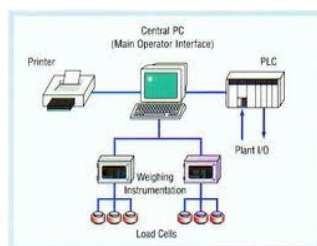




1. Introdução

A comunicação série RS232 continua a ser um dos protocolos de comunicação de referência e muito utilizado industrialmente. Inúmeros equipamentos utilizam a comunicação série para troca de dados, por exemplo pistolas para leitura de códigos de barras, impressoras de etiquetas de códigos de barras e códigos 2D (Datamatrix) conectados a PLC's e computadores. As comunicações série referidas são utilizadas também para implementar sistemas de rastreabilidade, para comunicar com PLC's e Comandos Numéricos para programação e diagnóstico.



A comunicação RS232 é uma comunicação ponto a ponto, ou seja, é realizada apenas entre dois equipamentos. Para que ambos os equipamentos comuniquem entre si, é necessário que ambos tenham os mesmos parâmetros de comunicação:

- **Velocidade de transmissão** de dados, definido em bits por segundo (bps), os valores mais utilizados são: **9600 bps**, 19200 bps, 38400 bps;

- **Controlo de paridade**, permite o controlo de erros na transmissão: ODD, EVEN e **NONE**.

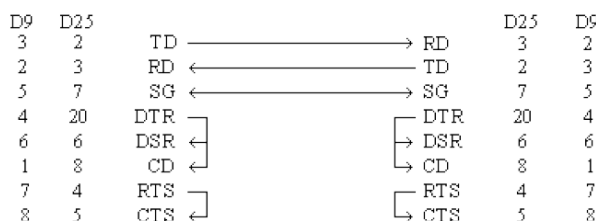
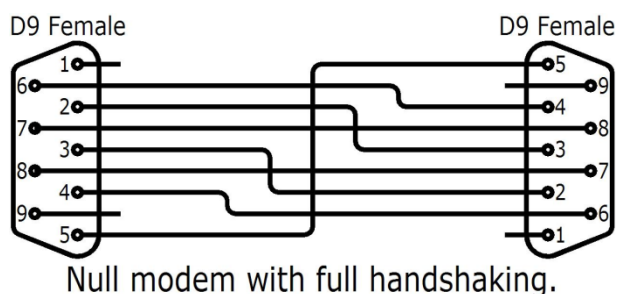
É adicionado um bit pelo dispositivo emissor ao carácter a enviar. O dispositivo recetor após leitura do carácter calcula o bit de paridade e, compara-o com o bit de paridade recebido do dispositivo emissor.

Quando é seleccionado o controlo de paridade par: o somatório de todos os bits a nível lógico "1" incluindo o bit de paridade é par. O controlo de paridade ímpar é semelhante.

- **Controlo de fluxo**: RTS, **Xon/Xoff**, Sem controlo de fluxo;

- **Stop Bits**, número de bits após envio de um carácter: 1 ou 2 stop bits; para definir o fim de transmissão do carácter.

Para os equipamentos comunicarem é necessário interligar os dois equipamentos através de um cabo (conjunto de condutores), como o indicado na figura:



Os computadores mais recentes não dispõem de porta série RS232, mas sim de portas USB, sendo por isso necessário utilizar um adaptador USB/RS232 e instalar o driver apropriado.

2. Objectivos

O trabalho prático pretende familiarizar o aluno com a comunicação série RS232, que permite a troca de informação entre computadores, entre computadores e PLC's ou entre PC e outros equipamentos, que disponibilizem o mesmo interface.

Familiarização com os **objetos de programação**:

- SerialPort, OpenFileDialog, SaveFileDialog, MenuStrip

Familiarização com as **instruções** de programação:

- MsgBox, If Then Else, Mid, Split, InStr, etc

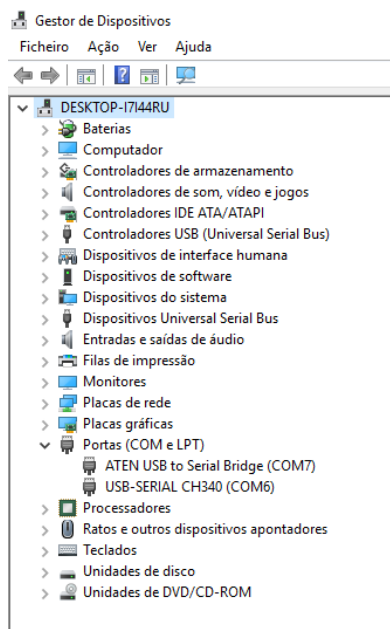
O trabalho prático será realizado em diferentes fases, utilizando inicialmente versões mais simples para comunicação entre computadores, até à realização de um programa de comunicação automático com detecção de tramas e mensagens especiais.

3. Instalação dos drivers dos conversores USB-RS232

As portas série RS232 já raramente equipam os computadores atuais. Estas foram substituídas por portas USB. Assim, é necessário transformar uma porta USB do computador numa porta série. Para isso utilizam-se os chamados conversores USB-RS232. Cada conversor tem um driver específico. Assim, o mais adequado será instalar os seguintes drivers windows para os três diferentes conversores:

<p>ATEN USB-RS232 Serial Converter</p> <p>http://www.aten.com/global/en/products/usb-&-thunderbolt/usb-converters/uc232a/#.WdlvjGhSyW9</p>	
<p>Prolific PL2303</p> <p>http://www.prolific.com.tw/US/ShowProduct.aspx?p_id=225&pcid=41</p>	
<p>CH340</p> <p>https://sparks.gogo.co.nz/ch340.html</p>	

Para verificar se os drivers estão a funcionar corretamente devem-se verificar as portas que estão a ser utilizadas usando o "device manager". Se, depois de ligar um equipamento, na secção das portas COM aparecer um símbolo amarelo quer dizer que não existe um driver para o equipamento que está ligado.



4. Instalação do com0com

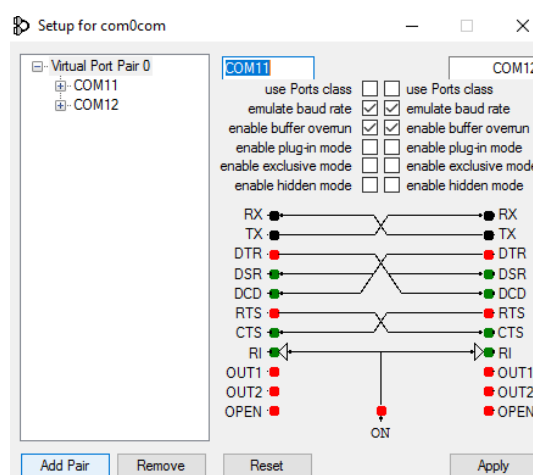
O com0com é um software que emula portas série no computador. Assim, permite que os alunos possam testar o programa em casa sem terem acesso aos equipamentos conversores USB-RS232.

Para isso é necessário instalar o software com0com (<http://com0com.sourceforge.net/>). Para instalar aceda ao link <https://sourceforge.net/projects/com0com/files/latest/download>

Depois de instalar a aplicação, confirme no gestor de dispositivos se foi criado um par de portas COM. Se for este o caso em princípio não necessita de fazer os passos seguintes. Caso contrário pode fazer a configuração do programa executando o seguinte ficheiro

C:\Program Files (x86)\com0com\setupg.exe

Na janela, devem seleccionar as opções assinaladas na seguinte figura:

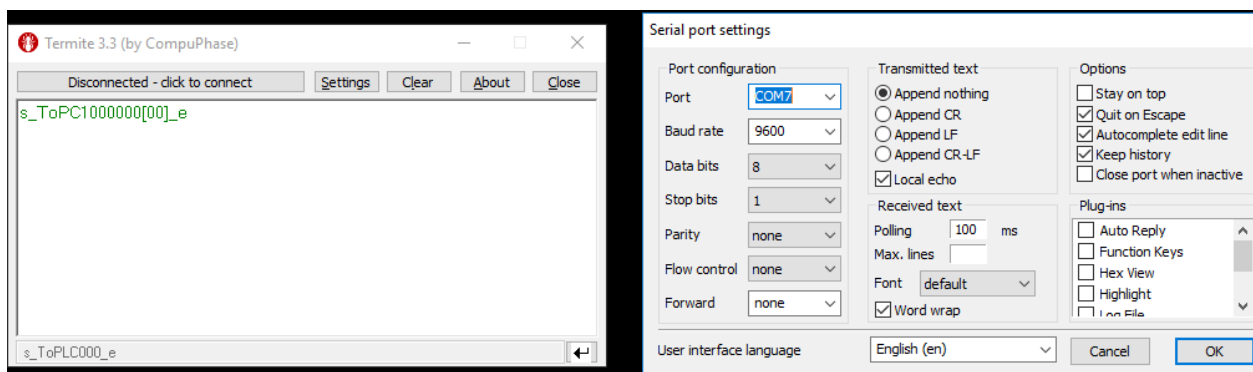


Caso não consiga criar corretamente as portas COM veja o seguinte manual:

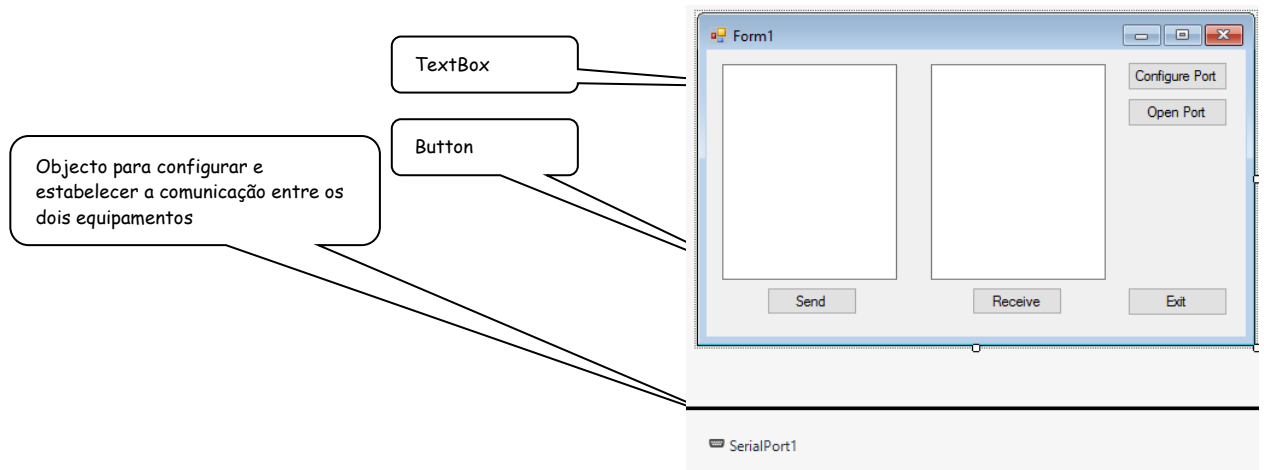
<http://www.softwx.com/weather/virtualvp/VirtualSerialPorts64bitHelp.pdf>

5. Software de comunicação série Termite

O Termite <https://www.compuphase.com/software/termite.htm> é um software que permite a comunicação do computador com outros equipamentos por porta série. Para instalar, fazer download aqui <https://www.compuphase.com/software/termite-3.3.exe> . Na imagem em baixo por ver-se o software. Para utilizar o programa é preciso configurá-lo primeiro, indicando os parâmetros de comunicação adequados. Para fazer isso clicar em settings e depois escolher os parâmetros corretos.



6. Exercício 1: Programa simples de comunicação



Desenvolvimento de uma aplicação simples para envio e receção de mensagens entre dois computadores. O programa deverá conter:

- **Botão para configurar a porta de comunicação**, atribuindo os parâmetros de comunicação 9600 bps, sem paridade (None), 8 bits de dados e um stop bit;
- **Botão para iniciar/fechar** a comunicação entre computadores. Utilizar o mesmo botão alterando o texto;
- **Caixa de texto** para **escrever e enviar a mensagem**;
- **Caixa de texto** para visualização da **mensagem recebida**;

Utilizar o objeto **SerialPort**, para configurar e estabelecer a comunicação entre computadores, principais propriedades:

SerialPort1.PortName="COM1"	- Nome
SerialPort1.ReadExisting	- Leitura dos dados recebidos
SerialPort1.Write (Txt_Envio.Text)	- Envio dos dados
SerialPort1.Isopen	- Verifica se está aberta
SerialPort1.Open	- Abrir porta
SerialPort1.Close	- Fechar Porta

No windows, para consultar as propriedades da porta de comunicação (COM) associada ao conversor USB/RS232, consultar o "Painel de Controlo" -> "System and Security" -> "Device Manager". Ver secção 2.

Listagem do programa:

```
...box\AulasUA\II_2017-2018\Pratica\Aula3\Ex1\Ex1\Form1.vb 1
1 Public Class Form1
2
3     Dim data_received As String
4
5     '-----
6     'Initialization routines
7     '-----
8     Private Sub Form1_Load(sender As Object, e As EventArgs) Handles ➤
9         MyBase.Load
10        TextBoxSend.Text = "My first message!"
11    End Sub
12
13    '-----
14    'Serial port configuration routines
15    '-----
16    'Configure serial communication parameters (hardcoded for now)
17    Private Sub ButtonConfigurePort_Click(sender As Object, e As EventArgs) ➤
18        Handles ButtonConfigurePort.Click
19        With SerialPort1
20            .PortName = "COM5" 'Check with device manager for the open port
21            .BaudRate = 9600
22            .Parity = IO.Ports.Parity.None
23            .DataBits = 8
24            .StopBits = IO.Ports.StopBits.One
25            .Encoding = System.Text.Encoding.UTF8
26        End With
27    End Sub
28
29    'Open and close serial port
30    Private Sub ButtonOpenClosePort_Click(sender As Object, e As EventArgs) ➤
31        Handles ButtonOpenClosePort.Click
32        If SerialPort1.IsOpen Then 'If opened, close. Change text on button
33            SerialPort1.Close()
34            ButtonOpenClosePort.Text = "Open Port"
35            ButtonOpenClosePort.BackColor = Color.LightGray
36        Else 'If closed, open. Change text on button
37            SerialPort1.Open()
38            ButtonOpenClosePort.Text = "Close Port"
39            ButtonOpenClosePort.BackColor = Color.Red
40        End If
41    End Sub
42
43    '-----
44    'Communication routines
45    '-----
46
47    'Send data in TextBoxSend
48    Private Sub ButtonSend_Click(sender As Object, e As EventArgs) Handles ➤
49        ButtonSend.Click
50        SerialPort1.Write(TextBoxSend.Text)
51    End Sub
52
53    Private Sub ButtonReceive_Click(sender As Object, e As EventArgs) Handles ➤
```

```

...box\AulasUA\II_2017-2018\Pratica\Aula3\Ex1\Ex1\Form1.vb 2
    ButtonReceive.Click
53     data_received = data_received & SerialPort1.ReadExisting 'read from serial port
54     TextBoxReceive.Text = data_received 'copy from data_received
55     data_received = "" 'erase text in data_received
56 End Sub
57
58 '-----
59 'Other routines
60 '-----
61 Private Sub ButtonExit_Click(sender As Object, e As EventArgs) Handles
    ButtonExit.Click
62     End
63 End Sub
64
65
66 End Class
67

```

7. Funções de interesse para processamento de strings

Função “Mid”

Permite separar uma *substring* a partir de uma *string* original.

A função recebe três parâmetros: uma variável do tipo *string*, contendo um conjunto de caracteres; um número com a posição do primeiro caractere a ler; e o número de caracteres consecutivos a ler

Mid-Exemplo:

```

Dim strOriginal As String = "0123456789"
MessageBox.Show(Mid(strOriginal, 1, 2)) 'devolve o valor "01"
MessageBox.Show(Mid(strOriginal, 5, 3)) 'devolve o valor "456"

```

A função “Split”

Divide uma *string* em *substrings*.

A função recebe uma *string*, um caractere delimitador e devolve um array de *strings*. Em cada posição do array devolvido vem uma parte da *string* original. Por exemplo, se a *string* “*String_original*” possuir um espaço em branco entre cada letra e o utilizador quiser separar as letras obtendo em cada posição do array uma só letra, pode indicar à função “split” para usar o caractere “espaço” como divisor da *string* original. A este caractere, usado para dividir a “*String_original*”, chama-se delimitador.

Split-Exemplo:

```

Dim strOriginal As String = "o l a"
Dim arrayDeStrings As String()
arrayDeStrings = Split(strOriginal, " ")
MessageBox.Show(arrayDeStrings(0))
MessageBox.Show(arrayDeStrings(1))
MessageBox.Show(arrayDeStrings(2))

```

A função “InStr”

Permite localizar a posição de uma *substring* numa *string* maior.

A função recebe duas *strings*: a *string* maior e a *substring* e devolve um inteiro com posição da *substring* na *string* maior. O valor devolvido pode ser (0) se a *substring* não for encontrada na *string* maior, ou (≥ 1)

InStr-Exemplo:

```
Dim strOriginal As String = "abcdefghi"  
Dim strChave As String = "cd"  
MessageBox.Show(InStr(strOriginal, strChave)) 'devolve o valor 3
```

As funções “Right” e “Left”

Devolve uma *substring* que contém um determinado número de caracteres, a contar da direita/esquerda, da *string* original.

Por existir uma propriedade do form com o mesmo nome é necessário utilizar o caminho completo *Microsoft.VisualBasic.Left* e *Microsoft.VisualBasic.Right*.

Right e Left-Exemplo:

```
Dim strOriginal As String = "abcdefghi"  
MessageBox.Show(Microsoft.VisualBasic.Left(strOriginal, 3)) 'devolve o valor "abc"  
MessageBox.Show(Microsoft.VisualBasic.Right(strOriginal, 2)) 'devolve o valor "hi"
```

A função “Len”

Devolve um inteiro com o número de caracteres que existem numa *string*.

A função recebe uma *string*, da qual queremos saber o seu comprimento, ou seja, o número de caracteres que a constituem e devolve um número.

InStr-Exemplo:

```
Dim strOriginal As String = "abcdefghi"  
MessageBox.Show(Len(strOriginal)) 'devolve o valor 9
```

A função “StrComp”

Compara duas *strings*.

A função recebe duas *strings* e devolve um valor inteiro, igual a zero se as duas *strings* forem iguais.

StrComp-Exemplo:

```
Dim palavra1 As String = "123"  
Dim palavra2 As String = "123"  
MessageBox.Show(StrComp (palavra1, palavra2)) 'devolve o valor 0
```

A função “CStr”

Permite converter um número numa *string*. A *string* obtida pode então ser atribuída a variáveis do tipo *string* ou usada em funções de manipulação de *strings*.

CStr-Exemplo:

```
Dim texto As String  
texto = CStr(123)
```

A função “Chr”

Converte um valor numérico no seu caractere correspondente, de acordo com a tabela ASCII

Chr-Exemplo:

```
MessageBox.Show(Chr(65)) 'devolve o valor "A"
```

A função “Asc”

Converte um caractere no seu valor decimal de acordo com a tabela ASCII.

Esta função recebe uma *string* e devolve um valor inteiro.

Asc-Exemplo:

```
MessageBox.Show(Asc("A")) 'devolve o valor 65
```


Leitura e Escrita de Ficheiros Texto

Função	Descrição
Fileopen(1, NomeFicheiro, OpenMode.xxx)	Abertura do ficheiro
OpenMode: Input, Output, Append, Binary, Randon	Entrada, Saída, Anexar, Binário, Randon de dados
TextBox1.Text = LineInput(1)	Leitura de uma linha do ficheiro Texto
PrintLine(1, TextBox2.Text)	Escrita da string TextBox2.Text no ficheiro
EOF(1)	Fim do ficheiro
Flush (1)	Transfere dados do buffer para o ficheiro
FileClose (1)	Fecha ficheiro

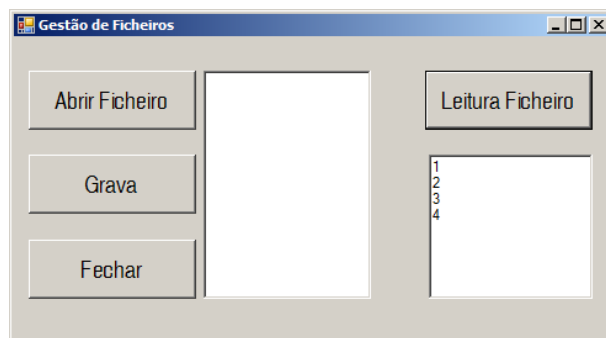
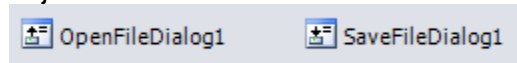
8. Outros Exemplos

Programa para escrita e leitura de dados num ficheiro de texto.

Para utilizar o acesso a ficheiros utilizando o *StreamWriter* e *StreamReader*, é necessário declarar no início do programa:

Imports System.IO

Para permitir seleccionar o ficheiro de leitura e/ou o ficheiro de escrita, necessitamos de utilizar os objectos:



Para ativar mensagens para o utilizador, utilizar a instrução:

`MsgBox("Cancelada a Abertura do Ficheiro")`

Instruções para gestão de ficheiros:

Leitura de Ficheiro	Escrita em ficheiro	
<code>OpenFileDialog (fic)</code>	<code>SaveFileDialog (fic)</code>	Declaração do Objecto: Leitura/Escrita
<code>fic.FileName) = ""</code>		Nome do Ficheiro (L/E)
<code>fic.InicialDirectory = "E:\"</code>		Diretório(Pasta) inicial
<code>fic.Filter = "Ficheiros Text (*.txt) *.txt Todos os Ficheiros (*.*) *.*"</code>		Filtro para a seleção dos ficheiros
<code>fic.FilterIndex=2</code>		Utiliza o 2º filtro por definição
<code>fic.Tiltle = "Abertura Ficheiro" ou "Nome Ficheiro a Gravar"</code>		Título da caixa de diálogo
<code>fic.ShowDialog ()</code>		Visualização da Cx de diálogo

fic.ShowDialog() = DialogResult.OK	Ficheiro selecionado OK
Fic.ShowDialog() = DialogResult.NOK	Anulada a Abertura de Ficheiro
fs = New System.IO.StreamReader(fic.filename) fs = New System.IO.StreamWriter(fic.filename)	Abertura do ficheiro Escrita no ficheiro
Txt = fs.ReadToEnd()	fs.Write(TextBox2.Text)
Fs.Close()	Fechar ficheiro

Listagem do programa:

```
Imports System.IO
Public Class Form1
    Dim ficheiro_guardar As System.IO.StreamWriter

    Private Sub Form1_Load
        ' Filtros para permitir apenas a abertura de alguns tipos de ficheiros
        OpenFileDialog1.Filter = "Ficheiros de texto | *.txt | Fic HTML | *.html|Todos os Fic|*.*" ' Filtro de abertura de ficheiros

        SaveFileDialog1.Filter = "Fic de Texto | *.txt" ' Filtro para gravar ficheiros com uma determinada extensão
    End Sub

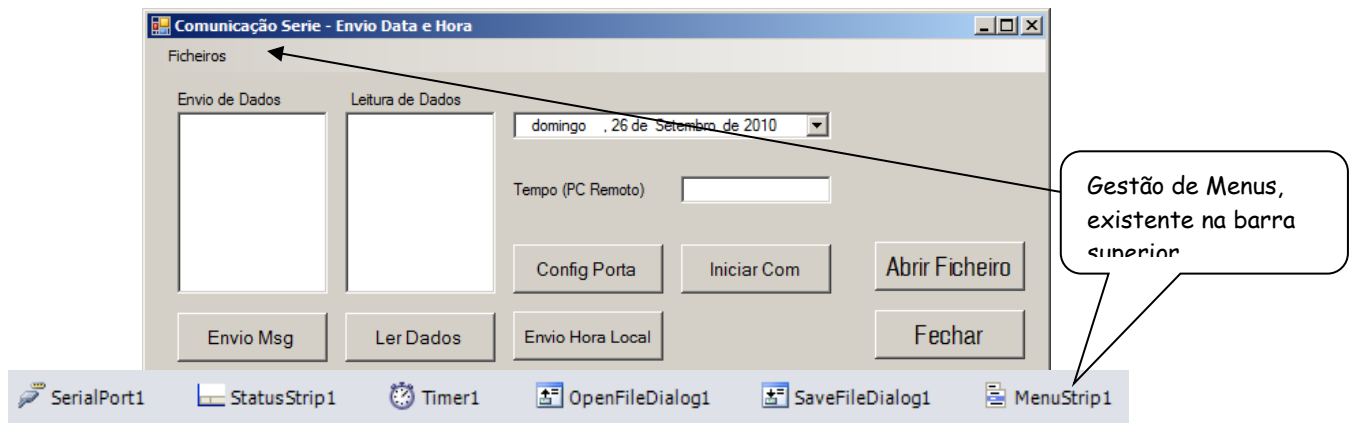
    Private Sub Bt_Ler_Click
        ' Leitura de um ficheiro de texto
        OpenFileDialog1.FileName = "" ' Anulada a abertura de ficheiro
        OpenFileDialog1.ShowDialog()
        If Windows.Forms.DialogResult.OK And OpenFileDialog1.FileName <> "" Then
            Dim streamabrir As StreamReader = New StreamReader(OpenFileDialog1.FileName)
            txt_leitura.Text = streamabrir.ReadToEnd()
            streamabrir.Close()
        Else
            MsgBox("Cancelada a Abertura do Ficheiro")
        End If
    End Sub

    Private Sub Bt_Abrir_Click
        ' Seleccionar ficheiro de texto
        SaveFileDialog1.FileName = "teste"
        SaveFileDialog1.ShowDialog() ' Abertura da janela de diálogo para abertura de ficheiros
        If Windows.Forms.DialogResult.OK Then
            ficheiro_guardar = New System.IO.StreamWriter(SaveFileDialog1.FileName)
        End If
    End Sub

    Private Sub Grava_Click
        ' Escreve dados em ficheiro
        ficheiro_guardar.Write(txt_escrita.Text)
        txt_escrita.Text = ""
    End Sub

    Private Sub Fechar_Click
        ' Fecha ficheiro de texto
        ficheiro_guardar.Close()
    End Sub
End Class
```

- a. **Envio/ Receção de Mensagens, envio da hora do computador, registo em ficheiro de texto e Gestão de Menus.**



Modificar o programa anterior, retirar alguns botões de comando para simplificar o interface com o utilizador, mantendo as mesmas funcionalidades.

O botão para sair, é realizado pela opção sair das opções existentes no menu superior. A visualização dos dados existentes no ficheiro será realizada numa janela independente (form2).

A **gestão de menus** é realizada utilizando o **objecto MenuStrip**. Algumas notas de utilização:

- Escrever o nome das opções, na respectiva caixa de diálogo;
- a utilização de "&" antes de uma letra, permite o acesso rápido a essa opção com Ctrl + Letra respectiva;
- A utilização de "-", permite criar uma linha de separação;

Para **gestão das forms**, poderá realizar e utilizar seguinte programa:

```
' Chamada de uma nova form
Form2.StartPosition = FormStartPosition.CenterParent
Form2.Text = "Janela 2"
Form2.ShowDialog()
```

Para fechar uma form:

```
' Voltar à página anterior
Close()
```

Listagem programa:

```
Public Class Form1
```

```
Private Sub Button1_Click...
```

```
' Chamada de uma nova form
Form2.StartPosition = FormStartPosition.CenterParent
Form2.Text = "Janela 2"
Form2.ShowDialog()
```

```
End Sub
```

```
Private Sub Btn_Janela3_Click...
```

```
Dim form3rect As New Rectangle(800, 400, 400, 450)
Form3.Text = "Inf Ind - Janela 3"
form3.FormBorderStyle = Windows.Forms.FormBorderStyle.FixedSingle
form3.StartPosition = FormStartPosition.Manual
form3.DesktopBounds = form3rect
Form3.ShowDialog()
```

```
End Sub
```

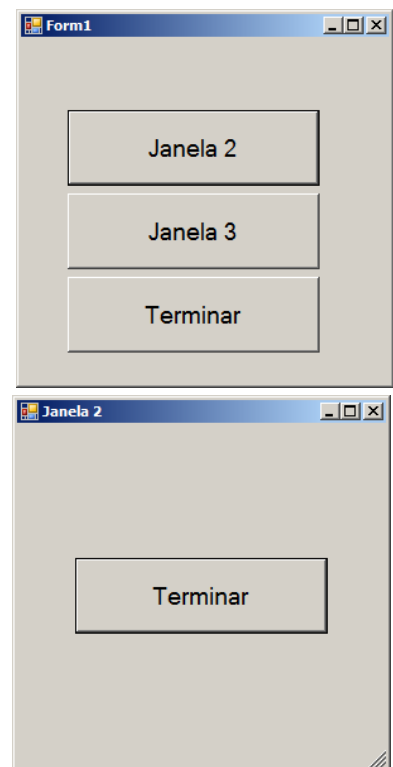
```
Private Sub Btn_Terminar_Click...
```

```
End
```

```
End Sub
```

```
Private Sub Form1_Load...
```

```
' Positionamento da Janela Principal
Me.StartPosition = FormStartPosition.Manual
Me.Location = New Point(600, 400)
```



```

' Posicionamento no centro da janela
Me.Left = (Screen.PrimaryScreen.WorkingArea.Width - Me.Width) / 2
Me.Top = (Screen.PrimaryScreen.WorkingArea.Height - Me.Height) / 2
End Sub
End Class

```

Public Class Form2

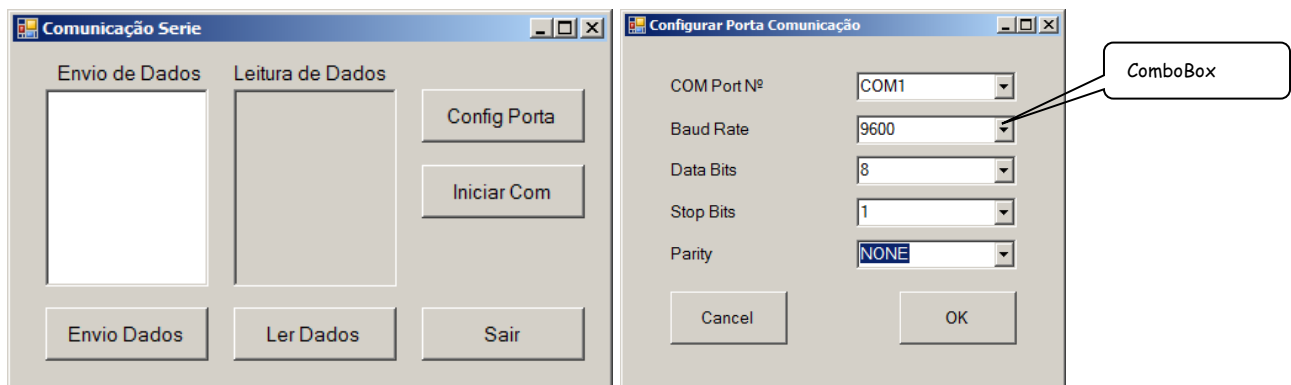
```

Private Sub Btn_Terminar_Click
    Close()
End Sub
End Class

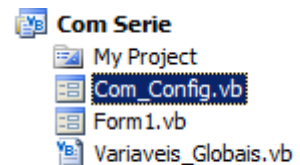
```

Acrescentar ao exemplo anterior de Comunicação RS232, uma nova “form” que permita a configuração dos parâmetros da porta série.

Na versão anterior acrescentar uma nova janela de configuração dos parâmetros de comunicação da porta série, como indicado no exemplo:



Para realizar este programa, acrescentar um novo módulo, para declaração das variáveis globais, e outra form para acrescentar as opções de configuração da porta série.



Listagem do programa:

```

Public Class Form1
    Dim rx As String
    Private Sub Form1_Load
        ' Atributos iniciais
        My.Forms.Com_Config.Text = "Configurar Porta Comunicação"
    End Sub
    Private Sub Bt_Config_Porta_Click
        ' Posicionamento da janela, no centro do monitor
        Com_Config.StartPosition = FormStartPosition.CenterScreen
        Com_Config.Show()
    End Sub
    Private Sub Bt_Iniciar_Com_Click
        ' Abrir/Iniciar Porta de Comunicação
        If Not SerialPort1.IsOpen Then
            ' Configurar Porta de Comunicação
            With SerialPort1
                .PortName = comm_port
                .BaudRate = comm_baudrate
                .Parity = comm_parity
                .DataBits = comm_databits
                .StopBits = comm_stopbits
            End With

            SerialPort1.Open()
            Bt_Iniciar_Com.Text = "Fechar Com"
        Else
            SerialPort1.Close()
            Bt_Iniciar_Com.Text = "Iniciar Com"
        End If
    End Sub
    Private Sub SerialPort1_DataReceived

```

```

        'Leitura de dados
        rx = rx & SerialPort1.ReadExisting
    End Sub
    Private Sub Bt_Envio_Click
        'Envio de dados
        SerialPort1.Write(TextBox1.Text)
    End Sub
    Private Sub Bt_Ler_Dados_Click
        ' Visualizar Dados Recebidos
        TextBox2.Text = rx
        rx = ""
    End Sub
    Private Sub Bt_Sair_Click
        ' Terminar Programa
    End
End Sub
End Class
Public Class Com_Config
    Private Sub Com_Config_Load
        ' Variáveis a visualizar em cada uma das
        cbbox_Port.Items.Clear()
        cbbox_Port.Items.Add("COM1")
        cbbox_Port.Items.Add("COM2")
        cbbox_Port.Items.Add("COM3")
        cbbox_Port.Items.Add("COM4")
        cbbox_Port.SelectedIndex = 0

        cbbox_Baud_rate.Items.Clear()
        cbbox_Baud_rate.Items.Add("4800")
        cbbox_Baud_rate.Items.Add("9600")
        cbbox_Baud_rate.Items.Add("19200")
        cbbox_Baud_rate.SelectedIndex = 0

        cbbox_Databits.Items.Clear()
        cbbox_Databits.Items.Add("7")
        cbbox_Databits.Items.Add("8")
        cbbox_Databits.SelectedIndex = 0

        cbbox_Stopbits.Items.Clear()
        cbbox_Stopbits.Items.Add("1")
        cbbox_Stopbits.Items.Add("1.5")
        cbbox_Stopbits.Items.Add("2")
        cbbox_Stopbits.SelectedIndex = 0

        cbbox_Parity.Items.Clear()
        cbbox_Parity.Items.Add("NONE")
        cbbox_Parity.Items.Add("ODD")
        cbbox_Parity.Items.Add("EVEN")
        cbbox_Parity.SelectedIndex = 0
    End Sub

    Private Sub BTcancel_Click
        ' Sair da janela de configuração
        Close()
    End Sub

    Private Sub BTOK_Click
        ' Validar os dados e sair da janela de configuração
        comm_port = cbbox_Port.Text
        comm_databits = CInt(cbbox_Databits.Text)
        comm_baudrate = CInt(cbbox_Baud_rate.Text)

        Select Case cbbox_Stopbits.Text
            Case "0"
                comm_stopbits = IO.Ports.StopBits.None
            Case "1"
                comm_stopbits = IO.Ports.StopBits.One
            Case "1.5"
                comm_stopbits = IO.Ports.StopBits.OnePointFive
            Case "2"
                comm_stopbits = IO.Ports.StopBits.Two
        End Select

        Select Case cbbox_Parity.Text
            Case "Even"
                comm_parity = IO.Ports.Parity.Even
            Case "Odd"
                comm_parity = IO.Ports.Parity.Odd
            Case "None"
                comm_parity = IO.Ports.Parity.None
        End Select
    End Sub
End Class

```

```
        Close()  
    End Sub  
End Class
```

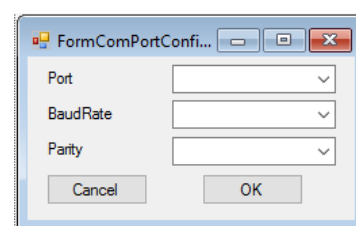
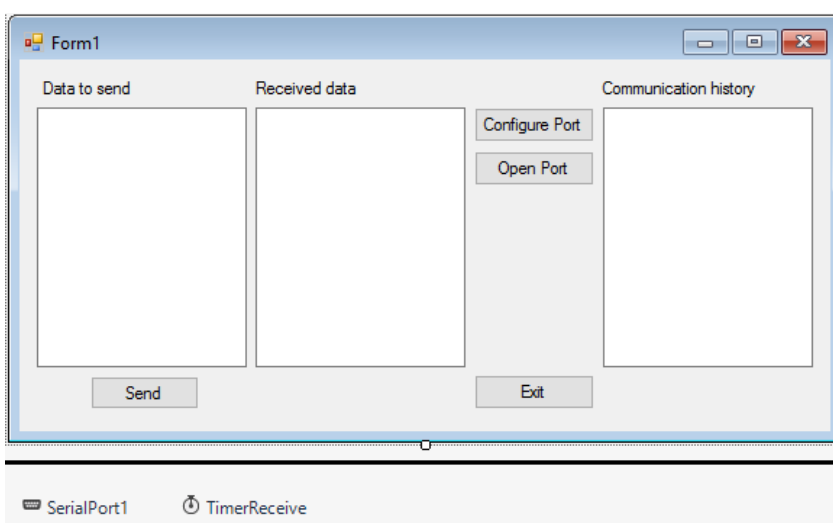
```
Module Variaveis_Globais
```

```
    'Configuração da porta'  
    Public comm_port As String  
    Public comm_databits As Byte  
    Public comm_stopbits As IO.Ports.StopBits  
    Public comm_baudrate As Integer  
    Public comm_parity As IO.Ports.Parity  
End Module
```






Pretende-se melhorar o programa do exemplo 1, acrescentando as seguintes funcionalidades:

- Leitura automática (periódica) das mensagens usando um objeto Timer e removendo o botão
- Implementação de um registo do histórico de mensagens, com indicação da hora de receção da mensagem
- Utilização de um elemento *Module* para definição das variáveis globais dos parâmetros da porta. Para simplificar o programa, serão configurados apenas os parâmetros: nome da porta, baud rate e paridade. Todos os outros devem ficar no seu estado default.
- Utilização de um *Form* adicional para configuração dos parâmetros da porta
- Utilização da variável *is_configuration_valid* para gestão dos comportamentos do programa, i.e., só abre a porta COM se a configuração dos parâmetros for válida.
- Fazer o disable dos botões do programa de forma contextual, i.e. enquanto a porta COM estiver fechada, o botão de de envio deve estar *disabled*.









Listagem do programa da Form1:

```
...box\AulasUA\II_2017-2018\Pratica\Aula3\Ex2\Ex2\Form1.vb 1
1 Public Class Form1
2
3     Dim data_received As String
4
5     '-----
6     'Initialization routines
7     '-----
8     Private Sub Form1_Load(sender As Object, e As EventArgs) Handles 
9         MyBase.Load
10         TextBoxSend.Text = "My first message!"
11
12         'Configure TimerReceive to tic every x millisecs
13         TimerReceive.Interval = 300
14         TimerReceive.Enabled = False 'Only tic when the Port is Open
15
16         'To make sure we do not open the Port without a valid configuration
17         ModuleComPortParameters.configuration_is_valid = False
18     End Sub
19
20     '-----
21     'Serial port configuration routines
22     '-----
23     'Configure serial communication parameters
24     Private Sub ButtonConfigurePort_Click(sender As Object, e As EventArgs) 
25         Handles ButtonConfigurePort.Click
26         FormComPortConfiguration.StartPosition = FormStartPosition.CenterParent
27         FormComPortConfiguration.ShowDialog()
28
29         'Now setup the object SerialPort1 using the variables from the 
30         ModuleComPortParameters
31         'Only use the vales if configuration was declared valid
32         If ModuleComPortParameters.configuration_is_valid = True Then
33             With SerialPort1
34                 .PortName = ModuleComPortParameters.port
35                 .BaudRate = ModuleComPortParameters.baud_rate
36                 .Parity = ModuleComPortParameters.parity
37             End With
38         End If
39     End Sub
40
41     'Open and close serial port
42     Private Sub ButtonOpenClosePort_Click(sender As Object, e As EventArgs) 
43         Handles ButtonOpenClosePort.Click
44         If ModuleComPortParameters.configuration_is_valid = True Then
45             If SerialPort1.IsOpen Then 'If opened, close. Change text on button
46                 SerialPort1.Close()
47                 ButtonOpenClosePort.Text = "Open Port"
48                 ButtonOpenClosePort.BackColor = Color.LightGray
49                 TimerReceive.Enabled = False 'Stop receiving periodically
50             Else 'If closed, open. Change text on button
51                 SerialPort1.Open()
52                 ButtonOpenClosePort.Text = "Close Port"
53                 ButtonOpenClosePort.BackColor = Color.Red
54                 TimerReceive.Enabled = True 'Start receiving periodically
```



```

53         End If
54     Else
55         MsgBox("Cannot open port with an invalid configuration!")
56     End If
57 End Sub
58
59 '-----
60 'Communication routines
61 '-----
62
63 'Send data in TextBoxSend
64 Private Sub ButtonSend_Click(sender As Object, e As EventArgs) Handles 
65     ButtonSend.Click
66     SerialPort1.Write(TextBoxSend.Text)
67 End Sub
68
69 'Receive data
70 Private Sub TimerReceive_Tick(sender As Object, e As EventArgs) Handles 
71     TimerReceive.Tick
72     data_received = data_received & SerialPort1.ReadExisting 'read from 
73     serial port
74
75     If Len(data_received) > 0 Then 'Copy from reception buffer to TextBox 
76         if there is something there
77         TextBoxReceive.Text = data_received 'copy from data_received
78         TextBoxHistory.Text = TimeOfDay.ToLongTimeString + ": " + 
79         data_received + vbNewLine +
80         TextBoxHistory.Text 'Append data to history
81         data_received = "" 'erase text in data_received
82     End If
83 End Sub
84
85 '-----
86 'Other routines
87 '-----
88 Private Sub ButtonExit_Click(sender As Object, e As EventArgs) Handles 
89     ButtonExit.Click
90 End Sub
91
92 End Class

```

```
...-2018\Pratica\Aula3\Ex2\Ex2\FormComPortConfiguration.vb 1
1 Public Class FormComPortConfiguration
2     '-----
3     'Initialization routines
4     '-----
5
6     Private Sub FormComPortConfiguration_Load(sender As Object, e As EventArgs) Handles MyBase.Load
7         'Configure the combo boxes with the appropriate options
8
9         'ComboBox Port Configuration
10        Dim available_ports As Array = IO.Ports.SerialPort.GetPortNames
11        Dim i As Integer 'cycle all available ports
12        For i = 0 To UBound(available_ports)
13            ComboBoxPort.Items.Add(available_ports(i))
14        Next
15        ComboBoxPort.SelectedIndex = 0 'Default value
16
17        'ComboBox Baud rate configuration
18        ComboBoxBaudRate.Items.Clear()
19        ComboBoxBaudRate.Items.Add("4800")
20        ComboBoxBaudRate.Items.Add("9600")
21        ComboBoxBaudRate.Items.Add("19200")
22        ComboBoxBaudRate.SelectedIndex = 2 'Default
23
24        'ComboBox parity configuration
25        ComboBoxParity.Items.Clear()
26        ComboBoxParity.Items.Add(IO.Ports.Parity.None)
27        ComboBoxParity.Items.Add(IO.Ports.Parity.Odd)
28        ComboBoxParity.Items.Add(IO.Ports.Parity.Even)
29        ComboBoxParity.SelectedIndex = 2
30
31    End Sub
32
33    '-----
34    'Button press routines
35    '-----
36
37    Private Sub ButtonCancel_Click(sender As Object, e As EventArgs) Handles ButtonCancel.Click
38        ModuleComPortParameters.configuration_is_valid = False
39        Me.Close()
40    End Sub
41
42    Private Sub ButtonOK_Click(sender As Object, e As EventArgs) Handles ButtonOK.Click
43        'Copy the values in the combo boxes to the variables in the ModuleComPortParameters
44        'In some cases, must use a switch case to map from string to IO.Ports type values
45
46        ModuleComPortParameters.port = ComboBoxPort.Text 'String to String mapping, no sweat
47        ModuleComPortParameters.baud_rate = ComboBoxBaudRate.Text 'String to Integer, automatic conversion
48
49        Select Case ComboBoxParity.Text 'String to IO.Ports.Parity, trickier
```

```

...-2018\Pratica\Aula3\Ex2\Ex2\FormComPortConfiguration.vb 2
50         Case "None"
51             ModuleComPortParameters.parity = IO.Ports.Parity.None
52         Case "Even"
53             ModuleComPortParameters.parity = IO.Ports.Parity.Even
54         Case "Odd"
55             ModuleComPortParameters.parity = IO.Ports.Parity.Odd
56     End Select
57
58     'Set configuration is valid flag
59     ModuleComPortParameters.configuration_is_valid = True
60
61     Me.Close()
62 End Sub
63 End Class

```

Listagem do programa da ModuleComPortParameters

```

...7-2018\Pratica\Aula3\Ex2\Ex2\ModuleComPortParameters.vb 1
1 Module ModuleComPortParameters
2     'Global variables for the communication parameters
3     Public port As String
4     Public baud_rate As Integer
5     Public parity As IO.Ports.Parity
6     Public configuration_is_valid As Boolean
7 End Module
8

```