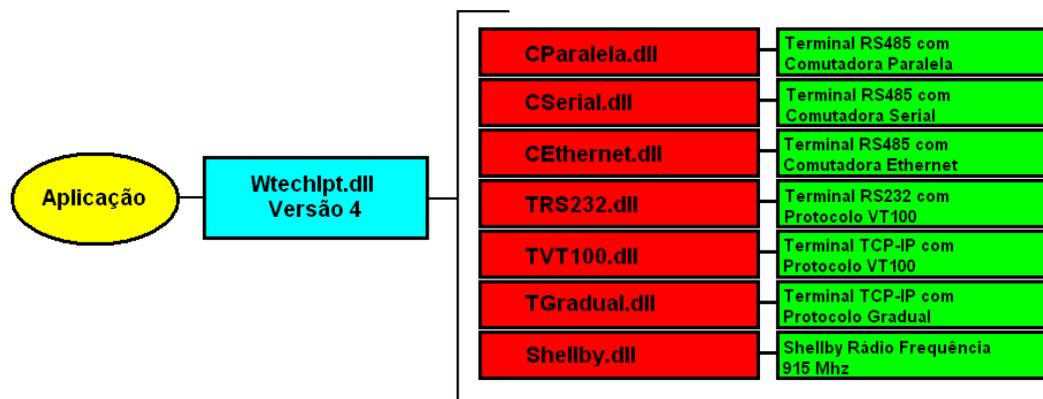


# Wilbor Tech – Gradual Tecnologia

## Manual da Dll Wtechlpt Versão 4.x.x.x



- APLICAÇÃO REALIZA CHAMADAS AS ROTINAS DENTRO DA WTECHLPT.DLL.  
 - WTECHLPT.DLL CONFORME A CONFIGURAÇÃO DO ARQUIVO DE INICIALIZAÇÃO DE CADA INTERFACE, REDIRECIONA A CHAMADA.

A Gradual disponibiliza uma Dll para comunicação com os Terminais. Deste modo, basta que sua linguagem de programação suporte o uso de Dll's para que o sistema comunique com os microterminais.

Inicialmente a Dll Wtechlpt foi projetada para fazer a leitura/escrita na porta paralela do PC, comunicando-se com uma comutadora paralela, usada em redes RS-485.

Com o passar do tempo, a Dll evoluiu para outros tipos de comunicação, e atualmente ela consegue comunicar com os seguintes equipamentos (inclusive simultaneamente):

- ✓ Terminais Shellby 915 (comunicação via RF – rádio frequência)
- ✓ Terminais TCP/IP (protocolo VT-100 ou Gradual)
- ✓ Terminais RS-485 (comutadora paralela, serial e/ou ethernet)
- ✓ Terminais RS-232

A versão 4.x.x.x da Wtechlpt trabalha em conjunto com outras Dll's, mostradas a seguir:

Equipamento	DLL	Arquivo .INI
Terminal Shellby 915	Shellby.dll	Shellby_Base1.ini
Comutadora Paralela	CParalela.dll	CParalela.ini
Comutadora Serial	CSerial.dll	CSerial.ini
Comutadora Ethernet	CEthernet.dll	CEthernet.ini
Terminal TCP/IP (VT-100)	TVT100.dll	TVT100.ini
Terminal TCP/IP (Gradual)	TGradual.dll	TGradual.ini
Terminal RS-232	TRS232.dll	TRS232.ini

## **Desenvolvimento do sistema**

Para manipulação dos terminais, como limpeza de tela, envio de mensagem para o display, posicionamento de cursor, etc, o sistema deverá chamar as funções contidas na Dll.

Basicamente, o sistema deverá possuir um Timer, que fará a varredura de teclado dos terminais, ou seja, em intervalos regulares verificará se alguma tecla do terminal foi pressionada e fazer o devido tratamento, ecoando ou não a informação no display. Se o sistema for construído através de Threads, deve-se tomar cuidado com seu gerenciamento, evitando conflitos entre elas.

A versão 4.x.x.x da Wtechlpt é compatível com suas versões anteriores, tendo as mesmas funções internas. Deste modo, o mesmo sistema que foi projeto para comunicar com a Comutadora Paralela funciona também com outros equipamentos da Gradual (conforme tabela), basta trocar a Dll e configurar o arquivo de inicialização.

## **Configuração de arquivos**

Cada Dll trabalha em conjunto com um arquivo INI, onde são informadas as configurações de interface e lista de terminais habilitados.

**IMPORTANTE:** Os arquivos de configuração INI devem ser colocados no mesmo diretório do executável (.exe) do sistema, e as Dll's devem permanecer nos diretórios de sistema do windows ou do executável.

Para uma descrição completa dos arquivos de configuração, veja em seu respectivo manual contida neste pacote.

## **Funcionamento simultâneo de equipamentos**

A versão 4.x.x.x da Wtechlpt permite a utilização de diferentes tipos de equipamentos ao mesmo tempo, utilizando as Dll's em conjunto.

### **Exemplo 1:**

- Equipamento: Comutadora Paralela
- Arquivos: Wtechlpt.dll, CParalela.dll e CParalela.ini

### **Exemplo 2:**

- Equipamentos: Comutadora Paralela e Terminais Shellby 915
- Arquivos: Wtechlpt.dll, CParalela.dll, CParalela.ini, Shellby.dll e Shellby\_Base1.ini

### **Exemplo 3:**

- Equipamento: Comutadora Serial
- Arquivos: Wtechlpt.dll, CSerial.dll e CSerial.ini

### **Exemplo 4:**

- Equipamento: Terminais Shellby 915
- Arquivos: Wtechlpt.dll, Shellby.dll e Shellby\_Base1.ini

### **Exemplo 5:**

- Equipamento: Terminal TCP/IP (protocolo VT-100)
- Arquivos: Wtechlpt.dll, TVT100.dll e TVT100.ini

### **Exemplo 6:**

- Equipamentos: Terminal TCP/IP (protocolo VT-100) e Terminais Shellby 915
- Arquivos: Wtechlpt.dll, TVT100.dll, TVT100.ini, Shellby.dll e Shellby\_Base1.ini

### **Aplicação RS485 com uso da Wtechlpt.dll versão 4 com outros modelos de terminais.**

### **IMPORTANTE:**

#### **Dll\_Close();**

Para evitar erro ao finalizar uma aplicação, o desenvolvimento do aplicativo deve seguir a seguinte seqüência:

- Configlpt(Porta, Timeout);
- Aplicação executando realizando chamadas as rotinas da Dll's;
- Dll\_Close();

Na carga da aplicação deve-se chamar Configlpt para carregar as configurações das Dll's auxiliares, os parâmetros Porta e Timeout somente serão utilizados quando Comutadora Paralela, outros equipamentos utilizarão os parâmetros de configuração no arquivo de inicialização. Ao finalizar a aplicação deve-se chamar a rotina **Dll\_Close()**; para descarregar corretamente as Dll's auxiliares evitando mensagens de erros por violação de acesso, principalmente se estiver utilizando equipamentos Gradual com comunicação TCP-IP.

### **Funções da Dll Wtechlpt**

Este material descreve o funcionamento da Dll em ambiente Windows. Para o uso da Wtechlpt.dll, é necessário que a linguagem de programação do sistema permita a utilização de Dll's, o que ocorre na grande maioria dos casos.

#### **●ConfigLpt(Endereço, Timeout)**

**Retorno:** Tipo boolean, *true* se conseguiu estabelecer a comunicação, e *false* se ocorreu algum erro no processo.

Iniciaiza a Dll para a comunicação com os Terminais. Os endereços mais comuns para as portas paralelas estão na tabela a seguir. Caso eles não sejam válidos para a máquina onde esteja a comutadora, será necessário verificar no "Setup" da máquina qual o endereço válido.

Porta	Endereço (em hexadecimal)
-----	
LPT1	378h
LPT2	278h

*Timeout:* A comutadora tem um processador interno que é mais lento que o processador do PC que a controla, por isso é necessário informar um valor de timeout para que o PC espere um tempo entre o envio de dois comandos para a mesma. Quanto mais rápido o PC maior deve ser esse valor. Se for especificado um

valor muito baixo pode haver perda de comandos enviados. Por outro lado se for especificado um valor muito alto, a performance do sistema pode ficar comprometida. Portanto para cada PC deve ser especificado um valor adequado para o timeout.

#### ●Dll\_Close()

A versão 4.x.x.x da Wtechlpt.dll possui uma varredura que garante a compatibilidade das diferentes interfaces com o sistema RS485 com uso da Comutadora Paralela. Para o sistema finalizar corretamente e sem mensagens de violação de acesso, deve-se interromper qualquer chamada a suas rotinas, normalmente Dll\_Get(), Dll\_Status() e/ou se houver dados para as interfaces LPT e COM do terminal, Dll\_Print() e Dll\_Serial(). Uma vez interrompida a comunicação com a DLL deve-se realizar a chamada a Dll\_Close() antes de encerrar o Sistema.

#### ●Dll\_PosCur(Terminal, Lin, Col)

Posiciona o cursor do Terminal na posição indicada por Lin, Col.

#### ●Dll\_Clear(Terminal)

Apaga o display do Terminal.

#### ●Dll\_Echo(Terminal, Dado)

Escreve um caracter no Display.

#### ●Dll\_Display(Terminal, Dado)

Escreve uma string no Display.

#### ●Dll\_Get(Terminal)

Recebe um caracter do buffer:

**Retorno:** Chr(0) - Caso não haja dados no buffer.  
Dado - Caso haja dado(s) no buffer, retorna o primeiro caracter do buffer.

O buffer representa a área de armazenamento de toda informação que entra no Terminal. Seja ela digitada no teclado, lida pelo leitor de código de barras ou pela porta serial. Portanto os dados são sempre lidos do buffer do Terminal, simplificando assim o desenvolvimento, pois não há a necessidade de escrever uma rotina em separado para cada dispositivo de entrada. O próprio Terminal já faz esse trabalho.

#### ●Dll\_Status(Terminal)

**Retorno:** Tipo Integer.

Retorna o Byte de status do Terminal:

bit 0 e 1: 00 teclado qwerty  
          01 teclado numérico  
          10 teclado 16  
          11 reservado  
bit 2: reservado  
bit 3: cts 1 off-line  
          0 on-line  
bit 4: reservado  
bit 5: busy 1 off-line  
          0 on-line  
bit 6: reservado  
bit 7: reservado

OBS: Pedido de status usando TVT100.dll deve desconsiderar o CTS e BUSY, estarão sempre on-line e o tipo de teclado será devolvido conforme configuração do arquivo INI.

### **Escrevendo nas portas paralela e serial do Terminal:**

Para enviarmos dados para as interfaces dos Terminais, deveremos fazê-lo caracter a caracter, efetuando uma checagem de status do envio de cada caracter de modo a garantirmos o envio correto ao Terminal. O programa deverá efetuar o sucesso da execução do comando para cada caracter escrito na porta do Terminal.

Retorno: 1 - Caracter escrito com sucesso  
          0 - Falha na escrita do caracter. Ele deve ser escrito novamente.

#### **•Dll\_Serial(Terminal, Dado)**

Escreve um caracter na interface serial do terminal.

**Retorno:** Tipo Integer.

#### **•Dll\_Print(Terminal, Dado)**

Escreve um caracter na interface paralela do terminal.

**Retorno:** Tipo Integer.

#### **•Dll\_Aciona(Terminal, Dado)**

Escreve um caracter na interface de acionamento do terminal.

**Retorno:** Tipo Integer.

Exemplo: Caso o Terminal possua leds de acionamento, podemos acioná-los enviando dados para a “porta de impressão“ de acionamento, é como se fosse uma impressora. O byte que escrevemos na porta irá então ligar ou desligar os leds correspondentes aos bits setados.

Envia-se os seguintes valores para acionar cada um dos leds:

‘0’ chr(48) – liga os dois leds.  
‘1’ chr(49) – liga o led 2 e desliga o led 1.  
‘2’ chr(50) – liga o led 1 e desliga o led 2.  
‘3’ chr(51) – desliga os dois leds.

OBS: Por questões de meio físico não é possível tornar o Terminal TCP com protocolo Gradual, totalmente compatível com RS485 mantendo a mesma velocidade quanto ao uso da Dll\_Serial e Dll\_Print. Quando usado para enviar um número grande de bytes pela SERIAL ou LPT do terminal, indicamos estar utilizando com o protocolo VT100.

### ●Dll\_Acesso(Comando)

**Retorno:** Tipo Integer.

Serve como compatibilidade para sistemas desenvolvidos com o Device Driver (grad20.sys). A função Dll\_Acesso recebe como parâmetro a mesma string de comando que é enviada para o Device driver, traduz e envia o comando.

Ex: Dll\_Acesso(Chr(254) + '00L')  
Esse comando apaga o visor do Terminal ID 00.

Assim é possível aproveitar o código já escrito hoje. A diferença fica para as funções de inicialização e finalização da Dll que são diferentes das funções de abertura e fechamento do Device driver.

As strings de comando para operação da comutadora possuem a seguinte sintaxe:

Chr(254) + NN + C + Parâmetros

Onde:

NN - Número do Terminal (entre "00" e "31")

C - Comando para a comutadora

- Posicionamento do cursor: **Chr(254) + NN + "C" + LPP**

NN - Número do Terminal (entre "00" e "31")

"C" - Letra "C" maiúscula

L - Linha do cursor ("0" ou "1")

PP - Coluna do cursor (entre "00" e "39")

- Apaga o Display: **Chr(254) + NN + "L"**

NN - Número do Terminal (entre "00" e "31")

"L" - Letra "L" maiúscula

- Escreve no Display: **Chr(254) + NN + "D" + XXXX**

NN - Número do Terminal (entre "00" e "31")

"D" - Letra "D" maiúscula

XXXX - String de dados a ser escrita (Ex: "Testando")

- Recebendo um caracter do buffer: **Chr(254) + NN + "K"**

NN - Número do Terminal (entre "00" e "31")

"K" - Letra "K" maiúscula

**Retorno:** Chr(0) - Caso não haja dados no buffer

Dado - Caso haja dado(s) no buffer, retorna o primeiro caracter do buffer.

O buffer representa a área de armazenamento de toda informação que entra no Terminal. Seja ela digitada no teclado, lida pelo leitor de código de barras ou pela porta serial. Portanto os dados são sempre lidos do buffer do Terminal, simplificando assim o desenvolvimento, pois não há a necessidade de escrever uma rotina em separado para cada dispositivo de entrada. O próprio Terminal já faz esse trabalho.

- Leitura do status: **Chr(254) + NN + "U"**

NN - Número do Terminal (entre "00" e "31")

"U" - Letra "U" maiúscula

Retorno: Byte de status do Terminal

### ***Declaração das funções em Delphi:***

- function ConfigLpt(Endereco, Timeout: Word): Boolean; stdcall; external 'WTechLpt.dll';
- procedure Dll\_Close(); stdcall; external 'WTechLpt.dll';
- procedure Dll\_PosCur(Terminal, Lin, Col: Byte); stdcall; external 'WTechLpt.dll';
- procedure Dll\_Clear(Terminal: Byte); stdcall; external 'WTechLpt.dll';
- procedure Dll\_Echo(Terminal: Byte; Dado: Char); stdcall; external 'WTechLpt.dll';
- procedure Dll\_Display(Terminal: Byte; Dado: string); stdcall; external 'WTechLpt.dll';
- procedure Dll\_Aciona(Terminal, Dado: Byte); stdcall; external 'WTechLpt.dll';
- function Dll\_Get(Terminal: Byte): Char; stdcall; external 'WTechLpt.dll';
- function Dll\_Status(Terminal: Byte): Byte; stdcall; external 'WTechLpt.dll';
- function Dll\_Print(Terminal: Byte; Dado: Char): Byte; stdcall; external 'WTechLpt.dll';
- function Dll\_Serial(Terminal: Byte; Dado: Char): Byte; stdcall; external 'WTechLpt.dll';

### ***Declaração das funções em VB:***

- Declare Function ConfigLpt Lib "WTechLpt.dll" (ByVal Endereco As Integer, ByVal Timeout As Integer) As Boolean
- Declare Sub Dll\_Close Lib "WTechLpt.dll" ()
- Declare Sub Dll\_Display Lib "WTechLpt.dll" (ByVal Terminal As Byte, ByVal Dado As String)
- Declare Function Dll\_Get Lib "WTechLpt.dll" (ByVal Terminal As Byte) As Byte
- Declare Function Dll\_Status Lib "WTechLpt.dll" (ByVal Terminal As Byte) As Byte
- Declare Sub Dll\_Aciona Lib "WTechLpt.dll" (ByVal Terminal As Byte, ByVal Dado As Byte)
- Declare Sub Dll\_Clear Lib "WTechLpt.dll" (ByVal Terminal As Byte)
- Declare Sub Dll\_PosCur Lib "WTechLpt.dll" (ByVal Terminal As Byte, ByVal Lin As Byte, ByVal Col As Byte)
- Declare Function Dll\_Acesso Lib "WTechLpt.dll" (ByVal Cmd As String) As Integer

### ***Declaração das funções em C/C++:***

- typedef bool \_stdcall (\*PtrConfigLpt)(byte Endereco, byte Timeout);
- typedef void \_stdcall (\*PtrDll\_Close);
- typedef void \_stdcall (\*PtrDll\_Echo)(byte Terminal, char Dado);
- typedef void \_stdcall (\*PtrDll\_Display)(byte Terminal, char\* Dado);
- typedef char \_stdcall (\*PtrDll\_Get)(byte Terminal);
- typedef byte \_stdcall (\*PtrDll\_Acesso)(char\* Comando);
- typedef byte \_stdcall (\*PtrDll\_Status)(byte Terminal);
- typedef byte \_stdcall (\*PtrDll\_Print)(byte Terminal, char Dado);
- typedef byte \_stdcall (\*PtrDll\_Serial)(byte Terminal, char Dado);
- typedef void \_stdcall (\*PtrDll\_Aciona)(byte Terminal, byte Dado);
- typedef void \_stdcall (\*PtrDll\_Clear)(byte Terminal);
- typedef void \_stdcall (\*PtrDll\_PosCur)(byte Terminal, byte Linha, byte Coluna);
- typedef void \_stdcall (\*PtrDll\_Close)(void);

### ***Declaração das funções em Xharbour:***

- DECLARE Integer ConfigLpt in Wtechlpt.dll Integer Porta, Integer Timeout
- DECLARE FUNCTION Dll\_Close in Wtechlpt.dll
- DECLARE FUNCTION Dll\_Clear in Wtechlpt.dll Integer Terminal
- DECLARE FUNCTION Dll\_PosCur in Wtechlpt.dll Integer Terminal, Integer Linha, Integer Coluna
- DECLARE FUNCTION Dll\_Display in Wtechlpt.dll Integer Terminal, String Dados
- DECLARE FUNCTION Dll\_Acesso in Wtechlpt.dll String Dados
- DECLARE Integer Dll\_Get in Wtechlpt.dll Integer Terminal
- DECLARE Integer Dll\_Print in Wtechlpt.dll Integer Terminal, String Dados
- DECLARE Integer Dll\_Serial in Wtechlpt.dll Integer Terminal, String Dados

### **Dúvidas ou maiores informações?**

Entre em contato com o departamento de Suporte:

- ✓Telefone (11) 4538-7733
- ✓E-mail: [suporte@gradual.com.br](mailto:suporte@gradual.com.br)
- ✓Atendimento On-Line: [www.gradual.com.br](http://www.gradual.com.br)