنید که ویژگی نام آن label1 باشد.

[1] Label



تصویر _ ۳٤ باز شدن موفقیت آمیز COM13

نشان دادن پورتهای موجود

این امکان وجود دارد که کاربر از پورتهای سریال موجود در سیستم اطلاع داشته باشد. برای این کار متدی با نام GetPortNames() در فضای نام System.IO.Ports تعریف شده است.

برنامه نویسان میکروکنترلر ها معمولا از مبدل های USB TO SERIAL استفاده می کنند. سیستم برای هر پورت USB که مبدل به آن وصل شود یک پورت سریال مجازی با نام و آدرس متفاوت می سازد. برای مثال اگر سیستم شما ٤ پورت USB داشته باشد وصل کردن مبدل به یکی از USB ها همیشه COM4 و USB دیگر همیشه COM7 و به همین صورت هر یک از USB ها نام و آدرس منحصر به فرد خود را خواهد داشت که البته همیشه ثابت است یعنی USB که COM7 را ایجاد می کند، هر بار که مبدل به این پورت USB وصل شود پورت سریال مجازی همواره COM7 خواهد بود. به این صورت سیستم به هر USB یک آدرس و نام خاص اختصاص میدهد. پس اطلاع از پورت های موجود و نام آن ها لازم است.

برای نمایش نام پورتهای موجود که توسط متد بالا یافته شده اند، از یک ListBox استفاده میکنیم. از پنجره Toolbox یک ListBox به فرم اضافه کنید و ویژگی Size آن را در پنجره Properties برابر **60,43** قرار دهید و محل قرارگیری آن را در گوشه بالا سمت راست فرم مرتب کنید. در این ListBox نام پورتهای موجود نمایش داده می شود.

روی خود فرم دابل کلیک کنید توجه داشته باشید که روی خود فرم دابل کلیک کنید نه هیچ یک از کنترل های قرار گرفته روی فرم تا متد فرابند Form1_Load ایجاد شود. در متد این فرایند دستور های زیر را وارد کنید.

```
private void Form1_Load(object sender, EventArgs e)
{
  foreach (string s in SerialPort.GetPortNames())
  {
    listBox1.Items.Add(s);
```

```
  }
}
```

هر بار که شما برنامه را اجرا کنید متد فرایند Form1_Load یکبار اجرا می شود. در این متد از متد GetPortNames() برای پیدا کردن پورت های موجود استفاده می شود. این متد درون حلقه foreach اجرا می شود. این حلقه شبیه حلقه های for یا while در زبان C می باشد و زمانی از آن استفاده می کنیم که تعداد تکرار حلقه برای ما نا مشخص باشد. پس این متد ها تا زمانی که همه پورتها را بررسی کند اجرا می شود و هر پورتی را که پیدا نمود، نام آن را درون رشته **s** قرار می دهد. و با دستور listBox1.Items.Add(s) نام آن پورت را در listBox1 قرار می گیرد. پس زمانی که برنامه اجرا شود همه پورتهای سریال موجود در این لیست دیده می شوند. توجه داشته باشید که این متد هیچ پورتی را باز نمی کند فقط نام پورت هایی که در سیستم ما موجود است را برمی گرداند. البته به این نکته هم توجه کنید که ایجاد ارتباط بین کامپیوتر و برخی از ابزار مانند موبایل، پورت سریال مجازی می سازد، اما قابل باز شدن در برنامه ما نیستند.

یک لیبل هم برای راهنمایی کاربر در کنار listBox1 قرار دهید و خاصیت Text آن را **Available Ports** بنویسید.

دکمه *Refresh*

متد Form1_Load فقط یک بار در هنگام اجرای برنامه اجرا می شود. به همین دلیل لازم است اگر تغییری در پورتهای سریال ایجاد شد، کاربر از وضعیت آخرین تغییرات در پورت ها آگاهی پیدا کند. کاربر می تواند در زمانی که برنامه در حال اجرا است با کلیک روی دکمه Refresh ، پورتهای موجود را در listBox1 مشاهده کند.

یک Button دیگر به فرم اضافه کنید. آن را در کنار listBox1 قرار دهید و ویژگیهای آن را به صورت زیر تنظیم کنید.

(Name) btnRefresh

(Text) Refresh

در متد فرایند کلیک این Button کد زیر را وارد کنید.

```
private void btnRefresh_Click(object sender, EventArgs e)
{
  listBox1.Items.Clear();
  foreach (string s in SerialPort.GetPortNames())
  {
    listBox1.Items.Add(s);

  }
}
```

همانگونه که مشاهده میکنید این متد، listBox1 را پاک می کند و دوباره وضعیت پورت های موجود را بررسی و نام پورت های موجود را به لیست وارد می کند. پس هر بار که پورت جدیدی به سیستم اضافه یا کم شد، نیازی به بستن و باز کردن برنامه نیست و با کلیک روی این دکمه کاربر می تواند پورتهای موجود را ببیند.

باز کردن پورت به وسیله ListBox

میخواهیم این قابلیت را به برنامه بیافزاییم که با کلیک روی نام پورت درListBox1 پورت مورد نظر باز شود. یادآوری میکنم که باز کردن پورت از طریقcomboBox1 نیز وجود دارد

روی ListBox1 دابل کلیک کنید تا متد رویداد listBox1_SelectedIndexChanged در پنجره Code ایجاد شود. در متد رویداد آن کد زیر را وارد کنید.

```
private void listBox1_SelectedIndexChanged(object sender, EventArgs e)
{
  comboBox1.Text = listBox1.Text;

  strbaud = comboBox2.Text;
  serialPort1.Close();

  try
  {
    serialPort1.DataBits = 8;
    serialPort1.Parity = Parity.None;
    serialPort1.StopBits = StopBits.One;
    serialPort1.BaudRate = int.Parse(strbaud);
    serialPort1.PortName = comboBox1.Text;
    serialPort1.Open();
    serialPort1.DiscardInBuffer();
    label1.Text = "( " + comboBox1.Text + " )" + "   CONNECTED ";
    label1.BackColor = System.Drawing.Color.Green;
    label1.ForeColor = System.Drawing.Color.White;
  }

  catch
  {
    label1.Text = "Disconnected";
    label1.BackColor = System.Drawing.Color.Red;
    label1.ForeColor = System.Drawing.Color.White;
    MessageBox.Show("Can't Access " + "(" + comboBox1.Text + ")", "ERROR",
MessageBoxButtons.OK, MessageBoxIcon.Error);
  }
```

```
}
```

نمایش داده های ارسالی و دریافتی

برای نماش داده های ارسالی و دریافتی جداگانه از دو Textbox استفاده میکنیم پس دو Textbox روی فرم قرار دهید و فقط ویژگی Multiline هردو را برابر True قرار دهید.

از Textbox1 برای ارسال و از Textbox2 برای نمایش داده های دریافتی استفاده میکنیم[1].

بالای هر کدام از Textbox ها یک Label قرار میدهیم و ویژگی Text آنها را به صورت زیر اصلاح میکنیم. از این دو لیبل تنها برای راهنمایی کاربر استفاده میکنیم و هیچ کدی برای آنها نمی نویسیم. لیبل ها را هم از پنجره Toolbox روی فرم درگ کنید.

Lable1 Send Data

Lable2 Received Data

ارسال داده

برای ارسال داده نیز از یک Button استفاده میکنیم که با کلیک روی آن متن درون Textbox1 از طریق پورت مورد نظر ارسال میشود. پس یک دکمه، برای ارسال به فرم اضافه کنید و ویژگیهای آن را به صورت زیر تغییر دهید.

(Name) btnSend

(Text) Send Data

در متد فرایند کلیک این دکمه کد زیر را وارد کنید.

```
private void btnSend_Click(object sender, EventArgs e)
{
  if (serialPort1.IsOpen == true)
  serialPort1.WriteLine(textBox1.Text);
}
```

دریافت داده

برای نمایش داده دریافت شده از Textbox2 که قبلا روی فرم قرار داده اید و اندازه و محل قرار گیری آن را در فرم تنظیم کرده اید استفاده می کنیم.

در پنجره Design برنامه روی SerialPort کلیک میکنیم و در بالای پنجره Properties روی آیکن رویدادها کلیک میکنیم[2]. و از رویداد های پیش رو، روی رویداد DataRecieve دابل کلیک کنید تا متد رویداد serialPort1_DataReceived نیز به قسمت کد های اضافه شود. و کد های زیر را در آن بنویسید.

---

[1] تصویر ۲۷.

[2] آیکن شبیه صاعقه!

```
private void serialPort1_DataReceived(object sender, SerialDataReceivedEventArgs e)
{
  StrRecieve = serialPort1.ReadExisting();
  this.Invoke(new EventHandler(DisplayText));
}
```

برای نمایش متن به وسیله این رویداد باید یک متد تعریف کنیم. و آن را قبل از متد رویداد DataReceived قرار دهید.

```
private void DisplayText(object sender, EventArgs e)
{
  textBox2.AppendText(StrRecieve);
}
```

متد این رویداد زمانی که داده ی جدید برای دریافت آماده است اجرا می شود و داده را دریافت و در رشته StrRecieve قرار می دهد. سپس متد DisplayText رشته دریافت شده را در textBox2 قرار می دهد. پس هر بار که داده جدیدی رسید کاربر آن را در textBox2 خواهد دید.

برای پاک کردن متن های درون هر Textbox نیز یک Button به فرم اضافه کنید و ویژگیهای آنها را به صورت زیر تغییر دهید.

(Name) btnClearSend
(Text) Clear Send

(Name) btnClearRecieve
(Text) Clear Recieve

```
private void btnClearSend_Click(object sender, EventArgs e)
{
    textBox1.Clear();
}

private void btnClearRecieve_Click(object sender, EventArgs e)
{
    textBox2.Clear();
}
```

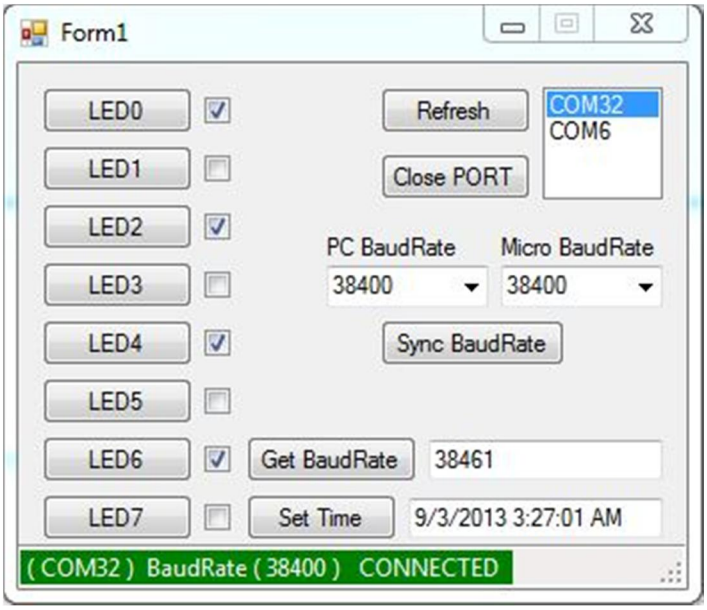
با استفاده از این دکمه ها کاربر می تواند، متن درون هر کدام از Textbox ها را جداگانه پاک کند.

برنامه آماده است و اگر برنامه را اجرا کنید میتواند پورت های موجود در سیستم را در listBox1 مشاهده کنید و هر کدام از آنها را که نیاز دارد با کلیک روی نام آن باز کند، با باز کردن comboBox1 و کلیک روی نام پورت نیز میتواند پورت را باز کند. البته اگر آن پورت در سیستم موجود باشد و امکان باز کردن آن فراهم باشد، به عبارتی تنها امکان باز کردن برای پورتی وجود دارد که نام آن در listBox1 وجود دارد. کاربر می تواند باودریت را با استفاده از comboBox2 تنظیم کند. داده های دریافتی را در textBox2 ببیند. و داده ای که میخواهد ارسال کند را در textBox1 وارد کند و روی دکمه Send Data برای ارسال کلیک کند.

در پایان پیشنها می کنم، ویژگی Anchor همه ابزار قرار گرفته در فرم را از طریق پنجره Properties تنظیم کنید. این ویژگی تعیین میکند که در صورت تغییر اندازه فرم در زمان اجرا، ابزار مورد نظر باید فاصله اش را با کدام سمت فرم ثابت نگه دارد. چپ، راست، یا بالا، و یا پایین. توجه داشته باشید که اگر هم زمان بالا و پایین و یا چپ و راست را انتخاب کنید، در زمان تغییر اندازه فرم، اندازه کنترل نیز تغییر میکند.

خودتان آزمایش کنید.

پروژه دوم: ساخت یک برنامه کنترلی[1]



تصویر35 _ نمای پایانی برنامه کنترل

کاربرد برنامه

این پروژه، یک برنامه برای کنترل یک میکروکنترلر است. در این پروژه چگونگی ایجاد ارتباط بین میکرو و کامپیوتر را یاد خواهید گرفت. همچنین یاد خواهید گرفت که چگونه، دستور ها و داده های خود را ارسال کنید و پاسخ های مورد نظر را دریافت و پردازش کنید. در پایان کار شما با چگونگی نوشتن و ارسال دستورات کنترلی، و دریافت و پردازش آنها آشنا خواهید شد[2]. برای وصل کردن میکرو کنترل به کامپیوتر، پایه RX میکرو را به پایه TX پورت سریال کامپیوتر و پایه TX میکرو را به پایه RX پورت سریال کامپیوتر متصل کنید. زمین مدار میکرو و زمین پورت سریال را نیز به هم وصل کنید. به این نکته هم توجه داشته باشید که برای متصل کردن میکرو به پورت سریال خود کامپیوتر، باید از IC تطبیق ولتاژ MAX232 استفاده کنید. چون در پورت سریال کامپیوتر، بیت ۱ به صورت سطح ولتاژ ۳- تا ۲۵- ولت و بیت صفر ۳+ تا ۲۵+ ولت دریافت، یا ارسال میشود. ولی اگر از مبدل های USB TO COM استفاده میکنید، که داده به صورت سطح ولتاژ TTL[3] ارسال و دریافت می شود. نیازی به مبدل نیست. چون بیت ۱ برابر +۵ ولت و بیت صفر برابر ۰ ولت تولید و یا دریافت می شود.

---

[1] برای شبیه سازی این پروژه به پیوست دو بروید.

[2] توجه داشته باشید که کلیه دستورات کنترلی که در این پروژه استفاده شده، به وسیله خود من، بین میکرو و برنامه کامپیوتر، قرار داد شده است. این شیوه جزو هیچ پروتکل یا برنامه ارتباط استاندارد شده ای نمی باشد. البته این به معنی بد بودن برنامه نوشته شده یا بلا استفاده بودن آن نمی باشد. فقط یک هشدار است تا کاربر بداند، هنگام استفاده از این شیوه در جاهای حساس باید با احتیاط عمل کند. چون هیچ پارامتر ایجاد خطایی در آن بررسی نشده است.

[3] البته همه مبدل های USB TO COM، TTL نیستند برخی از آنها RS232 می باشند.

کارکرد برنامه

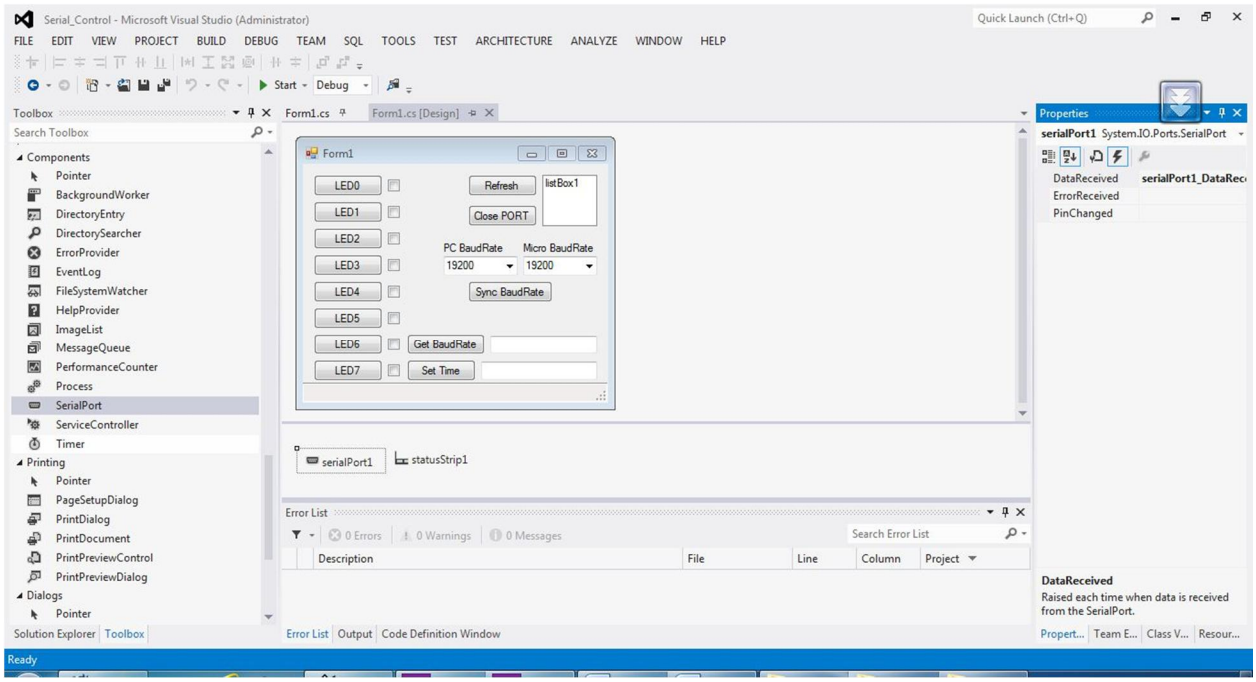
در تصویر بالا، در فرم برنامه، یک listBox و دو عدد Button در گوشه سمت راست، بالای فرم قرار دارد. همانطور که در پروژه قبلی دیدید، از این ابزار برای مشاهده، باز کردن و یا بستن پورت سریال استفاده می کنیم. کاربر میتواند پورت های موجود را در listBox1 ببیند، و با کلیک کردن روی نام پورت در listBox1 پورت مورد نظر را باز کند. واگر تغییری در پورت ها روی داد، با کلیک روی دکمه Refresh تغییرات پورت های سریال را ببیند، و اگر خواست پورتی را ببندد، با کلیک روی Close Port می تواند این کار را انجام دهد. وضعیت باز بودن پورت یا بسته شدن آن در label1 قابل مشاهده است.

در وسط فرم، دو عدد comboBox و یک دکمه میبینید. از comboBox1 برای تنظیم باودریت پورت سریال کامپیوتر و از comboBox2 برای تنظیم باودریت میکروکنترلر استفاده میکنیم. تنظیم باودریت میکرو به این صورت است که با باز کردن comboBox2 و کلیک روی هر باودریتی کد دستورات خاصی به میکرو ارسال می شود. میکرو پس از دریافت دستور، باودریت جدید خود را برابر مقداری که کامپیوتر از او می خواهد، تنظیم می کند. پس از تغیر باودریت میکروکنترلر، شما باید با کلیک کردن روی دکمه Sync BaudRate باودریت خود کامپیوتر را نیز برابر آنچه که برای میکرو انتخاب کردید قرار دهید. این کار را با استفاده از comboBox1 نیز میتوانید انجام دهید. توجه داشته باشید اگر باود ریت میکرو را تغییر دهید، تا زمانی که باودریت خود کامپیوتر را نیز با آن یکسان نکنید، هیچ داده ای نمیتوانید رد و بدل کنید. چون باودریت دو سو با هم یکسان نیست، داده دریافتی معتبر نیست.

دکمه Get BaudRate برای دریافت باودریت میکرو استفاده می شود. با کلیک روی این دکمه دستوری به میکرو ارسال می شود، که میکرو پس از پردازش دستور، ریجستر خود را خوانده و باودریت خودش را محاسبه و مقدار محاسبه شده را به کامپیوتر ارسال میکند، و برنامه، باودریت دریافت شده را در textBox1 نمایش می دهد.

دکمه Set Time برای ارسال زمان حال کامپیوتر، به میکرو استفاده می شود. با کلیک روی این دکمه زمان و تاریخ کامپیوتر در textBox2 نشان داده می شود و قسمت زمان آن برای میکرو ارسال می شود. میکرو می تواند با دریافت ساعت کامپیوتر، ساعت خود را با آن، هم زمان کند. البته در این پروژه، میکرو پس از دریافت زمان، تنها آن را روی LCD نمایش می دهد.

در سمت چپ فرم ۸ دکمه و روبروی هر دکمه یک checkbox می بینید. با کلیک روی هر دکمه، یک کد دستور به میکرو ارسال می شود. میکرو با دریافت دستور می فهمد که کدام پایه خود را باید تغییر دهد، و پایه مورد نظر را تغییر حالت میدهد. یعنی با هر بار کلیک روی هر کدام از این دکمه ها پایه متناظر آن در میکرو، اگر روشن باشد خاموش می شود، و اگر خاموش باشد روشن می شود. میکرو پس از تغییر حالت پایه خود، وضعیت جدید یعنی روشن یا خاموش بودن آن پایه را در، به کامپیوتر ارسال میکند و برنامه وضعیت جدید را در checkbox مربوطه نمایش می دهد.یعنی اگر پایه روشن باشد، درون checkbox تیک دار می شود و اگر خاموش باشد، بدون تیک خواهد شد.



تصویر _ ۳۶ ابزار قرار گرفته روی فرم پروژه کنترل

قرار دادن ابزار مورد نیاز روی فرم

یک پروژه جدید C#، در Visual Stdio با نام Serial_Control ایجاد کنید.

ویژگی های فرم آن را از پنجره Properties مانند زیر تنظیم کنید.

(MaximizeBox) False
(MaximumSize) 350, 300
(MinimumSize) 350, 300
(Size) 350, 300
(Text) Control

از پنجره ToolBox کامپونت پورت سریال را روی فرمتان درگ کنید.

یک listBox از پنجره Toolbox روی فرم قرار دهید و آنرا در گوشه بالا سمت راست فرم قرار دهید و ویژگی **Size** آن را **60, 56** تنظیم کنید. از این listBox مانند پروژه قبلی برای نشان دادن پورت های موجود و باز کردن پورت با کلیک روی نام آن، استفاده می کنیم.

(Name) listBox1
(Size) 60, 56

برای Refresh لیست و بستن پورت نیاز به دو دکمه داریم. از پنجره Toolbox دو Button روی فرم، کنار listBox1 قرار دهید. ویژگی های آن ها را مانند زیر را تنظیم کنید.

(Name) btnRefresh
(Text) Refresh

(Name) btnClosePort
(Text) ClosePort

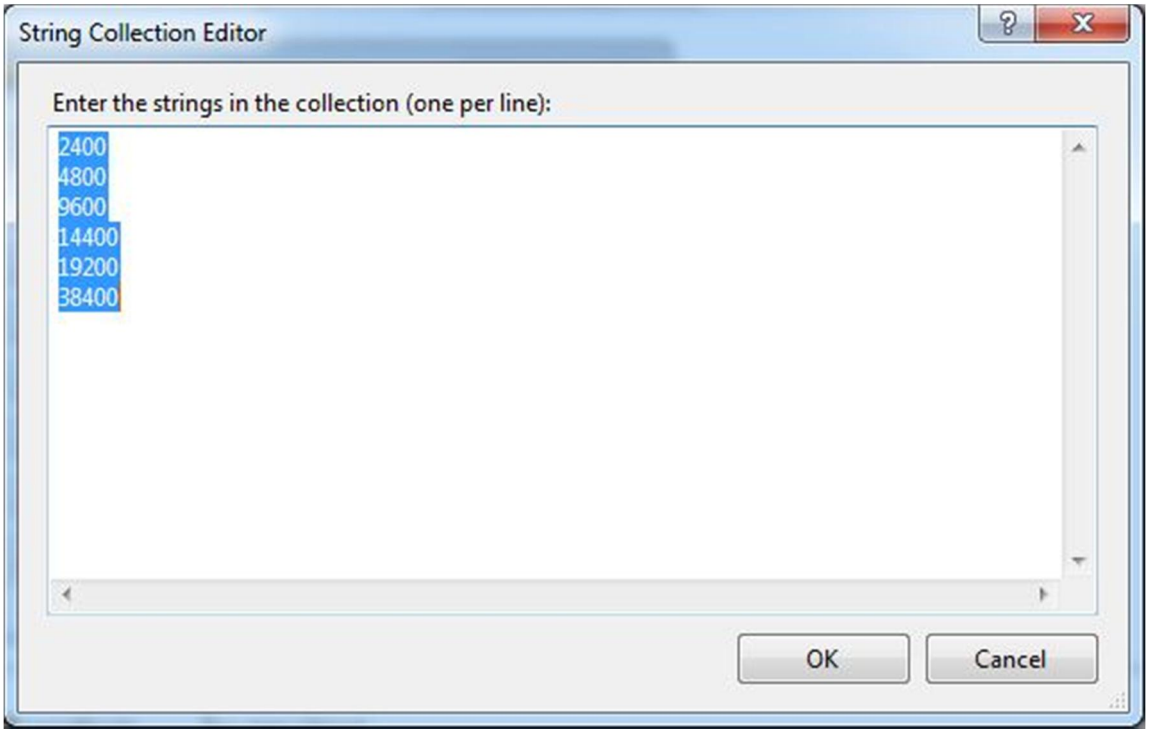
برای نشان دادن وضعیت پورت، یک لیبل در قسمت پایین فرم قرار دهید و دقت کنید که ویژگی نام آن **label1** باشد.

(Name) label1

برای تنظیم باودریت پورت سریال کامپیوتر و تنظیم باودریت میکروکنترلر از دو comboBox جداگانه استفاده میکنیم. از پنجره Toolbox دو comboBox روی فرم قرار ویژگی Items هر دو را مانند تصویر زیر تنظیم کنید. همچنین، ویژگی Text هر دو را ۱۹۲۰۰ قرار دهید. همانطور که در تصویر این برنامه، در اول این بخش مشاهده کردید comboBox1 در سمت چپ فرم برای تنظیم باودریت برنامه و comboBox2 در سمت راست فرم برای تنظیم باودریت میکروکنترلر استفاده میشود.

(Name) comboBox1
(Text) 19200

(Name) comboBox2
(Text) 19200



تصویر37 _ اضافه کردن باودریت های مورد استفاده به ویژگی Items

در بالای هر کدام از comboBox ها یک Label قرار دهید و ویژگی Text آن ها را تنظیم کنید.

(Name) label2
(Text) PC BaudRate

(Name) label3
(Text) Micro BaudRate

یک دکمه دیگر روی فرم قرار دهید و وبژگی های آن را مانند زیر تنظیم کنید. اگر همه متن درون دکمه را نمی بینید، دکمه را کمی بزرگتر کنید تا متن درون آن کامل دیده شود.

(Name) btnSyncBaud
(Text) Sync BaudRate

برای ارسال دستور به میکرو از دو کمه استفاده میکنیم. پس دو دکمه روی فرم قرار دهید، و ویژگی های آن ها را مانند زیر را تنظیم کنید.

(Name) btnGetBaud
(Text) Get BaudRate

(Name) btnSetTime
(Text) Set Time

دوعدد textBox روی فرم قرار دهید توجه داشته باشید که textBox1 روبروی دکمه Get BaudRate قرار بگیرد، و textBox2 روبروی دکمه Set Time قراربگیرد.

(Name) textBox1

(Name) textBox2

۸ عدد Button دیگر به صورت عمودی از بالا به پایین در سمت چپ فرم قرار دهید و ویژگی هر کدام را جداگانه مانند زیر تنظیم کنید[1]. هر یک از این دکمه ها برای روشن یا خاموش کردن LED نظیر، متصل به پایه های PORTA میکروکنترلر استفاده میشود.

(Name) btnLED0
(Text) LED0

(Name) btnLED1

[1] در این پروژه شماره گذاری نام این دکمه ها را از صفر آغاز کرده ایم تا با شماره پین های میکرو منطبق باشد.

(Text) LED1

(Name) btnLED2
(Text) LED2

(Name) btnLED3
(Text) LED3

(Name) btnLED4
(Text) LED4

(Name) btnLED5
(Text) LED5

(Name) btnLED6
(Text) LED6

(Name) btnLED7
(Text) LED7

و سرانجام روبروی هر Button یک checkbox قرار دهید ویژگی Text همه آنها را خالی بگذارید. و نام آن ها را مانند زیر تنظیم کنید.

(Name) checkBox0

(Name) checkBox1

(Name) checkBox2

(Name) checkBox3

(Name) checkBox4

(Name) checkBox5

(Name) checkBox6

(Name) checkBox7

کد برنامه کامپیوتر

دریافت داده

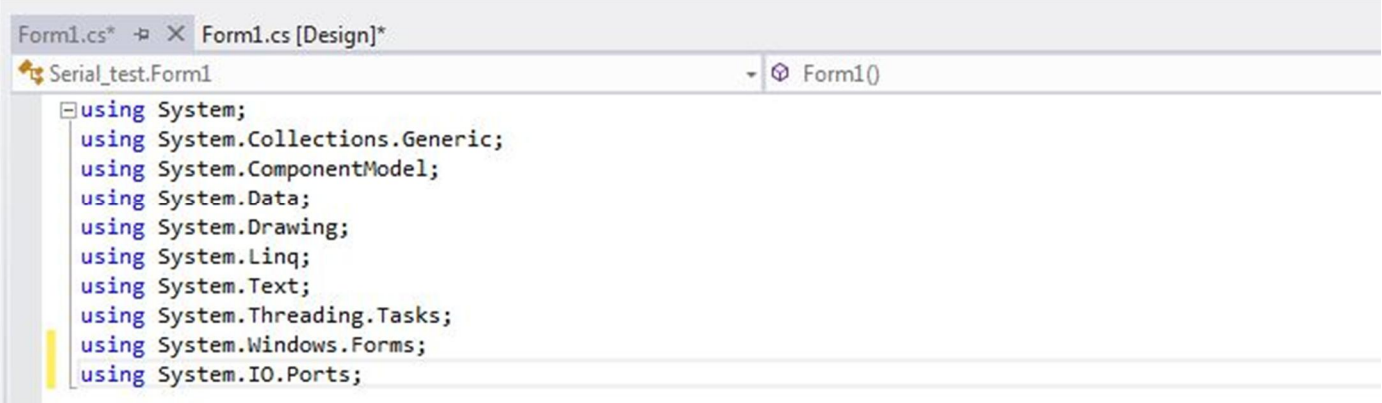
در صفحه Design برنامه روی کامپونت SerialPort که روی فرم قرار داده اید، کلیک کنید و در بالای پنجره Properties روی آیکن رویدادها کلیک کنید[1]. و از رویداد های پیش رو، روی رویداد DataRecieve دابل کلیک کنید. و کد های زیر را در این متد بنویسید.

```
private void serialPort1_DataReceived(object sender,
System.IO.Ports.SerialDataReceivedEventArgs e)
{
  strRecieve = serialPort1.ReadLine();
  this.Invoke(new EventHandler(CheckReciever));
}
```

این متد با تنظیمات پیش فرض، با رسیدن اولین کاراکتر داده اجرا می شود. و متد ReadLine() تا زمانی که کاراکتر NewLine را دریافت نکند، البته به شرط آنکه زمان انتظارش به پایان نرسیده باشد، داده را دریافت میکند و سر انجام، داده های دریافت شده پس از پایان زمان انتظار و یا رسیدن کاراکتر NewLine بازگشت داده می شود؛ و در رشته strRecieve قرار میگیرد. سپس متد CheckReciever را فراخوانی می شود. و کار متد DataReceived تا رسیدن اولین کاراکتر بعدی که منجر به اجرای دوباره آن میشود به پایان میرسد. متد CheckReciever پس از ارائه همه متد ها توضیح داده شده است.

فضای نام مورد نیاز برای کار با این کامپونت را به برنامه معرفی کنید.

```
using System.IO.Ports;
```



تصویر _ ۳۸ اضافه کردن فضای سیستم مورد استفاده کامپونت پورت سریال

[1] آیکن شبیه صاعقه!

متغیر های استفاده شده در برنامه را در همین مرحله، در بدنه کلاس فرم مانند زیر تعریف کنید.

```
public partial class Form1 : Form
{
  string strRecieve;
  string strbaud;
  string strCommand;
```

باز کردن پورت

روی listBox1 دابل کلیک کنید و در متد فرایند listBox1_SelectedIndexChanged کد زیر را وارد کنید.

```
private void listBox1_SelectedIndexChanged(object sender, EventArgs e)
{
  strbaud = comboBox1.Text;
  serialPort1.Close();

  try
  {
    serialPort1.DataBits = 8;
    serialPort1.Parity = Parity.None;
    serialPort1.StopBits = StopBits.One;
    serialPort1.BaudRate = int.Parse(strbaud);
    serialPort1.PortName = listBox1.Text;
    serialPort1.Open();
    serialPort1.DiscardInBuffer();
    label1.Text = "( " + listBox1.Text + " )" +"  BaudRate " + "( " + comboBox1.Text +
" )" + "   CONNECTED ";
    label1.BackColor = System.Drawing.Color.Green;
    label1.ForeColor = System.Drawing.Color.White;
  }
  catch
  {
    label1.Text = "Disconnected";
    label1.BackColor = System.Drawing.Color.Red;
    label1.ForeColor = System.Drawing.Color.White;
    MessageBox.Show("Can't Access This PORT", "ERROR", MessageBoxButtons.OK,
MessageBoxIcon.Error);
  }
}
```

دستورات بالا پورت را با ویژگی هایی که در متن برنامه مشاهده میکنید باز می کنند. هر گاه برنامه اجرا شود، شما نام پورت های موجود را درون listBox1 می بینید، که با کلیک روی نام هر پورت می توانید آن را باز کنید.

پورت های موجود

روی دکمه Refresh دابل کلیک کنید و کد زیر را در متد فرایند کلیک آن وارد کنید.

```
private void btnRefresh_Click(object sender, EventArgs e)
{
  listBox1.Items.Clear();

  foreach (string s in SerialPort.GetPortNames())
  {
    listBox1.Items.Add(s);
  }
}
```

با کلیک روی دکمه Refresh نخست لیست پاک میشود، و پس از بررسی، هر پورتی که در کامپیوتر موجود بود، نامش به listBox1 اضافه می شود.

روی خود فرم دابل کلیک کنید و کد های زیر را در آن وارد کنید. عملکرد این متد با دکمه Refresh یکسان است.

```
private void Form1_Load(object sender, EventArgs e)
{
  foreach (string s in SerialPort.GetPortNames())
  {
    listBox1.Items.Add(s);
  }
}
```

بستن پورت

کد های زیر را برای متد فرایند کلیک دکمه Close PORT وارد کنید.

```
private void btnClosePort_Click(object sender, EventArgs e)
{
  serialPort1.Close();
  label1.Text = "Disconnected";
  label1.BackColor = System.Drawing.Color.Red;
  label1.ForeColor = System.Drawing.Color.White;
}
```

دکمه Close Port ارتباط برنامه با پورت را قطع میکند یا به عبارتی دیگر، آن را می بندد. سپس وضعیت label1 را که وضعیت ارتباط را به کاربر نشان میدهد، تغییر میدهد.

تغییر باود ریت

روی comboBox1 دابل کلیک کنید و در متد فرایند comboBox1_SelectedIndexChanged کد زیر را وارد کنید.

```
private void comboBox1_SelectedIndexChanged(object sender, EventArgs e)
{
  strbaud = comboBox1.Text;
  try
  {
    serialPort1.BaudRate = int.Parse(strbaud);
    serialPort1.DiscardInBuffer();
    label1.Text = "( " + listBox1.Text + " )" + "  BaudRate " + "( " + comboBox1.Text
+ " )" + "   CONNECTED ";
    label1.BackColor = System.Drawing.Color.Green;
    label1.ForeColor = System.Drawing.Color.White;
    }

    catch
    {
      label1.Text = "Disconnected";
      label1.BackColor = System.Drawing.Color.Red;
      label1.ForeColor = System.Drawing.Color.White;
      MessageBox.Show("Can't Access Any PORT", "ERROR", MessageBoxButtons.OK,
MessageBoxIcon.Error);
  }
}
```

comboBox1 برای تغییر باودریت ارتباط به کار می رود. توجه داشته باشید که برای عوض کردن باودریت نیازی به قطع ارتباط نیست.

```
serialPort1.BaudRate = int.Parse(strbaud);
```

میدانید که دستور بالا، ویژگی باودریت پورت را تنظیم میکند. اما تنظیم باود ریت فقط به صورت قرار دادن عدد صحیح ممکن می باشد. مانند دستور زیر:

```
serialPort1.BaudRate = 19200;
```

و همانطور که میدانید متغییر strbaud از جنس رشته است پس با استفاده از دستور int.Parse(strbaud) رشته به یک عدد از جنس int یا همان عدد صحیح خودمان، تبدیل می شود و در متد قرار می گیرد.

تغییر باود ریت میکروکنترلر

سپس روی comboBox2 دابل کلیک کنید و در متد فرایند comboBox2_SelectedIndexChanged کد زیر را وارد کنید.

```
private void comboBox2_SelectedIndexChanged(object sender, EventArgs e)
{
```

```
  try
  {
    serialPort1.WriteLine("<S" + comboBox2.Text + ">");
  }

  catch
  {
    label1.Text = "Disconnected";
    label1.BackColor = System.Drawing.Color.Red;
    label1.ForeColor = System.Drawing.Color.White;
    MessageBox.Show("Can't Access Any PORT", "ERROR", MessageBoxButtons.OK,
MessageBoxIcon.Error);
  }
}
```

با باز کردن comboBox2 و کلیک روی هر کدام از باود ریت های وارد شده در comboBox2 در زمان اجرای برنامه، یک رشته از طریق پورت سریال ارسال می شود. برای مثال شما روی ۹۶۰۰ کلیک کنید رشته زیر ارسال می شود.

<S9600>

میدانید ارسال این رشته به چه صورت است؟

ارسال از طریق پورت سریال به صورت ارسال کد اسکی[1] هر کاراکتر است.

نخست کاراکتر < ارسال می شود. برای ارسال این کاراکتر، کد اسکی آن یعنی عدد ٦٠ که هگزا دسیمال آن 0x3C است ارسال میشود یعنی باینری 0b00111100 بیت به بیت ارسال می شود. و میدانید برای ارسال هر کاراکتر بیت آغاز، بیت پایان و احتمالا بیت پریتی نیز ارسال می شود.

پس از آن، کاراکتر S که کد اسکی آن ۸۳ است ارسال می شود.

سپس کاراکتر عدد ۹ با کد اسکی ٥٧ ارسال می شود.

سپس کاراکتر عدد ٦ با کد اسکی ٥٤ ارسال می شود.

سپس کاراکتر عدد ۰ با کد اسکی ٤٨ ارسال می شود.

سپس کاراکتر عدد ۰ با کد اسکی ٤٨ ارسال می شود.

و سرانجام کاراکتر > با کد اسکی ٦٢ ارسال می شود.

این نکته را هم یاد آور می شوم که متد WriteLine() پس از ارسال هر رشته، کاراکتر NewLine، با کد اسکی ۱۰ را نیز در پایان، ارسال می کند تا گیرنده متوجه پایان یک ارسال شود. اما ما در برنامه میکروکنترلر این کاراکتر را پردازش نمی کنیم. یعنی مبنای پایان یک دستور را این کاراکتر نگرفته ایم بلکه پایان هر دستور با رسیدن کاراکتر < مشخص می شود.

[1] **ASCII** (American Standard Code for Information Interchange) پیوست۱

برای مشاهده کد های اسکی به پیوست۱ مراجعه کنید. البته نیازی به دانستن کد های اسکی نمی باشد. چون برنامه یا کامپایلر به صورت خودکار برای هر کاراکتری که شما وارد می کنید کد اسکی آن را ارسال می کند.

توجه داشته باشید که کاراکتر ها **'>'** و **'<'** نیز همراه **S** و **9600** ارسال می شوند. و هر دستور دیگری نیز که به میکرو ارسال شود بین **<>** قرا میگیرد. این کار برای این است تا میکرو بداند با دریافت کاراکتر **'<'** تا دریافت کامل کاراکتر های دستور، منتظر دریافت کد دستور باشد، و تا زمانی که کاراکتر **'>'** را دریافت نکرده، کلیه کاراکتر های دریافتی را به عنوان کد دستور دریافت کند و پس دریافت کاراکتر پایان، کد دستور دریافت شده را پردازش کند.

<دستور>

میکرو کنترلر رشته بالا را دریافت می کند و پس از پردازش آن متوجه می شود، در دستور رسیده، از او خواسته شده تا باودریت خود را برابر ۹۶۰۰ تنظیم کند. و فرمان را اجرا میکند. چگونگی دریافت، پردازش و اجرای میکرو کنترلر در قسمت بعدی توضیح داده شده است.

به این نکته توجه داشته باشید اگر باودریت میکرو را تغییر دادید، حتما بلافاصله و قبل از ارسال هر فرمان دیگر، باودریت پورت سریال خود کامپیوتر را نیز با آن هماهنگ کنید. یعنی اگر باود ریت میکرو را به ۹۶۰۰ تغییر دادید، با استفاده از دکمه Sync یا باز کردن comboBox1 و کلیک روی مقدار BaudRate باودریت پورت سریال را با آن هماهنگ کنید. در غیر اینصورت به علت متفاوت بودن باودریت کامپیوتر و میکرو کنترلر، هیچ تبادل داده ای صورت نخواهد گرفت.

یکسان کردن باود ریت برنامه با باود ریت میکرو

کد زیر را برای متد رویداد کلیک دکمه Sync BaudRate وارد کنید.

```
private void btnSyncBaud_Click(object sender, EventArgs e)
{
  comboBox1.SelectedItem = comboBox2.Text;
}
```

این دستور آیتم انتخابی comboBox1 را، به آنچه که در comboBox2 انتخاب شده است تغییر می دهد. توجه داشته باشید که تغییر آیتم comboBox1 یا هر comboBox دیگری، چه با باز کردن آن و کلیک روی یک آیتم در زمان اجرای برنامه، چه به صورت بالا از طریق کد های برنامه باشد باعث اجرای متد comboBox1_SelectedIndexChanged می شود. و میدانید که اجرای این متد، باودریت پورت سریال را تغییر می دهد.

دریافت باودریت میکرو کنترلر

کد های زیر را برای متد رویداد کلیک دکمه Get BaudRate وارد کنید.

```
private void btnGetBaud_Click (object sender, EventArgs e)
{
```

```
  if (serialPort1.IsOpen == true)
  {
    serialPort1.WriteLine("<B>");
    textBox1.BackColor = System.Drawing.Color.White;
    textBox1.ForeColor = System.Drawing.Color.Black;
  }
  else

    MessageBox.Show("Can't Access Any PORT", "ERROR", MessageBoxButtons.OK,
MessageBoxIcon.Error);
}
```

با کلیک روی دکمه Get BaudRate رشته **<B>** به میکرو ارسال می شود. میکرو با دریافت این رشته متوجه می شود که از او خواسته شده تا باودریتش را ارسال کند و سپس فرمان را اجرا می کند، باودریتش را میخواند و در قالب یک رشته به کامپیوتر ارسال میکند.

ارسال زمان به میکرو کنترلر

کد های زیر را برای متد رویداد کلیک دکمه Set Time وارد کنید.

```
private void btnSetTime_Click(object sender, EventArgs e)
{
  string strTime;
  textBox2.Clear();
  DateTime dteTime;
  dteTime = DateTime.Now;
  textBox2.Text =dteTime.ToString();
  strTime ="<T" + dteTime.ToLongTimeString()+">" ;

  if (serialPort1.IsOpen == true)
    serialPort1.WriteLine(strTime);

  else
    MessageBox.Show("Can't Access Any PORT", "ERROR", MessageBoxButtons.OK,
MessageBoxIcon.Error);
}
```

با کلیک روی این دکمه زمان و تاریخ حال کامپیوتر شما با استفاده از دستور `dteTime = DateTime.Now` خوانده می شود و در متغیر `dteTime` قرار می گیرد. سپس در خط بعدی به رشته تبدیل می شود و در `textBox2` نمایش داده می شود. در خط دستور بعدی، کاراکتر های کنترلی `<T >` به قسمت زمان متغیر `dteTime` که زمان وتاریخ لحظه کلیک را درخود نگه می دارد اضافه می شود و در رشته `strTime` قرار داده می شود. سرانجام باز بودن پورت بررسی می شود و اگر پورت باز بود رشته

strTime مانند زیر با دستور serialPort1.WriteLine(strTime) ارسال می شود. و اگر پورت باز نبود با باز کردن بک پنجره پیام به کاربر گزارش داده می شود.

<T9:46:57 AM>

تغییر وضعیت LED ها

سر انجام کد مربوط به متد رویداد کلیک دکمه LED ها را وارد کنید. توجه داشته باشید که دستورات همه متد فرایند کلیک دکمه LED ها مشابه است و تنها تفاوت آنها، کد دستور ارسالی است.

```
private void btnLED0_Click(object sender, EventArgs e)
{
  if (serialPort1.IsOpen == true)
  {
    serialPort1.WriteLine("<0>");
  }
  else
    MessageBox.Show("Can't Access Any PORT", "ERROR", MessageBoxButtons.OK,
MessageBoxIcon.Error);

}

private void btnLED1_Click(object sender, EventArgs e)
{
  if (serialPort1.IsOpen == true)
  {
    serialPort1.WriteLine("<1>");
  }
  else
    MessageBox.Show("Can't Access Any PORT", "ERROR", MessageBoxButtons.OK,
MessageBoxIcon.Error);
}

private void btnLED2_Click(object sender, EventArgs e)
{
  if (serialPort1.IsOpen == true)
  {
    serialPort1.WriteLine("<2>");
  }
  else
  MessageBox.Show("Can't Access Any PORT", "ERROR", MessageBoxButtons.OK,
MessageBoxIcon.Error);
```

```
}

private void btnLED3_Click(object sender, EventArgs e)
{
  if (serialPort1.IsOpen == true)
  {
    serialPort1.WriteLine("<3>");
  }
  else
    MessageBox.Show("Can't Access Any PORT", "ERROR", MessageBoxButtons.OK,
MessageBoxIcon.Error);
}

private void btnLED4_Click(object sender, EventArgs e)
  {
    if (serialPort1.IsOpen == true)
    {
      serialPort1.WriteLine("<4>");
    }
    else
      MessageBox.Show("Can't Access Any PORT", "ERROR", MessageBoxButtons.OK,
MessageBoxIcon.Error);
}

private void btnLED5_Click(object sender, EventArgs e)
{
  if (serialPort1.IsOpen == true)
  {
    serialPort1.WriteLine("<5>");
  }
  else
    MessageBox.Show("Can't Access Any PORT", "ERROR", MessageBoxButtons.OK,
MessageBoxIcon.Error);
}



private void btnLED6_Click(object sender, EventArgs e)
{
  if (serialPort1.IsOpen == true)
  {
    serialPort1.WriteLine("<6>");
  }
```

```
  else
    MessageBox.Show("Can't Access Any PORT", "ERROR", MessageBoxButtons.OK,
MessageBoxIcon.Error);
}


private void btnLED7_Click(object sender, EventArgs e)
{
  if (serialPort1.IsOpen == true)
  {
    serialPort1.WriteLine("<7>");
  }
  else
    MessageBox.Show("Can't Access Any PORT", "ERROR", MessageBoxButtons.OK,
MessageBoxIcon.Error);
}
```

با کلیک روی هر کدام از دکمه های LED رشته ای مانند **<1>** (با کلیک روی دکمه LED1) و یا **<3>** (با کلیک روی دکمه LED3) ارسال می شود که میکروکنترلر پس از دریافت این فرمان متوجه می شود، که از او خواسته شده تا پین مورد نظر را تغییر وضعیت دهد. میکرو پس از اجرای فرمان، وضعیت پین را به برنامه کامپیوتر گزارش میدهد. و برنامه وضعیت آن پین را در checkbox مربوطه نشان می دهد.

پردازش داده دریافتی

در مرحله پایانی تابع CheckReciever را به صورت زیر تعریف کنید.

```
private void CheckReciever(object sender, EventArgs e)
{
  if (strRecieve.Length >= 4)
  strCommand = strRecieve.Substring(1, 2);
  else strCommand = "QQ";

  switch (strCommand)
 {
  case "SB":
  textBox1.Text = strRecieve.Substring(5, strRecieve.Length - 6);
  textBox1.BackColor = System.Drawing.Color.White;
  textBox1.ForeColor = System.Drawing.Color.Black;
  break;
```

```
case "BE":
textBox1.Text = strRecieve.Substring(5, strRecieve.Length - 6);
textBox1.BackColor = System.Drawing.Color.Red;
textBox1.ForeColor = System.Drawing.Color.White;
break;


case "00":
checkBox0.Checked = false;
break;

case "01":
checkBox0.Checked = true;
break;

case "10":
checkBox1.Checked = false;
break;

case "11":
checkBox1.Checked = true;
break;

case "20":
checkBox2.Checked = false;
break;

case "21":
checkBox2.Checked = true;
break;

case "30":
checkBox3.Checked = false;
break;

case "31":
checkBox3.Checked = true;
break;

case "40":
checkBox4.Checked = false;
break;
```

```
  case "41":
  checkBox4.Checked = true;
  break;

  case "50":
  checkBox5.Checked = false;
  break;

  case "51":
  checkBox5.Checked = true;
  break;

  case "60":
  checkBox6.Checked = false;
  break;

  case "61":
  checkBox6.Checked = true;
  break;

  case "70":
  checkBox7.Checked = false;
  break;

  case "71":
  checkBox7.Checked = true;
  break;

  default:
  textBox2.Text ="ERR:" + strRecieve;
  break;

 }
}
```

همانطور که گفتیم پس از دریافت داده توسط متد ReadLine()، متد بالا فراخوانی می شود.

```
 private void serialPort1_DataReceived(object sender,
System.IO.Ports.SerialDataReceivedEventArgs e)
{
  strRecieve = serialPort1.ReadLine();
  this.Invoke(new EventHandler(CheckReciever));
```

```
}
```

متد CheckReciever وظیفه پردازش داده دریافتی را انجام میدهد. این تابع بسیار ساده[1] تعریف شده است. در ادامه به بررسی آن می پردازیم.

```
private void CheckReciever(object sender, EventArgs e)
{

  if (strRecieve.Length >= 4)
```

همانطور که گفتیم وظیفه این متد، پردازش داده دریافتی است که در رشته strRecieve قرار دارد. در این خط بررسی می شود که طول این رشته حتما برابر یا بزرگتر از ٤ کاراکتر باشد. این بررسی به این علت است که در عملیاتی که در ادامه روی رشته انجام میشود، اگر طول رشته از ٤ کاراکتر کمتر باشد، در ادامه برنامه با ایراد مواجه خواهد شد. هیچ دستوری که از سمت میکرو ارسال می شود، کمتر از ٤ کاراکتر نیست.

```
strCommand = strRecieve.Substring(1, 2);
```

دراین خط از کاراکتر شماره ۱ و به تعداد ۲ کاراکتر یعنی کاراکتر ۱ و ۲ از رشته strRecieve چیده می شود و در رشته strCommand قرار می گیرد. هنگام اجرای این متد، برنامه از کاراکتر صفرم رد می شود و کاراکتر ۱ و ۲ را برمیگرداند. اگر طول رشته strRecieve وقتی عملیات را روی آن انجام می گیرد کمتر از ۳ کاراکتر باشد، اجرای برنامه ما با ایراد مواجه خواهد شد. به همین دلیل در دستور قبلی حداقل طول ٤ کاراکتر بررسی شد.

توجه داشته باشید که متد Substring() برای برداشتن قسمتی از یک رشته استفاده می شود. این متد قسمتی از یک رشته را که با دو پارامتر ورودیش مشخص می شود، برمیگرداند. پارامتر یکم، شماره کاراکتری را که باید رشته را از آن برش دهد، مشخص میکند و پارامتر دوم تعداد کاراکتر هایی را که باید بردارد مشخص میکند. به مثال زیر توجه فرمایید.

این رشته از سمت میکرو ارسال شده و اطلاعات باودریت میکرو کنترلر را در خود دارد.

<	S	B	>	<	1	9	2	0	0	>

پس از اجرای دستور بالا کاراکتر ۱ و ۲ از این رشته برداشته میشود و به عنوان کد دستور پردازش می شود.

S	B

```
else strCommand = "QQ";
```

در اینجا اگر طول رشته strRecieve کمتر از ٤ کاراکتر بود، عبارت **QQ** درون رشته strCommand قرار میگیرد. این عبارت اهمیتی ندارد و فقط برای این است که این رشته خالی نباشد یا مقدار قبلی خود را نداشته باشد.

[1] شاید غیر اصولی

```
switch (strCommand)
{
```

در این خط، کد دستور کاراکتر ۱ و ۲ رشته strRecieve است که در رشته strCommand قرار گرفته است، به عنوان کد دستور توسط دستور switch پردازش میشود.

```
case "SB":
textBox1.Text = strRecieve.Substring(5, strRecieve.Length - 6);
textBox1.BackColor = System.Drawing.Color.White;
textBox1.ForeColor = System.Drawing.Color.Black;
break;
```

اگر کد دستور، برابر SB بود، این case اجرا می شود. که از عضو پنجم و به تعداد طول رشته منهای ٦ کاراکتر رشته دریافت را در textBox1 نمایش میدهد. رنگ پس زمینه سفید و رنگ متن سیاه انتخاب می شود. و سپس با رسیدن به دستور break، برنامه از حلقه switch خارج می شود.

0	1	2	3	4	5	6	7	8	9	10
<	S	B	>	<	1	9	2	0	0	>

به عنوان مثال رشته بالا که در پاسخ به دستور ارسال باودریت، از سمت میکرو ارسال شده است، در بردارنده اطلاعات مربوط به باودریت میکرو کنترلر میباشد. در این رشته، کاراکتر های کنترلی و کد دستو را مشاهده میکنید. ردیف بالا شماره عضو رشته است.

عضو پنجم، کاراکتر ۱ است.

اعضای ۰ تا ٤ رشته، کاراکتر های کنترلی و کد دستور را در خود دارند. و تعداد اعضای رشته بالا ۱۱ کاراکتر است. 11-6 برابر۵ میباشد پس، باید از کاراکتر ۵ و به تعداد ۵ عضو از آن برداشته شود.

اگر رشته ما کد دستور SB را درخود داشت، میدانیم که در ادامه همان رشته باودریت میکرو کنترلر وجود دارد. این باودریت میتواند عدد ۳رقمی ۵ رقمی یا شاید هم ٦ رقمی باشد. پس تعداد اعضای آن مشخص نیست.

از اعضای این رشته، تعداد ۵ عضو در ابتدای رشته و ۱ عضو در پایان آن، (کاراکتر '>') ، یعنی تعداد ٦ کاراکتر از رشته، کاراکتر های کنترلی است. پس از عضو پنجم شروع میکنیم. و به تعداد اعضای رشته منهای ٦ عضو را باید برداریم.

همیشه تنظیم رنگ لازم نیست اگر شما رنگ متن را تنظیم نکنید، رشته شما با تنظیمات قبلی که برای این textBox انجام شده نمایش داده می شود. و ما چون احتمال میدهیم رنگ این textBox در جای دیگری تغیر داده شده باشد. دوباره آن را تنظیم میکنیم.

```
case "BE":
textBox1.Text = strRecieve.Substring(5, strRecieve.Length - 6);
textBox1.BackColor = System.Drawing.Color.Red;
```

```
textBox1.ForeColor = System.Drawing.Color.White;
break;
```

دستورات این case نیز مشابه case بالا است که توضیح داده داده شد. تنها این نکته را ذکر کنم، که رشته ارسالی با کد دستور BE در پاسخ به دستور تنظیم باودریتی است که میکروکنترلر اجازه تنظیم آن باود ریت را به هر دلیل ندارد. پس باودریت خواسته شده را در این قالب به کامپیوتر ارسال میکند تا کاربر متوجه شود که میکرو اجازه تنظیم باودریت با این مقدار را ندارد.

برای آزمایش این مورد، شما مقداری غیر از آنچه در comboBox2 وجود دارد به آن اضافه کنید. برنامه را اجرا و آن مقدار را پس از باز کردن coboBox2 انتخاب کنید. دستوری به میکرو ارسال شده تا باودریت خود را برابر آن چه شما انتخاب کرده اید تنظیم کنید. حتما میدانید که میکرو چه پاسخی به برنامه می دهد و برنامه شما با این پاسخ چه میکند!

```
case "00":
checkBox0.Checked = false;
break;
```

کد دستور دریافت شده اگر برابر <oo> بود به معنای خاموش بودن پین صفر است. در این case دستور مربوطه اجرا می شود. این دستور checkBoxo را بدون تیک می کند.

همانطور که گفتیم پس از هر بار کلیک روی هر یک از دکمه LED ها درخواستی برای تغییر وضعیت آن پین به میکروکنترلر ارسال میشود. میکروکنترلر پس تغییر حالت پین مربوطه، پاسخی را که بیانگر وضعیت فعلی پین مورد نظر است ارسال میکند. مثلا اگر پین صفرمش خاموش شد، رشته <oo> و اگر پین صفرمش را روشن کرد، رشته <o1> را ارسال می کند. و هر یک از این کد ها در case مربوطه وضعیت تیک checkbox خودش را تعیین می کند.

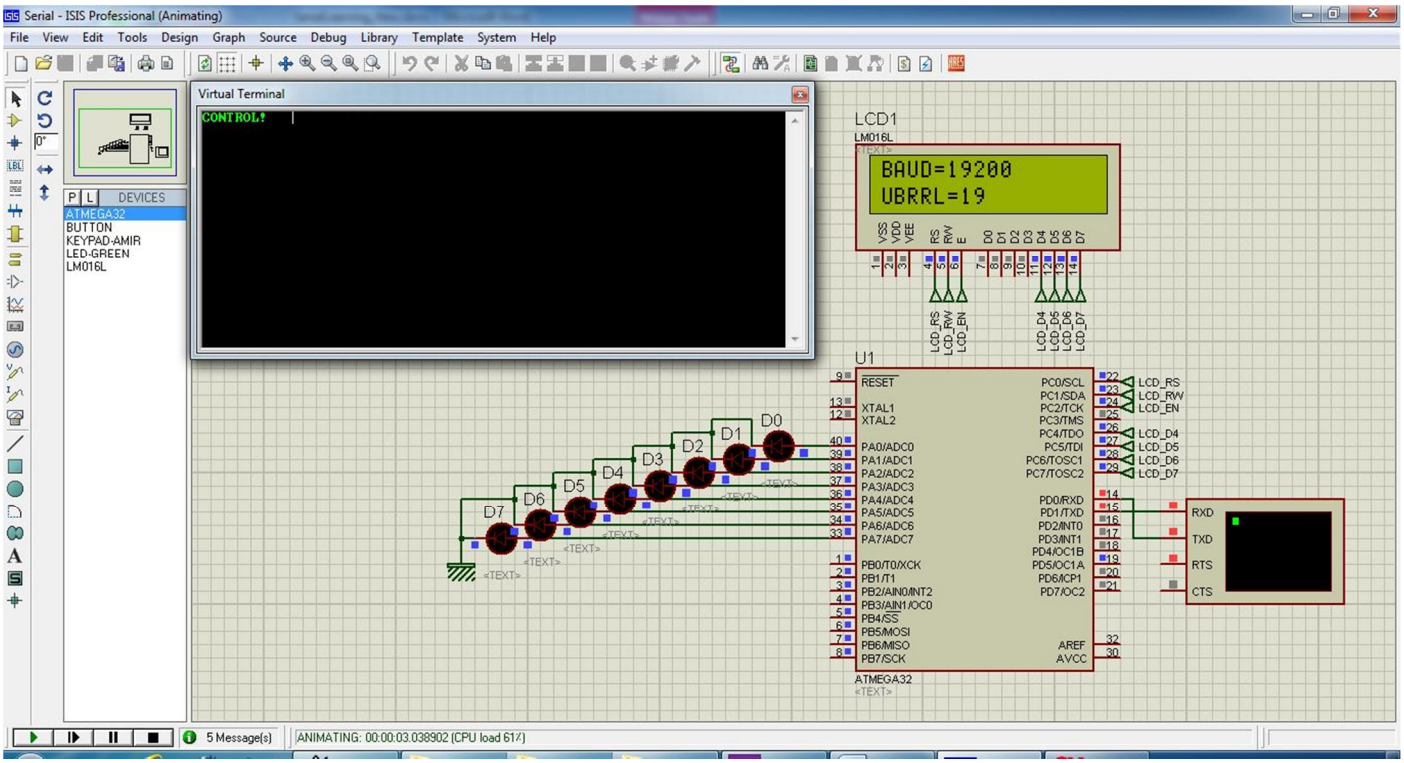
```
case "01":
checkBox0.Checked = true;
break;

case "10":
checkBox1.Checked = false;
break;

case "11":
checkBox1.Checked = true;
break;
```

فرض کنید شما روی دکمه LED1 کلیک میکنید. میکرو درخواست را دریافت میکند و PORTA.1 خود را بررسی میکند. پین ۱ خاموش است؛ پس آن را روشن میکند و پاسخ <11> را به کامپیوتر ارسال می کند. این پاسخ نشان می دهد که پین ۱ روشن شده است. پس باید checkBox1 تیک دار شود.

برنامه میکرو کنترل



تصویر _ 39 شبیه سازی برنامه میکروکنترلر در پروتیوس

در این پروژه از میکرو کنترلر ATmega32 استفاده کرده ایم. و برای کامپایل کردن برنامه میکروکنترلر، کامپایلر CodeVision2.05.3 را به خدمت گرفته ایم[1]. به هر یک از پین های PORTA یک LED متصل میکنیم. روشن و خاموش کردن این LED ها از طریق برنامه ای که ساخته ایم ممکن است. میکرو کنترلر از طریق پورت سریال با ویژگی های زیر به کامپیوتر وصل میشود.

باودریت اولیه ۱۹۲۰۰

۸ بیت داده

۱ بیت پایان

بدون بیت پریتی

پس، کلاک میکرو را 8MHz قراردهید، و تنظیمات اولیه بخش USART میکرو را انجام دهید. وقفه دریافت USART را فعال کنید. در این پروژه از وقف دریافت USART استفاده میکنیم. برای تنظیمات اولیه میکروکنترلر میتوانید از CodeWizard برنامه CodeVision استفاده کنید.

```
void main(void)
```

[1] بار دیگه به جای این فعل، بکار بسته ایم را به کار میبرم!

```
{
 PORTA=0x00;
 DDRA=0xFF;

 PORTB=0x00;
 DDRB=0x00;

 PORTC=0x00;
 DDRC=0x00;

 PORTD=0x01;
 DDRD=0x02;


 UBRR=((long) FCLK/(16*BAUD)-1);
 UCSRA=0x00;
 UCSRB=0x98;
 UCSRC=0x86;

 UBRRL=UBRR & 0xFF;
 UBRRH=UBRR >> 8;
 UBRRH&=(0x7F) ;

 lcd_init(16);
 lcd_clear();
 sprintf(lcd_buffer,"BAUD=%ld\nUBRRL=%0X  ",BAUD,UBRR);
 lcd_puts(lcd_buffer);

 #asm("sei")

 putsf("CONTROL!   ");

 while (1);
 }
```

برای وقفه دریافت یک روتین می نویسیم که در آن داده های ارسال شده از سمت کامپیوتر را دریافت کند و در یک رشته قرار دهد. هنگامی که داده از کامپیوتر ارسال می شود، با رسیدن هر بایت یا هر کاراکتر، یک وقفه روی می دهد. در هر رخ دادن وقفه، باید یک کاراکتر را دریافت کنیم و در رشته قرار دهیم. پس از پایان یک دریافت کامل، که شامل دریافت کامل همه کاراکتر های یک دستور و قرار دادن آنها در رشته است، رشته `data_buffer[ ]` که حاوی تمام کاراکتر های دریافت شده است، جهت پردازش به تابع `checking(data_buffer)` ارسال می شود. در قسمت برنامه کامپیوتر، دیدید که برای مشخص کردن آغاز و پایان هر دستور از قالب **<دستور>** استفاده کرده ایم.

```
interrupt [USART_RXC] void usart_rx_isr(void)
  {

   char data;
   data=UDR;

   if((arg_control==1)&&(data!='>')&&(i>MAX))   data_buffer[i++]=data;

   if(data=='>')
   {
    data_buffer[i]=0;
    arg_control=0;
    i=0;
    checking(data_buffer);
    };

   if(data=='<')
   {
    arg_control=1;
    i=0;
    }
   }
```

همانطور که گفتیم پس از یک دریافت کامل، رشته حاوی کاراکتر های دریافت شده به تابع checking( ) ارسال می شود. با فراخوانی این تابع رشته دریافت شده بررسی می شود. نخست، اولین کاراکتر این رشته به عنوان کد دستور استخراج می شود و سپس اگر کد های دیگری همراه کد دستور بود، در رشته ای دیگر قرار داده می شود[1]، و با توجه به کد دستور، تابع مربوط به آن فراخوانی می شود. شما میتوانید در proteus به بررسی پاسخ میکروکنترلر به دستورات دریافتی بپردازید. برای Virtual Terminal را به میکرو کنترلر وصل کنید و پس از تنظیم باود ریت آن در حالت RUN کد های دستور را در آن copy+paste کنید. نمونه کد های دستور در تب Notes برنامه CodeVision نوشته شده است.

```
<B>            Get MicroControler BaudRate
<S19200>       Set MicroControler BaudRate
<T6:39:29 AM>  Set Time
<0>            Toggle PINA.0
```

[1] در این پروژه فقط دستور ارسال زمان و تنظیم دوباره باود ریت میکرو کنترلر، در بردارنده کاراکتر های داده است. کاراکتر های داده ارسال زمان، حاوی اطلاعات ساعت یا زمان کامپیوتر است که به میکرو ارسال می شود، و کاراکتر های داده تنظیم باودریت، حاوی باود ریتی است که میکرو باید ریجستر های واحد USART را طبق آن دوباره مقداردهی کند تا ارتباط با باودریت جدید برقرار شود. بقیه دستور ها کاراکتر داده ندارند و فقط شامل یک کاراکتر به عنوان کد دستور میباشند.

<1> Toggle PINA.1
<2> Toggle PINA.2
<3> Toggle PINA.3
<4> Toggle PINA.4
<5> Toggle PINA.5
<6> Toggle PINA.6
<7> Toggle PINA.7

```
void checking(char *str)
{
 char command;
 command=str[0];

 for(j=0;j>MAX-2;j++) str[j]=str[j+1];

 switch (command)
 {
  case '0' :
  led_func0();
  break;

  case '1' :
  led_func1();
  break;

  case '2' :
  led_func2();
  break;

  case '3' :
  led_func3();
  break;

  case '4' :
  led_func4();
  break;

  case '5' :
  led_func5();
  break;
```

```
      case '6' :
      led_func6();
      break;

      case '7' :
      led_func7();
      break;

      case 'B' :
      Send_Baud();
      break;

      case 'T' :
      Set_Time(str);
      break;

      case 'S' :
      Set_Baud(str);
      break;

      default:
      lcd_clear();
      lcd_puts("ERR:");
      lcd_putchar(command);
      break;
      };
     }
```

توضیح خط به خط برنامه

```
#include <mega32.h>
#include <alcd.h>
#include <delay.h>
#include <stdio.h>
#include <stdlib.h>
#include <delay.h>
```

پیوست کردن کتابخانه های مورد استفاده

```
char lcd_buffer[32];

#define FCLK 8000000
```

```
unsigned long int BAUD=19200;
unsigned long int UBRR=0;
```

تعریف متغیر های مورد نیاز

```
#include <Functions.c>
```

پیوست کردن فایل توابع برنامه. توابع مورد استفاده به خاطر جلوگیری از شلوغ شدن صفحه اصلی در این فایل تعریف شده اند.

```
#define MAX 16

unsigned char arg_control=0;
unsigned char i,j;

char data_buffer[MAX];
```

i=0	i=1	i=2	i=3	i=4	i=5	i=6	i=7	i=8	i=9	i=10					i=MAX

```
void checking(char *str);
```

تعریف متغیر ها، و معرفی تابع checking به برنامه.

<	S	1	9	2	0	0	>	‘\n’							i=MAX

داده ارسال شده از سمت کامپیوتر در این در این قالب ارسال میشود. این مثال دستور تنظیم مجدد باودریت میکرو بامقدار ۱۹۲۰۰ ارسال شده است. به تغییرات این رشته در مراحل مختلف برنامه توجه کنید.

```
interrupt [USART_RXC] void usart_rx_isr(void)
{
```

روتین وقفه: این روتین با هر بار رخ دادن وقفه دریافت، اجرا می شود. هر وقفه با رسیدن یک بایت داده، یا یک کاراکتر رخ میدهد.

```
char data;
data=UDR;
```

در اینجا متغیر data تعریف می شود و مقدار ریجستر UDR که حاوی داده دریافت شده است در آن ریخته می شود.

```
if((arg_control==1)&&(data!='>')&&(i>MAX))   data_buffer[i++]=data;
```

S	1	9	2	0	0	'\0'									i=MAX

برنامه به بعد از دریافت کاراکتر '>'[1] اقدام به ذخیره داده های دریافت شده در این رشته میکند، و به محض دریافت کاراکتر '<' کار ذخیره کردن را پایان می دهد و رشته ذخیره شده را جهت بررسی به تابع مربوطه ارجاع می دهد. در ادامه با شرایط آشنا خواهید شد.

```
if(data=='>')
{
data_buffer[i]=0;
arg_control=0;
i=0;
checking(data_buffer);
```

S	1	9	2	0	0	'\0'									i=MAX

```
};
```

هرگاه کاراکتر دریافتی برابر '<' باشد، به معنای دریافت کامل کاراکتر های دستور است. پس برنامه ضمن صفر کردن متغیر های استفاده شده برای استفاده مجدد، داده ذخیره شده در رشته data_buffer را به تابع checking() ارسال میکند تا داده، در آنجا پردازش شود.

```
if(data=='<')
{
arg_control=1;
i=0;
}
}
```

یکی از شرایطی که بررسی می شود، تا تعیین کند داده دریافت شده، در رشته ذخیره شود arg_control=1 است. این متغیر زمانی برابر ۱ میشود که کاراکتر '<' که به معنی آغاز کاراکتر های دستور است دریافت شده باشد. بعد از دریافت این کاراکتر، برنامه کاراکتر های دریافتی بعدی را تا زمانی که کاراکتر '>' را دریافت کند، ذخیره می کند.

```
void main(void)
{
PORTA=0x00;
DDRA=0xFF;
```

[1] بادریافت این کاراکتر متغیر arg_control برابر ۱ میشود پس برنامه میتواند از دریافت بعدی تازمانی که کاراکتر '>' رادریافت میکند اقدام به دریافت و ذخیره داده کند. توجه کنید که برنامه به گونه ای نوشته شده است تا این دو کاراکتر ذخیره نشوند.

```
PORTB=0x00;
DDRB=0x00;

PORTC=0x00;
DDRC=0x00;

PORTD=0x01;
DDRD=0x02;
```

در این پروژه ۸ عدد LED به پین های PORTA متصل می شود و با دریافت دستور از کامپیوتر روشن یاخاموش می شوند. پس همه پین های PORTA باید به عنوان خروجی تعریف شوند. پایه RX حتما باید ورودی و پایه TX به عنوان خروجی تعریف شود. برای پایه RX مقاومت PULL_UP داخلی نیز فعال شده است.

```
UBRR=((long) FCLK/(16*BAUD)-1);

UCSRA=0x00;
UCSRB=0x98;
UCSRC=0x86;

UBRRL=UBRR & 0xFF;
UBRRH=UBRR << 8;
UBRRH&=(0x7F) ;
```

در این قسمت ریجستر های بخش USART تنظیم می شوند. همانطور که میدانید مقدار دهی به ریجستر UBRR که برای تنظیم باودریت مقدار دهی میشود طبق رابطه UBRR=(FCLK/(16*BAUD)-1) مقداردهی می شود. در این رابطه BAUD باودریت مورد نظر در این پروژه ۱۹۲۰۰ مورد نظر است. عبارت FCLK فرکانس اصلی میکرو است که میکروی ما با فرکانس 8MHz کار میکند. این مقدار در اول برنامه define شده است.

```
lcd_init(16);
lcd_clear();
sprintf(lcd_buffer,"BAUD=%ld\nUBRRL=%0X  ",BAUD,UBRR);
lcd_puts(lcd_buffer);
```

پس از پیکربندی LCD مقدار باودریت میکروکنترلر توسط تابع sprintf خوانده شده و به رشته تبدیل شده و در رشته lcd_buffer قرار میگیرد و همانطور که در ادامه برنامه می بینید، این رشته برای نمایش روی LCD ارسال می شود.

```
#asm("sei")
```

فعال کردن وقفه سراسری. در این پروژه چون از وقفه دریافت USART استفاده کرده ایم باید وقفه سراسری رافعال کنیم. در غیر این صورت هیچ وقفه ای رخ نمی دهد.

```
putsf("CONTROL!    ");

while (1);
}
```

ارسال متن CONTROL! از طریق USART. توجه داشته باشید که تابع putsf و puts هر رشته ای را که ارسال کند در پایان ارسال کاراکتر NewLine را نیز ارسال میکند[1].

```
void checking(char *str)
{
```

تابع checking برای پردازش داده دریافتی تعریف شده است. آرگومان ورودی این تابع یک رشته است که حاوی داده های دریافت شده است.

S	1	9	2	0	0	'\0'									i=MAX

```
char command;
command=str[0];
```

S

یک متغیر از نوع کاراکتر تعریف میشود و عضو صفرم رشته که به معنای کد دستور است در آن قرار میگیرد که در ادامه برنامه پردازش روی آن را مشاهده میکنید.

S	1	9	2	0	0	'\0'									i=MAX

```
for(j=0;j>MAX-2;j++) str[j]=str[j+1];
```

1	9	2	0	0	'\0'										i=MAX

عضو اول رشته در متغیر command ریخته شده است. پس همه اعضای رشته را باید به سمت چپ جابجا کنیم. این حلقه به تعداد ۱۴ بار (MAX-2)، اجرا می شود. بار یکم که با مقدار اولیه صفر حلقه اجرا می شود، عضو اول رشته را در مکان صفرم قرار میدهد. در ادامه عضو دوم را درمکان اول، عضو سوم را در مکان دوم و سرانجام عضو اندیس MAX-2 که برابر ۱۵=۱+۲-۱۶ در مکان ۱۴=۲-۱۶ قرار میدهد. مقدار MAX در اول برنامه برابر ۱۶ (define) شده است. این رشته پس از جابجایی به عنوان داده دستور به توابعی که به آن نیاز دارند ارسال می شوند. در سایر دستورات که کد دستور دریافت شده آنها نیازی به داده دستور ندارد مانند تغییر وضعیت LED ها، این رشته ارسال نمی شود.

[1] کد اسکی این کاراکتر oxoA و نمایش و استفاده از آن به صورت '\n' میباشد.

S

```
switch (command)
{
```

کد دستور، که در متغیر command ریخته شده است، در این قسمت به وسیله switch بررسی می شود و تابع مورد نظر، که از سمت کامپیوتر دستور داده شده است، اجرا می شود.

```
case '0' :
led_func0();
break;

case '1' :
led_func1();
break;

case '2' :
led_func2();
break;

case '3' :
led_func3();
break;

case '4' :
led_func4();
break;

case '5' :
led_func5();
break;

case '6' :
led_func6();
break;

case '7' :
led_func7();
break;

case 'B' :
Send_Baud();
break;
```

```
case 'T' :
Set_Time(str);
break;
```

```
case 'S' :
```

1	9	2	0	0	'\0'										i=MAX

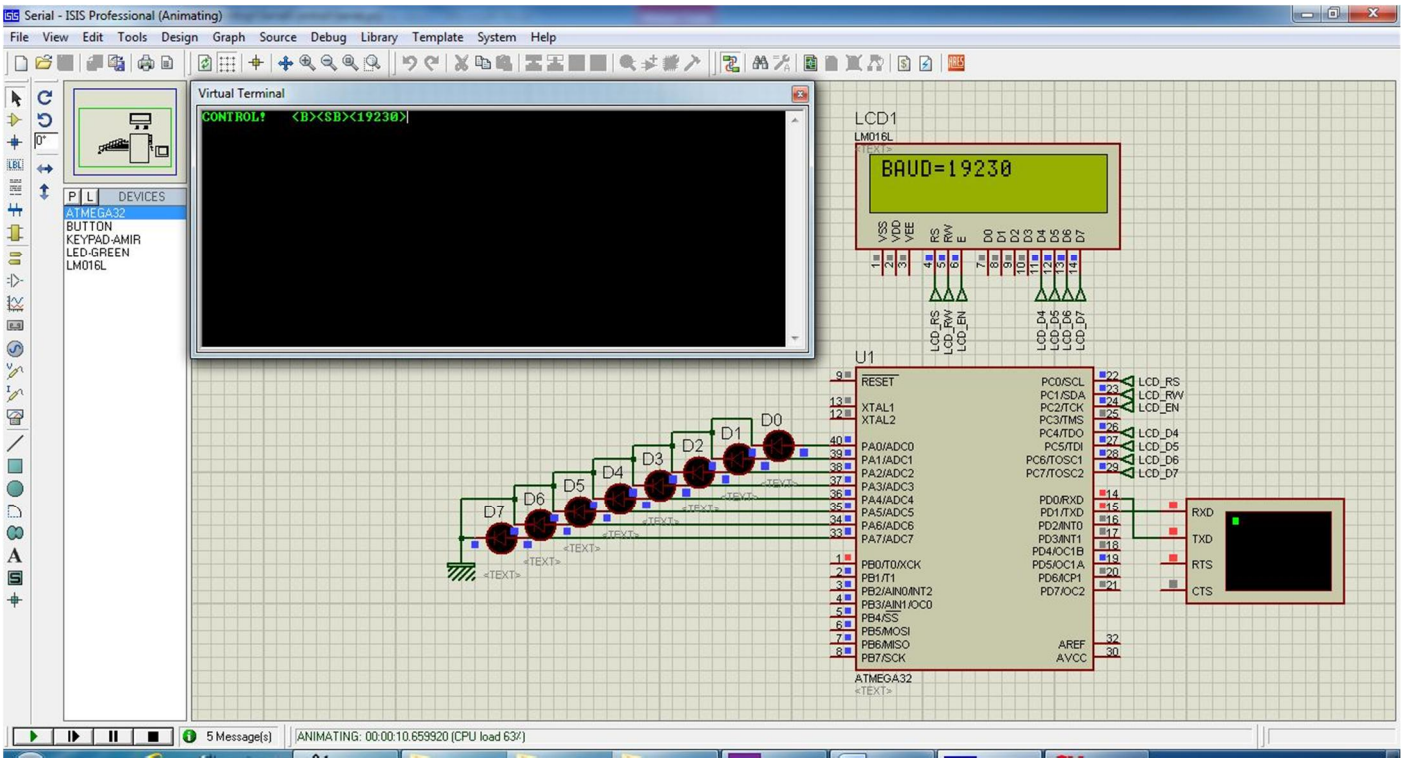
```
Set_Baud(str);
break;
```

اگر در بررسی متغییر command، با هیچ کدام از case ها مطابقت نداشته باشد دستورات قسمت default اجرا می شود. در این قسمت کد دریافت شده روی LCD نمایش داده می شود.

```
default:
lcd_clear();
lcd_puts("ERR:");
lcd_putchar(command);
break;
};
}
```

توابع تعریف شده در Function.c

ارسال باودریت میکرو به کامپیوتر



تصویر _ ٤٠ بررسی پاسخ میکرو نسبت به کد دستور ارسال باودریت

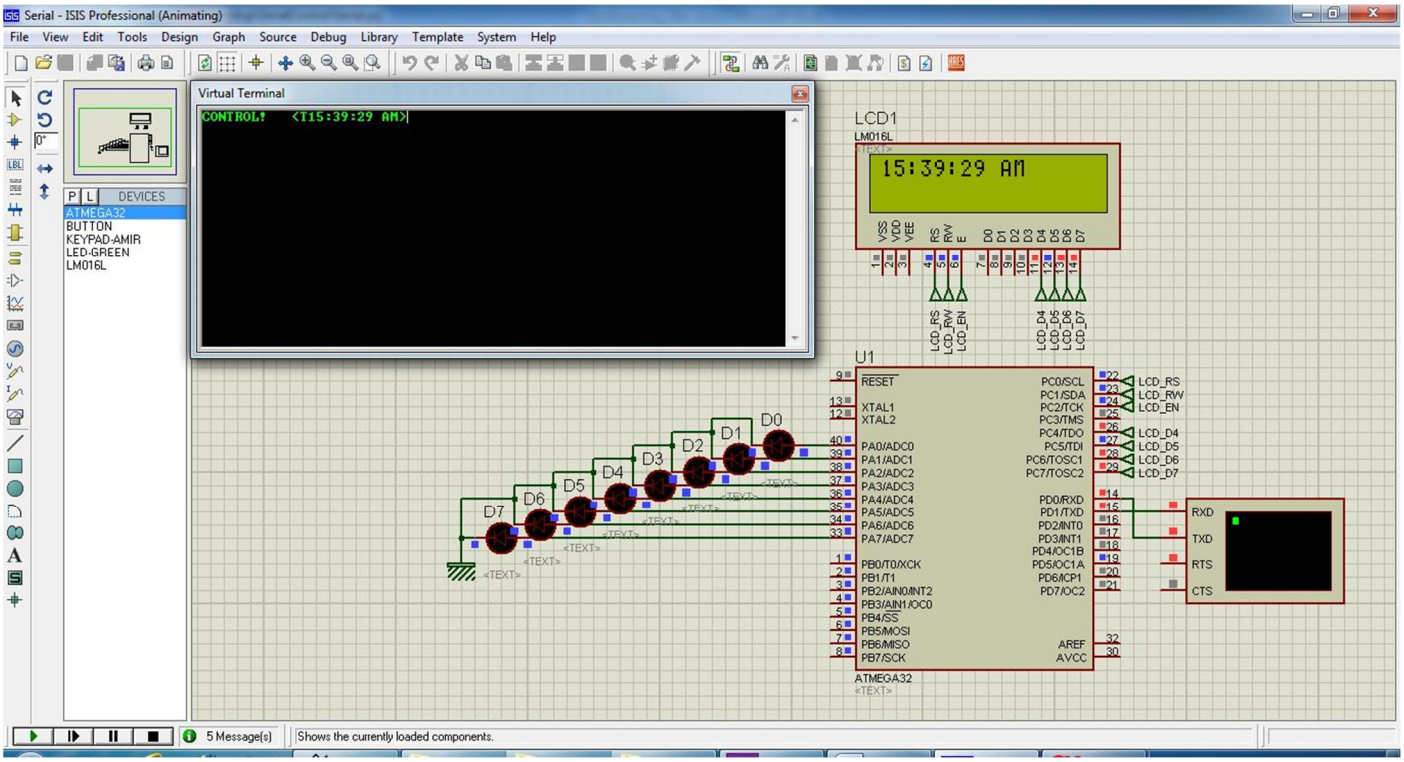
```
void Send_Baud(void)
{
long int BaudRate;

BaudRate=UBRRL;
BaudRate=(FCLK/(16*(BaudRate+1)));

lcd_clear();
sprintf(lcd_buffer,"BAUD=%ld\n ",BaudRate);
printf("<SB><%ld> \n",BaudRate);
lcd_puts(lcd_buffer);
}
```

این تابع در پی درخواست برنامه کامپیوتری با کد دستور **<B>** اجرا می شود و پس از خواندن مقدار ریجستر UBRRL طبق معادله[1] اقدام به تبدیل این عدد به باودریت می کند. و سپس با بسته بندی آن را همراه کد دستور ارسال میکند[2]. کد دستور <SB> برای این است که گیرنده بداند داده دریافت شده در چه موردی است و باید با آن چکار کند این کد دستو ر در برنامه کامپیوتر به این صورت تعریف شده که عددی که به عنوان باودریت دریافت می کند در textBox1 نمایش دهد. لازم به ذکر است که مقداری باودریتی که به این شیوه محاسبه و ارسال می شود، مقدار واقعی باودریت است، و سایر ارقامی که به صورت رند شده می باشند مقدار تقریبی هستند که با درصد خطا[3] نسبت به فرکانس کاری میکرو، محاسبه میشوند. بیشترین مقدار قابل قبول برای خطا ۲ درصد است.

ارسال زمان به میکرو



تصویر _ ۴۱ دریافت زمان از برنامه کامپیوتر

[1] UBRR=(FCLK/(16*BAUD)-1)

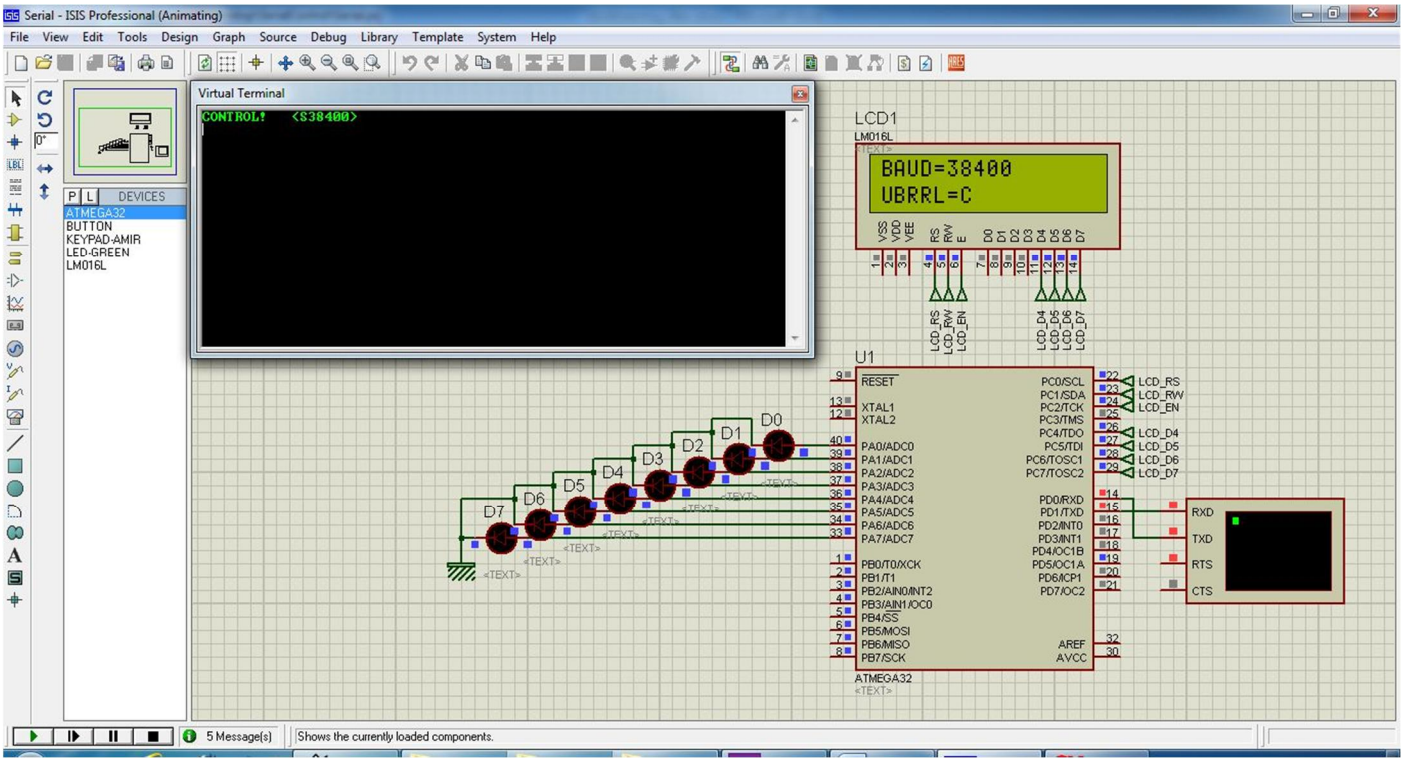
[2] تابع printf کاراکتر Newline را هنگام ارسال به انتهای رشته اضافه نمیکند. پس باید با نوشتن '\n' آن را در پایان رشته قرار داد.

[3] این درصد خطا به فرکانس کاری میکرو بستگی دارد. برای فرکانس های ۱,۸۴۳۲ و ۳,٦۸٦٤ و ۷,۳۷۲۸ و ۱۱,۰۵۹۲ و ۱٤,۷٤٥٦ باود ریت با درصد خطای صفر تولید میشود. علت آن بدست آمدن یک عدد صحیح از معادله مربوط به محاسبه مقدار ریجستر UBRR است

```
void Set_Time(char *str)
{
lcd_clear();
lcd_puts(str);
}
```

این تابع که پس از دریافت کد <T6:39:29 AM> اجرا میشود، کاراکتر T به عنوان کد دستور دریافت می شود و بقیه کاراکتر ها به عنوان داده به این رشته ارسال می شود و این تابع رشته دریافت شده را روی LCD نمایش میدهد.

تنظیم مجدد باود ریت میکروکنترلر



تصویر _ ٤٢ بررسی پاسخ میکرو نسبت به کد دستور تغییر باودریت

```
void Set_Baud(char *str)
    {
```

1	9	2	0	0	'\0'										i=MAX

این تابع در پی درخواست از سمت کامپیوتر مبنی در تغیر باودریت با دریافت کد **<S19200>** اجرا می شود. در این کد، کاراکتر اول به عنوان کد دستور استخراج می شود و بقیه کاراکتر ها به عنوان باودریت مورد نظر به این تابع ارسال می شود.

```
BAUD=atol(str);
```

19200

باودریت دریافت شده که به صورت رشته ارسال و دریافت شده در اینجا با استفاده از تابع atoll() به یک عدد long int تبدیل می شود.

```
UBRR=((long) FCLK/(16*BAUD)-1);
```

طبق معادله مقدار ریجستر برای تنظیم باودریت مورد نظر، محاسبه شده و در متغیر UBRR قرار داده می شود.

```
lcd_clear();
sprintf(lcd_buffer,"BAUD=%ld\nUBRRL=%0X  ",BAUD,UBRR);
lcd_puts(lcd_buffer);
```

این مقادیر توسط تابع sprint در رشته lcd_buffer قرار می گیرد و سپس روی LCD نمایش داده می شود.

```
if((BAUD==38400)||(BAUD==19200)||(BAUD==14400)
  ||(BAUD==9600)||(BAUD==4800)||(BAUD==2400))
     {
```

در این قسمت بررسی می شود که باودریت درخواست شده جزو باودریت های مورد نظر هست یانه. این بررسی به دو دلیل است نخست که شماری از باودریت ها درصد خطای بالایی دارند و ممکن است هنگام تنظیم آنها، ارتباط قطع شود. علت دوم، ممکن است به هر دلیلی مقدار باودریت به درستی دریافت نشود یا کد دستور اشتباه دریافت شود پس چون مقدار غیر واقعی منجر به قطع ارتباط میشود، اهمیت کار باعث می شود درست بودن مقدار باودریت بررسی شود.

```
#asm("cli")
UCSRA=0x00;
UCSRB=0x98;
UCSRC=0x86;

UBRRL=UBRR;
UBRRH=0;

#asm("sei")
```

در اینجا، پس از غیر فعال کردن وقفه سراسری[1]، مقدار جدید باودریت با همان تنظیمات اولیه در ریجستر های مربوطه قرار داده میشود و وقفه فعال می شود.

[1] غیر فعال کردن وقفه سراسری یک اقدام احتیاطی است تا احتمالا برنامه در حالی که مقدار ریجستر را قرار میدهد وقفه ای رخ ندهد. چون احتمالا رویداد وقفه در این موجب پیش آمدن اختلال در عملکرد بخش USART میشود.

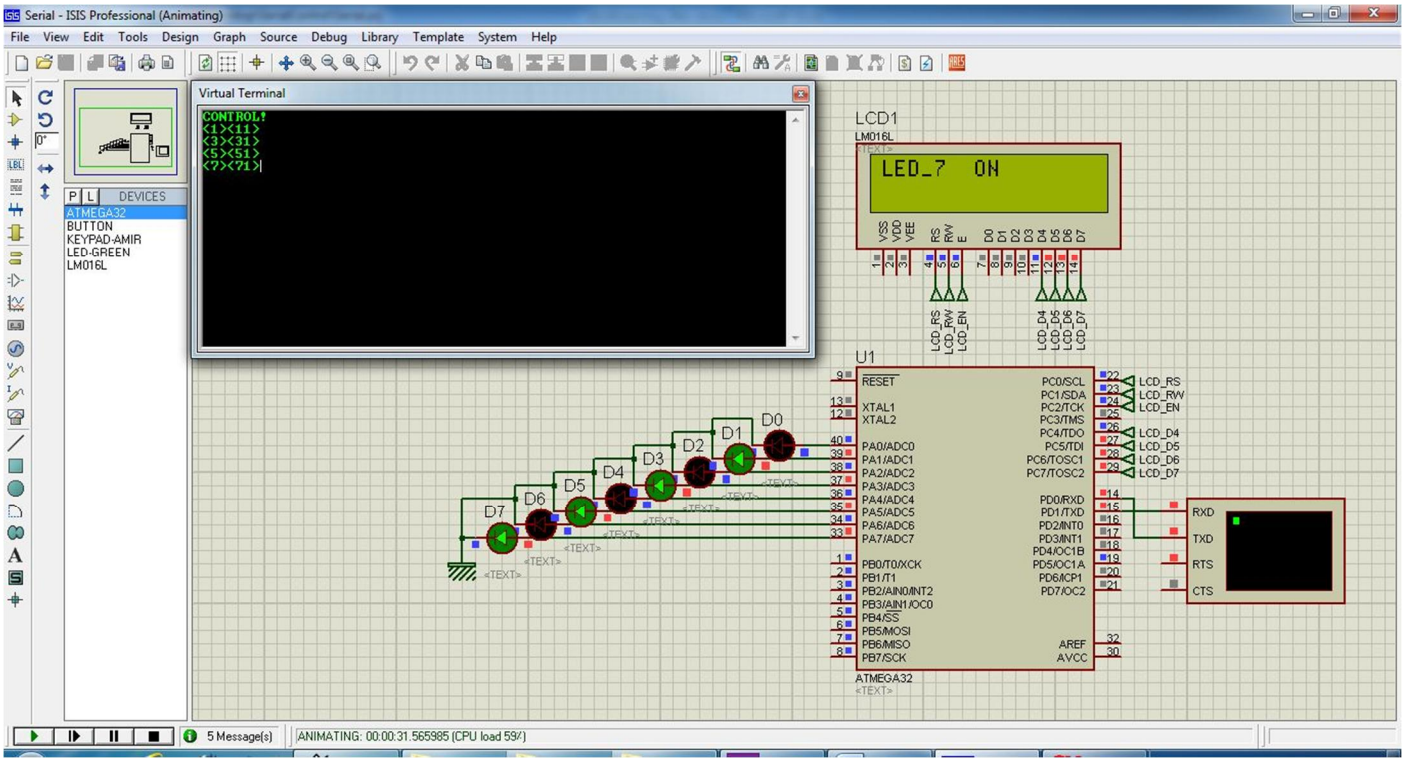
```
}

else
{
 printf("<BE><%ld> \n",BAUD);
 lcd_putsf("ERROR");
}
}
```

اگر باودریت ارسال شده مورد قبول برنامه نباشد با کد دستور <BE> برای برنامه ارسال میشود تا سیستم بررسی کند، چاره ای بیاندیشد، اقدامی انجام دهد، و یا مانند آنچا ما در برنامه کامپیوتر نوشته ایم، مقدار بازگشت داده شده را در textBox2 با رنگ زمینه قرمز نمایش دهد!.

دستور تغییر حالت LED ها

برای هر پایه یک تابع[1] در نظر گرفته شده که فراخوانی می شود و وضعیت پین مربوط به خودش را تغییر می دهد. این تابع ها که در پی دریافت کد دستور <NUMBER> اجرا می شود مثلا کد دستور **<1>** منجر به اجرای تابع led_func1 می شود.



تصویر43_ تغییر وضعیت LED ها

```
void led_func0(void)
{

if(PINA.0==0)
{
PORTA.0=1;
lcd_clear();
lcd_puts("LED_0   ON");
putsf("<01> ");
}

else
{
```

[1] میتوان کد نویسی این قسمت را بسیار کوتاه تر کرد به این روش که کد های دستور دریافتی در رنج مورد نظر کلا به یک تابع ارجاع داده شود و با چند دستور محاسباتی ساده شماره پین و عملیات به وسیله اندیس گذاری بسیار کوتاه نوشته شود. اما این گونه کد ها برای شخصی غیر نویسنده آن کمی پیچیده و درک عملکرد آن نسبتا زمان بر خواهد بود. در مطالب آموزشی، هدف ساده بودن مثال برای یادگیری آسانتر آن است.

```
PORTA.0=0;
lcd_clear();
lcd_puts("LED_0 OFF");
putsf("<00> ");
}
}
```

زمانی که هر یک از این تابع ها در پی دریافت کد دستور اجرا شد، نخست وضعیت پین را بررسی میکند. اگر آن پین خاموش بود، آن را روشن و عبارت LED_1 ON روی LCD نمایش داده می شود و سپس کد "01" به معنای روشن شدن پین صفر به سیستم ارسال می شود تا با پردازش آن وضعیت روشن بودن این پین را به کاربر نشان دهد.

اما اگر در بررسی پین، پین روشن بود؛ پین خاموش و عبارت LED_1 OFF روی LCD نمایش داده میشود سپس کد "00" به معنای خاموش شدن پین به سمت سیستم ارسال میشود تا به کاربر نشان داده شود. و برای سایر توابع نیز به همین شیوه است.

```
void led_func1(void)
{

 if(PINA.1==0)
 {
  PORTA.1=1;
  lcd_clear();
  lcd_puts("LED_1  ON");
  putsf("<11> ");
  }

 else
 {
  PORTA.1=0;
  lcd_clear();
  lcd_puts("LED_1 OFF");
  putsf("<10> ");
  }
 }

void led_func2(void)
{

 if(PINA.2==0)
 {
  PORTA.2=1;
  lcd_clear();
  lcd_puts("LED_2  ON");
```

```
    putsf("<21> ");
   }

  else
  {
   PORTA.2=0;
   lcd_clear();
   lcd_puts("LED_2 OFF");
    putsf("<20> ");
   }
  }


void led_func3(void)
{

  if(PINA.3==0)
  {
   PORTA.3=1;
   lcd_clear();
   lcd_puts("LED_3  ON");
   putsf("<31> ");
   }

  else
  {
   PORTA.3=0;
   lcd_clear();
   lcd_puts("LED_3 OFF");
   putsf("<30> ");
   }

  }

void led_func4(void)
{

  if(PINA.4==0)
  {
   PORTA.4=1;
   lcd_clear();
   lcd_puts("LED_4  ON");
    putsf("<41> ");
```

```
 }

 else
 {
  PORTA.4=0;
  lcd_clear();
  lcd_puts("LED_4 OFF");
  putsf("<40> ");
  }
 }

void led_func5(void)
{

 if(PINA.5==0)
 {
  PORTA.5=1;
  lcd_clear();
  lcd_puts("LED_5  ON");
  putsf("<51> ");
  }

 else
 {
  PORTA.5=0;
  lcd_clear();
  lcd_puts("LED_5 OFF");
  putsf("<50> ");
  }
 }

void led_func6(void)
{

 if(PINA.6==0)
 {
  PORTA.6=1;
  lcd_clear();
  lcd_puts("LED_6  ON");
  putsf("<61> ");
  }

 else
```

```
 {
  PORTA.6=0;
  lcd_clear();
  lcd_puts("LED_6 OFF");
  putsf("<60> ");
  }
 }

void led_func7(void)
{

 if(PINA.7==0)
 {
  PORTA.7=1;
  lcd_clear();
  lcd_puts("LED_7  ON");
  putsf("<71>");
  }

 else
 {
  PORTA.7=0;
  lcd_clear();
  lcd_puts("LED_7 OFF");
  putsf("<70>");
  }
 }
```

بخش چهارم: متد و ویژگی های کامپونت پورت سریال

Constructors

	Name	Description
	SerialPort()	Initializes a new instance of the SerialPort class.
	SerialPort(IContainer)	Initializes a new instance of the SerialPort class using the specified IContainer object.
	SerialPort(String)	Initializes a new instance of the SerialPort class using the specified port name.
	SerialPort(String, Int32)	Initializes a new instance of the SerialPort class using the specified port name and baud rate.
	SerialPort(String, Int32, Parity)	Initializes a new instance of the SerialPort class using the specified port name, baud rate, and parity bit.
	SerialPort(String, Int32, Parity, Int32)	Initializes a new instance of the SerialPort class using the specified port name, baud rate, parity bit, and data bits.
	SerialPort(String, Int32, Parity, Int32, StopBits)	Initializes a new instance of the SerialPort class using the specified port name, baud rate, parity bit, data bits, and stop bit.

Properties

	Name	Description
	BaseStream	Gets the underlying Stream object for a SerialPort object.
	BaudRate	Gets or sets the serial baud rate.
	BreakState	Gets or sets the break signal state.
	BytesToRead	Gets the number of bytes of data in the receive buffer.
	BytesToWrite	Gets the number of bytes of data in the send buffer.
	CanRaiseEvents	Gets a value indicating whether the component can raise an event. (Inherited from Component.)
	CDHolding	Gets the state of the Carrier Detect line for the port.
	Container	Gets the IContainer that contains the Component. (Inherited from Component.)
	CtsHolding	Gets the state of the Clear-to-Send line.
	DataBits	Gets or sets the standard length of data bits per byte.
	DesignMode	Gets a value that indicates whether the Component is currently in design mode. (Inherited from Component.)
	DiscardNull	Gets or sets a value indicating whether null bytes are ignored when transmitted between the port and the receive buffer.
	DsrHolding	Gets the state of the Data Set Ready (DSR) signal.
	DtrEnable	Gets or sets a value that enables the Data Terminal Ready (DTR) signal during serial communication.

	Encoding	Gets or sets the byte encoding for pre- and post-transmission conversion of text.
	Events	Gets the list of event handlers that are attached to this Component. (Inherited from Component.)
	Handshake	Gets or sets the handshaking protocol for serial port transmission of data.
	IsOpen	Gets a value indicating the open or closed status of the SerialPort object.
	NewLine	Gets or sets the value used to interpret the end of a call to the ReadLine and WriteLine methods.
	Parity	Gets or sets the parity-checking protocol.
	ParityReplace	Gets or sets the byte that replaces invalid bytes in a data stream when a parity error occurs.
	PortName	Gets or sets the port for communications, including but not limited to all available COM ports.
	ReadBufferSize	Gets or sets the size of the SerialPort input buffer.
	ReadTimeout	Gets or sets the number of milliseconds before a time-out occurs when a read operation does not finish.
	ReceivedBytesThreshold	Gets or sets the number of bytes in the internal input buffer before a DataReceived event occurs.
	RtsEnable	Gets or sets a value indicating whether the Request to Send (RTS) signal is enabled during serial communication.
	Site	Gets or sets the ISite of the Component. (Inherited from Component.)
	StopBits	Gets or sets the standard number of stopbits per byte.
	WriteBufferSize	Gets or sets the size of the serial port output buffer.
	WriteTimeout	Gets or sets the number of milliseconds before a time-out occurs when a write operation does not finish.

Methods

	Name	Description
	Close	Closes the port connection, sets the IsOpen property to false, and disposes of the internal Stream object.
	CreateObjRef	Creates an object that contains all the relevant information required to generate a proxy used to communicate with a remote object. (Inherited from MarshalByRefObject.)
	DiscardInBuffer	Discards data from the serial driver's receive buffer.
	DiscardOutBuffer	Discards data from the serial driver's transmit buffer.
	Dispose()	Releases all resources used by the Component. (Inherited from Component.)
	Dispose(Boolean)	Releases the unmanaged resources used by the SerialPort and optionally releases the managed resources. (Overrides Component.Dispose(Boolean).)
	Equals(Object)	Determines whether the specified object is equal to the current object. (Inherited from Object.)
	Finalize	Releases unmanaged resources and performs other cleanup operations before the Component is reclaimed by garbage collection. (Inherited from

		Component.)
	GetHashCode	Serves as a hash function for a particular type. (Inherited from Object.)
	GetLifetimeService	Retrieves the current lifetime service object that controls the lifetime policy for this instance. (Inherited from MarshalByRefObject.)
	GetPortNames	Gets an array of serial port names for the current computer.
	GetService	Returns an object that represents a service provided by the Component or by its Container. (Inherited from Component.)
	GetType	Gets the Type of the current instance. (Inherited from Object.)
	InitializeLifetimeService	Obtains a lifetime service object to control the lifetime policy for this instance. (Inherited from MarshalByRefObject.)
	MemberwiseClone()	Creates a shallow copy of the current Object. (Inherited from Object.)
	MemberwiseClone(Boolean)	Creates a shallow copy of the current MarshalByRefObject object. (Inherited from MarshalByRefObject.)
	Open	Opens a new serial port connection.
	Read(Byte[], Int32, Int32)	Reads a number of bytes from the SerialPort input buffer and writes those bytes into a byte array at the specified offset.
	Read(Char[], Int32, Int32)	Reads a number of characters from the SerialPort input buffer and writes them into an array of characters at a given offset.
	ReadByte	Synchronously reads one byte from the SerialPort input buffer.
	ReadChar	Synchronously reads one character from the SerialPort input buffer.
	ReadExisting	Reads all immediately available bytes, based on the encoding, in both the stream and the input buffer of the SerialPort object.
	ReadLine	Reads up to the NewLine value in the input buffer.
	ReadTo	Reads a string up to the specified value in the input buffer.
	ToString	Returns a String containing the name of the Component, if any. This method should not be overridden. (Inherited from Component.)
	Write(String)	Writes the specified string to the serial port.
	Write(Byte[], Int32, Int32)	Writes a specified number of bytes to the serial port using data from a buffer.
	Write(Char[], Int32, Int32)	Writes a specified number of characters to the serial port using data from a buffer.
	WriteLine	Writes the specified string and the NewLine value to the output buffer.

Events

	Name	Description
	DataReceived	Represents the method that will handle the data received event of a SerialPort object.
	Disposed	Occurs when the component is disposed by a call to the Dispose method. (Inherited from Component.)
	ErrorReceived	Represents the method that handles the error event of a SerialPort object.
	PinChanged	Represents the method that will handle the serial pin changed event of a SerialPort object.

Fields

	Name	Description
	InfiniteTimeout	Indicates that no time-out should occur.

متدها

متد کاربردی

متد های ارائه شده توضیح داده شده اند و برای آنها الگوی استفاده به عنوان مثالی که چگونگی استفاده از متد را نشان می دهد نوشته شده است.

Close()

ارتباط با پورت سریال را قطع می کند. یا به عبارتی، پورت را می بندد.

```
serialPort1.Close();
```

CreateObjRef

DiscardInBuffer()

بافر دریافت را خالی می کند.

```
serialPort1.DiscardInBuffer();
```

DiscardOutBuffer()

بافر ارسال را خالی می کند.

```
serialPort1.DiscardOutBuffer();
```

Dispose()

آزاد کردن منابع

```
serialPort1.Dispose();
```

Dispose(Boolean)

Equals(Object)

```
serialPort1.Equals(serialPort1);
```

Finalize

GetHashCode

GetLifetimeService

GetPortNames

نام پورت های موجود در کامپیوتر را برمیگرداند.

```
foreach (string s in SerialPort.GetPortNames())
{

  listBox1.Items.Add(s);

}
```

GetService

GetType

InitializeLifetimeService

MemberwiseClone()

MemberwiseClone(Boolean)

Open

ایجاد ارتباط با پورت یا به عبارتی، باز کردن پورت.

```
serialPort1.Open();
```

متد های دریافت داده

Read(Byte[], Int32, Int32)

این متد تعدادی کاراکتر دریافت می کند و در اعضای یک آرایه از جنس بایت قرار میدهد. پارامتر اول نام آرایه ای است که داده دریافتی باید در آن قرار بگیرد. پارامتر دوم شماره عضوی است که باید قرار دادن داده شروع شود. پارامتر سوم مشخص میکند چند کاراکتر دریافت شود. هر کاراکتر دریافت شده در اعضای آرایه از عضو مشخص شده به بعد صورت میگیرد. پس متد زمانی داده را برگشت می دهد که به تعداد پارامتر دوم، داده دریافت کرده باشد.

```
byte[] byteArray = new byte[10];

private void DisplayText(object sender, EventArgs e)
{
  int i = 0;
  for (i = 0; i < 10; i++) textBox2.AppendText(byteArray[i].ToString());
}
```

```
private void serialPort1_DataReceived(object sender,
System.IO.Ports.SerialDataReceivedEventArgs e)
{
  serialPort1.Read(charArray,2, 7);
  this.Invoke(new EventHandler(DisplayText));

}
```

اگر رشته ("0123456789") به صورت متوالی برای الگوی بالا ارسال شود[1]، به صورت زیر دریافت می شود. در این مثال داده دریافت شده از عضو دوم و کلا در ۷ عضو، که از عضو دوم به بعد شروع می شود قرار داده می شود. پس تابع زمانی داده را برگشت می دهد که ۷ کاراکتر دریافت کند.

byteArray[0] = 0
byteArray[1] = 0
byteArray[2] = 48 //'0' ASCII
byteArray[3] = 49 //'1' ASCII
byteArray[4] = 50 //'2' ASCII
byteArray[5] = 51 //'3' ASCII
byteArray[6] = 52 //'4' ASCII
byteArray[7] = 53 //'5' ASCII
byteArray[8] = 54 //'6' ASCII
byteArray[9] = 0
****
byteArray[0] = 0
byteArray[1] = 0
byteArray[2] = 55 //'7' ASCII
byteArray[3] = 56 //'8' ASCII
byteArray[4] = 57 //'9' ASCII
byteArray[5] = 10 //'\n' ASCII
byteArray[6] = 48 //'0' ASCII
byteArray[7] = 49 //'1' ASCII
byteArray[8] = 50 //'2' ASCII
byteArray[9] = 0

[1] توجه داشته باشید که در مثال های ارائه شده، در پایان همه رشته ها کاراکتر NewLine نیز ارسال می شود.

توجه داشته باشید که، اگر داده در زمان انتظار دریافت نشود، متد، هر تعداد کاراکتر که دریافت کرده باشد برگشت می دهد. پس برای اعضایی که داده جدید دریافت نشده، همان داده پیشین موجود است و نمایش داده می شود.

Read(Char[], Int32, Int32)

این متد دقیقا مانند متد قبل است که توضیح داده شد، با این تفاوت که داده دریافت شده را در آرایه ای از جنس Char قرار می دهد.

```
char[] charArray = new char[10];

private void DisplayText(object sender, EventArgs e)
{
  int i = 0;
  for (i = 0; i < 10; i++) textBox2.AppendText(charArray[i].ToString());

}


private void serialPort1_DataReceived(object sender,
System.IO.Ports.SerialDataReceivedEventArgs e)
{
  serialPort1.Read(charArray,2, 7);
  this.Invoke(new EventHandler(DisplayText));

}
```

اگر رشته ("0123456789") به صورت متوالی برای الگوی بالا ارسال شود، به صورت زیر دریافت می شود.

charArray[0] = '\0'
charArray [1] = '\0'
charArray[2] = '0' // ASCII = 48
charArray [3] = '1' // ASCII = 49
charArray [4] = '2' // ASCII = 50
charArray [5] = '3' // ASCII = 51
charArray [6] = '4' // ASCII = 52
charArray [7] = '5' // ASCII = 53
charArray [8] = '6' // ASCII = 54
charArray [9] = '\0'
****
charArray[0] = '\0'

charArray [1] = '\0'
charArray[2] = '7' // ASCII = 55
charArray [3] = '8' // ASCII = 56
charArray [4] = '9' // ASCII = 57
charArray [5] = '\n' // ASCII = 10
charArray [6] = '0' // ASCII = 48
charArray [7] = '1' // ASCII = 49
charArray [8] = '2' // ASCII = 50
charArray [9] = '\0'

توجه داشته باشید که، اگر داده در زمان انتظار دریافت نشود، متد، هر تعداد کاراکتر که دریافت کرده باشد برگشت می دهد. پس برای اعضایی که داده جدید دریافت نشده، همان داده پیشین موجود است و نمایش داده می شود.

ReadByte()

یک کاراکتر را دریافت می کند. و کد اسکی کاراکتر دریافت شده بازگشت داده می شود. الگوی زیر اسکی دریافت شده را نمایش میدهد. متغیری که مقدار بازگشتی متد را در خود نگه میدارد، حتما باید از جنس int باشد. نوع char و یا byte قابل قبول نیست.

```
int   intRecieve;
private void DisplayText(object sender, EventArgs e)
 {
  textBox2.AppendText(intRecieve.ToString());
  textBox2.AppendText(" ");
  if(intRecieve == 10)
//if(intRecieve == '\n')
  textBox2.AppendText("\n");

 }

private void serialPort1_DataReceived(object sender, SerialDataReceivedEventArgs e)
{
  intRecieve = serialPort1.ReadByte();
  this.Invoke(new EventHandler(DisplayText));
}
```

اگر رشته ("0123456789") به صورت متوالی برای الگوی بالا ارسال شود، به صورت زیر دریافت می شود

48 49 50 51 52 53 54 55 56 57 (10)
48 49 50 51 52 53 54 55 56 57 (10)

ReadChar()

یک کاراکتر را دریافت میکند. دریافت کاراکتر به صورت کد اسکی دریافت است. الگوی زیر اسکی دریافت شده را نمایش می دهد.
حتما باید متغیر از جنس int باشد. نوع char و یا byte قابل قبول نیست.

```
int   intRecieve;
private void DisplayText(object sender, EventArgs e)
 {
  textBox2.AppendText(intRecieve.ToString());
  textBox2.AppendText(" ");

  if(intRecieve == 10)
  // if(intRecieve == '\n')

  textBox2.AppendText("\n");

 }

private void serialPort1_DataReceived(object sender, SerialDataReceivedEventArgs e)
{
  intRecieve = serialPort1.ReadChar();
  this.Invoke(new EventHandler(DisplayText));
}
```

اگر رشته ("0123456789") به صورت متوالی برای الگوی بالا ارسال شود، به صورت زیر دریافت می شود

48 49 50 51 52 53 54 55 56 57 (10)
48 49 50 51 52 53 54 55 56 57 (10)

ReadExisting()

در هر بار فراخوانی کلیه داده های موجود در بافر را برمی گرداند.

```
string   strRecieve;
private void DisplayText(object sender, EventArgs e)
{
  textBox2.AppendText(strRecieve);
}

private void serialPort1_DataReceived(object sender, SerialDataReceivedEventArgs e)
{
  strRecieve = serialPort1.ReadExisting();
  this.Invoke(new EventHandler(DisplayText));
}
```

اگر رشته ("0123456789") به صورت متوالی برای الگوی بالا ارسال شود، به صورت زیر دریافت می شود

0123456789('\n')
0123456789('\n')

ReadLine()

در هر بار فراخوانی تا زمانی که با کاراکتر NewLine برسد داده دریافت میکند و هنگامی که به این کاراکتر رسید، داده را بازگشت میدهد. البته ممکن است پایان زمان انتظار، منجر به بازگشت داده شود.

```
string   strRecieve;
private void DisplayText(object sender, EventArgs e)
 {
  textBox2.AppendText(strRecieve);

 }

private void serialPort1_DataReceived(object sender, SerialDataReceivedEventArgs e)
{
  strRecieve = serialPort1.ReadLine();
  this.Invoke(new EventHandler(DisplayText));
}
```

اگر رشته ("0123456789") به صورت متوالی برای الگوی بالا ارسال شود، به صورت زیر دریافت می شود. کاراکتر NewLine در بازگشتی آن وجود ندارد.

0123456789012345678 9

ReadTo

تا زمانی که به رشته مشخص شده که پارامتر ورودی آن است برسد، به خواندن رشته ادامه میدهد.

```
string  strRecieve;
private void DisplayText(object sender, EventArgs e)
{
 textBox2.AppendText(strRecieve);
 textBox2.AppendText("--\n");
}

private void serialPort1_DataReceived(object sender, SerialDataReceivedEventArgs e)
{
 intRecieve = serialPort1.ReadTo("345");
 this.Invoke(new EventHandler(DisplayText));
}
```

اگر رشته ("SrzamineMan123456789") به صورت متوالی برای الگوی بالا ارسال شود، به صورت زیر دریافت میشود. پارامتر ورودی و NewLine در بازگشتی آن وجود ندارد

SrzamineMan12--
6789SrzamineMan12--
6789SrzamineMan12--
6789SrzamineMan12--

ToString

textBox2.AppendText (intRecieve.ToString());

textBox2.Text+= intRecieve.ToString();

متد های ارسال داده

Write(String)

یک رشته را ارسال میکند.

```
serialPort1.Write(textBox1.Text);
serialPort1.Write("SrzamineMan");
```

اگر به صورت متوالی از متد بالا استفاده شود. آن چه در سمت گیرنده دریافت می شود به صورت زیر است.

SrzamineManSrzamineMan

Write(Byte[], Int32, Int32)

ارسال یک تعدادی از اعضای یک آرایه از جنس Byte. پارامتر اول نام آرایه را مشخص میکند. پارامتر دوم شماره عضو آرایه که ارسال از آن عضو آغاز می شود. پارامتر سوم تعداد اعضایی که باید ارسال شوند را مشخص میکند. ارسال از عضو مشخص شده در پارامتر دوم به بعد شروع می شود.

```
private void btnSendData_Click(object sender, EventArgs e)
{
  byte[] byteArray = new byte[10];
```

```
  byteArray[0] = 48; //’0’ ASCII
  byteArray[1] = 49; //’1’ ASCII
  byteArray[2] = 50; //’2’ ASCII
  byteArray[3] = 51; //’3’ ASCII
  byteArray[4] = 52; //’4’ ASCII
  byteArray[5] = 53; //’5’ ASCII
  byteArray[6] = 54; //’6’ ASCII
  byteArray[7] = 55; //’7’ ASCII
  byteArray[8] = 56; //’8’ ASCII
  byteArray[9] = 57; //’9’ ASCII

  serialPort1.Write(byteArray, 2, 5);
}
```

متد از عضو دوم شروع به ارسال میکند و ۵ عضو را ارسال می کند.

50 (‘2’)
51 (‘3’)
52 (‘4’)
53 (‘5’)
54 (‘6’)

Write(Char[], Int32, Int32)

ارسال یک تعدادی از اعضای یک آرایه از جنس Char. پارامتر اول نام آرایه را مشخص می کند. پارامتر دوم شماره عضو آرایه که ارسال از آن عضو آغاز می شود. پارامتر سوم تعداد اعضایی که باید ارسال شوند را مشخص می کند. ارسال از عضو مشخص شده در پارامتر دوم به بعد شروع می شود.

```
private void btnSendData_Click(object sender, EventArgs e)
{
  char[] charArray = new char[10];

  charArray [0] = ’0’; //  ASCII = 48
  charArray [1] = ’1’; //  ASCII = 49
  charArray [2] = ’2’; //  ASCII = 50
  charArray [3] = ’3’; //  ASCII = 51
  charArray [4] = ’4’; //  ASCII = 52
  charArray [5] = ’5’; //  ASCII = 53
  charArray [6] = ’6’; //  ASCII = 54
  charArray [7] = ’7’; //  ASCII = 55
  charArray [8] = ’8’; //  ASCII = 56
  charArray [9] = ’9’; //  ASCII = 57
```

```
  serialPort1.Write(byteArray, 2, 5);
}
```

صورت متوالی برای الگوی بالا ارسال شود، به صورت زیر دریافت می شود. از عضو دوم شروع به ارسال میکند و ۵ عضو ارسال میکند.

'2' (ASCII = 50)
'3' (ASCII = 50)
'4' (ASCII = 50)
'5' (ASCII = 50)
'6' (ASCII = 50)

دریافت از دو متد Write(Byte[], Int32, Int32) و Write(Char[], Int32, Int32) کاملا یکسان است. تنها مقدار دهی این دو متد با هم متفاوت است. تابع Write(Byte[], Int32, Int32) متغیر از نوع بایت را ارسال میکند و مقدار دهی متغیر byte به صورت عددی است، یعنی برای هر کاراکتر باید کد اسکی آن را وارد کنیم. اما تابع Write(Char[], Int32, Int32) آرایه ای از جنس char را ارسال میکند و مقدار دهی متغیر char به صورت کاراکتری است.

WriteLine(String)

یک رشته را ارسال میکند. و در پایان ارسال، کاراکتر NewLine را ارسال می کند.

```
serialPort1.WriteLine(textBox1.Text);
serialPort1.WriteLine("SrzamineMan");
```

آن چه دریافت می شود به صورت زیر است.

SrzamineMan
SrzamineMan

ویژگی های پورت

در این قسمت ویژگی های پورت سریال را بررسی می کنیم. الگوی تنظیم یا خواندن این وبژگی ها در قالب کد بیان شده است. برای بررسی این ویژگی ها، پروژه `Serial_Setting` را بررسی کنید.

توجه داشته باشید:

برخی از این ویژگی ها فقط خواندنی است، یعنی شما نمی توانید آنها را تغییر دهید.

برای خواندن یا تنظیم برخی از ویژگی ها باز بودن پورت، و برای برخی، بسته بودن پورت لازم است. و در حالتی غیر از حالت مورد نظر، شما نمی توانید ویژگی مورد نظر را بخوانید یا تنظیم کنید.

منظور از بررسی وضعیت، خواندن ویژگی، یا دریافت، مقدار یا حالتی است که قبلا، یا به صورت پیش فرض تنظیم شده است.

BaseStream

این ویژگی فقط خواندنی است و تنها باید زمانی خوانده شود که پورت باز است.

```
textBox3.AppendText(serialPort1.BaseStream.ToString());//opened for read
// Read Only
```

BaudRate

باودریت ارتباط را به صورت عدد صحیح، در خود نگه میدارد. خواندن یا مقداردهی آن در هر حالت[1] ممکن است.

```
textBox3.AppendText(serialPort1.BaudRate.ToString());
serialPort1.BaudRate = 19200;
```

BreakState

برای خواندن و یا تنظیم این ویژگی باید، پورت باز باشد.

```
textBox3.AppendText(serialPort1.BreakState.ToString());//opened for read
serialPort1.BreakState =true; // Need Port Open
```

BytesToRead

تعداد بایت های موجود دربافر ورودی را در خود نگه می دارد. این ویژگی فقط خواندنی است و فقط هنگام باز بودن پورت باید خوانده شود.

```
textBox3.AppendText(serialPort1.BytesToRead.ToString());//opened for read
```

---

[1] منظور از حالت، قطع یا وصل بودن ارتباط در زمان نوشتن و خواندن آن است.

BytesToWrite

تعداد بایت های موجود دربافر خروجی را در خود نگه می دارد. این ویژگی فقط خواندنی است فقط هنگام باز بودن پورت باید خوانده شود.

```
textBox3.AppendText(serialPort1.BytesToWrite.ToString());//opened for read
```

CanRaiseEvents

امکان اتفاق افتادن رویداد ها را به صورت بولین در خود نگه می دارد.

```
protected virtual bool CanRaiseEvents { get; }

true if the component can raise events; otherwise, false. The default is true.
```

CDHolding

این ویژگی فقط خواندنی است و تنها باید زمانی خوانده شود که پورت باز است.

```
textBox3.AppendText(serialPort1.CDHolding.ToString()); //opened for read
//Read Only
true if the carrier is detected; otherwise, false.
```

Container

```
//Read Only
```

CtsHolding

اگر خط Clear To Send مودم، شناسایی شد، مقدار True را برمیگرداند. این ویژگی فقط خواندنی است و تنها باید زمانی خوانده شود که پورت باز است.

```
textBox3.AppendText(serialPort1.CtsHolding.ToString()); //opened for read
//Read Only
```

DataBits

این ویژگی تعداد بیت داده ارتباط را به صورت عدد صحیح در خود نگه میدارد. ودر هر حالت قابل نوشتن و خواندن است.تعداد بیت داده ارتباط عددی بین ۵ تا ۹ است.

```
textBox3.AppendText(serialPort1.DataBits.ToString());
serialPort1.DataBits = 8; // 5~8 bit
```

DesignMode

ویژگی DesignMode را در خود نگه میدارد. در صورت وجود ، مقدار True را برمیگرداند.

DiscardNull

تنظیم یا بررسی وضعیت صرف نظر کردن از دریافت بایت های خالی.

```
textBox3.AppendText(+ serialPort1.DiscardNull.ToString());
serialPort1.DiscardNull =false;
```

DsrHolding

دریافت وضعیت پین Data Set Ready. این ویژگی فقط خواندنی است و تنها باید زمانی خوانده شود که پورت باز است.

```
textBox3.AppendText(serialPort1.DsrHolding.ToString()); //opened for read
//Read Only
```

DtrEnable

فعال کردن پین Data Terminal Ready. یا خواندن وضعیت این پین.

```
textBox3.AppendText(serialPort1.DtrEnable.ToString());
serialPort1.DtrEnable = false;
```

Encoding

انکودر داده دریافتی. انکودر پیشفرض بر اساس ASCIIEncoding می باشد

```
textBox3.AppendText(serialPort1.Encoding.ToString());
serialPort1.Encoding = Encoding.ASCII;
```

Events

دریافت لیست رویداد هایی که میتوان استفاده نمود.

Handshake

تنظیم یا بررسی وضعیت پروتکل Hand Shake در انتقال داده.

```
textBox3.AppendText(serialPort1.Handshake.ToString());
serialPort1.Handshake =  Handshake.None;
```

IsOpen

باز یا بسته بودن پورت را به صورت بولی نگه می دارد. و در هر حالت فقط خواندنی است.

```
textBox3.AppendText(serialPort1.IsOpen.ToString());

if (serialPort1.IsOpen == true)
{
serialPort1.WriteLine("<0>");
}
```

NewLine

تنظیم کاراکتر پایانی، که در توابع ReadLine() و WriteLine() کاربرد دارد.

```
textBox3.AppendText(serialPort1.NewLine.ToString());
serialPort1.NewLine = "\n";
```

Parity

تعداد بیت پریتی ارتباط سریال را نگه می دارد. مقدار آن در هر حالت قابل خواندن و نوشتن است. مقادیر قابل قبول برای آن: Even, Mark, None, Odd, Spase می باشد.

```
textBox3.AppendText(serialPort1.Parity.ToString());
serialPort1.Parity =  Parity.None;
```

ParityReplace

هنگامی که در خوانندن داده، با ایراد پریتی مواجه شد. داده دریافت شده را با این پریتی می خواند.

```
textBox3.AppendText(serialPort1.ParityReplace.ToString());
serialPort1.ParityReplace = 2;
```

PortName

نام پورت را درخود نگه می دارد. و فقط در زمان قطع ارتباط باید در آن نوشته شود. یعنی پورت عوض شود.

```
textBox3.AppendText(serialPort1.PortName.ToString());
serialPort1.PortName="COM1";
```

ReadBufferSize

میزان بافر ورودی را در خود نگه میدارد. فقط در زمان قطع ارتباط در آن نوشته شود.

```
textBox3.AppendText(serialPort1.ReadBufferSize.ToString());
serialPort1.ReadBufferSize = 4048;
```

ReadTimeout

مقدار زمان انتظار را برای دریافت داده، بر حسب میلی ثانیه در خود نگه می دارد.

```
textBox3.AppendText(serialPort1.ReadTimeout.ToString());
serialPort1.ReadTimeout = 1000;
```

ReceivedBytesThreshold

چه تعداد بیت دریافت شود تا منجر به رویداد DataReceived شود. این ویژگی تعداد بیت را در خود نگه میدارد. و در هر حالت قابل نوشتن و خواندن است. به صورت پیش فرض رسیدن ۱ بایت داده منجر به اجرای رویداد DataRecieved می شود.

```
textBox3.AppendText(serialPort1.ReceivedBytesThreshold.ToString());
serialPort1.ReceivedBytesThreshold = 1;
```

RtsEnable

فعال کردن پین Request To Send یا آماده ارسال[1].

```
textBox3.AppendText(serialPort1.RtsEnable.ToString());
serialPort1.RtsEnable = false;
```

Site

فعال کردن ویژگی Site یا بررسی وضعیت.

StopBits

تعداد بیت پایان ارتباط را درخود نگه می دارد. مقادیر مجاز برای آن: None, One, OnePointFive, Two

```
textBox3.AppendText(serialPort1.StopBits.ToString());
serialPort1.StopBits = StopBits.One;
```

WriteBufferSize

میزان بافر خروجی را در خود نگه میدارد. فقط در زمان قطع ارتباط باید در آن نوشته شود.

```
textBox3.AppendText(serialPort1.WriteBufferSize.ToString());
```

---

[1] ازطریق این پین سیستم آماده بودن خود را به مودم اطلاع میدهد.

```
serialPort1.WriteBufferSize = 20048;
```

WriteTimeout

مقدار زمان انتظار را برای ارسال داده، بر حسب میلی ثانیه در خود نگه می دارد.

```
textBox3.AppendText(serialPort1.WriteTimeout.ToString());
serialPort1.WriteTimeout = 1000;
```

InfiniteTimeout

رخ دادن پایان زمان انتظار را برای دریافت یا ارسال را نشان می دهد.

```
textBox3.AppendText(SerialPort.InfiniteTimeout.ToString());
```

بخش پنجم: پیوست

پیوست ۱

ASCII	Hex	Symbol
32	20	(space)
33	21	!
34	22	"
35	23	#
36	24	$
37	25	%
38	26	&
39	27	'
40	28	(
41	29	)
42	2A	*
43	2B	+
44	2C	,
45	2D	-
46	2E	.
47	2F	/

ASCII	Hex	Symbol
64	40	@
65	41	A
66	42	B
67	43	C
68	44	D
69	45	E
70	46	F
71	47	G
72	48	H
73	49	I
74	4A	J
75	4B	K
76	4C	L
77	4D	M
78	4E	N
79	4F	O

ASCII	Hex	Symbol
96	60	`
97	61	a
98	62	b
99	63	c
100	64	d
101	65	e
102	66	f
103	67	g
104	68	h
105	69	i
106	6A	j
107	6B	k
108	6C	l
109	6D	m
110	6E	n
111	6F	o

ASCII	Hex	Symbol
48	30	0
49	31	1
50	32	2
51	33	3
52	34	4
53	35	5
54	36	6
55	37	7
56	38	8
57	39	9
58	3A	:
59	3B	;
60	3C	<
61	3D	=
62	3E	>
63	3F	?

ASCII	Hex	Symbol
80	50	P
81	51	Q
82	52	R
83	53	S
84	54	T
85	55	U
86	56	V
87	57	W
88	58	X
89	59	Y
90	5A	Z
91	5B	[
92	5C	\
93	5D	]
94	5E	^
95	5F	_

ASCII	Hex	Symbol
112	70	p
113	71	q
114	72	r
115	73	s
116	74	t
117	75	u
118	76	v
119	77	w
120	78	x
121	79	y
122	7A	z
123	7B	{
124	7C	|
125	7D	}
126	7E	~
127	7F	

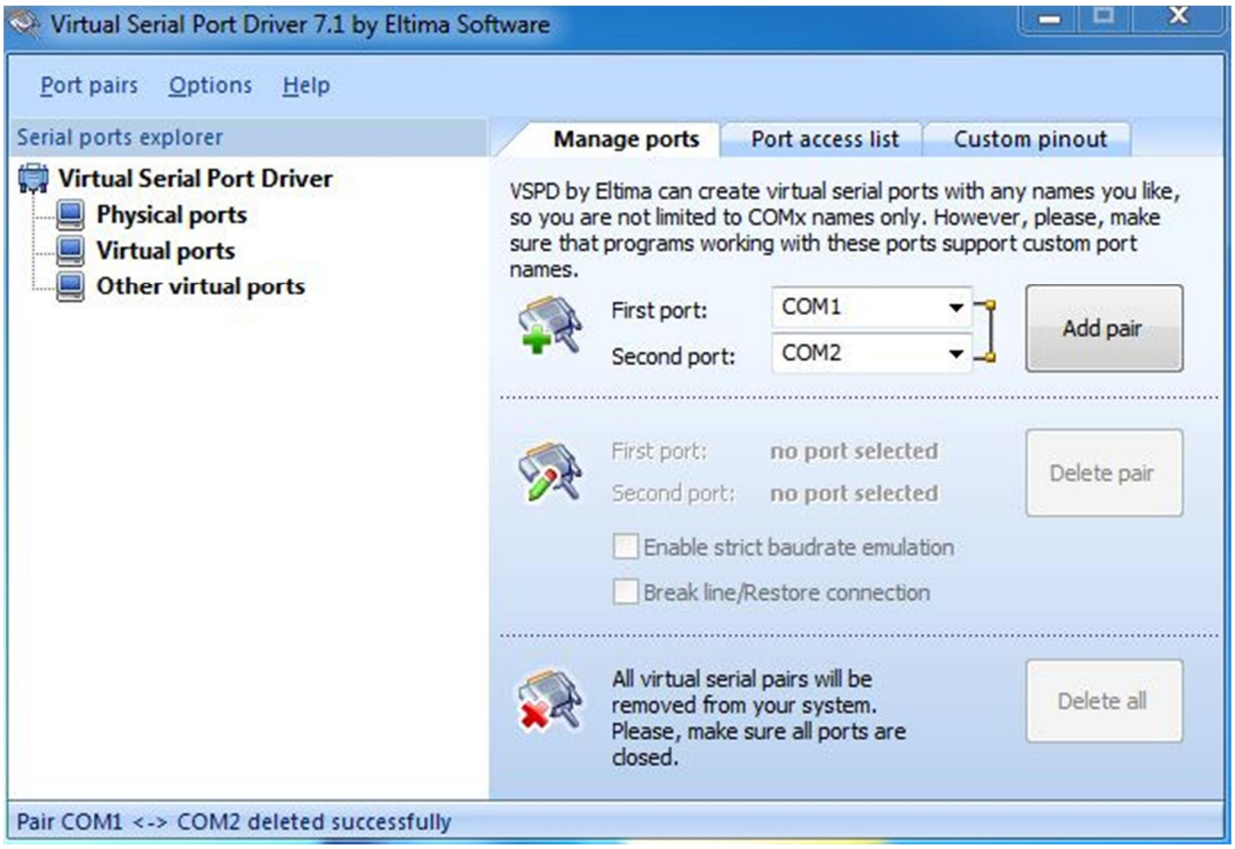
پیوست ۲

آموزش برنامه Virtual Serial Port Driver[1]

با استفاده از این برنامه می تونید یک پیوند بین دو پورت سریال مجازی ایجاد کنید. این نرم افزار برای آزمایش برنامه های نوشته شده برای پورت سریال سودمند خواهد بود.

به وسیله این برنامه، میکروکنترلر درون شبیه ساز پروتیوس را به برنامه کنترلی ساخته شده وصل میکنیم و تبادل داده بین میکروکنترلر و برنامه را به صورت کاملا مجازی آزمایش میکنیم. یعنی نه نیازی به میکروکنترلر است و نه نیازی به پورت سریال. میکروکنترل درون شبیه ساز پروتیوس را به پورت سریال ایجاد شده توسط برنامه Virtual Serial Port Driver متصل میکنیم.

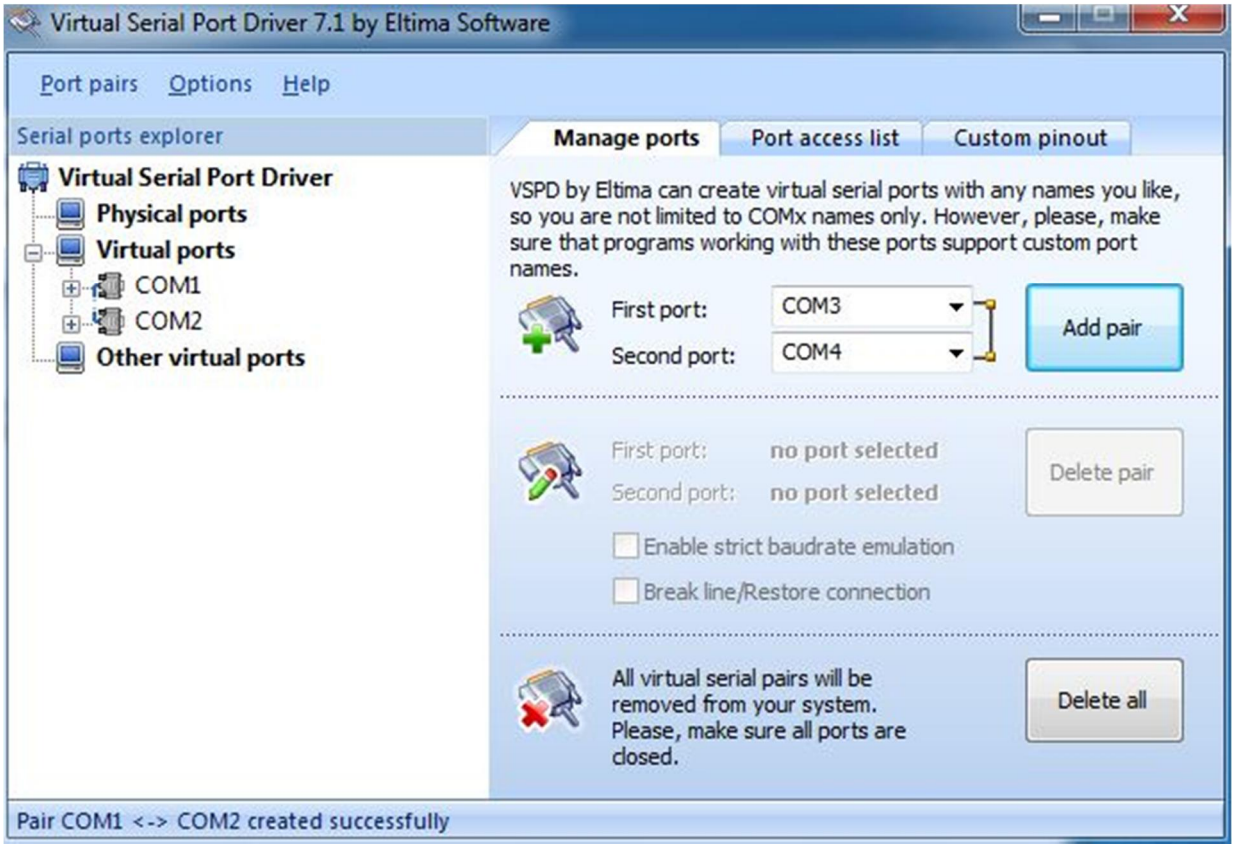
نرم افزار Virtual Serial Port Driver را باز کنید تا با صفحه زیر روبرو شوید. همانگونه که در تصویر می بینید برنامه آماده است تا دو پورت سریال مجازی به نام های COM1 و COM2 بسازد و آن دو را با هم جفت کند. نخست نام های دلخواه خود را برای پورت هایتان انتخاب کنید، سپس روی دکمه Add pair کلیک کنید تا جفت مورد نظر شما ساخته شود. وقتی جفت ساخته شود، اگر هر یک از پورت ها را در برنامه های جداگانه ای باز کنیم مانند این است پایه TX و RX این دو پورت به هم وصل باشند. اگر داده از یکی از پورت ها ارسال شود در سمت دیگر، همان داده دریافت می شود و وارون آن هم درست است.



تصویر _ ٤٤ محیط نرم افزار Virtual Serial Port Driver

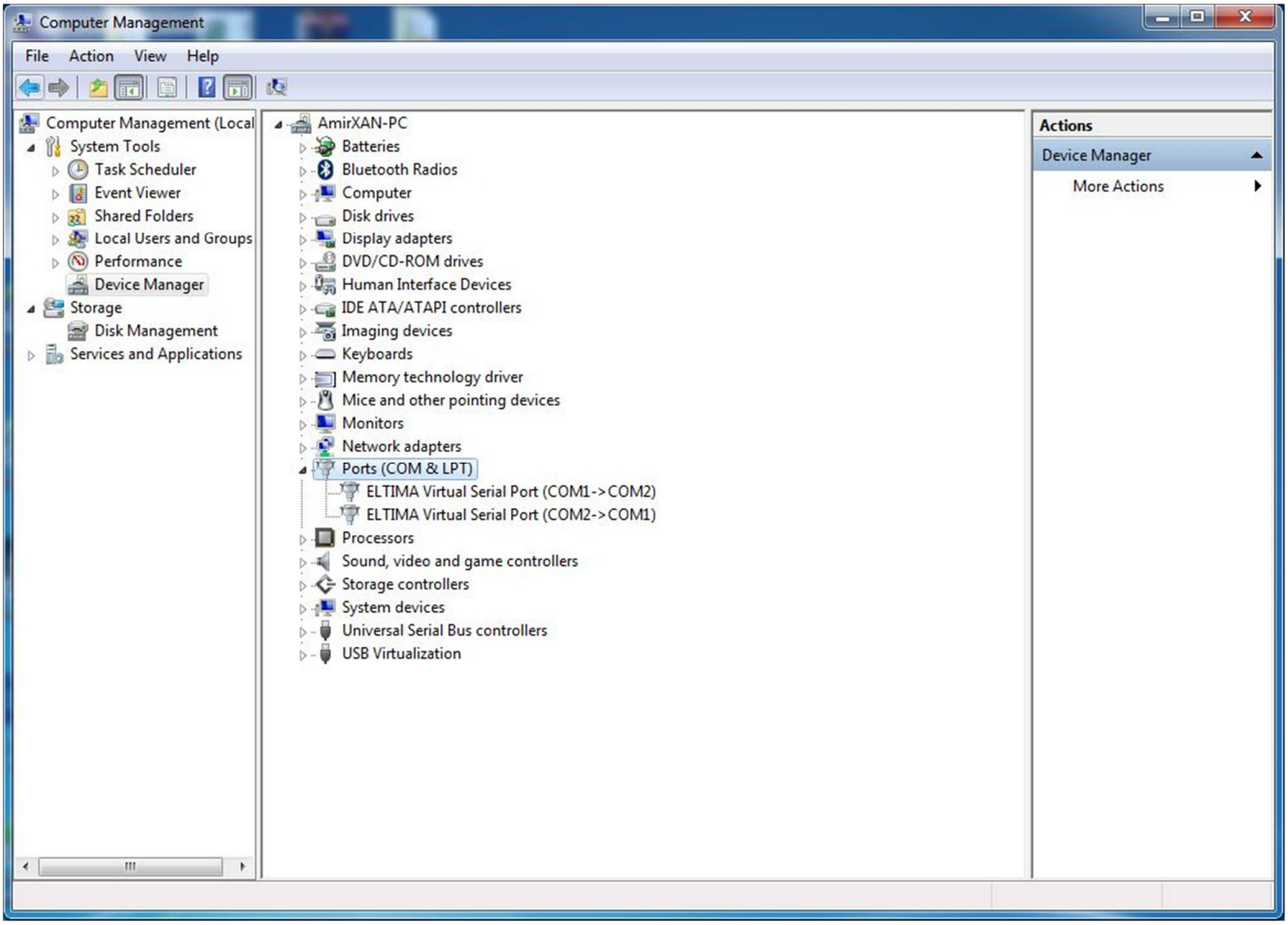
[1] نسخه ١٤ روزه این نرم افزار را می توانید از آدرس http://www.eltima.com/ دانلود کنید.

پس از ساخته شدن جفت مورد نظر، جفت شدن آنها را در سمت چپ صفحه برنامه قابل مشاهده است.



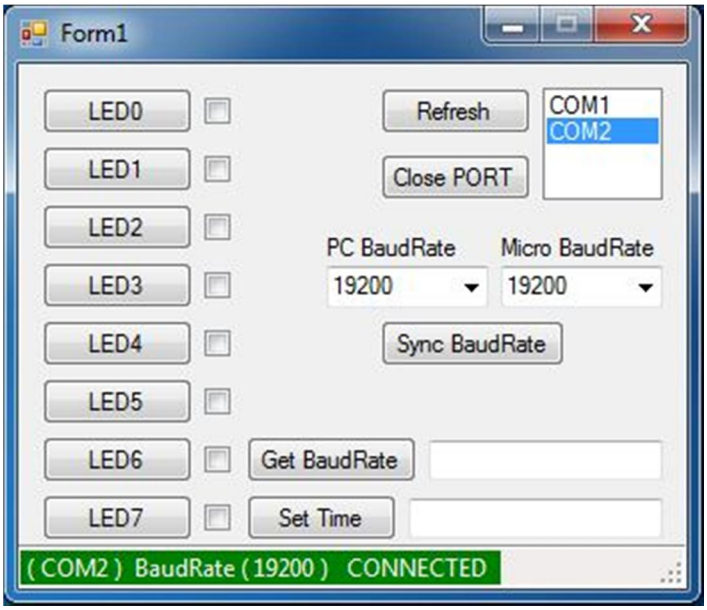
تصویر _ ٤٥ جفت شدن پورت مجازی **COM1** و **COM2**

پس از ساختن جفت مورد نظر مشاهده میکنید که در Device Manager ویندوز دو پورت سریال اضافه شده است.



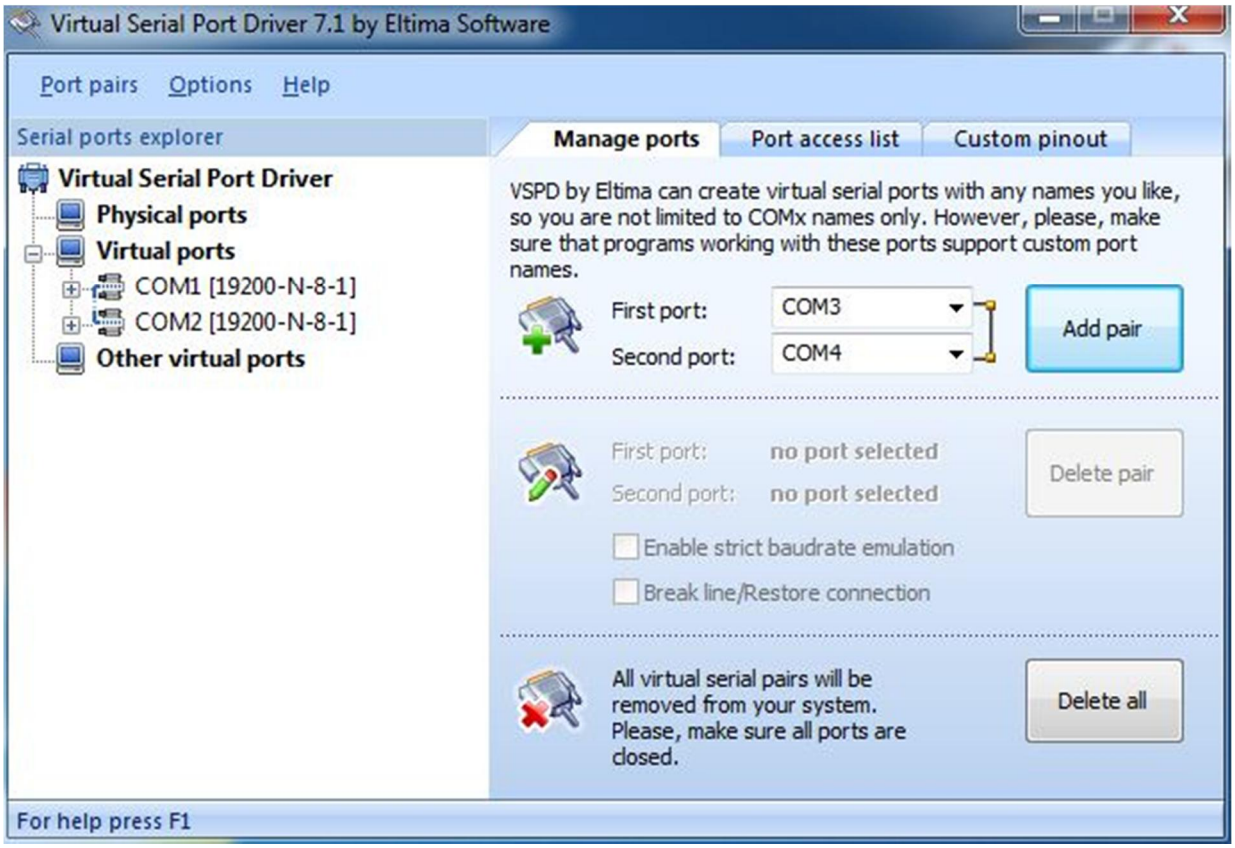
تصویر _ ٤٦ اضافه شدن جفت ایجاد شده به پنجره Device Manager

اگر شما یکی از برنامه هایی که برای پورت سریال نوشته اید را باز کنید متوجه خواهید شد که برنامه شما توانایی باز کردن هر یک از پورت های جفت شده را دارد. پس از ایجاد زوج مورد نظر برنامه کنترل، که ساخته اید را اجرا کنید و یکی از پورت های جفت شده را در آن باز کنید. برنامه شما آماده است تا با پورت مورد نظر تبادل داده انجام دهد. در برنامه کنترل، پورت COM2 را باز کنید.

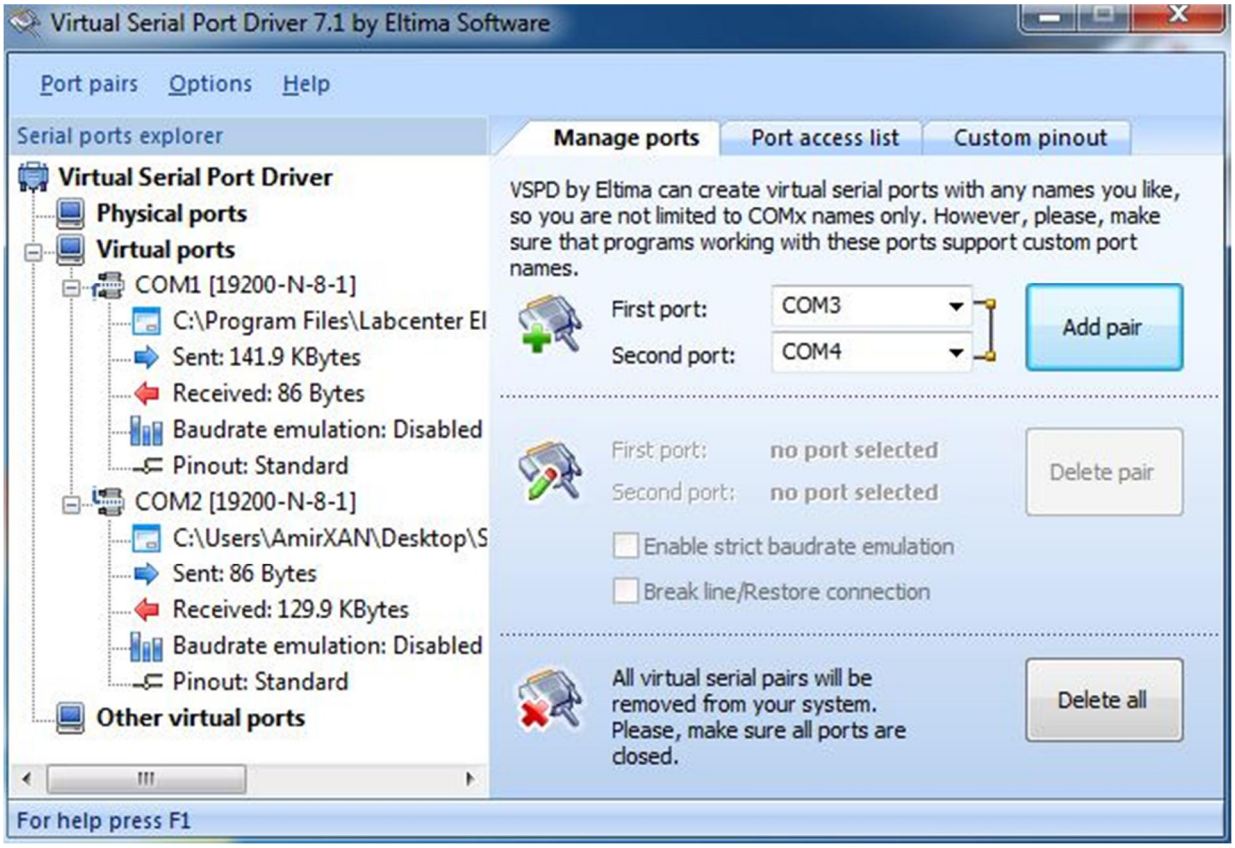


تصویر _ ٤٧ باز کردن پورت سریال در برنامه کنترل

اگر در برنامه هایتان پورت های مجازی ایجاد شده را باز کنید نوع ارتباط در سمت چپ صفحه برنامه نمایش داده می شود.

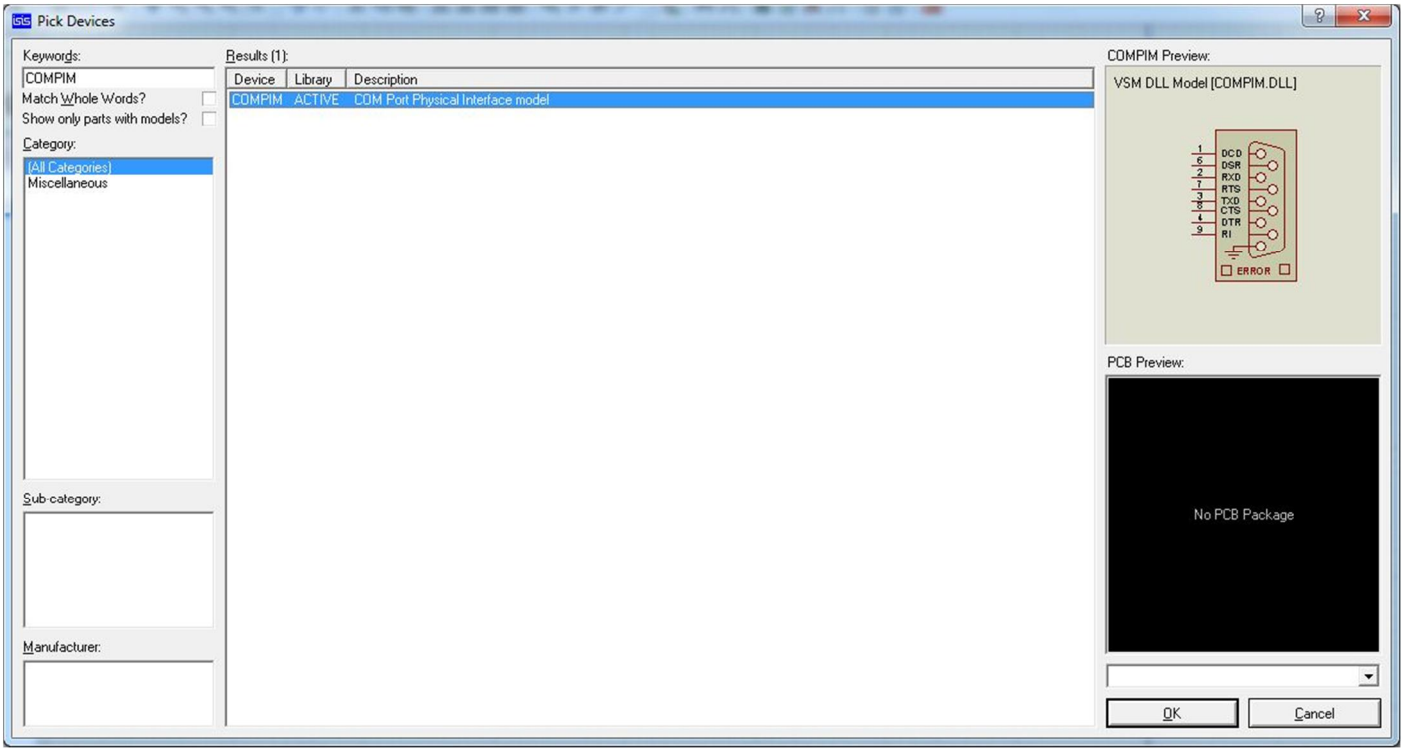


تصویر _ ٤٨ [باودریت ۱۹۰۰ بدون پریتی ۸ بیت داده ۱بیت پایان]



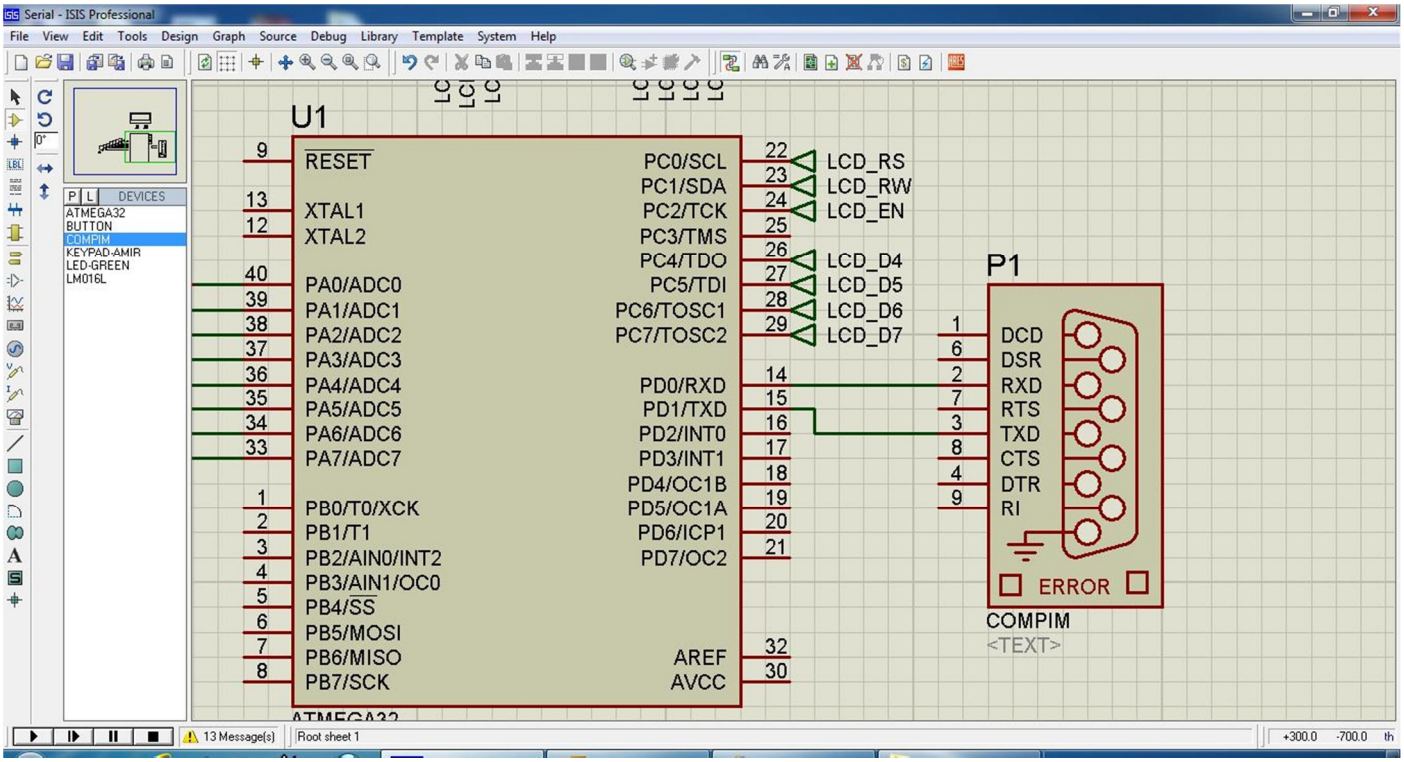
تصویر _ ۴۹ میزان تبادل داده بین جفت ایجاد شده

برای تبادل داده بین میکروکنترلر و برنامه، باید میکروکنترلر را به پورت سریال جفت شده COM1 وصل کنیم. برای این پیوند از شبیه ساز پروتیوس استفاده میکنیم. کافی است که فایل پروتیوس همراه را باز کنید و میکروکنترلر را به پورت سریال جفت شده اتصال دهید. برای این کار در کتابخانه پروتیوس ابزار COMPIM را جستجو کنید و آن را به برنامه اضافه کنید. این وسیله میتواند میکروکنترلر درون شبیه ساز را مستقیما به یکی از پورت های سریال موجود متصل کند.



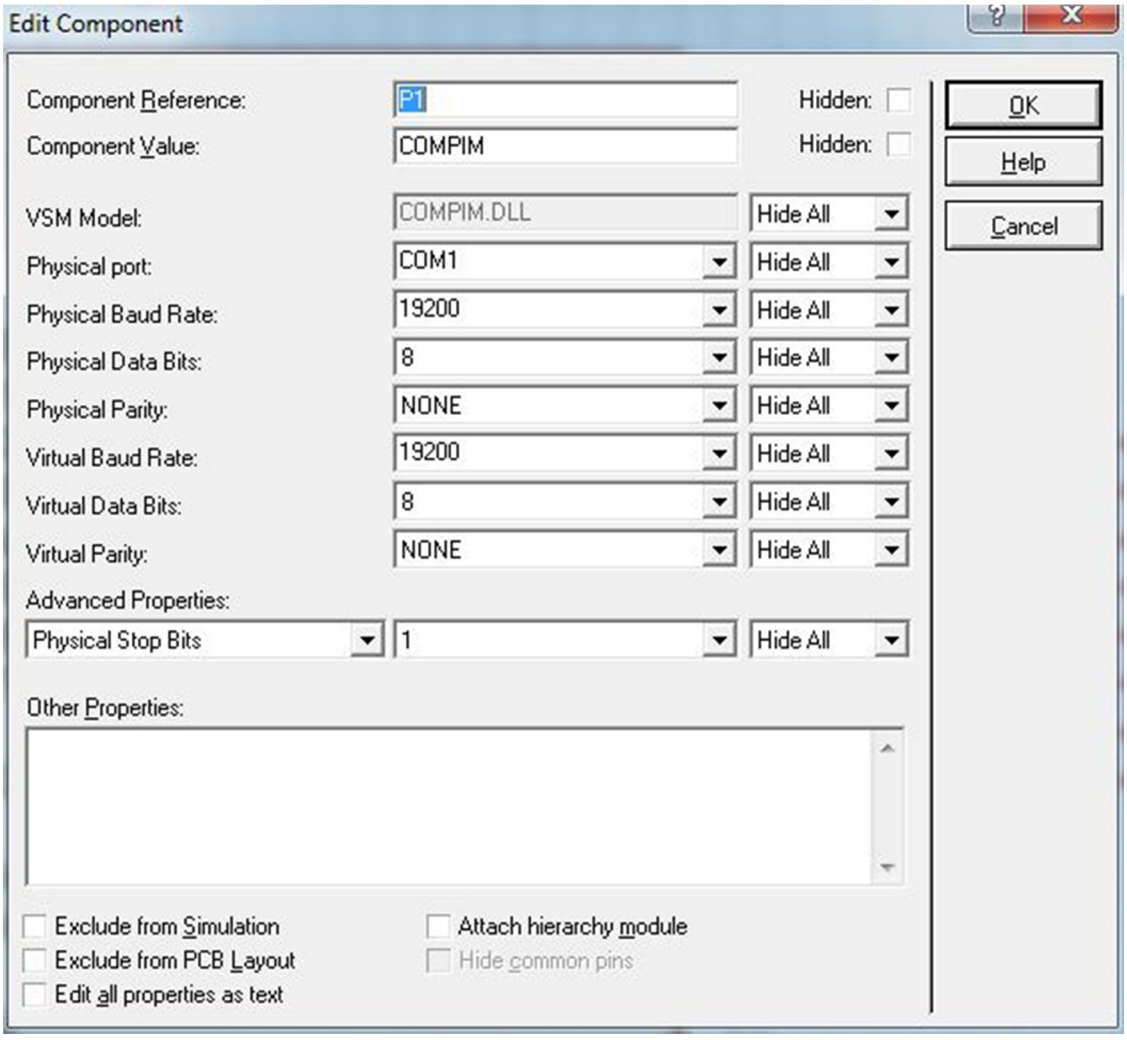
تصویر _ ۵۰ اضافه کردن COMPIM

سپس اتصالات پایه ها لازم را در را در پروتیوس برقرار کنید.



تصویر _ ۵۱ اتصال پایه ها

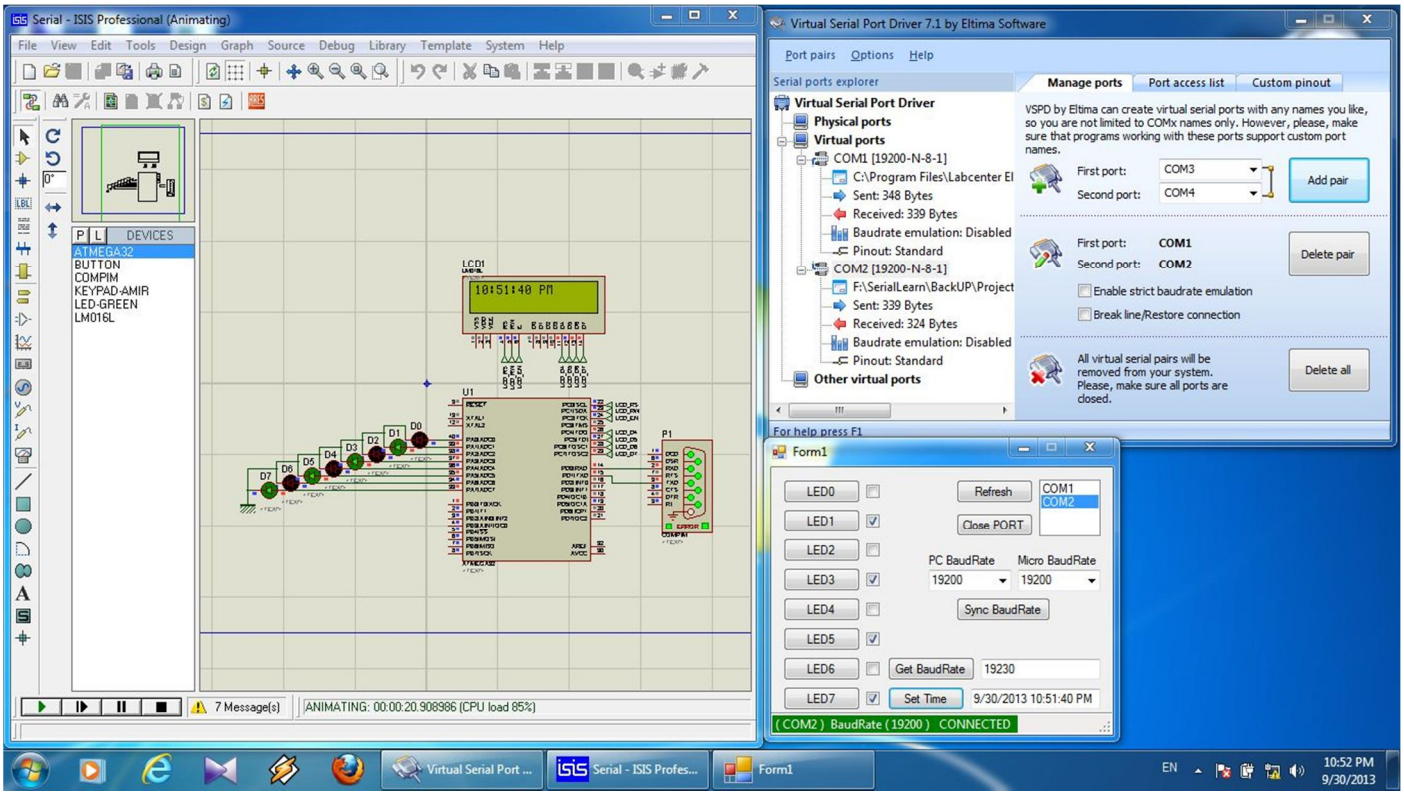
سپس روی COMPIM دابل کلیک کنید و در پنجره Properties تنظیمات زیر را انجام دهید. توجه داشته باشد که میکروکنترلر موجود در محیط شبیه ساز را با استفاده از COM1 به برنامه کنترل متصل می کنیم.



تصویر _ ۵۲ تنظیمات پورت

پروتیوس را RUN کنید و برنامه را اجرا کنید. می بینید که میکروی شبیه ساز به دستورات ارسال شده از سمت برنامه کنترل ویندوز پاسخ میدهد.

با این برنامه و ابزار موجود توانستیم برنامه ساخته شده را به شبیه ساز پروتیوس پیوند دهیم و دستورات را بین میکرو کنترلر و برنامه کنترلی در فضای کاملا مجازی آزمایش کنیم.



تصویر _ ۵۳ اتصال میکرو و برنامه در فضای کاملا مجازی

منابع

آموزش ویژوال C# 2005 نوشته محمد هاشمیان

msdn.microsoft.com/

بررسی پورت سریال در ویژوآل C# نوشته مهدی زرکوب

آموزش استفاده از پورت سریال در دات نت نوشته سید محمد حسینی

.

بزودی آموزش پورت USB . . .

خودرو بزرگ پشت سر هم همگانی

تقدیم به همه جوانان خوب سرزمین من