Interface Comutadora Paralela para Microterminais RS-485

Gradual tecnologia Ltda. Manual Comutadora Paralela Novembro 2003.

Índice

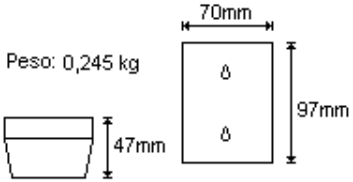
Características do Aparelho:

Comutadora Paralela

Descrição:

- Interface para conexão de Microterminais RS-485 ao PC.
- Até 32 pontos de Microterminais.
- Até 1.000 metros de cabo para montagem da rede.
- A Comutadora Paralela tem protocolo Gradual para programação em qualquer ambiente.

Características mecânicas



Funcionamento

A Interface comutadora paralela permite o tratamento de um barramento RS-485 com protocolo Gradual.

Ela se comporta como um periférico escravo, que responderá aos comandos do programa que trata a rede de Microterminais no Microcomputador.

Para realizar essa comunicação, a Gradual desenvolveu ferramentas que fazem o tratamento da porta e dos dados, que é feito através da chamada doas funções.

Estas ferramentas são:

- Grad20.sys : Device-Driver de 16 bits
  - Funciona em DOS ou Windows 95 e 98, permite programas em DOS acessarem os Microterminais.

- WtechLpt: Dll de 32 bits
  - Funciona em Windows 95, 98, ME, NT e XP, e permite que apenas aplicativos windows tratem os Microterminais.

Manual Grad20.sys

Device-Driver para Microterminais RS-485
Versão 2.0

Este material descreve o funcionamento do Device-Driver para tratamento de Microterminais GRADUAL RS-485, em sistema operacional baseados em DOS. Este considera que o tratamento multiterminal será efetuado pelo usuário, funcionando assim apenas como um dispatcher para a comutadora.

A Instalação do programa deverá ser feita adicionando-se as seguintes informações no arquivo CONFIG.SYS:

DEVICE=GRAD20.SYS

Quando da instalação inicial, o driver ira considerar valores default de paralela e timeout de envio, LPT1 e timeout de 130. Estes valores poderão ser modificados através dos comandos que detalhamos a seguir. Esta instalação deixará o arquivo GRADEVIC disponível para acesso aos Microterminais.

ACESSOS AOS MICROTERMINAIS

Os comandos de controle são efetuados através de escritas no arquivo instalado no sistema através do config.sys e a linha de comando deverão conter o caracter de escape 254 em decimal(0FE em hexa) como inicializador. Simbolizaremos o 254 por 'þ'.

Os comandos possuem a seguinte sintaxe:

þNNCLPPxxxxxxxx

onde: þ - Caracter de escape
NN - Número do Terminal para envio da mensagem/comando
C - Tipo de Comando
L - Número da linha (caso posicionamento do cursor)
PP - Número da coluna (caso posicionamento do cursor)
xx - Mensagem (envio para o display)

- POSICIONA CURSOR

þNNCLPP

NN - Número do Terminal ('00' a '31')
C - Caracter 'C'
L - Linha: L='0' 1a. Linha.
L='1' 2a. Linha.
PP - Coluna: PP='00' 1a. coluna
PP='39' ultima coluna.

- APAGA DISPLAY

þNNL

NN - Número do Terminal ('00' a '31')
L - Caracter 'L'

- ESCREVE NO DISPLAY

þNNDxxxxx

NN - Número do Terminal ('00' a '31')
D - Caracter 'D'
xx - String de dados a ser escrita

- SELECIONA LPT

þPN

P - Caracter 'P'
N - Número da LPT onde da comutadora.
Valor default '1'.

- DEFINE VALOR DE TIMEOUT

þTVVVVV

T - Caracter 'T'
VVVVV - Valor do Timeout de escrita no display
Valor default '00130'.

Obs: O valor de timeout e dependente de software, seu valor deverá ser testado empiricamente até se encontrar o menor valor em que o Microterminal funcione sem perder caracter nem que o display fique descontrolado. Valores para referência

486DX2-66 = 050
485DX4-100= 100
P100 = 250

Estes valores são apenas referências iniciais, para facilitar a inicialização. Tente sempre encontrar o melhor valor para cada maquina instalada.

UTILIZANDO AS INTERFACES DOS MICROTERMINAIS

Para enviarmos dados para as interfaces dos Microterminais, deveremos faze-lo caracter a caracter, efetuando uma checagem de status do envio de cada caracter de modo a garantirmos o envio correto ao terminal.

Após o envio do comando inicial o driver irá considerar todos os bytes enviados a ele para a interface daquele terminal. Porém a cada envio o programa deverá efetuar uma leitura do driver e testar o seu retorno:

Chr(0) - dado não escrito
Chr(1) - dado enviado

- ESCREVE NA PARALELA

þNNIx

NN - Número do Terminal ('00' a '31')
I - Caracter 'I'
x - Caracter a ser enviado para a impressora

- ESCREVE NA SERIAL

þNNRx

NN - Número do Terminal ('00' a '31')
R - Caracter 'R'
x - Caracter a ser enviado para a serial

RECEBENDO DADOS DOS MICROTERMINAIS

Existem dois comandos de leitura de informações dos Microterminais.

O uso dos dois comandos deverá seguir a seqüência de escrita do comando, seguida de uma leitura do arquivo para receber o retorno.

- LEITURA DE STATUS

þNNU

NN - Número do Terminal ('00' a '31')
U - Caracter 'U'
Retorno: Byte de status do Microterminal
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

Em caso de retorno do Chr(0), a comutadora gerou o timeout e o terminal não está instalado na rede.

- LEITURA DE TECLADO

þNNK

NN - Número do Terminal ('00' a '31')
K - Caracter 'K'
Retorno: Chr(0) - em caso da não existência de tecla.
outros - caracter recebido do terminal.

EXEMPLO DE PROGRAMA DE ACESSO AO DEVICE-DRIVER

```
/*
** Acesso aos Microterminais através de Device-Driver Arquivo.
*/
#include <string.h>
#include <fcntl.h>
#include <dos.h>
#include <io.h>
#include <stdio.h>
#include <conio.h>

#define CR 0x0d
#define DEL 0x7f

int file,ti;
unsigned char s[500],r[150],v[10];
unsigned char i,c;

void main( void ){
      file=_open( "gradevic", O_RDWR|O_BINARY ); /* Abre o
arquivo */
      strcpy( s, "þT00100");  /* Define o valor de timeout em 100 */
      _write( file, &s, strlen(s) );
      clrscr();
      if( file!=-1 ){
        while( !kbhit() ){
            for( i=0; i<32; i++ ){
                  sprintf( s, "þ%02iK", i ); /* Pede teclado */
                  _write( file, s, strlen(s) );
                  _read( file, s, 2 );  /* Lê o retorno */
                  switch( s[0] ){
                        /* Não tem tecla pra tratar */
                        case 0   : break;
                        /* Recebeu o ENTER echo 'cr' */
                        case CR  : sprintf( s, "þ%02iDcr", i );
                                   _write( file, s, strlen(s) );
                                   break;
                        /* Recebeu DEL apaga o display */
                        case DEL : sprintf( s, "þ%02iL", i );
                                   _write( file, s, strlen(s) );
                                   break;
                        /* Vai enviar string para Serial */
                        case 'R' :
                              sprintf( s, "þ%02iR", i );
                              _write( file, s, strlen(s) );
                              strcpy( s, "Teste de impressao
Serial\x0d\x0a" );
                              c=0;
                              while( s[c] ){
                                    _write( file, &s[c], 1 );
                                    _read( file, v, 2 );
                                    if( v[0] ) c++;
                              }
                              break;
                        /* Vai enviar string para Paralela */
                        case 'P' :
                              strcpy( s, "þ%02iI", i );
                              _write( file, s, strlen(s) );
                              strcpy( s, "Teste de impressao
Paralela\x0d\x0a" );
                              c=0;
                              while( s[c] ){
                                    _write( file, &s[c], 1 );
                                    _read( file, v, 2 );
                                    if( v[0] ) c++;
                              }
                              break;
                        /* Envia uma string para o display */
                        case 'L' :
                              for( i=0; i<32; i++ ){
                                    sprintf( s, "þ00Lþ00DTeste da nova
versao do Device-Driver.%02i",i );
```

```
                        _write( file, &s, strlen(s) );
                    }
                    break;
                /* Pede o status */
                case 'U' : sprintf( s, "þ%02iU");
                    _write( file, s, strlen(s) );
                    _read( file, v, 2 );
                    sprintf(s, "þ%02iLþ%02iD", i);
                    if( v[0] ){
                      strcat( s, "Terminal não ligado" );
                    }
                    else{
                      if( v[0]&0x20 )
                        strcat( s, "LPT ocupada" );
                      else
                        strcat( s, "LPT livre" );
                      if( v[0]&0x08 )
                        strcat( s, "Serial ocupada" );
                      else
                        strcat( s, "Serial livre" );
                    }
                    break;
                /* Ecoa o caracter recebido no display */
                default  : _write( file, s, 1 );
                    break;
            }
        }

      }
      close( file );
    }
}
```


Manual WtechLpt

Dll de operação da Comutadora Paralela com os Microterminais RS-485 Versão 1.0

Este material descreve o funcionamento da "Dll" para operação da Comutadora Paralela com os Microterminais Wilbor Tech RS-485 em ambiente Windows. Este considera que o tratamento multiterminal será efetuado pelo usuário, funcionando assim apenas como um "dispatcher" para a Comutadora Paralela.

A Instalação do programa deverá ser feita via programa de instalação anexo com o pacote da Dll. Ver Apêndice D.

A Dll pode ser operada de duas maneiras:

1) Mantendo compatibilidade com os programas existentes que usam o Device Driver "Grad20.sys" (apenas com alterações mínimas no código fonte do programa) onde o envio e recebimento de comandos são feitos via strings (as mesmas do Device driver).
2) Realizando chamadas diretas as funções da Dll. Ver Apêndices A, B e C.

➢ ***Operação da Dll mantendo compatibilidade com o "Device Driver"***

A Compatibilidade com o Device Driver é feita na Dll com o Comando Dll_Acesso. Ele recebe como parâmetro à mesma string de comando que é enviada para o Device driver, traduz e envia o comando para a comutadora.

Ex: Dll_Acesso(Chr(254) + '00L')
Esse comando apaga o visor do Microterminal.
* Qualquer dúvida, sobre as strings de comando do Device driver, ver arquivo Grad20.doc

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 Microterminal (entre "00" e "31")
C - Comando para a comutadora

- Posicionamento do cursor: **Chr(254) + NN + "C" + LPP**
NN - Número do Microterminal (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 Microterminal (entre "00" e "31")
"L" - Letra "L" maiúscula

- Escreve no Display: **Chr(254) + NN + "D" + XXXX**
NN - Número do Microterminal (entre "00" e "31")
"D" - Letra "D" maiúscula
XXXX - String de dados a ser escrita (Ex: "Testando")

- Escreve na porta de Acionamento: **Chr(254) + NN + "A" + X**
NN - Número do Microterminal (entre "00" e "31")
"A" - Letra "A" maiúscula
X - Byte a ser escrito na porta de acionamento

Exemplo: Caso o Microterminal 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.

- Recebendo um caracter do buffer: **Chr(254) + NN + "K"**
NN - Número do Microterminal (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 Microterminal. 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 Microterminal, simplificando assim o desenvolvimento, pois não há a necessidade de escrever uma rotina em separado para cada dispositivo de entrada. O próprio Microterminal já faz esse trabalho.

- Leitura do status: **Chr(254) + NN + "U"**
NN - Número do Microterminal (entre "00" e "31")
"U" - Letra "U" maiúscula
Retorno: Byte de status do Microterminal
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

Escrevendo nas portas paralela e serial do Microterminal:

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

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

- Escrita na porta Paralela: **Chr(254) + NN + "I" + X**
NN - Número do Microterminal (entre "00" e "31")
"I" - Letra "I" maiúscula
X - Caracter a ser escrito na porta
Retorno: Chr(1) - Caracter escrito com sucesso
Chr(0) - Falha na escrita do caracter. Ele deve ser escrito novamente.

- Escrita na porta serial: **Chr(254) + NN + "R" + X**
NN - Número do Microterminal (entre "00" e "31")
"R" - Letra "R" maiúscula
X - Caracter a ser escrito na porta
Retorno: Chr(1) - Caracter escrito com sucesso
Chr(0) - Falha na escrita do caracter. Ele deve ser escrito novamente.

- Inicializa o uso da porta onde está a comutadora:
ConfigLpt(Endereço, Timeout)

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.

➢ ***Operação da Dll realizando chamadas diretas as funções***

- Inicializa o uso da porta onde está a comutadora:
ConfigLpt(Endereco, Timeout)

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.

- Posicionamento do cursor:

Dll_PosCur(Terminal, Lin, Col)

Posiciona o cursor do Microterminal na posição indicada por Lin, Col. Onde Lin pode variar de 0 até 1 e Col pode variar de 0 até 39.

- Apaga o Display:

Dll_Clear(Terminal)

- Escreve um caracter no Display:

Dll_Echo(Terminal, Dado)

- Escreve uma string no Display:

Dll_Display(Terminal, Dado)

- Recebendo um caracter do buffer:

Dll_Get(Terminal)

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 Microterminal. 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 Microterminal, simplificando assim o desenvolvimento, pois não há a necessidade de escrever uma rotina em separado para cada dispositivo de entrada. O próprio Microterminal já faz esse trabalho.

- Leitura do status:

Dll_Status(Terminal)

Retorno: Byte de status do Microterminal
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

Escrevendo nas portas paralela e serial do Microterminal:

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

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

- Escrita na porta Paralela:

Dll_Print(Terminal, Dado)

- Escrita na porta Serial:

Dll_Serial(Terminal, Dado)

- Escreve na porta de Acionamento:

Dll_Aciona(Terminal, Dado)

Exemplo: Caso o Microterminal 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.

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_Echo Lib "WTechLpt.dll" (ByVal Terminal As Byte, ByVal Dado As String)
- 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 String
- Declare Function Dll_Status Lib "WTechLpt.dll" (ByVal Terminal As Byte) As Byte
- Declare Function Dll_Print Lib "WTechLpt.dll" (ByVal Terminal As Byte, ByVal Dado As String) As Byte

- Declare Function Dll_Serial Lib "WTechLpt.dll" (ByVal Terminal As Byte, ByVal Dado As String) 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++

__declspec(dllimport) Bool ConfigLpt(word Endereco, Timeout); stdcall;
__declspec(dllimport) void Dll_PosCur(byte Terminal, Lin, Col); stdcall;
__declspec(dllimport) void Dll_Clear(byte Terminal); stdcall;
__declspec(dllimport) void Dll_Echo(byte Terminal, char Dado); stdcall;
__declspec(dllimport) void Dll_Display(byte Terminal, char Dado); stdcall;
__declspec(dllimport) void Dll_Aciona(byte Terminal, Dado); stdcall;
__declspec(dllimport) char Dll_Get(byte Terminal); stdcall;
__declspec(dllimport) byte Dll_Status(byte Terminal); stdcall;
__declspec(dllimport) byte Dll_Print(byte Terminal, char Dado); stdcall;
__declspec(dllimport) byte Dll_Serial(byte Terminal, char Dado); stdcall;

Apêndice D – Instalação de Dll em Windows 2000, XP e Vista

1. Logar no PC como administrador ou com alguma conta com privilégios administrativos. Esta conta deverá possuir uma senha, mesmo que seja um único usuário. Caso não possua, deverá criar uma senha:
   1.1. Selecionar: Iniciar -> Configurações -> Painel de Controle -> Contas de usuário;
   1.2. Selecionar a conta "Administrador", em seguida "Criar uma senha" e preencher os dados. E por fim "Criar senha".

Esse procedimento é importante, pois evita-se maiores transtornos em relação a instalação e funcionamento do port95nt.

2. Caso o Sistema Operacional seja o Windows Vista 32 bits, copie o arquivo DlPortio.sys para Windows\System32\Drivers\
3. Instalar o Port95NT.exe;
4. Reiniciar o PC;
5. Confirmar se a instalação foi bem sucedida, certificando que o driver foi instalado:
   5.1. Selecionar: Painel de Controle -> Sistemas -> Hardware -> Gerenciador de Dispositivos -> Menu Exibir -> Mostrar dispositivos ocultos -> Drivers que não são Plug and Play: DriverLINX Port I/O .
6. Tomar o cuidado de que as Dll's DlPortIO.dll e WtechLpt.dll se encontrem no mesmo diretório do software ou na pasta system32 do Windows.

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:_____