Base RF para
Terminais Shellby 915

Índice

Apresentação

Este manual tem por objetivo informar as características e funcionamento da Base RF, a forma de comunicação com o computador e a compatibilidade de sistemas.

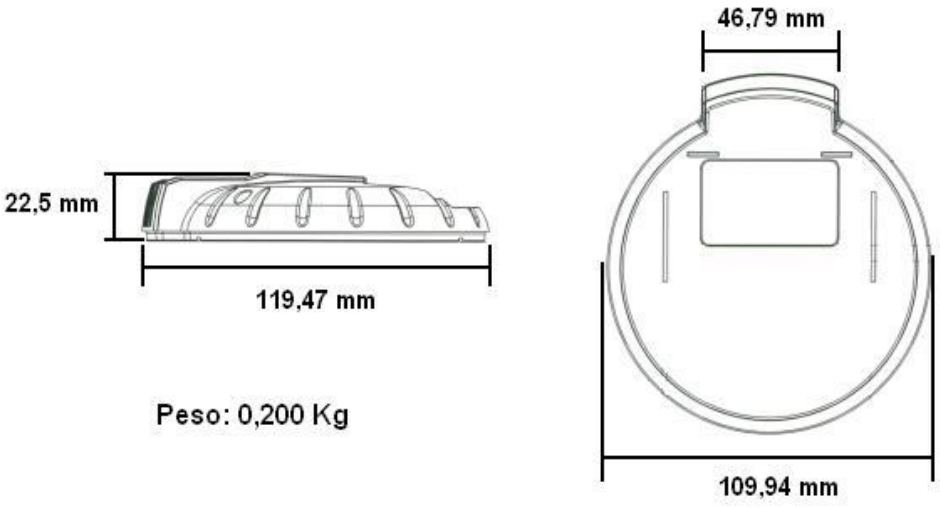
A Base RF é um equipamento com comunicação através de rádio freqüência na faixa de 915 MHz, sendo responsável pelo envio e recebimento de dados dos Terminais Shellby 915, bem como o gerenciamento de pacotes da rede RF.

Características Elétricas

A Base RF possui as seguintes características:

- Suporte para até 16 Terminais Shellby 915;
- Velocidade de comunicação: 57600 bps;
- Interface Serial RS-232;
- Alimentação: 5Vdc

Características Mecânicas



Configuração da Base RF

A configuração dos parâmetros de funcionamento da Base RF é feito através da Dll fornecida pela Gradual Tecnologia. Esta Dll trabalha em conjunto com o arquivo “Shellby_Base1.ini”, onde são informados os campos de configuração:

➢*Número de Terminais*

Valores válidos: 0 a 15.

Função: Configura o número de terminais gerenciados pela Base RF.

➢*Número de Base*

Valores válidos: 64 a 96.

Função: Determina o endereço lógico da Base na rede RF.

➢*Canal*

Valores válidos: 1 a 62.

Função: Determina a freqüência de operação na rede RF. Cada canal possui uma freqüência, deste modo é possível configurar um canal para cada rede.

➢*Timeout*

Valores válidos: 1 a 10.

Função: Determina o tempo de espera para recebimento de um pacote dos terminais. Recomenda-se fixar o valor de Timeout para 3.

➢*Retry*

Valores válidos: 1 a 10.

Função: Determina o número de tentativas para re-transmissão de um pacote na rede RF. Recomenda-se fixar o valor de Timeout para 4.

➢*Discover*

Valores válidos: 1 a 100.

Função: Determina o tempo entre envio de comando de busca por novos terminais na rede RF. Este valor é multiplicado por 50 ms.

➢*Sync*

Valores válidos: 1 a 50.

Função: Determina o tempo entre o recebimento de pacotes dos terminais na rede RF. Este valor é multiplicado por 50 ms.

➢*Dispatch*

Valores válidos: 1 a 50.

Função: Determina o tempo entre o envio de pacotes para os terminais na rede RF. Este valor é multiplicado por 50 ms.

IMPORTANTE: Os valores de Discover, Sync e Dispatch influenciam diretamente no desempenho da rede. Valores muito altos podem deixar a rede mais lenta, e valores muito baixos provocam um fluxo desnecessário de pacotes na rede RF.

Conexão ao Computador

A Base RF para os Terminais Shellby 915 poderá ser conectada ao computador das seguintes formas:

- ✓**Serial RS-232:** diretamente a uma porta existente no computador;
- ✓**USB:** através de um Conversor Serial USB;

✓**Ethernet:** através de um Conversor Ethernet RS-232.

A Dll Wtechlpt-Shellby dá suporte para até 8 Bases RF instaladas ao mesmo tempo. Isto permite a configuração de sub-redes e/ou aumento da área de cobertura do link de rádio para funcionamento dos terminais.

Os terminais possuem uma configuração de operação em multi-base que permite que os mesmos procurem um novo link de rádio quando ficar fora do alcance da base original. Esta migração de base é totalmente transparente a aplicação, sendo gerenciada integralmente pela Dll.

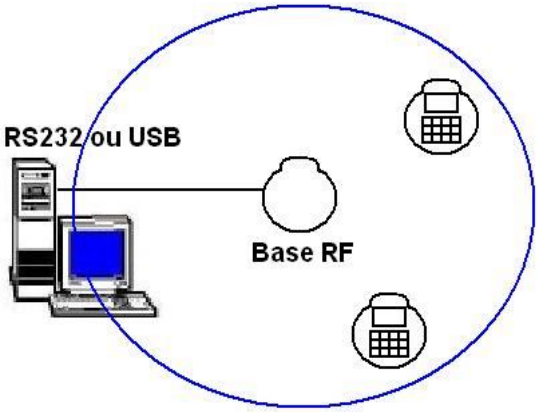


Figura 1: *Base RF conectada diretamente na porta serial do PC*

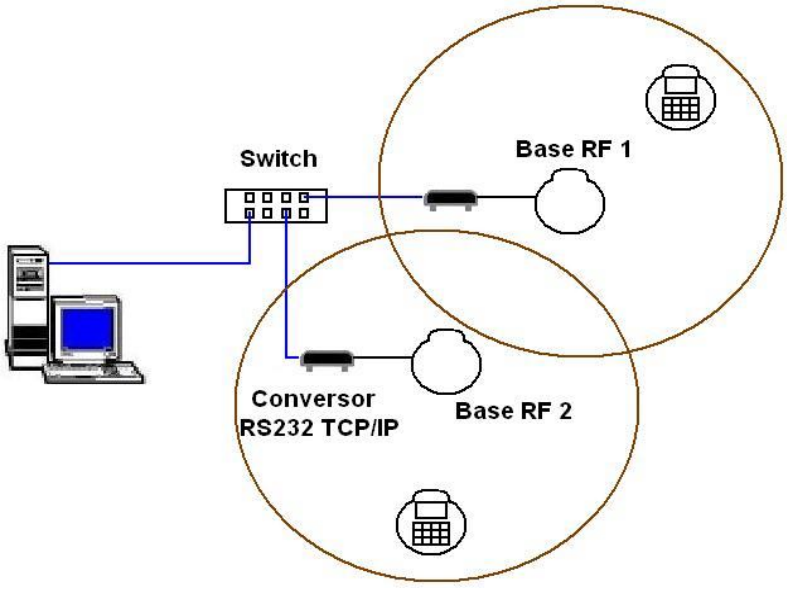


Figura 2*: Base RF conectada a um Conversor RS-232 TCP/IP*

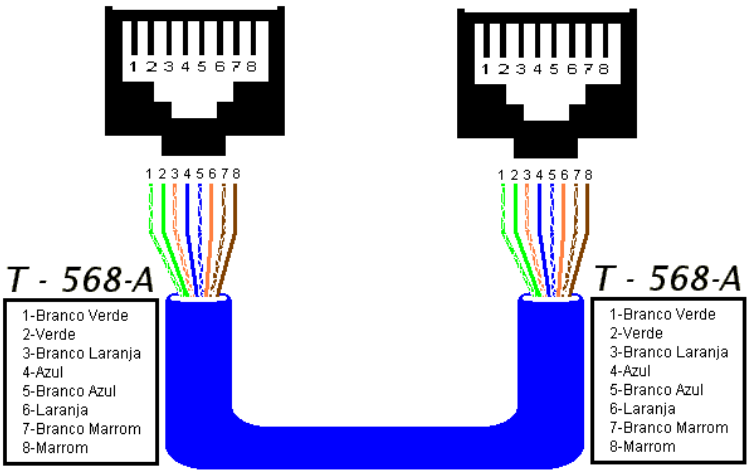


Figura 3: *Desenho ilustrativo sobre a montagem de um cabo UTP (T-568A).*

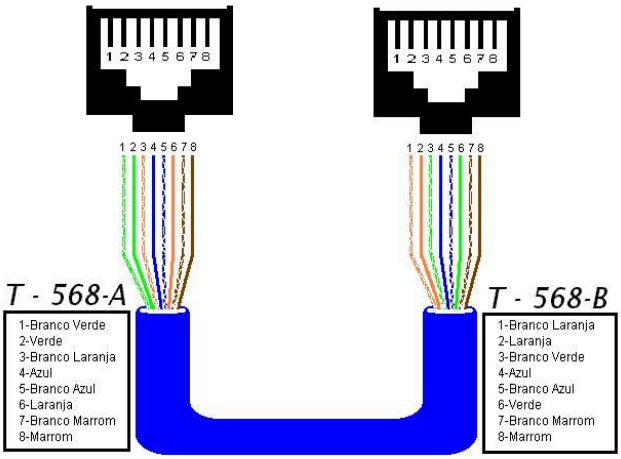


Figura 4: *Desenho ilustrativo sobre a montagem de um "cabo Cross".*

Compatibilidade de Sistemas

Para comunicação com a Base RF e Terminais Shellby 915, é necessário um sistema gerenciador de terminais que deverá processar as informações de acordo com os dados digitados nos terminais.

Nos próximos tópicos serão mostradas situações onde já se utiliza um sistema para comunicação com algum equipamento da Gradual e suas compatibilidades com os Terminais Shellby:

- Terminal RS485 com Comutadora (Paralela, Serial ou Ethernet)
- Terminal TCP/IP

1. Sistema com Comutadora Paralela

Os sistemas desenvolvidos para Comutadora Paralela utilizando a Dll Wtechlpt podem ser usados para comunicação dos Terminais Shellby 915 com total compatibilidade. Pela comunicação ser baseada na Wtechlpt, basta trocar a Dll e configurar um arquivo de inicialização para que o sistema realize a comunicação com a Base RF.

A Dll para comunicação dos Terminais Shellby possui as mesmas funções utilizadas anteriormente, tais como Dll_Acesso, Dll_Clear, Dll_Display, Dll_Get, entre outras. Portanto não haverá alteração de código no sistema.

Veja no tópico “Manual – Dll Wtechlpt” para a descrição completa das funções e nos apêndices A, B, C e D as declarações da Dll para uso no software.

Figura 5: *Sistema c/ comutadora paralela*

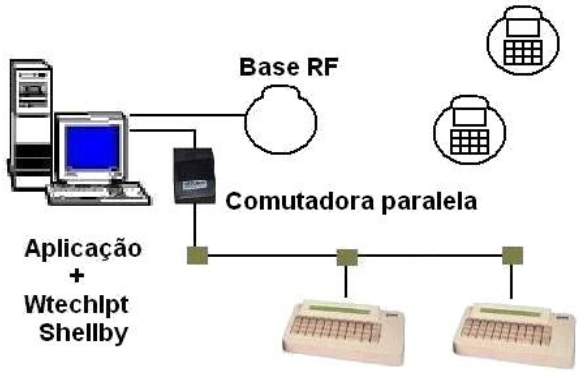


Figura 6: *Sistema c/ Terminais Shellby*

2. Sistema com Comutadora Serial

Para que os sistemas desenvolvidos para Comutadora Serial comuniquem com os Terminais Shellby, é necessário utilizar o aplicativo RS232 Shellby. Este software funciona como uma ponte entre o sistema gerenciador de terminais e a Base RF.

Desta forma, o sistema não sofre alterações e inicialmente já realiza a comunicação com os Terminais Shellby.

Veja no manual do aplicativo RS232 Shellby para uma descrição completa de sua configuração.

O sistema que foi projeto para Comutadora Serial envia e recebe os dados na porta COM do PC, então é necessário que os dados realmente saiam da porta serial e sejam direcionados para outro lugar. O RS232 Shellby funciona através de outra porta serial, então é necessário que o PC tenha duas portas seriais livres para que o esquema acima funcione (Figura 7). Para os casos em que o PC tenha somente uma porta COM, porém tenha portas USB, pode-se alternativamente instalar um Conversor USB Serial (Figura 8).

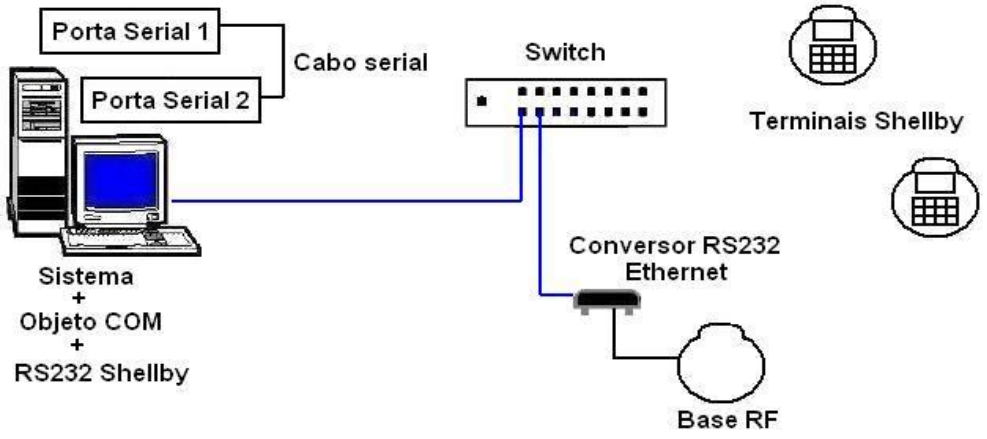


Figura 7: *Conexão Serial para Serial*

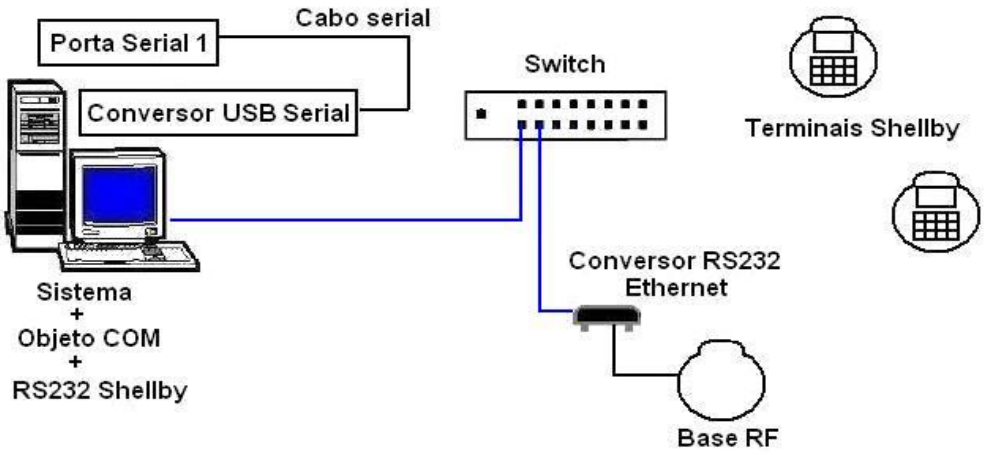


Figura 8: *Conexão usando um Conversor USB-Serial*

3. Sistema com Comutadora Ethernet

Para que os sistemas desenvolvidos para Comutadora Ethernet comuniquem com os Terminais Shellby, é necessário utilizar o aplicativo CEthernet Shellby. Este software funciona como uma ponte entre o sistema gerenciador de terminais e a Base RF.

Desta forma, o sistema não sofre alterações e inicialmente já realiza a comunicação com os Terminais Shellby. O aplicativo CEthernet Shellby pode funcionar como Client ou Server, semelhante à Comutadora Ethernet. Desta forma, se o sistema funcionar como Server (aguarda conexão numa porta), então o software deve ser configurado como Client. Caso o sistema funciona como Client (faz um pedido de conexão a um endereço IP e porta), o software deve ser configurado como Server.

Veja no manual do aplicativo CEthernet Shellby para uma descrição completa de sua configuração.

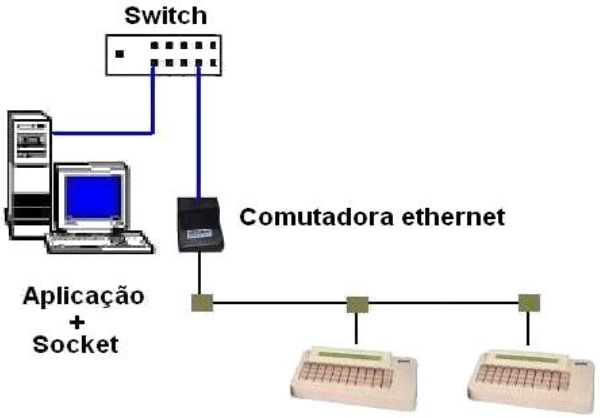


Figura 9:
Aplicação socket e Comutadora Ethernet

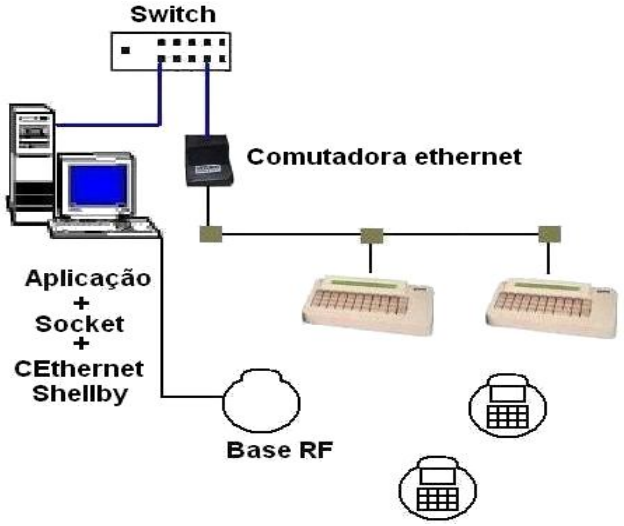


Figura 10:
Aplicação socket e software CEthernet Shellby

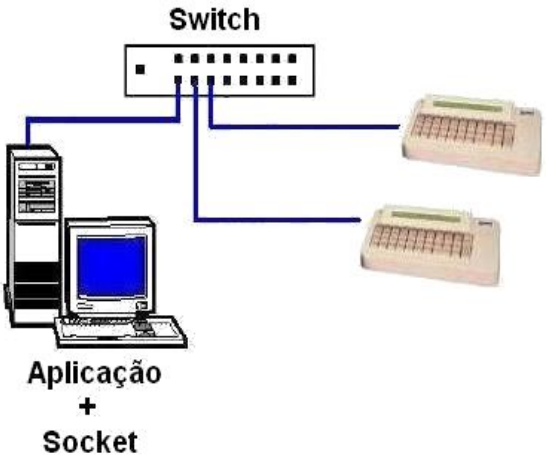
4. Sistema com Terminais TCP/IP usando comunicação Socket

Os sistemas desenvolvidos para Terminais TCP/IP (protocolo VT-100 ou Gradual) usando comunicação socket (sem uso de Dll), podem ser utilizados com os Terminais Shellby com total compatibilidade e sem alteração no código fonte. Para isso, é necessário utilizar o aplicativo TCPShellby. Este software funciona como uma ponte entre o sistema gerenciador de terminais e a Base RF.

Desta forma, o sistema não sofre alterações e inicialmente já realiza a comunicação com os Terminais Shellby. O aplicativo TCPShellby pode funcionar como Client ou Server, semelhante ao Terminal TCP/IP. Desta forma, se o sistema funcionar como Server (aguarda conexão numa porta), então o software deve ser configurado como Client. Caso o sistema funciona como Client (faz um pedido de conexão a um endereço IP e porta), o software deve ser configurado como Server.

Deste modo, é possível manter a rede de Terminais TCP/IP, e adicionar Terminais Shellby.

Veja no manual do aplicativo TCPShellby para uma descrição completa de sua configuração e funcionamento.



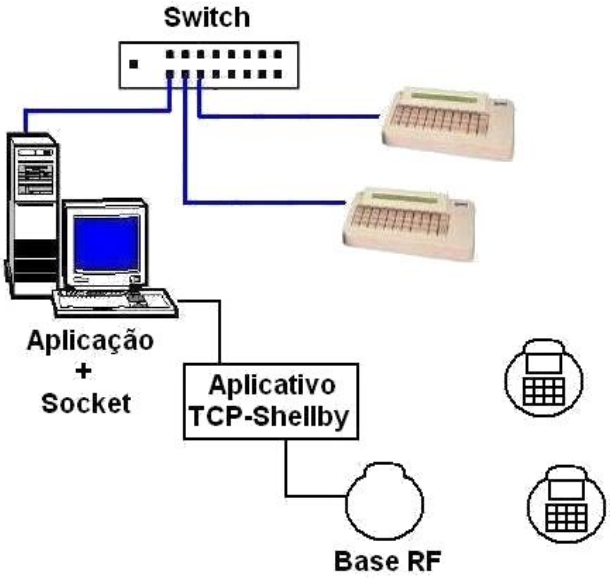


Figura 11:
Aplicação socket e Rede TCP/IP

Figura 12:
Aplicação socket e software TCP-Shellby

Adequação Visual no Display

Pelo fato dos Terminais Shellby 915 possuirem display de 7x20, as mensagens enviadas através dos sistemas desenvolvidos para Microterminais de 44 teclas com display de 2x40 serão particionadas em linhas de 20 colunas.

Então o sistema terá compatibilidade total de funcionamento, porém em alguns casos terão que ser feitos adequação visual das mensagens.

Caso o sistema funcione com Microterminais de 16 teclas, não há necessidade de alteração das mensagens no display, visto que o mesmo possui 2x16.

Manual - Dll Wtechlpt

Este material descreve o funcionamento da Dll para operação da Base RF com os Terminais Shellby 915 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.

Inicialmente a 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
- ✓ Terminais TCP/IP (protocolo VT-100 ou Gradual)
- ✓ Terminais RS-485 (comutadora paralela, serial e/ou ethernet)
- ✓ Terminais RS-232

Internamente, a Dll possui objetos para comunicação socket, COM e LPT. O que o sistema deve fazer são chamadas as funções contidas na Dll para manipulação dos terminais. Então, para limpar o display usa-se Dll_Clear, mensagens no display Dll_Display, e assim por diante. Veja a descrição completa abaixo:

- **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 a Base RF. Caso o sistema trate apenas Terminais Shellby, os parâmetros Endereço e Timeout serão ignorados, caso contrário serão utilizados para comunicação com a Comutadora Paralela.

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_PosCur(Terminal, Lin, Col)**

Posiciona o cursor do Terminal na posição indicada por Lin, Col. Onde Lin pode variar de 0 até 7 e Col pode variar de 0 até 19.

- **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)**

Recebendo 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.

  Pedido de Status para o Terminal. Retorna zero se estiver desligado (off-line), e um valor diferente de zero para ligado (on-line).

- **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 quase na totalidade 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. Todo o resto é 100% compatível.

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

Apêndice A: Declaração das funções em Delphi

```
function ConfigLpt(Endereco, Timeout: Word): Boolean; 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';
```

Apêndice B: Declaração das funções em VB

```
Declare Function ConfigLpt Lib "WTechLpt.dll" (ByVal Endereco As Integer, ByVal Timeout As Integer) As Boolean
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

Apêndice C: Declaração das funções em C/C++

```
typedef  bool _stdcall (*PtrConfigLpt)(byte Endereco, byte Timeout);
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);
```

Apêndice D: Declaração das funções em Xharbour

- DECLARE Integer ConfigLpt in Wtechlpt.dll Integer Porta, Integer Timeout
- 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

Termo de Garantia

A **Gradual Tecnologia Ltda**., garante a qualidade do produto adquirido, pelo prazo de 01 (hum) ano a contar da data da compra descrita na Nota Fiscal.

Este Termo garante contra defeitos de fabricação e/ou material, comprometendo-se a vendedora a reparar o produto ou substituí-lo por outro da mesma espécie, ou, ainda, por outro de igual função. O serviço de reparação ou a substituição será executado, exclusivamente, nas dependências da **Gradual Tecnologia Ltda**.

Será de responsabilidade do comprador, o abaixo descrito:

- Apresentar a Nota Fiscal de venda;
- Anexar à N.F., um descritivo do defeito apresentado;
- Enviar o produto devidamente embalado;
- Os custos de transporte, ida e volta.

Esta garantia perde a eficácia, nos seguintes casos:

- Utilizar o produto fora das especificações;
- Acidentes, mau uso e desgastes de partes consumíveis;
- Sofrer qualquer alteração, modificação ou adaptação, sem o consentimento expresso da Gradual Tecnologia Ltda;
- Assistência Técnica e/ou manutenção, através de terceiros não autorizados pela Gradual Tecnologia Ltda;
- Alteração ou violação do n.º de série.

Equipamento:_____
No. de Série:_____
Nota Fiscal:_____