Documentação DLL SC504 V2.1

SC504.DLL

rev agosto/2004

Inicialização e Finalização

tc_startserver

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

Esta é a primeira função que deve ser chamada. Se tiver sucesso na sua chamada retorna 1, e então 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.

tc_finishserver

void __stdcall tc_finishserver(void);

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

Funções Diversas

tc_version

char __stdcall tc_version(void);

Retorna versão da DLL. Ex.: 0x20 corresponde a versão 2.0.

tc_gethostip

char * __stdcall tc_gethostip(char *oip);

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

tc_inet_ntoa

char * __stdcall tc_inet_ntoa(DWORD oip);

Retorna IP em ASCII formatado por pontos.
oip: IP no formato de rede (DWORD).

tc_inet_addr

DWORD __stdcall tc_inet_addr(char *oip);

Retorna IP no formato de rede (DWORD).
oip: array de bytes em ASCII formatado por pontos.

tc_ipfromid

char * __stdcall tc_ipfromid(int ID);

Retorna IP formatado por pontos de um terminal.
ID: ID do terminal.

Comandos para o terminal

tc_sendlive

int __stdcall tc_sendlive(int ID);

Envia o comando de vivo para o terminal (IDvLive).
retorna: 0 houve erro, 1 se comando realizado com sucesso.

tc_sendalwayslive

int __stdcall tc_sendalwayslive(int ID);

Envia o comando de sempre vivo para o terminal (IDvAlwaysLive).
retorna: 0 houve erro, 1 se comando realizado com sucesso.

tc_reqconfig

int __stdcall tc_reqconfig(int ID);

Requisita configuração do terminal (IDvGetSetupTCP). Para receber a configuração utilize a função tc_getconfig.
retorna: <1 houve erro, >0 se comando realizado com sucesso.

tc_sendconfig

int __stdcall tc_sendconfig(int ID,ARG_SETUP_TCP *config);

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

```
typedef struct
{
   DWORD dwMY_IP_ADD;     // Endereço IP do terminal
   DWORD dwServer_IP;     // Endereço IP do servidor
   DWORD dwNetMask;       // Máscara de rede
   DWORD dwGateway;       // Endereço do Gateway
   DWORD dwNameServer;    // Endereço do servidor de nomes
   char TCName[32];       // Nome do terminal (string terminada em caracter nulo
                          // usada na linguagem C)
   WORD wPortsv;          // Porta de comunicação com o servidor
   char FTPs[100];        // String com o endereço do servidor de atualização
                          // ("http://..." ou "ftp://....")
   char FTPu[30];         // Nome do usuário para o servidor de FTP (atualização)
   char FTPp[30];         // Senha do usuário para o servidor de FTP
   DWORD dwDHCP;          // Se o terminal usar IP Dinâmico, esse valor será 1. Se
                          // não 0.
   DWORD dwSearchServer; // Se a busco do servidor for automática, esse valor
                          // será 1.
}ARG_SETUP_TCP;
```

tc_sendconfigserial

int __stdcall tc_sendconfigserial(int ID, int sercom, ARG_SERIAL_CFG *config);

Envia configuração da serial para o terminal (IDvConfigSerialA se sercom = 0 ou IDvConfigSerialB se sercom = 1).
retorna: <1 houve erro, >0 se comando realizado com sucesso.

```
typedef struct
{
   DWORD dwOpen;    // 0 = abrir, 1 = fechar
   DWORD dwBaud;    // Velocidade da porta serial
   BYTE bParity;    // Paridade
   BYTE bDataBits; // Databits
   WORD wTimeOut;  // reservado
}ARG_SERIAL_CFG;
```

tc_writeserial

int __stdcall tc_writeserial(int ID, int sercom, int tambuf, char *sbuf);

Escreve dados na serial do terminal.
sercom: 0 = serial IDvWriteSerialA, 1 = serial IDvWriteSerialB.
tambuf: número de bytes a serem escritos
sbuf: array de bytes contendo bytes a serem escritos
retorna: <1 houve erro, >0 se comando realizado com sucesso.

tc_sersetstatus

int __stdcall tc_sersetstatus(int ID, int sercom, unsigned char sts);

```
typedef struct
{
   BYTE aserial;    // 0 = COM 1, 1 = COM 2
```

```
   BYTE status;     // Estado de controle
}ARG_SERIAL_CFG;
```

Envia o estado do RTS e DTR da serial (IDvSetStatus).
Sercom: 0 = COM 1, 1 = COM 2;
Status: Bit0: RTS;
         Bit1: DTR;

tc_serreqstatus

int __stdcall tc_serreqstatus(int ID, int sercom);

Requisita estado do CDC, DSR e CTS da serial (IDvGetStatus). Para receber o estado da serial utilize a função tc_sergetstatus.
Sercom: 0 = COM 1, 1 = COM 2.

tc_updatesoft

int __stdcall tc_updatesoft(int ID);

Faz com que o terminal seja atualizada no endereço pre configurado (IDUpdateSoft).
retorna: <1 houve erro, >0 se comando realizado com sucesso.

tc_restart

int __stdcall tc_restart(int ID);

Faz com que o terminal se reinicialize (IDRestart).
retorna: <1 houve erro, >0 se comando realizado com sucesso.

tc_stopadv

int __stdcall tc_stopadv(int ID);

Faz com que o terminal para de exibir o loop de imagens (IDStopAdv).
retorna: <1 houve erro, >0 se comando realizado com sucesso.

tc_reloadadv

int __stdcall tc_reloadadv(int ID, unsigned long doreload);

Faz com que o terminal recarregue seu loop de imagens (IDReloadAdv).
doreload: deve ser sempre igual a 1.
retorna: <1 houve erro, >0 se comando realizado com sucesso.

tc_deleteadv

int __stdcall tc_deleteadv(int ID);

Apaga os arquivos do loops de imagens (IDvDeleteAdv).
retorna: <1 houve erro, >0 se comando realizado com sucesso.

tc_reloadpreimg

int __stdcall tc_reloadpreimg(int ID, unsigned long doreload);

Faz com que o terminal recarregue suas imagens predefinidas (IDReloadPreImg). Se o número de imagens for maior do que o anterior, é recarregado também, o loop de imagens.
doreload: deve ser sempre igual a 1.
retorna: <1 houve erro, >0 se comando realizado com sucesso.

tc_deletepreimg

int __stdcall tc_deletepreimg(int ID);

Apaga os arquivos de imagens predefinidas (IDvDeletePreImg).
retorna: <1 houve erro, >0 se comando realizado com sucesso.

tc_sendimage

int __stdcall tc_sendimage(int ID, char *APaleta, char *AImagem);

Envia uma imagem na tela do terminal (IDvShowIMG).
APaltea: array de bytes contendo a paleta de cores (256 x 3 = 768 bytes).
AImagem: array de bytes contendo a imagem (320 x 240 = 768000 bytes).
retorna: <1 houve erro, >0 se comando realizado com sucesso.

tc_sendimgfromfile

int __stdcall tc_sendimgfromfile(int ID, char *filename);

Envia uma imagem na tela do terminal (IDvShowIMG).
filename: caminha da imagem (.BMP ou .JPG) que será enviada para o terminal.
retorna: <1 houve erro, >0 se comando realizado com sucesso.

tc_sendimageblock

int __stdcall tc_sendimageblock(int ID, int x, int y, int width, int height, char *AImagem);

Envia uma região de imagem na tela do terminal (IdvShowImageBlock).
x,y,width,height: posição, largura e altura de onde será desenhada a imagem.
AImagem: array de byte contendo a imagem a ser mostrada (width x height bytes).
retorna: <1 houve erro, >0 se comando realizado com sucesso.

tc_enviapaleta

int __stdcall tc_enviapaleta(int ID, char *APaleta);

Envia paleta de cores para o terminal (IDvSendPalette).
APaleta: array de bytes contendo a paleta de cores (256 x 3 = 768 bytes).
retorna: <1 houve erro, >0 se comando realizado com sucesso.

tc_sendtext

int __stdcall tc_sendtext(int ID, ARG_DISPALY_TEXT *otexto);
retorna: <1 houve erro, >0 se comando realizado com sucesso.

```
typedef struct
{
   WORD wPosX;       // Posição X
   WORD wPosY;       // Posição Y
   char sText[128]; // Texto a ser escrito
   char sFont[32];  // Caminho da fonte utilizada (ex: \fonts\lucida12.bmp
   WORD wSize;       // reservado
   WORD wColor;      // Cor do texto
   WORD wBGColor;    // Cor do fundo
}ARG_DISPLAY_TEXT;
```

tc_fillscreen

int __stdcall tc_fillscreen(int ID, short color);

Escreve texto na tela do terminal (IDvShowText).
Color: cor de preenchimento.
retorna: <1 houve erro, >0 se comando realizado com sucesso.

tc_loadimage

int __stdcall tc_loadimage(int ID, int frame);

Mostra uma imagem precarregada na tela do terminal (IDShowFrame).
frame: número do frame que será mostrado no terminal (o primeiro frame é o 0).
retorna: <1 houve erro, >0 se comando realizado com sucesso.

tc_reqfile

int __stdcall tc_reqfile(int ID, char *filename);

Requisita um arquivo do terminal (IDvRecvFile). Para receber o arquivo utilize a função tc_getfile.
filename: caminho do arquivo do terminal que se deseja receber.
retorna: <1 houve erro, >0 se comando realizado com sucesso.

tc_sendfileptr

int __stdcall tc_sendfileptr(int ID, char *filename, char *filedata, unsigned long filesize);

Envia um arquivo para o terminal (IDvSendFile).
filename: caminho do arquivo que será escrito no terminal.
filedata: array de bytes contendo conteúdo do arquivo.
filesize: numero de bytes que será escritos no arquivo.
retorna: <1 houve erro, >0 se comando realizado com sucesso.

tc_sendfile

int __stdcall tc_sendfile(int ID, char *srcfilename, char *destfilename);

Envia um arquivo para o terminal (IDvSendFile).
srcfilename: caminho do arquivo de origem.
destfilename: caminho do arquivo destino.
retorna: <1 houve erro, >0 se comando realizado com sucesso.

tc_requid

int __stdcall tc_requid(int ID);

Requisita o MAC Address e o nome do terminal (IDvGetUID). Para receber os dados, utilize a função tc_getuid.
retorna: <1 houve erro, >0 se comando realizado com sucesso.

tc_reqsc

int __stdcall tc_reqsc(int ID);

Requisita a identificação de segurança do terminal (IDSecretCode). Para receber os dados, utilize a função tc_getsc.
retorna: <1 houve erro, >0 se comando realizado com sucesso.

tc_reqimageupdate

int __stdcall tc_reqimageupdate(int ID);

Requisita configuração de atualização do terminal (IDGetConfigAdvServer). Para receber os dados, utilize a função tc_getimageupdateconfig

tc_sendimageupdate

int __stdcall tc_sendimageupdate(int ID, ARG_IMGUPD *imgupd);

Envia configuração de atualização de imagens para o terminal (IDSetConfigAdvServer).

```
typedef struct
{
   long imgupdenable;       // 0 = desabilita, 1 = habilita
   char imgupdserver[100];  // endereço do servidor de atualização de imagens
   long imgupddeftime;      // tempo padrão de atualização
   long imgupdcurtime;      // tempo atual de atualiação
} ARG_IMGUPD;
```

tc_sendupdateimages

int __stdcall tc_sendupdateimages(int ID);

```
typedef struct
{
   BYTE aserial;    // 0 = COM 1, 1 = COM 2
   BYTE status;     // Estado de controle
}ARG_SERIAL_CFG;
```

Faz com que o terminal realize a atualização das imagens no servidor configurado.
Status: Bit0: indefinido
Bit1: DCD;
Bit2: DSR;
Bit3: CTS;

Recebendo dados dos terminais

tc_getserial

int __stdcall tc_getserial(int ID, int *sercom, char *buf);

Recebe dados da serial enviados pelo terminal (IdvReadSerialA/B).
sercom: se leu dados da serial A, sercom=0, se foi da B, sercom = 1.
buf: array de bytes contendo dados lidos da serial.
retorna: número de bytes lidos.

tc_sergetstatus

int __stdcall tc_sergetstatus(int ID, int sercom, ARG_SERIAL_STS *serialsts);

Recebe o estado da porta serial do terminal (IDvGetStatus).

tc_getmag

int __stdcall tc_getmag(int ID, char *buf1, char *buf2);

Recebe dados do leitor de cartão magnético enviados pelo terminal (IDbReadBuffLEC).
buf1,buf2: array de bytes contendo dados ligos das trilhas 1,2.
retorna: 0 se buffer vazio, 1 se conseguiu pegar dados.

tc_getconfig

int __stdcall tc_getconfig(int ID, ARG_SETUP_TCP *config);

Recebe configuração do terminal requisitada pela função tc_reqconfig.
config: estrutura contendo configuração do terminal.
retorna: 0 se buffer vazio, 1 se conseguiu pegar dados.

tc_getfile

int __stdcall tc_getfile(int ID, char *ofilename, int *ofilesize, char *oarquivo);

Recebe arquivo requisitado pela função tc_retfile.
ofilename: nome do arquivo requisitado.
ofilesize: tamanho do arquivo requisitado.
oaquivo: array de bytes com o conteúdo o arquivo.
retorna: -1 se ID inválido, 0 se arquivo não existir, >1 se conseguiu pegar dados.

tc_getuid

int __stdcall tc_getuid(int ID, char *macaddr, char *tcname);

Recebe o MAC Address e o nome do terminal requisitados pela função tc_requid.
macaddr: array de bytes contendo o mac address (6 bytes).
tcname: array de bytes contendo o nome do terminal.
retorna: 0 se buffer vazio, 1 se conseguiu pegar dados.

tc_getsc

int __stdcall tc_getsc(int ID, char *sc);

Recebe a identificação de segurança requisitados pela função tc_reqsc.
sc: array de bytes contendo as informações de segurança do terminal.
retorna: 0 se buffer vazio, 1 se conseguiu pegar dados.

tc_getident

unsigned long __stdcall tc_getident(int ID);

Recebe identificação do terminal.

retorna: 4 bytes com a identificação do terminal.

tc_ getimageupdateconfig

int __stdcall tc_getimageupdateconfig(int ID,ARG_IMGUPD *imgupd);

Recebe a configuração de atualização de imagens requisitados pela função tc_reqimageupdate.
retorna: 0 se buffer vazio, 1 se conseguiu pegar dados.

Compatibilidade com a versão 1.0

Para manter compatibilidade com a DLL 1.0, foram criadas funçõs que fazer a interface com a DLL 2.0. Porém, é preciso observar que as chamadas para tais funções sofreram pequenas alterações, como a diretiva "__stdcall", que foi implementado para haver maior compatibilidade da DLL entre diversas linguagens de programação e algumas funções que tinham como parâmetro, array de bytes, foram substituidos por ponteiro de bytes, portanto, para utilizar a DLL 2.0 nos sevidores que utilizam a 1.0 é preciso recompilar o programa, reajustando as chamadas de funções. Segue abaixo, protótipo das funções.

Funções em C

- void __stdcall GetIPFromHost(char *ret);
- void __stdcall TCinet_ntoa(DWORD nIP, char *buf);
- DWORD __stdcall TCinet_addr(char *buf);
- TTABSOCK __stdcall GetTabConectados(void);
- void __stdcall EnviaVivo(int ID);
- void __stdcall EnviaSempreVivo(int ID);
- void __stdcall SendConfig(int ID, char *ClienteAdd, char *ServerAdd, char *NetMaskAdd, char *GatewayAdd, char *NameserverAdd, char *TCNameAdd, WORD PortsvAdd, char *FTPServerAdd, char *FTPUserAdd, char *FTPPAssAdd, DWORD bDHCP, DWORD AutoFind);
- void __stdcall PedeConfig(int ID);
- void __stdcall EnviaTexto(int ID, WORD Posx, WORD Posy, char *Text2Show, char *Fonte, WORD TextColor, WORD BgColor);
- void __stdcall ClearDisplay(int ID, int Color);
- void __stdcall EnviaImagem(int ID, char *APaleta, char *tAImagem);
- void __stdcall EnviaImagemDoArquivo(int ID, char *filename);
- void __stdcall SetaTempoExib(int ID, WORD TempoEx);
- void __stdcall PedeTempoExib(int ID);
- void __stdcall UpdateSoft(int ID);
- void __stdcall StopAdv(int ID);
- void __stdcall DeleteAdv(int ID);
- void __stdcall HabilitaTeclado(int ID, DWORD HabSim);
- void __stdcall PedeHabilitaTeclado(int ID);
- void __stdcall AtualizaAdv(int ID, DWORD DoReload);
- void __stdcall HabilitaLEC(int ID, DWORD HabSim);
- void __stdcall PedeHabilitaLEC(int ID);
- void __stdcall AbreSerialCOM(int ID, BYTE SerCOM, BYTE SimAbre, int Baud, BYTE parity, BYTE databits);
- void __stdcall EscreveSerial(int ID, BYTE SerCOM, int TamBuf, char *sbuf);
- void __stdcall EnviaArquivo(int ID, char *localfilename, char *destfilename);
- bool __stdcall GetSerial(int *ID, int *Porta, char *buffer, int *Nbr);
- bool __stdcall GetLEC(int *ID, char *ptrilha, int *errocode);

- bool __stdcall Get_KBD_char(int *ID, char *caracter);
- bool __stdcall GetConfig(int *pID, char *pmyip, char *pserverip, char *pnetmask, char *pgateway, char *pnameserver, char *ptcname, WORD *pport, char *pupdserv, char *pupduser, char *pupdpass,char *pdynamicip, char *pfindserverchar);
- void __stdcall StartServerTC504(void);
- void __stdcall CloseTC504(void);
- void __stdcall InitTC504(void);

Funções em Pascal

- procedure GetIPFromHost(var res : byte); far; stdcall; external 'sc504.dll';
- procedure TCinet_ntoa(nIP : DWORD; var buf: byte); far; stdcall; external 'sc504.dll';
- function TCinet_addr(var buf: byte): DWORD; far; stdcall; external 'sc504.dll';
- procedure EnviaVivo(ID: Integer); far; stdcall; external 'sc504.dll';
- function GetConfig(var pID: integer; var pmyip: byte; var pserverip: byte; var pnetmask: byte; var pgateway: byte; var pnameserver: byte; var ptcname: byte; var pport : WORD; var pupdserv: byte; var pupduser: byte; var pupdpass: byte; var pdynamicip: byte; var pfindserver: byte): boolean; far; stdcall; external 'sc504.dll';
- procedure SendConfig(ID: Integer; var ClienteAdd: byte; var ServerAdd : byte; var NetMaskAdd: byte; var GatewayAdd: byte; var NameserverAdd: byte; var TCNameAdd: byte; PortsvAdd: WORD; var FTPServerAdd: byte; var FTPUserAdd: byte; var FTPPAssAdd: byte; bDHCP: DWORD; AutoFind: DWORD); far; stdcall; external 'sc504.dll';
- procedure PedeConfig(ID: Integer); far; stdcall; external 'sc504.dll'; procedure EnviaTexto(ID: Integer; Posx: Word; Posy: Word; var Text2Show: byte; var Fonte: byte; TextColor: Word; BgColor: Word); far; stdcall; external 'sc504.dll';
- procedure ClearDisplay(ID: Integer; Color : Integer); far; stdcall; external 'sc504.dll';
- procedure EnviaImagem(ID: Integer; var APaleta: byte; var AImagem: byte); stdcall; external 'sc504.dll';
- procedure SetaTempoExib(ID: Integer; TempoEx: word); far; stdcall; external 'sc504.dll';
- procedure PedeTempoExib(ID: Integer); far; stdcall; external 'sc504.dll';
- procedure UpdateSoft(ID: Integer); far; stdcall; external 'sc504.dll';
- procedure StopAdv(ID: Integer); far; stdcall; external 'sc504.dll';
- procedure DeleteAdv(ID: Integer); far; stdcall; external 'sc504.dll';
- procedure HabilitaTeclado(ID: Integer; HabSim: DWORD); far; stdcall; external 'sc504.dll';
- procedure PedeHabilitaTeclado(ID: Integer); far; stdcall; external 'sc504.dll';
- procedure AtualizaAdv(ID: Integer; DoReload: DWORD); far; stdcall; external 'sc504.dll';
- procedure HabilitaLEC(ID: Integer; HabSim: DWORD); far; stdcall; external 'sc504.dll';
- procedure PedeHabilitaLEC(ID: Integer); far; stdcall; external 'sc504.dll';
- procedure AbreSerialCOM(ID: Integer; SerCOM: byte; SimAbre: byte; Baud: integer; parity: byte; databits: byte); far; stdcall; external 'sc504.dll';
- procedure EscreveSerial(ID: Integer; SerCOM : byte; TamBuf: integer; var sbuf: byte); far; stdcall; external 'sc504.dll';
- procedure EnviaArquivo(ID: Integer; var localfilename: byte; var destfilename: byte); far; stdcall; external 'sc504.dll';
- procedure StartServerTC504; far; stdcall; external 'sc504.dll';
- procedure InitTC504; far; stdcall; external 'sc504.dll';
- function GetTabConectados(nada: integer): TTABSOCK; far; stdcall; external 'sc504.dll';
- function GetSerial(var ID: integer; var Porta: integer;var buffer: byte; var Nbr: integer): boolean; far; stdcall; external 'sc504.dll';
- function GetLEC(var ID: integer; var ptrilha: byte; var errcode: integer): boolean; far; stdcall; external 'sc504.dll';

function Get_KBD_char(var ID: integer; var caracter: byte): boolean; far; stdcall; external 'sc504.dll';

A troca de mensagens do programa principal com a DLL

Para aumentar a agilidade de troca de informação da DLL com o programa principal do servidor e evitar processamento desnecessário, foi implementado a troca de mensagens. A DLL envia mensagens para o programa principal 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 **tc_startserver** da seguinte forma:

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

tc_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 mensagem. Para isso devemos seguir os seguintes passos:

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

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

2) Devemos então, "reescrever" esta função (unit1.cpp):

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

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

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