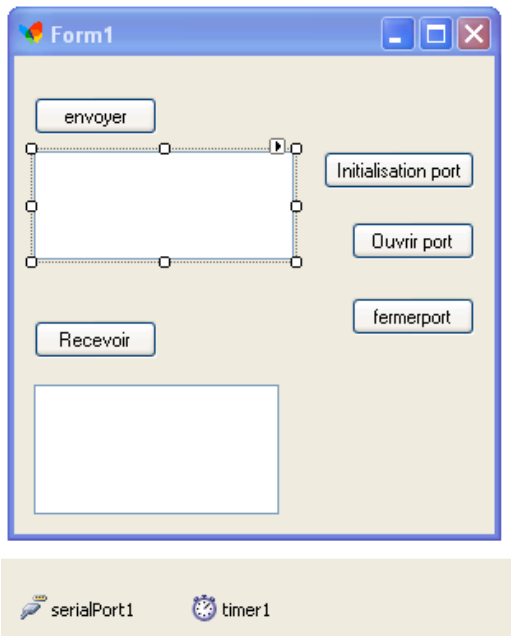
TP Gestion du port série en C# avec un Thread.

I) Création du projet

- Créer un nouveau projet C#, nommé PortSerieAvecTimer.



II) Interface graphique suivante:

- Réaliser l'interface graphique suivante:
- Ne pas oublier de renommer les différents contrôles

III) Contrôles

- Placer le contrôle serialPort.
- Placer le contrôle Timer.

IV) Code

ATTENTION LE CODE DONNEE CI-DESSOUS EST UN EXEMPLE, IL FAUT L'ADAPTER A VOS NOM DE VARIABLES

- **Compléter la zone de déclaration des classes avec** using System.Threading.
- Compléter la classe de formulaire avec le code donné ci-dessous (attributs et constucteur):

```
public partial class Form1 : Form
{

    string MaChaineAEmettre;
    string MaChaineCompleteRecue;
    Thread myThread;


    public Form1()
    {
        InitializeComponent();
        timer1.Enabled = false;
        MaChaineCompleteRecue = "";
        myThread = new Thread(new ThreadStart(ThreadLoop));

    }
```

- Ecrire le code de la fonction ThreadLoop()

```
public void ThreadLoop()
{
    // Tant que le thread n'est pas tué, on travaille
    while (Thread.CurrentThread.IsAlive)
    {
        string MaChaineRecue = "";
        MaChaineRecue = serialPort1.ReadExisting();
        MaChaineCompleteRecue = MaChaineCompleteRecue + MaChaineRecue;

    }
}
```

- Créer le gestionnaire clic du bouton "Initialisation port", et compléter le code:

ATTENTION: dans notre cas utiliser "COM1"

```
private void BInit_Click(object sender, EventArgs e)
{
    serialPort1.BaudRate = 9600;
    serialPort1.PortName = "COM2";
    serialPort1.Parity = System.IO.Ports.Parity.None;
    serialPort1.StopBits = System.IO.Ports.StopBits.One;


}
```

- Créer le gestionnaire clic du bouton "Ouvrir port", et compléter le code:

```
private void button1_Click(object sender, EventArgs e)
{
    serialPort1.Open();
}
```

- Créer le gestionnaire clic du bouton "Fermer port", et compléter le code:

```
private void button3_Click(object sender, EventArgs e)
{
    //Bouton Fermer port
    timer1.Enabled = false;
    serialPort1.Close();
    myThread.Abort();
}
```

- Créer le gestionnaire clic du bouton "Envoyer", et compléter le code:

```
private void button2_Click(object sender, EventArgs e)
{
    serialPort1.Write(MaChaineAEmettre);
}
```

- Créer le gestionnaire de l'événement **TextChanged** pour la TextBox d'envoi, et compléter le code:

```
private void textBox1_TextChanged(object sender, EventArgs e)
{

    MaChaineAEmettre = textBox1.Text;

}
```

- Créer le gestionnaire clic du bouton "Recevoir", et compléter le code:

```
private void buttonReception_Click(object sender, EventArgs e)
{
    //Demarrage du Timer pour le Rafraichissement de l'affichage
    timer1.Enabled = true;
    //Démarrage du Thread de réception des caractères.
    myThread.Start();

}
```

- Créer le gestionnaire Tick du Timer (par double-clic) et compléter le code:

```
private void timer1_Tick(object sender, EventArgs e)
{
    textBox2.Text = MaChaineCompleteRecue;
}
```

V) améliorations

- Modifier le code afin de réaliser un fonctionnement par événement pour le port série, tout en conservant le thread, (le code de celui-ci sera modifié: voir pdf ObjetsSynchronisationCSharp)

Modifier votre application afin de pouvoir choisir:

- Utiliser un seul bouton pour l'initialisation du port, l'ouverture du port, et la réception.
- Le port COM1, ou COM2 (case à cocher)
- Le débit de 9600 à 115200 bits/s.
- Le nombre de bit de stop.
- Le type de parité.

- Ajouter un deuxième thread pour le rafraichissement de l'affichage à la place du timer.