Manual do Desenvolvedor

Microterminal 721

DLL PMTG V1.5

R00.22

Sumário

Inicialização e Finalização

Este manual descreve as funções da DLL PMTG e os retornos esperados para as mesmas.

mt_startserver

int __stdcall mt_startserver(HWND mywhnd, int conecmsg, int commumsg);

Esta é a primeira função que deve ser chamada. Se tiver sucesso na sua chamada, os terminais já conectarão ao servidor.

mywhnd: Handle para a janela principal do programa do servidor, que é para onde a DLL irá mandar as mensagens para troca de dados. Se não quiser receber as mensagens deve seu valor deve ser NULL.

conecmsg: Valor da mensagem que a DLL enviará quando um terminal conectar/desconectar.
commumsg: Valor da mensagem que a DLL enviará quando terminal enviar dados.
retorna: 1 se servidor inicializado com sucesso, 0 se houve algum erro.

mt_finishserver

void __stdcall mt_finishserver(void);

Após chamar esta função, a DLL libera a memória armazenada, desconecta todos os terminais e para de aceitar novas conexões.

mt_version

char __stcall mt_version(void);

typedef struct
{
DWORD ip[255];
} TTABSOCK;
TTABSOCK.ip[0]=0x00000000
TTABSOCK.tab[1]=0xB600A8C0
TTABSOCK.tab[2]=0x00000000
...
Indica que no ID “1”, existe um terminal conectado com o IP “B600A8C0” (192.168.0.182).

Funções de tratamento de IP

mt_gethostip

char * __stcall mt_gethostip(char *oip);

Retorna o IP da máquina local em ASCII formatada por pontos.
oip: array de bytes onde será escrito os dados.

mt_inet_ntoa

char *__stdcall mt_inet_ntoa(DWORD oip);
Retorna IP em ASCII formatado por pontos.
Oip: IP no formato de rede (DWORD).

mt_inet_ntoa_inv

char *__stdcall mt_inet_ntoa_inv(CWORD oip);
Retorna IP em ASCII formatado por pontos, invertendo byte mais significativo pelo menos significativo.
Oip: IP no formato de rede (DWORD).

mt_inet_addr

DWORD __stdcall mt_inet_addr(char *oip);
Retorna IP no formato de rede (DWORD).
Oip: array de bytes em ASCII formatado por pontos.

mt_inet_addr_inv

DWORD __stdcall mt_inet_addr_inv(char *oip);
Retorna IP no formato de rede (DWROD), invertendo byte mais significativo pelo menos significativo.
Oip: array de bytes em ASCII formatado por pontos.

mt_ipfromid

char *__stdcall mt_ipfromid(int ID);
Retorna IP formatado por pontos de um terminal.
ID: ID do terminal.

Comandos básicos para o terminal

mt_sendlive

int __stdcall mt_sendlive(int ID);
Envia o comando de vivo para o terminal (IDLive).
Retorna: <=0 houve erro, 1 se comando realizado com sucesso.

mt_restart

int__stdcall mt_restart(int ID);

Faz com que o terminal se reinicialize (IDRestart).

retorna: <=0 houve erro, 1 se comando realizado com sucesso.

mt_ftpmode

int__stdcall mt_ftpmode(int ID);

Faz com que o terminal entre em modo FTP (IDvFTPMode).

retorna: <=0 houve erro, 1 se comando realizado com sucesso.

Rede

mt_sendconfig

int__stdcall mt_sendconfig(int ID, TSetupTCP *config); Envia configuração para o terminal (IDvSetSetupTCP). retorna: <=0 houve erro, 1 se comando realizado com sucesso.

```
typedef struct
{
DWORD microT_IP;                // Endereço IP do terminalDWORD server_IP;
      // Endereço IP do servidor DWORD msknet_IP;            // Máscara de rede
DWORD bDHCP; // 1 = IP dinâmico, 0 = IP fixo.

}TSetupTCP;
```

mt_reqconfig

int__stdcall mt_reqconfig(int ID);

Requisita configuração do terminal (IDvGetSetupTCP). A DLL enviará uma mensagem cada vez que receber aresposta a este comando. Para receber a configuração utilize a função mt_getconfig.

retorna: <=0 houve erro, 1 se comando realizado com sucesso.

mt_getconfig

int__stdcall mt_getconfig (int ID, TSetupTCP* config);

Recebe os dados enviados pelo terminal ao enviar o coamando IDvGetSetupTCPpara o terminal. A DLL enviaráuma mensagem cada vez que tiver dados a serem recebidos.

config: estrutura onde será salvo os dados recebidos.

retorna: <=0 buffer vazio, 1 se comando realizado com sucesso.

mt_sendexconfig

int__stdcall mt_sendexconfig(int ID, TExSetupTCP *config);

Envia configuração estendida para o terminal (IDvSetExSetupTCP).retorna: <=0 houve erro, 1 se comando realizado com sucesso.

```
typedef struct
{
DWORD gateway;      // IP do gateway

DWORD nameserver; // IP do servidor de nomesDWORD myname;  // Nome do terminal
}TExSetupTCP;
```

mt_reqconfig

int__stdcall mt_reqexconfig(int ID);

Requisita configuração do terminal (IdvGetExSetupTCP). A DLL enviará uma mensagem cada vez que
receber a resposta a este comando. Para receber a configuração utilize a função mt_getexconfig.

retorna: <=0 houve erro, 1 se comando realizado com sucesso.

mt_getexconfig

int__stdcall mt_getexconfig (int ID, TExSetupTCP* config);

Recebe os dados enviados pelo terminal ao enviar o coamando IDvGetExSetupTCP para o terminal. A DLL enviaráuma mensagem cada vez que receber a resposta ao comando.

config: estrutura onde será salvo os dados recebidos.

retorna: <=0 buffer vazio, 1 se comando realizado com sucesso.

Display

mt_backspace

int__stdcall mt_backspace(int ID);

Envia o comando de BackSpace para o terminal (IDvBackSpace).retorna: <=0 houve erro, 1 se comando realizado com sucesso.

mt_carret

int__stdcall mt_carret (int ID);

Envia o comando de CarriageReturn para o terminal (IDvCarRet).retorna: <=0 houve erro, 1 se comando realizado com sucesso.

mt_linefeed

int__stdcall mt_linefeed (int ID);

Envia o comando de LineFeed para o terminal (IDvLineFeed). retorna: <=0 houve erro, 1 se comando realizado com sucesso.

mt_formfeed

int__stdcall mt_formfeed (int ID);

Envia o comando de FormFeed para o terminal (IDvFormFeed).retorna: <=0 houve erro, 1 se comando realizado com sucesso.

mt_gotoxy

int__stdcall mt_gotoxy (int ID, int lin, int col);

Envia o comando de GotoXY para o terminal (IDvGoToXY). lin/col: Linha e Coluna onde será posicionado o cursor. retorna: <=0 houve erro, 1 se comando realizado com sucesso.

mt_gotoxyref

int__stdcall mt_gotoxyref(int ID, int lin, int col);

Envia o comando de GotoXYRef para o terminal (IDvGoToXYRef).

lin/col: Linha e Coluna onde será posicionado o cursor a partir da posição atual.retorna: <=0 houve erro, 1 se comando realizado com sucesso.

mt_dispstr

int__stdcall mt_dispstr(int ID, char *str);

Envia o comando de DisplayString para o terminal (IDvDispStr).str: string que será enviada para o display.
retorna: <=0 houve erro, 1 se comando realizado com sucesso.

mt_dispch

int__stdcall mt_dispch (int ID, char ch);
Envia o comando de DisplayCharacter para o terminal (IDvDispCh).
ch: Caracter que será enviado para o display.
retorna: <=0 houve erro, 1 se comando realizado com sucesso.

mt_dispclrln

int__stdcall mt_dispclrln(int ID, int lin);

Envia o comando de DisplayClearLine para o terminal (IDvDispClrLn).lin: linha do display que será apagada.
retorna: <=0 houve erro, 1 se comando realizado com sucesso.

mt_seteditstring

int__stdcall mt_seteditstring(int ID, BYTE *String, BOOL OnOff, BOOL PassWord);Envia o comando de EditString para o terminal (IDvSetEditString).
String: String inicial da edição de texto.
OnOff: ativa (1) e desativa (0) o modo de edição de string.

PassWord: habilita (1) e desabilita (0) o modo de edição em modo protegido (senha).retorna: <=0 houve erro, 1 se comando realizado com sucesso.

mt_reqeditstring

int__stdcall mt_reqeditstring(int ID);

Requisita o estado do EditString do terminal (IDbGetEditString). A DLL enviará uma mensagem com aresposta deste comando.
retorna: <=0 houve erro, 1 se comando realizado com sucesso.

mt_geteditstring

int__stdcall mt_geteditstring (int ID, char *buftecla);
Recebe os dados enviados pelo terminal ao enviar o coamando IDvSetEditString para o terminal. A DLL
enviará uma mensagem cada vez que tiver dados a serem recebidos.

buftecla: string que será recebida após pressionar a tecla Enter no microterminal.retorna: <=0 buffer vazio, 1 se comando realizado com sucesso.

Teclado

mt_setenablekey

int__stdcall mt_setenablekey(int ID, const BOOL OnOff);

Envia o comando de EnableKey para o terminal (IDvSetEnableKey). OnOff: ativa (1) e desativa (0) o teclado do terminal.
retorna: <=0 houve erro, 1 se comando realizado com sucesso.

mt_getenablekey

int__stdcall mt_getenablekey(int ID);

Requisita o estado do EnableKey do terminal (IDbGetEnableKey). A DLL enviará uma mensagem com a resposta deste comando.

retorna: <=0 houve erro, 1 se comando realizado com sucesso.

mt_reset

int__stdcall mt_reset(int ID);
Envia o comando de Reset para o terminal (IDvReset).
retorna: <=0 houve erro, 1 se comando realizado com sucesso.

mt_setcapslock

int__stdcall mt_setcapslock(int ID, const BOOL OnOff);

Envia o comando de SetCapsLock para o terminal (IDvSetCapsLock). OnOff: ativa (1) e desativa (0) o CapsLock.
retorna: <=0 houve erro, 1 se comando realizado com sucesso.

mt_getcapslock

int__stdcall mt_getcapslock(int ID);

Requisita o estado do CapsLock do terminal (IDbGetCapsLock). A DLL enviará uma mensagem com a resposta deste comando.

retorna: <=0 houve erro, 1 se comando realizado com sucesso.

mt_setnumlock

int__stdcall mt_setnumlock(int ID, const BOOL OnOff);
Envia o comando de SetNumLock para o terminal (IDvSetNumLock). OnOff: ativa (1) e desativa (0) o NumLock.
retorna: <=0 houve erro, 1 se comando realizado com sucesso.

mt_getnumlock

int__stdcall mt_getnumlock(int ID);

Requisita o estado do NumLock do terminal (IDbGetNumLock). A DLL enviará uma mensagem com a resposta deste comando.
retorna: <=0 houve erro, 1 se comando realizado com sucesso.

mt_programkbd

int__stdcall mt_programkbd(int ID, BYTE *Codigo);

Envia o comando de ProgramKeyboard para o terminal (IDbProgramKbd). A DLL enviará uma mensagem com aresposta deste comando.

Codigo: ponteiro para onde está o conteúdo do arquivo .bin com tabela de códigos do teclado.
retorna: <=0 houve erro, 1 se comando realizado com sucesso.

mt_setbeep

int__stdcall mt_setbeep(int ID, const BOOL OnOff);

Envia o comando de SetBeep para o terminal (IDvSetBeep). OnOff: ativa (1) e desativa (0) o Beep.
retorna: <=0 houve erro, 1 se comando realizado com sucesso.

mt_setbeepkey

int__stdcall mt_setbeepkey(int ID, const BOOL OnOff);

Envia o comando de SetBeepkey para o terminal (IDvSetBeepKey).

OnOff: ativa (1) e desativa (0) o Beep de tecla.

retorna: <=0 houve erro, 1 se comando realizado com sucesso.

mt_getkey

int__stdcall mt_getkey(int ID, char *buf); Recebe uma tecla do terminal (IDcGetCharTerm). buf: tecla recebida.
retorna: <=0 houve erro, 1 se comando realizado com sucesso.

Serial

mt_setenableserial

int__stdcall mt_setenableserial(int ID, const BYTE COM, const BOOL OnOff);

Envia o comando de SetEnableSerial para o terminal (IDvSetEnableSerial). COM: porta serial, deve ser sempre = 0 (COM1).
OnOff: ativa (1) e desativa (0) a serial.

retorna: <=0 houve erro, 1 se comando realizado com sucesso.

mt_getenableserial

int__stdcall mt_getenableserial(int ID, const BYTE COM);

Requisita o estado do EnableSerial do terminal (IDvGetEnableSerial). A DLL enviará uma mensagem coma resposta deste comando.

COM: porta serial, deve ser sempre = 0 (COM1).
retorna: <=0 houve erro, 1 se comando realizado com sucesso.

mt_sendbinserial

int__stdcall mt_sendbinserial(int ID, const BYTE COM, LPBYTE Bin, BYTE tam);Envia o comando de SendBinSerial para o terminal (IDbSendBinSerial).
COM: porta serial, deve ser sempre = 0 (COM1).

Bin: dados que serão enviados para a serial do terminal.tam: quantidade de dados que serão enviados.
retorna: <=0 houve erro, 1 se comando realizado com sucesso.

mt_getserial

int__stdcall mt_getserial(int ID, int *sercom, char *buf);

Envia o comando de GetSerial para o terminal (IDbGetBinSerial). A DLL enviará uma mensagem cadavez que tiver dados a serem recebidos.

sercom: de qual porta serial foram lidos os dados 0 = COM1.buf: os dados da serial serão escritos nesta variavel retorna: numero de bytes lidos da serial.

mt_sendconfigserial

int__stdcall mt_sendconfigserial(int ID, ARG_COM_SETUPSERIAL *config);Envia o comando de SendConfigSerial para o terminal (IDbSetSetupSerial). Config: estrutura onde será salvo os dados recebidos.
retorna: 0 houve erro, outro valor se comando realizado com sucesso.

```
typedef struct {
unsigned long baud;     // baudrate: 300 a 115.200unsigned short bits;
      // data bits
unsigned short parity;  // paridade
unsigned short stops;   // stop bits
unsigned char handshaking; // 0 = sem handshaking, 1 = RTS/CTS
} TSetupSerial;typedef struct {
```

unsigned char Com;
TSetupSerial Setup;
}ARG_COM_SETUPSERIAL;

mt_reqconfigserial

int__stdcall mt_reqconfigserial(int ID, BYTE COM);

Requisita a configuração da serial do terminal (IDvGetSetupSerial). A DLL enviará uma mensagem cadavez

que receber a resposta a este comando. Para receber a configuração utilize a função mt_getconfig.

COM: porta serial, deve ser sempre = 0 (COM1).

retorna: <=0 houve erro, 1 se comando realizado com sucesso.

mt_getconfigserial

int__stdcall mt_getconfigserial(int ID, ARG_COM_SETUPSERIAL *config);

Recebe os dados enviados pelo terminal ao enviar o coamando IDvGetSetupSerial para o terminal. ADLL enviará uma mensagem cada vez que tiver dados a serem recebidos.

config: estrutura onde será salvo os dados recebidos.

retorna: <=0 buffer vazio, 1 se comando realizado com sucesso.

mt_settermserial

int__stdcall mt_settermserial(int ID, const BYTE COM, const BYTE TERM, const BOOL OnOff);

Configura o carcter de terminador da porta serial ao enviar o coamando IdvSetTermSerial para oterminal.

COM: porta serial, deve ser sempre = 0 (COM1). TERM: Byte de terminador.
OnOff: ativa (1) e desativa (0) esta funcionalidade.

retorna: <=0 houve erro, 1 se comando realizado com sucesso.

Cartão Magnético

mt_sendsetcard

int__stdcall mt_sendsetcard(int ID, const BOOL OnOff);Envia o comando de IDvSetCard para o terminal. OnOff: ativa (1) e desativa (0) a serial.
retorna: <=0 houve erro, 1 se comando realizado com sucesso.

mt_reqgetcard

int__stdcall mt_reqgetcard(int ID);

Requisita o estado do cartão magnético do terminal (IDbGetCard). A DLL enviará uma mensagem com a respostadeste comando.
retorna: <=0 houve erro, 1 se comando realizado com sucesso.

mt_getcardbuf

int__stdcall mt_getcardbuf(int ID, ARG_CARD *cardbuf);

Recebe os dados do cartão magnéticos enviados pelo terminal com o comando IDbReadBuffCard. A DLL enviaráuma mensagem cada vez que tiver dados a serem recebidos.

config: estrutura onde serão salvos, os dados recebidos. retorna: <=0 buffer vazio, 1 se comando realizado com sucesso.

```
typedef struct {
unsigned char card[128];unsigned long status;
}ARG_CARD;
```

Impressora

mt_sendinitprn

int__stdcall mt_sendinitprn(int ID);
Envia o comando de IDcInitPrn para o terminal.

retorna: <=0 houve erro, 1 se comando realizado com sucesso.

mt_reqgetstatusprn

int__stdcall mt_reqgetstatusprn(int ID);

Requisita o estado da impressora do terminal (IDcGetStatusPrn). A DLL enviará uma mensagem com a respostadeste comando.

retorna: <=0 houve erro, 1 se comando realizado com sucesso.

mt_sendbinprn

int__stdcall mt_sendbinprn(int ID, LPBYTE Bin, BYTE tam);

Envia o comando de IDcSendPrn para o terminal (IDbSendBinSerial). Bin: dados que serão enviados para a serial do terminal.
tam: quantidade de dados que serão enviados.
retorna: <=0 houve erro, 1 se comando realizado com sucesso.

A troca de mensagens do programa principal com aDLL

Para aumentar a agilidade de troca de informação da DLL com o programa principal do servidor e evitarprocessamento desnecessário, foi implementado a troca de mensagens. A DLL envia mensagens para o programaprincipal sempre que ocorrer em evento entre o terminal e a DLL. Para facilitar o entendimento, utilizaremos o C++ Builder® como exemplo.
Para receber estas mensagens, o servidor deve chamar a função da DLL mc_startserver da seguinte forma:

```
#define COMUNICATION_MSG  WM_USER + 1#define CONNECT_MSG   WM_USER + 2
```

mc_startserver(Form1−>Handle,CONNECT_MSG,COMUNICATION_MSG);
Onde Form1 corresponde ao formulário (janela) principal, CONNECT_MSG corresponde à mensagem que o servidor enviará quando um terminal conectar/desconectar e COMUNICATION_MSG corresponde à mensagem que o servidor enviará quando um terminal enviar dados.

Devemos “redefinir” a função WndProc do formulário para prodermos receber as mensagens. Para isso devemosseguir os seguintes passos:

No arquivo de header (geralmente unit1.h) devemos adicionar na classe do formulário, a chamadapara a função:

```
...
private:    // User declarations
virtual void__fastcall WndProc(Messages::TMessage &Message);
...
```

Devemos então, “reescrever” esta função (unit1.cpp):

```
void__fastcall TForm1::WndProc(Messages::TMessage &Message)
{
if (Message.Msg == COMUNICATION_MSG)
{
//recebe mensagens enviadas pelo terminalreturn;
}
else if (Message.Msg == CONNECT_MSG)
{
//recebe mensagens quando um terminal conectou/desconectoureturn;
}

TForm::WndProc(Message); //chama WndProc antiga
```

}

Para saber como tratar as mensagens recebidas, fornecemos o código fonte do servidor para sertomado como exemplo.

CONTATOS GERTEC

Assistência Técnica Gertec
astecnica@gertec.com.br
Telefone (11) 2173-6500
Suporte Técnico Gertec
suporte@gertec.com.br
Telefone (11) 2575-1000.