BIBLIOTECA DE COMUNICAÇÃO

TRIX

XPComLib

Versão deste manual: 2.02 (08/1999)

Introdução

A Biblioteca de Comunicação da TRIX - XPComLib foi desenvolvida com o intuito de facilitar o desenvolvimento de aplicativos em microcomputadores que se comunicam com os Coletores de Dados de fabricação da TRIX Tecnologia Ltda. Todos os modelos de coletores de dados disponíveis dispõem de pelo menos uma interface de comunicação serial assíncrona padrão RS-232C, RS-485 ou RS-422 que permitem a transferência de informações com outros equipamentos. Esta transferência pode ser feita utilizando vários tipos de protocolos e meios físicos, dependendo do modelo de coletor sendo utilizado.

A XPComLib foi desenvolvida para utilização com aplicativos desenvolvidos para microcomputadores compatíveis com o PC da IBM e que utilizem o Sistema Operacional MS-DOS ou o Ambiente Operacional Windows ® 3.1, Windows for WorkGroups ® 3.11, Windows ® 95/98/NT. Para permitir o desenvolvimento em várias linguagens foram desenvolvidas versões da XPComLib específicas para cada linguagem e ambiente operacional suportado. Esta versão da XPComLib suporta os seguintes ambientes de desenvolvimento:

Sistema Operacional MS-DOS:

- Compilador C da Microsoft (versão 6.0 ou superior);
- Compilador C da Borland (versão 3.1 ou superior);
- Compilador Clipper ® versão 5.XX, EXOSPACE só é suportado em canais usando interrupções de hardware abaixo de 8.

Ambiente Operacional Windows ®:

Qualquer linguagem que permita utilização de funções de uma DLL (*Dynamic Link Library*) como por exemplo:

- Visual C++ da Microsoft versão 1.5;
- Visual Basic da Microsoft versão 3.0, acompanha exemplo para 16 bits;
- Visual Basic da Microsoft versão 4.0, acompanha exemplo para 32 bits;
- Borland C/C++ versão 3.1;
- Delphi versão 1.0, acompanha exemplo para 16 bits;
- Delphi versão 2.0, acompanha exemplo para 32 bits.

A biblioteca para Windows ® possui duas versões de DLL uma para 16 bits usada com Windows ® 3.X e outra para 32 bits usada com Windows ® 95/98/NT.

Instalação da Biblioteca a partir de Ambiente DOS

Para iniciar a instalação do programa o cursor deverá estar no aviso do DOS. Insira o disquete de instalação no drive A: ou B: e digite <unidade do drive>:INSTALA. Por exemplo para instalar a XPCOMLIB a partir do drive A: digite:

A:INSTALA

Digite <enter>, em seguida o programa de instalação solicitará o nome do diretório onde será instalado a XPCOMLIB.

Caso seja feita a instalação no mesmo diretório que já contém uma versão anterior da XPCOMLIB será solicitada a confirmação para sobrescrever os arquivos já existentes.

Em computadores com monitores monocromáticos que alteram as cores originais pode ser necessário executar o comando MODE CO80 no prompt do DOS antes de executar o instalador da XPCOMLIB.

Instalação da Biblioteca a partir de Ambiente WINDOWS

Insira o disquete de instalação no drive A: ou B: e execute o instalador digitando <unidade do drive>:INSTALA.

Para instalar a XPCOMLIB a partir do drive A: no ambiente Windows 3.X entre no Gerenciador de Programa, selecione a opção “Executar...” do menu “Arquivos” e digite:

A:INSTALA

Para instalar a XPCOMLIB a partir do drive A: no ambiente Windows ® 95/98/NT Selecione o botão “Iniciar”, escolha a opção “Executar...” e digite:

A:INSTALA

Selecione o botão OK e siga as instruções de instalação..

Caso seja feita a instalação no mesmo diretório que já contém uma versão anterior da XPCOMLIB será solicitada a confirmação para sobrescrever os arquivos já existentes.

Estrutura da Biblioteca de Comunicação

A “Biblioteca de Comunicação TRIX” consiste em três bibliotecas independentes para os ambientes suportados:

Biblioteca de funções de comunicação para linguagem C
Biblioteca de funções de comunicação para linguagem Clipper ®
Biblioteca de funções de comunicação para ambiente Windows ®

Cada uma destas bibliotecas está dividida em dois conjuntos de funções:

Funções Básicas de Comunicação que fazem a interface diretamente com o *hardware*.
Funções de nível superior com a implementação de protocolos e conversores.

As “Funções Básicas de Comunicação” fornecem interface para desenvolvimento de protocolos de comunicação ou acesso ao hardware. Permitem manipular os sinais de controle de comunicação e o recebimento ou envio de caracteres ou mensagens sem nenhum protocolo.

As “Funções de Nível Superior” fornecem a um aplicativo interface para utilização do protocolo XPnet, do protocolo XMODEM, da Rede de Rádio Freqüência (somente para Windows em 32 bits) e para conversão de formato de arquivos. Estas funções utilizam as “Funções Básicas de Comunicação” para a implementação dos protocolos de comunicação.

A “Biblioteca de Comunicação TRIX” pode também ser utilizada em aplicativos executados em sistemas operacionais multi-tarefa ou em redes locais.

Protocolo XPNET

A “Biblioteca de Comunicação TRIX” suporta uma configuração de rede utilizando como equipamento concentrador um microcomputador compatível com IBM-PC/XT/AT/386/486, com uma placa de expansão MOS-485, utilizando as interfaces elétricas RS-485 ou RS-422. Também são suportadas placas de expansão compatíveis com a IBM Serial Adapter, utilizando interface elétrica RS-232C.

A interface RS-485 só pode ser usada com o protocolo XPnet ou um outro protocolo proprietário, desenvolvido pelo usuário. Com outros protocolos só é possível ligar um único coletor ao concentrador e é obrigatório o usu da interface elétrica RS-232 ou RS-422.

A interface elétrica RS-232C somente pode ser utilizada em configurações ponto a ponto, isto é, permite somente um coletor conectado na rede e ligado diretamente ao concentrador. Utilizando um conversor RS-232 para RS-422 pode-se ter uma rede com mais de um coletor ligado a uma saída RS232 do concentrador ou ligado a um modem remoto acessado pelo concentrador via linha telefônica.

Utilizando as “Funções de Comunicação com a Rede XPnet” o usuário pode controlar uma rede com o protocolo XPnet. Esta biblioteca permite a transferência de dados entre o equipamento concentrador e até 32 dispositivos que utilizem o protocolo XPnet, como por exemplo os terminais de coleta de dados XTM-101, XTM-101 Plus, XTM-102 Plus, XTM-Compact, XTM-Compact Plus e Smart-Block da TRIX.

Como o protocolo XPnet baseia-se em um meio físico compartilhado por diversos dispositivos e o equipamento concentrador, faz-se necessário que todos os equipamentos a ela conectados possuam uma identificação lógica individual. Esta identificação é feita por um número entre 1 e 32, sendo o concentrador o número 0.

Todos os dispositivos, com exceção do equipamento concentrador, conectados na rede utilizando o protocolo XPnet são passivos, isto é, não iniciam um procedimento de comunicação. Dessa forma qualquer operação de comunicação na rede deve ser controlada pelo equipamento concentrador, utilizando as “Funções de Comunicação com a Rede XPnet” em seu aplicativo.

Como o protocolo XPnet somente permite comunicação entre o equipamento concentrador e dispositivos conectados a uma rede, qualquer transferência de dados entre dispositivos deve ser feita pelo aplicativo do equipamento concentrador através de procedimento próprio.

Protocolo XMODEM

Este protocolo deve ser utilizado na comunicação ponto a ponto. Permite que um coletor conectado diretamente ao concentrador receba ou envie arquivos de dados. É utilizado normalmente com coletores portáteis e necessita que tanto o coletor quanto o concentrador sejam colocados no modo de transmissão de dados.

Protocolo para Rede de Rádio Freqüência

Este protocolo foi desenvolvido com o intuito de facilitar o desenvolvimento de aplicativos em microcomputadores que se comunicam via radio com os Coletores de Dados de fabricação da TRIX Tecnologia Ltda.

Existe hoje apenas uma versão da DLL que suporta RF, que é a XPCOM32.DLL para Windows ® 95/98/NT.

Nesta rede o computador, chamado de Host, comunica-se com um equipamento concentrador denominado Base de RF, via interface de comunicação serial ponto a ponto assíncrona, padrão RS-232C ou RS-422. A Base de RF, por sua vez, comunica-se diretamente com os coletores portáteis via link de RF ou, se o coletor estiver fora de alcance, redireciona as mensagens para outras bases repetidoras, via interface de comunicação assíncrona RS-485. Essas outras bases, denominadas Repetidores, tentarão por sua vez se comunicar com os coletores via link de RF. Os coletores portáteis são denominados Terminais, neste documento.

A comunicação entre Host e Base de FR é feita utilizando protocolo com correção de erro.

Essas funções suportam uma configuração de rede ponto a ponto, utilizando como host um computador compatível com IBM-PC/XT/AT/386/486 com uma placa de expansão MOS-485, utilizando a interface elétrica RS-422. Também são suportadas placas de expansão compatíveis com a IBM Serial Adapter, utilizando interface elétrica RS-232C. A interface elétrica RS-485 não é suportada pela Base de RF para comunicação com o Host.

Utilizando as Funções de Comunicação de RF o usuário pode controlar uma rede composta de uma Base de RF, interligada em até 31 Repetidores que se comunicam por rádio com até 63 Terminais de coleta de dados JOBBY da TRIX. As funções de concentrador são desempenhadas unicamente pela Base de RF.

Como a rede de RF baseia-se em dois meios físicos compartilhados pelos Repetidores, pelos Terminais e a Base de RF (um dos meios físicos é o Link de Cabo entre a Base de RF e os Repetidores e o outro é o Link de Rádio entre Repetidores e Terminais ou Base de RF e Terminais), faz-se necessário que todos os equipamentos a ela conectados possuam uma identificação lógica individual. No Link de Rádio, esta identificação é feita por um número entre 0 e 63, tendo a Base de RF o número 0. No Link Base de RF-Repetidores, a identificação é feita por um número de 0 a 31, tendo a Base de RF o número 0.

No Link Base de RF-Repetidores, os repetidores são passivos e a base é ativa, ou seja, somente ela inicia o procedimento de comunicação com os coletores, através de uma operações de polling. O tráfego de mensagens pelo link é transparente para o microcomputador, só sendo mencionado nas funções de estatísticas e configuração.

No Link de Rádio os terminais, diferentemente da rede Xpnet, são dispositivos ativos, isto é, têm poder de iniciar um procedimento de comunicação com a Base de RF e consequentemente com o Host.

No Link Base de RF-Host, os dois são ativos, com comunicação full-duplex. A comunicação half-duplex, com o Host funcionando como elemento ativo é possível, mas não está disponível nesta versão da XPComlib.

Como o protocolo RF somente permite comunicação entre a Base de RF e os terminais, qualquer transferência de dados entre terminais deve ser feita pelo aplicativo do equipamento host, através de procedimento próprio.

Funções de conversão de formato de arquivo

Estas funções permitem que arquivos sejam convertidos entre o formato XPBASIC e arquivos no formato TEXTO, ASC, DBF ou LOTUS (somente em DOS).

Os arquivos na memória dos coletores TRIX têm o formato XPBASIC e devem ser convertidos após serem recolhidos para o concentrador.

Antes de enviar um arquivo ao coletor é necessário que ele seja convertido ao formato XPBASIC.

Biblioteca de Funções de Comunicação para Ambiente Windows ®

Apresentação

A “Biblioteca de Funções de Comunicação para Ambiente Windows ®” estão em arquivos tipo “DLL” (*Dynamic Link Library*), ou seja, as bibliotecas de funções que fornecem a um aplicativo interface de comunicação com os coletores de dados TRIX, utilizando como equipamento concentrador um microcomputador compatível com IBM-PC AT/386/486.

Existem hoje duas versões da DLL:

XPCOM16.DLL para Windows ® 3.1 e Windows for WorkGroups ® 3.11;
XPCOM32.DLL para Windows ® 95/98/NT.

Ambas DLLs foram testadas com aplicativos desenvolvidos em Visual C++ 1.5 e 4.0, Visual Basic 3.0 e 4.0 e Delphi 1.0, 2.0 e 3.0. Pelo fato de terem sido desenvolvidas como DLLs padrão devem funcionar com qualquer linguagem de desenvolvimento que suporte a chamada de funções em uma DLL.

O tipo de interface elétrica suportado por cada DLL é mostrado abaixo:

Tipo de interface	Serial RS-232C	Placa MOS-485	Placa MOS-485	Placa MOS-485 Rev E

DLL		2 fios	4 fios	ou superior, com 2 ou 4 fios
XPCOM16.DLL	OK	OK	OK	OK
XPCOM32.DLL	OK	----- (*)	OK	OK

(*) Notar que a versão da DLL para Windows ® 95/98/NT (XPCOM32.DLL) não suporta operação a dois fios com a placa MOS-485 até revisão "D". A Revisão "E" da placa MOS-485 permite a operação a dois fios com Windows ® 95/98/NT bem como melhora a performance de operação no Windows ® 3.X, deve ser configurada com o *jumper* de modo AUTO em OFF (aberto).

Acompanham a "Biblioteca de Funções de Comunicação para Ambiente Windows ®" os arquivos abaixo que incluem arquivos de *header* para serem incluídos na aplicação, em versões codificadas em Delphi ®, C e Visual Basic ®:

XPCOM16.DLL	DLL para Windows ® 3.x, 16 bits
XPCOM16.H	Arquivo com declarações para linguagem C (Visual C 1.5) , 16 bits
XPCOM16.LIB	Arquivo para Link com linguagem C (Visual C 1.5) , 16 bits
XPCOM16.BAS	Arquivo com declarações para Visual Basic 3.0, 16 bits
XPCOM16.PAS	Arquivo com declarações para Delphi 1.0, 16 bits
XPCOM32.DLL	DLL para Windows ® 95/98/NT, 32 bits
XPCOM32.H	Arquivo com declarações para linguagem C (Visual C 4.0) , 32 bits
XPCOM32.LIB	Arquivo para Link com linguagem C (Visual C 4.0) , 32 bits
XPCOM32.BAS	Arquivo com declarações para Visual Basic 4.0, 32 bits
XPCOM32.PAS	Arquivo com declarações para Delphi 2.0, 32 bits

Analogamente, deverão ser criadas versões destes arquivos conforme a linguagem de programação que será utilizada na aplicação.

Compilação e *Linkagem*

O arquivo "XPCOM16.DLL" para Windows ® 3.1x ou "XPCOM32.DLL" para Windows ® 95/98/NT, que é a biblioteca propriamente dita, deve estar localizado num subdiretório que o programa tenha acesso. Pode ser seu próprio diretório, o diretório SYSTEM do Windows ® ou outro qualquer que esteja no *path* do sistema.

A Compilação e *Linkagem* serão feitas de acordo com a linguagem de programação utilizada.

Os exemplos ao final de cada função são programas escritos em Visual C 1.5. Para criar o arquivo executável a partir dos fontes em C seguir os passos abaixo:

Microsoft C:

- No compilador abrir um projeto novo do tipo "Windows application (.EXE)".
- Na versão 1.5 (para 16 bits) entrar no menu "Option", opção "Project" clicar no botão "Linker", escolher a categoria "Input" e incluir na linha "Libraries" a biblioteca "xpcom16.lib".
- Na versão 5.0 (para 32 bits) entrar no menu "Project", opção "Settings", selecionar a página "Linker", escolher a categoria "Input" e incluir na linha "Libraries" a biblioteca "xpcom32.lib".
- Copiar o exemplo para um arquivo com extensão ".C" e incluir no projeto. Alterar no início deste arquivo o nome do arquivo de inclusão de "xpcom16.h" para "xpcom32.h" conforme estiver trabalhando em ambiente de 16 ou 32 bits.
- Copiar o arquivo "xpcom16.h" ou "xpcom32.h" para o mesmo diretório do arquivo em C conforme estiver trabalhando em ambiente de 16 ou 32 bits.
- Compilar e lincar.

Visual Basic:

Deve ser incluído no projeto as declarações das funções da biblioteca que estão no módulo "XPCOM16.BAS" ou "XPCOM32.BAS".

Delphi:

Deve ser incluído no projeto as declarações das funções da biblioteca que estão no módulo "XPCOM16.PAS" ou "XPCOM32.PAS".

Outras linguagens:

Para outras linguagens deve ser criado um módulo com a declaração das funções da biblioteca.

Programa de demontração XPNETW

O programa exemplo XPNETW fornecido com a biblioteca mostra uma aplicação em Visual Basic e outra em Delphi utilizando praticamente todas as funções da DLL, permitindo a transmissão e recepção de mensagens dos terminais, ativação/desativação de polling, transferência de arquivos/programa e atualização de data e hora dos terminais. O programa XPNETW trabalha em conjunto com o programa TXRX.BAS (XPBASIC) que pode ser executado nos coletores da TRIX.

Este programa é apresentado em quatro versões: Visual Basic em 16 bits, Visual Basic em 32 bits, Delphi em 16 bits e Delphi em 32 bits.

As versões distribuidas em Visual Basic não contém os controles (VBX ou OCX) necessários para a aplicação, portanto o usuário precisa ter o Visual Basic instalado em seu computador para utilizar estes programas.

Cada uma das versões contém um projeto completo com todos os fontes que se localizam nos subdiretórios da biblioteca descritos abaixo:

WINDOWS\DEMO\VB16	Arquivos do projeto XPNETW em Visual Basic 3.0 (16 bits para Windows 3.x). Dentro do visual basic carregar o arquivo de projeto XPNETW.MAK e executar. Ocorre um erro ao carregar XPNETW.FRM. Corrigir sua propriedade "Icon" colocando o arquivo XPNETW.ICO existente no diretório onde foi instalado o projeto. O arquivo XPCOM16.BAS deve ser removido do projeto e incluído a partir de sua localização real.
WINDOWS\DEMO\VB32\XPNETW	Arquivos do projeto XPNETW em Visual Basic 4.0 (32 bits para Windows 95/98/NT). Dentro do Visual Basic carregar o arquivo de projeto XPNETW.MAK e executar. O arquivo XPCOM32.BAS deve ser removido do projeto e incluído a partir de sua localização real. Para outras versões do Visual Basic criar um novo projeto e incluir todos os arquivos necessários.
WINDOWS\DEMO\DELPHI16	Arquivos do projeto XPNETW em Delphi 1.0 (16 bits para Windows 3.X). Dentro do delphi carregar o arquivo de projeto XPNETW.DPR e executar. Foi incluido também o executável XPNETW.EXE já compilado. O arquivo XPCOM16.PAS deve ser removido do projeto e incluído a partir de sua localização real.
WINDOWS\DEMO\DELPHI32\XPNETW	Arquivos do projeto XPNETW em Delphi 2.0 (32 bits para Windows 95/98/NT). Dentro do delphi carregar o arquivo de projeto XPNETW.DPR e executar. Foi incluido também o executável XPNETW.EXE já compilado. O arquivo XPCOM32.PAS deve ser removido do projeto e incluído a partir de sua localização real.
XPBASIC\TXRX.BAS	Programa em XPBASIC para ser utilizado junto com o XPNETW. Exibe no display do coletor a mensagem recebida e permite transmissão de mensagem ao pressionar tecla.

Funções disponíveis no programa XPNETW:

Funções para operação off-line

Para o uso destas funções é necessário primeiramente abrir o canal de comunicação onde o coletor está ligado.

Funções para operação on-line

Para o uso destas funções é necessário primeiramente abrir o canal de comunicação onde o coletor está ligado.

Funções para conversão de arquivos

Para o uso destas funções não é necessário abrir o canal de comunicação.

Executar o programa XPNETW.EXE. No menu "Canal" escolher a opção "Abre Canal". Será apresentada uma tela mostrando os canais de comunicação, ainda não utilizados, que estão configurados no painel de controle do Windows. Selecionar o canal que será utilizado na comunicação com o coletor. Manter os demais parâmetros com o seu valor padrão (modo Auto a 9600). Abrir o canal selecionando o botão OK.

Caso ocorra algum erro, verifique se o canal está com o endereço e interrupção corretos, não está em conflito com outros canais, não está sendo usado por outros programas e está configurado corretamente no painel de controle do Windows.

Resetar o coletor para garantir que não há programas nem dados em sua memória e ele está preparado para receber comandos remotos. Observe no canto inferior esquerdo do display o número do coletor.

Entrar no menu "Transferências" e selecionar a opção "Atualiza Data e Hora", selecionar o número do coletor que está conectado no canal serial e escolher o botão OK.

Entrar no menu "Transferências" e selecionar a opção "Transmite Programa", selecionar o número do coletor que está conectado no canal serial e procurar o programa XPBASIC\TXRX a ser transmitido para o coletor. Este programa apresenta na tela do coletor o número de mensagens que foram transmitidas e o numero de mensagens recebidas. A cada tecla pressionada será enviada uma mensagem para o concentrador. Cada mensagem recebida do concentrador será exibida na segunda linha do display e o contador de mensagens recebidas será incrementado.

Para enviar mensagens, manualmente, ao concentrador, entrar no menu "Mensagens", opção "Transmissão de Mensagens", selecionar o número do coletor, digitar a mensagem e pressionar o botão OK.

Para receber mensagens, manualmente, no concentrador, entrar no menu “Mensagens”, opção “Recepção de Mensagens”, selecionar o número do coletor, utilizar o botão “Envia Polling” para enviar uma solicitação ao coletor e retirar uma mensagem do coletor e trazer para o buffer de recepção de mensagens da biblioteca de comunicação, no concentrador. Utilizar o botão “Retira Mensagem” para retirar uma mensagem do buffer de recepção da biblioteca para ser exibida, pelo programa aplicativo, na lista de mensagens recebidas.

Para executar automaticamente a recepção de mensagens e a retransmissão ao coletor, executar os passos:

Entrar no menu “Mensagens”, opção “Ativa Polling” para habilitar a varredura automática dos coletores.
Entrar no menu “Mensagens”, opção“Terminais no Polling” e colocar o coletor na varredura automática.
Entrar no menu “Mensagens”, opção “Recepção Automática de Mensagens” e ativar a opção “Retransmite Mernsagem Recebida”.

Programa de exemplo de aplicação de Comunicação com Rádio Freqüência

Este exemplo de aplicação, para Base de RF com coletores Jobby, mostra como deve ser feita uma aplicação para comunicação com terminais utilizando rádio freqüência e pode ser utilizado para desenvolvimento de outros programas para operação on-line.

Programa Exemplo RFTEST

O programa exemplo RFTEST fornecido com a biblioteca mostra uma aplicação em Delphi utilizando praticamente todas as funções da DLL para comunicação via Rádio Freqüência, permitindo a transmissão e recepção de mensagens dos terminais, configuração da Base de RF, etc. O programa RFTEST trabalha em conjunto com o programa RF.BAS, escrito em XPBASIC, que deve ser executado nos coletores JOBBY da TRIX.

Este programa está escrito em duas versões, uma em Delphi 3.0 e outra em Visual Basic 4.0. Cada uma das versões contém um projeto completo com todos os fontes que se localizam nos subdiretórios da biblioteca descritos abaixo:

WINDOWS\DEMO\VB32\RFTEST	Arquivos do projeto RFTEST em Visual Basic 4.0 (32 bits para Windows 95/98/NT). Dentro do visual basic carregar o arquivo de projeto RFTEST.MAK e executar. Para outras versões do Visual Basic criar um novo projeto e incluir todos os arquivos necessários. Será necessário incluir o arquivo XPCOM32.BAS no projeto e colocar a XPCOM32.DLL no diretório do projeto ou em outro diretório que esteja no path do sistema.
WINDOWS\DEMO\DELPHI32\RFTEST	Arquivos do projeto RFTEST em Delphi 3.0 (32 bits para Windows 95/98/NT). Dentro do delphi carregar o arquivo de projeto RFTEST.DPR e executar. Foi incluído também o executável RFTEST.EXE já compilado. Será necessário incluir o arquivo XPCOM32.PAS no projeto e colocar a XPCOM32.DLL no diretório do projeto ou em outro diretório que esteja no path do sistema.
XPBASIC\RF.BAS	Programa em XPBASIC para ser utilizado junto com o RFTEST. Exibe no display do coletor a mensagem recebida e permite transmissão de mensagem ao pressionar tecla.

Este programa permite exercitar as funções da Base de RF e da XPcom32.dll, através de funções de abertura e fechamento de comunicação, configuração de parâmetros, telas de estatísticas, transmissão de mensagens e recepção com retransmissão automatizada.

Executar o programa RFTEST.EXE. No menu “Canal” escolher a opção “Abre Canal”. Será apresentada uma tela mostrando os canais de comunicação, ainda não utilizados, que estão configurados no painel de controle do Windows. Selecionar o canal que será utilizado na comunicação com o rádio. Manter os demais parâmetros com o seu valor padrão. Abrir o canal selecionando o botão OK.

Caso ocorra algum erro, verifique se a base de rádio freqüência está ligada, conectada ao canal correto e configurada corretamente.

Enviar o programa do coletor usando um cabo serial para ligar o PC ao coletor:

Entrar no menu “Transferência” e selecionar a opção “Envia Programa” para enviar o programa RF.BAS ao coletor. Resetar o coletor e digitar RXPROG. Após o término da transferência do programa digitar RUN. Será apresentado um menu com três opções:

1-MESMA MSG.
2-MSGS. VARIAVEIS
3-APAGA ESTATISTICA

Ao escolher a poção 1-MESMA MSG, será solicitada a mensagem a ser enviada continuamente ao concentrador. Digite uma mensagem com no máximo 40 caracteres e o intervalo de tempo a ser aguardado entre cada transmissão, em intervalos de 0,1s.

Ao escolher a opção 2-MSGS. VARIAVEIS, o coletor solicita ao usuário uma letra ou um símbolo, o tamanho da mensagem inicial e o intervalo entre cada transmissão.

O coletor Jobby inicia a transmissão contínua de mensagens para a base, aguarda a aplicação RFTEST receber a mensagem e retransmiti-la de volta ao coletor, em seguida compara a mensagem recebida com a emviada. Acumula então estatísticas de erros de transmissão, erros de mensagem não recebida e erros de mensagem recebida diferente da transmitida.

Será exibida uma tela com a informação:

TAM	Tamanho da mensagem enviada
TX	Número de vezes que a mensagem foi transmitida
ET	Número de transmissões não efetuadas
RX	Número de mensagens recebidas
ER	Número de erros de recepção
LP	Número de loops, mensagens que foram transmitidas e recebidas corretamente.
EL	Erros de loop

Biblioteca de Funções de Comunicação para Linguagem C em Ambiente DOS

Apresentação

A "Biblioteca de Funções de Comunicação para Linguagem C" destina-se a aplicativos que serão executados em sistema operacional MS-DOS 5.0 ou superior, ou sistema operacional compatível para fornecer interface de comunicação com os coletores de dados TRIX.

Esta biblioteca apresenta duas versões:

- 'XPCOMMSC': compatível com código compilado pelo Microsoft C Compiler versão 6.0 ou superior, sendo que a *linkagem* deve ser feita utilizando-se o Microsoft Link versão 5.1 ou superior;
- 'XPCOMBC': compatível com código compilado pelo Borland C ++ versão 3.1 ou superior, sendo que a *linkagem* deve ser feita utilizando-se o Turbo Link versão 5.1 ou superior.

Outros produtos poderão ser parcialmente compatíveis.

São fornecidos com a "Biblioteca de Funções de Comunicação para Linguagem C" dois arquivos *header* a serem incluídos nos arquivos fonte codificados em C. Estes arquivos são:

- XPCOMLIB.H: contém a definição das constantes, tipos e declarações das funções;
- XPCOMDEF.H: contém a definição de constantes auxiliares.

Faz-se necessário lembrar que a biblioteca fornecida utiliza recursos de interrupção de *hardware* do microcomputador, sendo portanto imprevisível o comportamento de um outro aplicativo que faz uso dos mesmos recursos.

Compilação e *Linkagem*

Devido a algumas características da "Biblioteca de Funções de Comunicação para Linguagem C" os arquivos fonte codificados em C, que utilizam esta biblioteca devem ser compilados utilizando os modelos de memória *large* ou *huge*.

Para o Microsoft C Compiler 6.0 deve ser usado como parâmetro de compilação:

/AL - modelo *large* de memória ou
/AH - modelo *huge* de memória.

Um exemplo de compilação do arquivo ROTINA.C é o seguinte:

```
CL /c /AH ROTINA.C
```

Para o Borland C++ 3.1 deve ser usado como parâmetro de compilação:

/ml - modelo *large* de memória ou
/mh - modelo *huge* de memória.

Um exemplo de compilação do arquivo ROTINA.C é o seguinte:

```
BCC /c /ml ROTINA.C
```

Para ambos os compiladores, podem ser utilizados outros parâmetros de acordo com a aplicação do código a ser gerado, contanto que não sejam incompatíveis com os modelos de memória utilizados.

A "Biblioteca de Funções de Comunicação para Linguagem C" deve ser *linkada* com os módulos de um aplicativo e com outras bibliotecas, utilizando-se tanto o Microsoft Link versão 5.1, quanto o Turbo Link versão 5.1. Deve-se ressaltar que os modelos de memória utilizados pelos módulos e pelas bibliotecas devem ser os mesmos.

Abaixo é apresentado exemplos de *link-edição* do arquivo ROTINA.C, para cada tipo:

- Microsoft Link versão 5.1:

  LINK ROTINA , , , XPCOMMSC,,

- Turbo Link versão 5.1:

  TLINK ROTINA , , , XPCOMBC

Funções para Windows e Linguagem C para DOS por categoria

Como a "Biblioteca de Funções de Comunicação para Ambiente Windows ®" possui as mesmas características gerais da "Biblioteca de Funções de Comunicação para Linguagem C", com exceção das características particulares ao ambiente Windows ®, a descrição das funções pertencentes às duas bibliotecas são feitas na mesma seção. No início de cada função é mostrado um quadro indicando, com uma marca na coluna apropriada, a que sistema operacional cada função se aplica:

A descrição das funções da "Biblioteca de Funções de Comunicação para Windows e para Linguagem C" é feita utilizando o seguinte formato:

- nome da função em negrito;
- sumário mostrando o modelo sintático a ser utilizado;
- compatibilidade com Windows e DOS indicada com uma marca na coluna apropriada;

Windows 16	Windows 32	DOS

- descrição contendo as ações da mesma e argumentos de entrada;
- o valor retornado pela função e sua descrição;
- sugestão de consulta a funções similares ou relacionadas;
- exemplo de utilização da função descrita.

Como a biblioteca fornecida utiliza recursos de interrupção de *hardware* do microcomputador, se alguma ocorrência causar o término do aplicativo, antes da finalização operacional da rede, o sistema terá um comportamento errático e imprevisível, sendo necessário desligá-lo. Dessa forma, faz-se necessário que o aplicativo não permita intervenções para seu término através da digitação de "CTRL-BREAK" ou "CTRL-C" , o que pode ser feito interceptando o *handler* da interrupção do Coltrol-Break.

É necessário também que o aplicativo não permita ser interrompido através de erros críticos do DOS, como por exemplo o erro gerado na leitura de um disquete quando a porta de seu *drive* estiver aberta, o que pode ser feito interceptando o *handler* das interrupções que geram esses erros.

Não é recomendável que o aplicativo execute *shells* do DOS permitindo a execução de outros aplicativos com a rede em operação. Aconselha-se neste caso finalizar operacionalmente a rede antes da execução do shell, e ao seu término reinicializá-la com os argumentos originais.

Os exemplos de programas em C que seguem cada função foram escritos para o ambiente DOS ou Windows conforme a função for para uso exclusivo em um dos ambientes. No caso de funções que são iguais para o DOS e para o Windows os exemplos foram escritos para Windows. Neste último caso as funções de exemplo precisam ser adaptadas para o ambiente DOS como segue:

1. No ambiente Windows não existe a função ComAddress(), que associa a cada porta COM1 a COM16 o endereço do canal de comunicação e sua interrupção de hardware. Caso o canal de comunicação, a ser utilizado, não seja o padrão do PC esta configuração deve ser feita no próprio Windows:
   Windows versão 3.x procurar na janela "Main" o ícone "Control Panel" item "Ports". Escolher port de COM1 a COM4 que será modificado, entrar na opção "Settings...", entrar na opção "Advanced...", modificar o endereço base do port e o número da linha de requisição de interrupção (IRQ) de acordo com o cartão de comunicação instalado. Confirmar com OK, sair do Windows e reinicializar para usar a nova configuração.
   Windows 95 procurar na janela "My Computer" o ícone "Control Panel" item "Add New Hardware". Escolher o botão "Next" e responder "Não" à pergunta: "Do you want Windows to search for your new hardware?", selecionar o botão "Next". Escolher opção

"Ports (COM & LPT)" e selecionar o botão "Next". No campo "Manufacturer" deixar selecionado "Standard port types" e no campo "Models" deixar selecionado "Comunications Port". Escolher botão "Next" e aceitar o endereço do canal (Input/Output range) e sua interrupção (Interrupt request) conforme mostrado na tela. (será modificado adiante). Selecionar o botão "Next" e "Finish", responder "No" à pergunta: "Do you want to shut down your computer now?". Selecionar o ícone "System" do "Control Panel", entrar no "Device Manager" opção "Ports (COM & LPT)". Escolher a última porta configurada. Entrar em "Resources" e na opção "Setting based on:" escolher a partir de "Basic Configuration 4" um canal que possa ser modificado quando tentar entrar em "Change Setting". Selecionar "Input/Output Range", entrar em "Change setting..." e modificar endereço em "Value" conforme hardware instalado. Selecionar "Interrupt Request", entrar em "Change setting..." e modificar interrupção em "Value" conforme hardware instalado. Sair do Windows e reinicializar para usar a nova configuração.

Windows NT procurar na janela "My Computer" o ícone "Control Panel" item "Ports". Escolher botão "Add..." e escolher o valor para os parâmetros:

"COM port number:" - escolher port COM1, COM2 etc. que não esteja sendo utilizado.
"Base I/O Port Address:" - entrar o endereço base do port conforme configurado no hardware.
"Interrupt Request Line (IRQ): - entrar o número da interrupção conforme configurado no hardware.

Confirmar com OK, sair do Windows e reinicializar para usar a nova configuração.

Nos ambiente para DOS usar a função ComAddress() para associar uma porta de comunicação ao endereço de hardware e interrupção correspondente caso estes sejam diferentes dos valores padrão de COM1 a COM4.

2. Não existe a função ComTest() no ambiente Windows e por isso utilizamos a função ComOpen() tanto para abrir quanto para verificar se um canal existe e está livre. Se houver erro na abertura do canal ou ele não existe ou não está livre.
3. Algumas funções devido a diferenças nos valores de seus argumentos ou número de argumentos têm versões específicas para DOS e Windows e portanto todos os exemplos que as utilizam precisam ser adaptados para serem executados no DOS. Veja como exemplo a função ComOpen() que no ambiente DOS tem um parâmetro a mais.
4. Os arquivos de inclusão da biblioteca XPComLib para linguagem C no ambiente DOS são xpcomlib.h e xpcomdef.h e substituem xpcom16.h ou xpcom32.h usados com o Windows. Não é necessário o arquivo de inclusão windows.h no ambiente DOS.
5. Ao escrever um programa que espera a resposta do terminal antes de prosseguir, por exemplo ao solicitar status interno do terminal, é necessário permitir que outras aplicações do Windows sejam executadas enquanto o programa aguarda. Esta espera deve ser feita usando a função XPnetWaitCom no ambiente Windows. No ambiente DOS pode-se fazer um loop de espera com a função XPnetmComStatus ou usar a função XPnetmWaitTx que incorpora este loop de espera. Não deve ser feito, no ambienta Windows, um loop de espera com a função XPnetmComStatus pois como as demais aplicações não são executadas, também a DLL de comunicação não será e o programa ficará parado num loop infinito.
6. Mudar função WinMain para void main(void).

Função de Identificação da Biblioteca

Esta função é destinada para identificar a versão da biblioteca de comunicação utilizada.

XPComVersion

Sumário

```
void FAR PASCAL XPComVersion (char far * sBuffer                    // Ponteiro para buffer com ao menos 25 caracteres
                              );
```

Windows 16	Windows 32	DOS
•	•	

Descrição

A função 'XPComVersion' retorna, no *buffer* cujo endereço foi passado como parâmetro, a versão da DLL sendo usada no momento.

O argumento 'sBuffer' é um ponteiro que define a localização do *buffer* de destino. Deve ter o tamanho de pelo menos 25 caracteres.

Valor Retornado

Não há valor retornado

Consulte

ComOpen, ComClose.

Exemplo

```
#include <windows.h>
```

```
#include "xpcom16.h"

char far  sBuffer[25];

int PASCAL WinMain(HANDLE hInstance, HANDLE hPrevInstance, LPSTR lpCmdLine, int nCmdShow)
        {
        XPComVersion(sBuffer);
        MessageBox((HWND)NULL, sBuffer, "Exemplo", MB_OK);
        return TRUE;
        }
```

Funções Básicas de Comunicação

Estas funções são destinadas à inicialização, transferência de dados, verificação da comunicação e finalização através do canal selecionado. São utilizadas pelas funções que implementam o protocolo XPnet e podem ser utilizadas para implementar outros protocolos de comunicação.

ComAddress

Sumário

```
int ComAddress (int nCanal,                     // Numero do canal de comunicacao
                unsigned int wPort,             // Port base
                unsigned int nIrq               // Interrupcao
                );
```

Windows 16	Windows 32	DOS
		•

Descrição

A função 'ComAddress' associa um número de canal ao endereço de hardware do canal de comunicação e à interrupção associada a ele. Antes desta associação é feito um teste para verificar que no endereço de hardware especificado está instalado um canal de comunicação. Todas as demais funções de comunicação recebem o número do canal como parâmetro. Os canais COM1, COM2, COM3 e COM4 já estão previamente definidos com os endereços padrões do IBM-PC, mas podem ser reconfigurados se desejado.

O argumento 'nCanal' especifica o canal de comunicação, possibilita o uso de no máximo 16 canais (0 a 15), cada qual permitindo conectar até 32 terminais. As constantes, já definidas para cada canal, são apresentadas abaixo:

COM1, COM2, COM3, COM4, . . . , COM16.

O argumento 'wPort' define o *port* base de “I/O” da “UART” da placa de comunicação utilizada, sendo alguns dos valores possíveis para o argumento mostrados abaixo em decimal e hexadecimal:

```
552 = 0x0228,                   808 = 0x0328,
568 = 0x0238,                   824 = 0x0338,
616 = 0x0268,                   872 = 0x0368,
632 = 0x0278 (PRINTER),         888 = 0x0378 (PRINTER),
680 = 0x02A8,                   936 = 0x03A8,
696 = 0x02B8,                   952 = 0x03B8 (MONOCHROME ADAPTER),
744 = 0x02E8 (COM3),           1000 = 0x03E8 (COM4),
760 = 0x02F8 (COM2),           1016 = 0x03F8 (COM1).
```

O argumento 'nIrq' define o nível de interrupção da placa de comunicação utilizada, sendo alguns dos valores possíveis para o argumento:

```
2,
3 (COM2 e COM4),
4 (COM1 e COM3),
5 (LPT2),
6 (FLOPPY DISK),
7 (LPT1),
```

10,
11,
12,
15.

Valor Retornado

A função retorna zero quando sua execução for bem sucedida, caso contrário retorna um código de erro sempre menor do que zero. O significado de cada código de erro retornado é fornecido no item “Códigos de Erro”.

Consulte

ComOpen, ComClose.

Exemplo

```
#include <stdlib.h>
#include “xpcomlib.h”
#include “xpcomdef.h”

void main(void)
    {
    if (ComAddress(COM1,0x238,3))
        exit(1);
    if (ComOpen(COM1,9600,8,1,COM_NONEP,COM_NOFLOW,COM_NOFLOW,256,256, FALSE))
        exit(1);
    if (ComClose(COM1))
        exit(1);
    }
```

ComTest

Sumário

```
int ComTest (int nCanal                    // Numero do canal de comunicacao
        );
```

Windows 16	Windows 32	DOS
		•

Descrição

A função ‘ComTest’ testa o *hardware* do canal de comunicação e identifica o tipo de UART. O canal não pode estar aberto.

O argumento 'nCanal' especifica o canal de comunicação, possibilita o uso de no máximo 16 canais (COM1 a COM16).

Valor Retornado

A função retorna o tipo de UART, conforme definido no arquivo *header* ‘XPCOMDEF.H’ quando sua execução for bem sucedida:

UART8250: UART 8250;
UART16450: UART 16450 com scratchpad;
UART16550: UART 16550 com FIFOs;

Caso contrário retorna um código de erro sempre menor do que zero. O significado de cada código de erro retornado é fornecido no item “Códigos de Erro”.

Consulte

ComAddress.

Exemplo

```
#include <stdlib.h>
#include <stdio.h>
#include “xpcomlib.h”
#include “xpcomdef.h”

int nUart;

void main(void)
        {
        if (ComAddress(COM1,0x238,3))
                exit(1);
        nUart = (ComTest(COM1))
        switch(nUart)
                {
                case UART8250
                        printf(“\nUART 8250 instalada.\n”);
                        break;
                case UART16450
                        printf(“\nUART 16450 instalada.\n”);
                        break;
                case UART16550
                        printf(“\nUART 16550 instalada.\n”);
                        break;
                default:
                        exit(1);
                }
        }
```

ComOpen - Versão para Windows

Sumário

```
int FAR PASCAL ComOpen (int nCanal,              Numero do canal de comunicacao
                        int nBaud,               // Taxa de comunicacao
                        int nBits,               // Numero de bits por caractere:  7 ou 8
                        int nStop,               // Numero de stop-bits:  1 ou 2
                        int nParid,              // Tipo de paridade
                        int nTxFlow,             // Tipo de controle de fluxo na transmissao
                        int nRxFlow,             // Tipo de controle de fluxo na recepcao
                        unsigned int wTxBuffer,  // Tamanho do buffer de transmissao
                        unsigned int wRxBuffer   // Tamanho do buffer de recepcao
                       );
```

Windows 16	Windows 32	DOS
•	•	

Descrição

A função 'ComOpen' abre o canal de comunicação, definindo os parâmetros para sua operação.

O argumento 'nCanal' especifica o canal de comunicação, possibilita o uso de no máximo 16 canais (COM1 a COM16).

O argumento 'nBaud' define a taxa de comunicação a ser utilizada com 8 *bits*, 1 *stop-bit* e sem paridade, sendo os valores possíveis para o argumento mostrados abaixo em *baud*:

110, 300, 600, 1200, 2400, 4800, 9600, 19200, 38400.

O argumento ‘nBits’ informa o número de *bits* por caractere: 7 ou 8.

O argumento ‘nStop’ informa o número de *stop-bits* 1 ou 2.

O argumento ‘nParid’ informa o tipo de paridade, conforme as constantes definidas em ‘XPCOMDEF.H’:

COM_NONEP	- Sem paridade,
COM_EVENP	- Paridade par;
COM_ODDP	- Paridade ímpar.

O argumento ‘nTxFlow’ informa o tipo de controle de fluxo na transmissão, conforme as constantes definidas em ‘XPCOMDEF.H’:

COM_NOFLOW	- Sem controle de fluxo,
COM_RTSCTS	- Controle de fluxo via RTS/CTS,
COM_XONXOFF	- Controle de fluxo via XON/XOFF.

O argumento ‘nRxFlow’ informa o tipo de controle de fluxo na recepção, conforme as constantes definidas em ‘XPCOMDEF.H’:

COM_NOFLOW	- Sem controle de fluxo,
COM_RTSCTS	- Controle de fluxo via RTS/CTS,
COM_XONXOFF	- Controle de fluxo via XON/XOFF.

O argumento 'wTxBuffer' define o espaço a ser alocado para o *buffer* de transmissão de mensagens, podendo ser um valor de 16 *bytes* a 65500 *bytes*.

O argumento 'wRxBuffer' define o espaço a ser alocado para o *buffer* de recepção de mensagens, podendo ser um valor de 16 *bytes* a 65500 *bytes*.

Valor Retornado

A função retorna zero quando sua execução for bem sucedida, caso contrário retorna um código de erro sempre menor do que zero. O significado de cada código de erro retornado é fornecido no item “Códigos de Erro”.

Consulte

ComClose, ComConfig.

Exemplo

```
#include <windows.h>
#include “xpcom16.h”

int PASCAL WinMain(HANDLE hInstance, HANDLE hPrevInstance, LPSTR lpCmdLine, int nCmdShow)
    {
    if (ComOpen(COM1,9600,8,1,COM_NONEP,COM_NOFLOW,COM_NOFLOW,256,256))
        return FALSE;
    if (ComClose(COM1))
        return FALSE;
    return TRUE;
    }
```

ComOpen - Versão para DOS

Sumário

```
int ComOpen (int nCanal,                    // Numero do canal de comunicacao
             int nBaud,                     // Taxa de comunicacao
             int nBits,                     // Numero de bits por caractere:  7 ou 8
             int nStop,                     // Numero de stop-bits:  1 ou 2
             int nParid,                    // Tipo de paridade
             int nTxFlow,                   // Tipo de controle de fluxo na transmissao
```

```
  int nRxFlow,                          // Tipo de controle de fluxo na recepcao
  unsigned int wTxBuffer,               // Tamanho do buffer de transmissao
  unsigned int wRxBuffer,               // Tamanho do buffer de recepcao
  bool b16550                           // Tipo da UART
);
```

Windows 16	Windows 32	DOS
		•

Descrição

A função 'ComOpen' abre o canal de comunicação, definindo os parâmetros para sua operação. Esta função deve ser executada após 'ComAddress' que inicializa a rede para uma configuração de *hardware* específica. Não é necessário executar 'ComAddress' para utilizar os canais COM1 a COM4 na sua configuração padrão.

O argumento 'nCanal' especifica o canal de comunicação, possibilita o uso de no máximo 16 canais (COM1 a COM16).

O argumento 'nBaud' define a taxa de comunicação a ser utilizada com 8 *bits*, 1 *stop-bit* e sem paridade, sendo os valores possíveis para o argumento mostrados abaixo em *baud*:

600, 1200, 2400, 4800, 9600, 19200.

O argumento 'nBits' informa o número de *bits* por caractere: 7 ou 8.

O argumento 'nStop' informa o número de *stop-bits* 1 ou 2.

O argumento 'nParid' informa o tipo de paridade, conforme as constantes definidas em 'XPCOMDEF.H':

COM_NONEP	- Sem paridade,
COM_EVENP	- Paridade par;
COM_ODDP	- Paridade ímpar.

O argumento 'nTxFlow' informa o tipo de controle de fluxo na transmissão, conforme as constantes definidas em 'XPCOMDEF.H':

COM_NOFLOW	- Sem controle de fluxo,
COM_RTSCTS	- Controle de fluxo via RTS/CTS,
COM_XONXOFF	- Controle de fluxo via XON/XOFF.

O argumento 'nRxFlow' informa o tipo de controle de fluxo na recepção, conforme as constantes definidas em 'XPCOMDEF.H':

COM_NOFLOW	- Sem controle de fluxo,
COM_RTSCTS	- Controle de fluxo via RTS/CTS,
COM_XONXOFF	- Controle de fluxo via XON/XOFF.

O argumento 'wTxBuffer' define o espaço a ser alocado para o *buffer* de transmissão de mensagens, podendo ser um valor de 16 *bytes* a 65500 *bytes*.

O argumento 'wRxBuffer' define o espaço a ser alocado para o *buffer* de recepção de mensagens, podendo ser um valor de 16 *bytes* a 65500 *bytes*.

O argumento 'b16550' pode ser TRUE ou FALSE e especifica que iremos usar o FIFO da UART 16550 caso seja detectada no canal de comunicação.

Valor Retornado

A função retorna zero quando sua execução for bem sucedida, caso contrário retorna um código de erro sempre menor do que zero. O significado de cada código de erro retornado é fornecido no item "Códigos de Erro".

Consulte

ComAddress, ComClose, ComConfig.

Exemplo

```
#include <stdlib.h>
#include “xpcomlib.h”
#include “xpcomdef.h”

void main(void)
        {
        if (ComAddress(COM1,0x238,3))
                exit(1);
        if (ComOpen(COM1,9600,8,1,COM_NONEP,COM_NOFLOW,COM_NOFLOW,256,256, FALSE))
                exit(1);
        if (ComClose(COM1))
                exit(1);
        }
```

ComTxHandler

Sumário

```
int ComTxHandler (int nCanal,                                   // Numero do canal de comunicacao
                  int *pNewTxHandler(struct ComStateS *)        // Ponteiro para novo handler de transmissao ou NULL
                 );
```

Windows 16	Windows 32	DOS
		•

Descrição

A função ‘ComTxHandler’ permite instalar um novo *handler* (programa para tratamento de interrupção) de transmissão, diferente do utilizado pela biblioteca.

O argumento 'nCanal' especifica o canal de comunicação, possibilita o uso de no máximo 16 canais (COM1 a COM16).

O argumento ‘*pNewTxHandler’ é um ponteiro para o novo *handler* de transmissão a ser utilizado pelo canal de comunicação especificado, ou NULL para voltar a instalar o *handler* de transmissão original. O *handler* retira um caractere do *buffer* de transmissão e envia ao canal, após retorna TRUE se o caractere foi transmitido ou FALSE caso contrário. A rotina do *handler* tem como parâmetro um ponteiro para estrutura do tipo ‘ComStateS’ conforme definido em ‘XPCOMLIB.H’.

Valor Retornado

A função retorna zero quando sua execução for bem sucedida, caso contrário retorna um código de erro sempre menor do que zero. O significado de cada código de erro retornado é fornecido no item “Códigos de Erro”.

Consulte

ComAddress, ComOpen, ComRxHandler.

Exemplo

```
#include <stdlib.h>
#include “xpcomlib.h”
#include “xpcomdef.h”

void main(void)
        {
        if (ComAddress(COM1,0x238,3))
                exit(1);
        if (ComOpen(COM1,9600,8,1,COM_NONEP,COM_NOFLOW,COM_NOFLOW,256,256, FALSE))
```

```
        exit(1);
    if (ComTxHandler(COM1,pNewTxHandler))
        exit(1);
    if (ComClose(COM1))
        exit(1);
    }
```

ComRxHandler

Sumário

```
int ComRxHandler (int nCanal,                                   // Numero do canal de comunicacao
                  void  *pNewRxHandler(struct ComStateS *)      // Ponteiro para novo handler de recepcao ou NULL
                  );
```

Windows 16	Windows 32	DOS
		•

Descrição

A função ‘ComRxHandler’ permite instalar um novo *handler* (programa para tratamento de interrupção) de recepção, diferente do utilizado pela biblioteca.

O argumento 'nCanal' especifica o canal de comunicação, possibilita o uso de no máximo 16 canais (COM1 a COM16).

O argumento ‘*pNewRxHandler’ é um ponteiro para o novo *handler* de recepção a ser utilizado pelo canal de comunicação especificado, ou NULL para voltar a instalar o *handler* de recepção original. O *handler* recebe um caractere do canal e insere no *buffer* de recepção. A rotina do *handler* tem como parâmetro um ponteiro para estrutura do tipo ‘ComStateS’ conforme definido em ‘XPCOMLIB.H’.

Valor Retornado

A função retorna zero quando sua execução for bem sucedida, caso contrário retorna um código de erro sempre menor do que zero. O significado de cada código de erro retornado é fornecido no item “Códigos de Erro”.

Consulte

ComAddress, ComOpen, ComTxHandler.

Exemplo

```
#include <stdlib.h>
#include “xpcomlib.h”
#include “xpcomdef.h”

void main(void)
    {
    if (ComAddress(COM1,0x238,3))
        exit(1);
    if (ComOpen(COM1,9600,8,1,COM_NONEP,COM_NOFLOW,COM_NOFLOW,256,256, FALSE))
        exit(1);
    if (ComRxHandler(COM1,pNewRxHandler))
        exit(1);
    if (ComClose(COM1))
        exit(1);
    }
```

ComConfig - Versão para Windows

Sumário

```
int FAR PASCAL ComConfig (int nCanal,        // Numero do canal de comunicacao
                          int nBaud,         // Taxa de comunicacao
                          int nBits,              // Numero de bits por caractere:  7 ou 8
                          int nStop,         // Numero de stop-bits:  1 ou 2
                          int nParid,        // Tipo de paridade
                          int nTxFlow,       // Tipo de controle de fluxo na transmissao
                          int nRxFlow        // Tipo de controle de fluxo na recepcao
                         );
```

Windows 16	Windows 32	DOS
•	•	

Descrição

A função ’ComConfig’ reconfigura um canal de comunicação já aberto, definindo os parâmetros para sua operação. Esta função deve ser executada quando se deseja alterar a configuração previamente estabelecida para um determinado canal de comunuicação.

O argumento 'nCanal' especifica o canal de comunicação, possibilita o uso de no máximo 16 canais (COM1 a COM16).

O argumento 'nBaud' define a taxa de comunicação a ser utilizada com 8 *bits*, 1 *stop-bit* e sem paridade, sendo os valores possíveis para o argumento mostrados abaixo em *baud*:

110, 300, 600, 1200, 2400, 4800, 9600, 19200.

O argumento ‘nBits’ informa o número de *bits* por caractere: 7 ou 8.

O argumento ‘nStop’ informa o número de *stop-bits* 1 ou 2.

O argumento ‘nParid’ informa o tipo de paridade, conforme as constantes definidas em ‘XPCOMDEF.H’:

COM_NONEP	- Sem paridade,
COM_EVENP	- Paridade par;
COM_ODDP	- Paridade ímpar.

O argumento ‘nTxFlow’ informa o tipo de controle de fluxo na transmissão, conforme as constantes definidas em ‘XPCOMDEF.H’:

COM_NOFLOW	- Sem controle de fluxo,
COM_RTSCTS	- Controle de fluxo via RTS/CTS,
COM_XONXOFF	- Controle de fluxo via XON/XOFF.

O argumento ‘nRxFlow’ informa o tipo de controle de fluxo na recepção, conforme as constantes definidas em ‘XPCOMDEF.H’:

COM_NOFLOW	- Sem controle de fluxo,
COM_RTSCTS	- Controle de fluxo via RTS/CTS,
COM_XONXOFF	- Controle de fluxo via XON/XOFF.

Valor Retornado

A função retorna zero quando sua execução for bem sucedida, caso contrário retorna um código de erro sempre menor do que zero. O significado de cada código de erro retornado é fornecido no item “Códigos de Erro”.

Consulte

ComOpen.

Exemplo

```
#include <windows.h>
#include “xpcom16.h”
```

```
int PASCAL WinMain(HANDLE hInstance, HANDLE hPrevInstance, LPSTR lpCmdLine, int nCmdShow)
        {
        if (ComOpen(COM1,9600,8,1,COM_NONEP,COM_NOFLOW,COM_NOFLOW,256,256))
                return FALSE;
        if (ComConfig(COM1,2400,8,1,COM_NONEP,COM_NOFLOW,COM_NOFLOW))
                return FALSE;
        if (ComClose(COM1))
                return FALSE;
        return TRUE;
        }
```

ComConfig - Versão para DOS

Sumário

```
int ComConfig (int nCanal,              // Numero do canal de comunicacao
               int nBaud,               // Taxa de comunicacao
               int nBits,               // Numero de bits por caractere:  7 ou 8
               int nStop,               // Numero de stop-bits:  1 ou 2
               int nParid,              // Tipo de paridade
               int nTxFlow,                     // Tipo de controle de fluxo na transmissao
               int nRxFlow              // Tipo de controle de fluxo na recepcao
              );
```

Windows 16	Windows 32	DOS
		•

Descrição

A função 'ComConfig' reconfigura um canal de comunicação já aberto, definindo os parâmetros para sua operação. Esta função deve ser executada quando se deseja alterar a configuração previamente estabelecida para um determinado canal de comunuicação.

O argumento 'nCanal' especifica o canal de comunicação, possibilita o uso de no máximo 16 canais (COM1 a COM16).

O argumento 'nBaud' define a taxa de comunicação a ser utilizada com 8 *bits*, 1 *stop-bit* e sem paridade, sendo os valores possíveis para o argumento mostrados abaixo em *baud*:

600, 1200, 2400, 4800, 9600, 19200.

O argumento 'nBits' informa o número de *bits* por caractere: 7 ou 8.

O argumento 'nStop' informa o número de *stop-bits* 1 ou 2.

O argumento 'nParid' informa o tipo de paridade, conforme as constantes definidas em 'XPCOMDEF.H':

COM_NONEP	- Sem paridade,
COM_EVENP	- Paridade par;
COM_ODDP	- Paridade ímpar.

O argumento 'nTxFlow' informa o tipo de controle de fluxo na transmissão, conforme as constantes definidas em 'XPCOMDEF.H':

COM_NOFLOW	- Sem controle de fluxo,
COM_RTSCTS	- Controle de fluxo via RTS/CTS,
COM_XONXOFF	- Controle de fluxo via XON/XOFF.

O argumento 'nRxFlow' informa o tipo de controle de fluxo na recepção, conforme as constantes definidas em 'XPCOMDEF.H':

COM_NOFLOW - Sem controle de fluxo,
COM_RTSCTS - Controle de fluxo via RTS/CTS,
COM_XONXOFF - Controle de fluxo via XON/XOFF.

Valor Retornado

A função retorna zero quando sua execução for bem sucedida, caso contrário retorna um código de erro sempre menor do que zero. O significado de cada código de erro retornado é fornecido no item “Códigos de Erro”.

Consulte

ComAddress, ComOpen.

Exemplo

```
#include <stdlib.h>
#include “xpcomlib.h”
#include “xpcomdef.h”

void main(void)
    {
    if (ComAddress(COM1,0x238,3))
        exit(1);
    if (ComOpen(COM1,9600,8,1,COM_NONEP,COM_NOFLOW,COM_NOFLOW,256,256, FALSE))
        exit(1);
    if (ComConfig(COM1,2400,8,1,COM_NONEP,COM_NOFLOW,COM_NOFLOW))
        exit(1);
    if (ComClose(COM1))
        exit(1);
    }
```

ComClose

Sumário

```
int FAR PASCAL ComClose (int nCanal                 // Numero do canal de comunicacao
                        );
```

Windows 16	Windows 32	DOS
•	•	•

Descrição

A função ‘ComClose’ fecha o canal de comunicação aberto anteriormente.

O argumento 'nCanal' especifica o canal de comunicação, possibilita o uso de no máximo 16 canais (COM1 a COM16).

Valor Retornado

A função retorna zero quando sua execução for bem sucedida, caso contrário retorna um código de erro sempre menor do que zero. O significado de cada código de erro retornado é fornecido no item “Códigos de Erro”.

Consulte

ComAddress, ComOpen.

Exemplo

```
#include <windows.h>
#include “xpcom16.h”
```

```
int PASCAL WinMain(HANDLE hInstance, HANDLE hPrevInstance, LPSTR lpCmdLine, int nCmdShow)
        {
        if (ComOpen(COM1,9600,8,1,COM_NONEP,COM_NOFLOW,COM_NOFLOW,256,256))
                return FALSE;
        if (ComClose(COM1))
                return FALSE;
        return TRUE;
        }
```

ComGetConfig - Versão para Windows

Sumário

```
long FAR PASCAL ComGetConfig (int nCanal,       // Numero do canal de comunicacao
                              int nInfo         // Tipo de informacao desejada
                              );
```

Windows 16	Windows 32	DOS
•	•	

Descrição

A função 'ComGetConfig' é utilizada para se obter informações diversas sobre a configuração do canal

O argumento 'nCanal' especifica o canal de comunicação, possibilita o uso de no máximo 16 canais (COM1 a COM16).

O argumento 'nInfo' determina a informação desejada, é definido conforme as constantes abaixo, que já estão declaradas no arquivo *header* 'XPCOMDEF.H' da biblioteca:

COMCFG_OPEN:	Informa se o canal de comunicação está aberto;
COMCFG_RATE:	Informa o valor da taxa de comunicação;
COMCFG_NBITS:	Informa o número de *bits* por caractere 7 ou 8;
COMCFG_NSTOP:	Informa o número de *stop bits* 1 ou 2;
COMCFG_PAR:	Informa o tipo de paridade;
COMCFG_TXFLOW:	Informa o tipo do controle de fluxo na transmissão;
COMCFG_RXFLOW:	Informa o tipo do controle de fluxo na recepção;
COMCFG_BUFRX:	Informa o tamanho do *buffer* de recepção;
COMCFG_BUFTX:	Informa o tamanho do *buffer* de transmissão;

Valor Retornado

O valor retornado pela função varia de acordo com o especificado no argumento 'nInfo', relacionados abaixo:

COMCFG_OPEN:	Retorna TRUE ou 1 se o canal está aberto, FALSE ou 0 caso contrário;
COMCFG_RATE:	Retorna o valor da taxa de comunicação, caso contrário retorna um código de erro;
COMCFG_NBITS:	Retorna o número de *bits* por caractere 7 ou 8, caso contrário retorna um código de erro;
COMCFG_NSTOP:	Retorna o número de *stop bits* 1 ou 2, caso contrário retorna um código de erro;
COMCFG_PAR:	Retorna o tipo de paridade conforme as constantes definidas em 'XPCOMDEF.H': COM_NONEP - Sem paridade, COM_EVENP - Paridade par; COM_ODDP - Paridade ímpar, Caso contrário retorna um código de erro;
COMCFG_TXFLOW:	Retorna o tipo do controle de fluxo na transmissão conforme as constantes definidas em 'XPCOMDEF.H': COM_NOFLOW - Sem controle de fluxo, COM_RTSCTS - Controle de fluxo via RTS/CTS, COM_XONXOFF - Controle de fluxo via XON/XOFF, Caso contrário retorna um código de erro;
COMCFG_RXFLOW:	Retorna o tipo do controle de fluxo na recepção conforme as constantes definidas em 'XPCOMDEF.H': COM_NOFLOW - Sem controle de fluxo, COM_RTSCTS - Controle de fluxo via RTS/CTS, COM_XONXOFF - Controle de fluxo via XON/XOFF, Caso contrário retorna um código de erro;

COMCFG_BUFRX: Retorna o tamanho do *buffer* de recepção, caso contrário retorna um código de erro;
COMCFG_BUFTX: Retorna o tamanho do *buffer* de transmissão, caso contrário retorna um código de erro;
Caso contrário retorna um código de erro;

O valor do código de erro retornado é sempre menor do que zero. Seu significado é fornecido no item “Códigos de Erro”.

Consulte

ComOpen, ComConfig.

Exemplo

```
#include <windows.h>
#include <stdlib.h>
#include “xpcom16.h”

long lInfo;
char cBuffer[10];

int PASCAL WinMain(HANDLE hInstance, HANDLE hPrevInstance, LPSTR lpCmdLine, int nCmdShow)
	{
	if (ComOpen(COM1,9600,8,1,COM_NONEP,COM_NOFLOW,COM_NOFLOW,256,256))
		return FALSE;
	lInfo = ComGetConfig(COM1,COMCFG_RXFLOW);
	switch (lInfo)
		{
		case COM_NOFLOW:
			{
			MessageBox((HWND)NULL, “Sem controle de fluxo”, "Exemplo", MB_OK);
			break;
			}
		case COM_RTSCTS:
			{
			MessageBox((HWND)NULL, “Controle de fluxo via RTS/CTS”, "Exemplo", MB_OK);
			break;
			}
		case COM_XONXOFF:
			{
			MessageBox((HWND)NULL, “Controle de fluxo via XON/XOFF”, "Exemplo", MB_OK);
			break;
			}
		default:
		return FALSE;
		}
	lInfo = ComGetConfig(COM1,COMCFG_RATE);
	if (lInfo < 0)
		return FALSE;
	MessageBox((HWND)NULL, ltoa(lInfo, cBuffer, 10), “Taxa de transmissao”, MB_OK);
	if (ComClose(COM1))
		return FALSE;
	return TRUE;
	}
```

ComGetConfig - Versão para DOS

Sumário

```
long ComGetConfig (int nCanal,                  // Numero do canal de comunicacao
                   int nInfo                    // Tipo de informacao desejada
                  );
```

Windows 16	Windows 32	DOS
		•

Descrição

A função ‘ComGetConfig’ é utilizada para se obter informações diversas sobre a configuração do canal

O argumento 'nCanal' especifica o canal de comunicação, possibilita o uso de no máximo 16 canais (COM1 a COM16).

O argumento 'nInfo' determina a informação desejada, é definido conforme as constantes abaixo, que já estão declaradas no arquivo *header* ‘XPCOMDEF.H’ da biblioteca:

COMCFG_RATE:	Informa o valor da taxa de comunicação;
COMCFG_NBITS:	Informa o número de *bits* por caractere 7 ou 8;
COMCFG_NSTOP:	Informa o número de *stop bits* 1 ou 2;
COMCFG_PAR:	Informa o tipo de paridade;
COMCFG_TXFLOW:	Informa o tipo do controle de fluxo na transmissão;
COMCFG_RXFLOW:	Informa o tipo do controle de fluxo na recepção;
COMCFG_BUFRX:	Informa o tamanho do *buffer* de recepção;
COMCFG_BUFTX:	Informa o tamanho do *buffer* de transmissão;
COMCFG_UART:	Informa o tipo de UART detectada;

Valor Retornado

O valor retornado pela função varia de acordo com o especificado no argumento 'nInfo', relacionados abaixo:

COMCFG_RATE:	Retorna o valor da taxa de comunicação, caso contrário retorna um código de erro;
COMCFG_NBITS:	Retorna o número de *bits* por caractere 7 ou 8, caso contrário retorna um código de erro;
COMCFG_NSTOP:	Retorna o número de *stop bits* 1 ou 2, caso contrário retorna um código de erro;
COMCFG_PAR:	Retorna o tipo de paridade conforme as constantes definidas em ‘XPCOMDEF.H’: COM_NONEP - Sem paridade, COM_EVENP - Paridade par; COM_ODDP - Paridade ímpar, Caso contrário retorna um código de erro;
COMCFG_TXFLOW:	Retorna o tipo do controle de fluxo na transmissão conforme as constantes definidas em ‘XPCOMDEF.H’: COM_NOFLOW - Sem controle de fluxo, COM_RTSCTS - Controle de fluxo via RTS/CTS, COM_XONXOFF - Controle de fluxo via XON/XOFF, Caso contrário retorna um código de erro;
COMCFG_RXFLOW:	Retorna o tipo do controle de fluxo na recepção conforme as constantes definidas em ‘XPCOMDEF.H’: COM_NOFLOW - Sem controle de fluxo, COM_RTSCTS - Controle de fluxo via RTS/CTS, COM_XONXOFF - Controle de fluxo via XON/XOFF, Caso contrário retorna um código de erro;
COMCFG_BUFRX:	Retorna o tamanho do *buffer* de recepção, caso contrário retorna um código de erro;
COMCFG_BUFTX:	Retorna o tamanho do *buffer* de transmissão, caso contrário retorna um código de erro;
COMCFG_UART:	Retorna o tipo de UART detectada conforme as constantes definidas em ‘XPCOMDEF.H’: UART8250: UART 8250, UART16450: UART 16450 com scratchpad, UART16550: UART 16550 com FIFOs, Caso contrário retorna um código de erro;

O valor do código de erro retornado é sempre menor do que zero. Seu significado é fornecido no item “Códigos de Erro”.

Consulte

ComTest, ComOpen, ComConfig.

Exemplo

```
#include <stdlib.h>
#include <stdio.h>
#include “xpcomlib.h”
```

```
#include “xpcomdef.h”

long lInfo;

void main(void)
        {
        if (ComAddress(COM1,0x238,3))
                exit(1);
        if (ComOpen(COM1,9600,8,1,COM_NONEP,COM_NOFLOW,COM_NOFLOW,256,256, FALSE))
                exit(1);
        lInfo = ComGetConfig(COM1,COMCFG_RXFLOW);
        switch (lInfo)
                {
                case COM_NOFLOW:
                        {
                        printf(“\nSem controle de fluxo\n”);
                        break;
                        }
                case COM_RTSCTS:
                        {
                        printf(“\nControle de fluxo via RTS/CTS\n”);
                        break;
                        }
                case COM_XONXOFF:
                        {
                        printf(“\nControle de fluxo via XON/XOFF\n”);
                        break;
                        }
                default:
                        exit(1);
                }
        lInfo = ComGetConfig(COM1,COMCFG_RATE);
        if (lInfo < 0)
                exit(1);
        else
                printf(“\nTaxa de transmissao: %u \n”, lInfo);
        if (ComClose(COM1))
                exit(1);
        }
```

ComGetnRx

Sumário

```
long FAR PASCAL ComGetnRx (int nCanal            // Numero do canal de comunicacao
                                );
```

Windows 16	Windows 32	DOS
•	•	•

Descrição

A função ComGetnRx é utilizada para se obter o número de *bytes* no *buffer* de recepção.

O argumento 'nCanal' especifica o canal de comunicação, possibilita o uso de no máximo 16 canais (COM1 a COM16).

Valor Retornado

A função retorna o número de *bytes* no *buffer* de recepção quando sua execução for bem sucedida, caso contrário retorna um código de erro sempre menor do que zero. O significado de cada código de erro retornado é fornecido no item “Códigos de Erro”.

Consulte

ComGetnTx, ComGetfRx, ComGetfTx.

Exemplo

```
#include <windows.h>
#include <stdlib.h>
#include “xpcom16.h”

long lInfo;
char cBuffer[10];

int PASCAL WinMain(HANDLE hInstance, HANDLE hPrevInstance, LPSTR lpCmdLine, int nCmdShow)
        {
        if (ComOpen(COM1,9600,8,1,COM_NONEP,COM_NOFLOW,COM_NOFLOW,256,256))
                return FALSE;
        lInfo = ComGetnRx(COM1);
        if (lInfo < 0)
                return FALSE;
        MessageBox((HWND)NULL, ltoa(lInfo, cBuffer, 10), “Numero de bytes no buffer de recepcao”, MB_OK);
        if (ComClose(COM1))
                return FALSE;
        return TRUE;
        }
```

ComGetnTx

Sumário

```
long FAR PASCAL ComGetnTx (int nCanal            // Numero do canal de comunicacao
                           );
```

Windows 16	Windows 32	DOS
•	•	•

Descrição

A função ComGetnTx é utilizada para se obter o número de *bytes* no *buffer* de transmissão

O argumento 'nCanal' especifica o canal de comunicação, possibilita o uso de no máximo 16 canais (COM1 a COM16).

Valor Retornado

A função retorna o número de *bytes* no *buffer* de transmissão quando sua execução for bem sucedida, caso contrário retorna um código de erro sempre menor do que zero. O significado de cada código de erro retornado é fornecido no item “Códigos de Erro”.

Consulte

ComGetnRx, ComGetfRx, ComGetfTx.

Exemplo

```
#include <windows.h>
#include <stdlib.h>
#include “xpcom16.h”

long lInfo;
char cBuffer[10];

int PASCAL WinMain(HANDLE hInstance, HANDLE hPrevInstance, LPSTR lpCmdLine, int nCmdShow)
```

```
{
if (ComOpen(COM1,9600,8,1,COM_NONEP,COM_NOFLOW,COM_NOFLOW,256,256))
        return FALSE;
lInfo = ComGetnTx(COM1);
if (lInfo < 0)
        return FALSE;
MessageBox((HWND)NULL, ltoa(lInfo, cBuffer, 10), "Numero de bytes no buffer de transmissao", MB_OK);
if (ComClose(COM1))
        return FALSE;
return TRUE;
}
```

ComGetfRx

Sumário

```
long FAR PASCAL ComGetfRx (int nCanal          // Numero do canal de comunicacao
                           );
```

Windows 16	Windows 32	DOS
•	•	•

Descrição

A função ComGetfRx é utilizada para se obter o número de *bytes* vagos no *buffer* de recepção.

O argumento 'nCanal' especifica o canal de comunicação, possibilita o uso de no máximo 16 canais (COM1 a COM16).

Valor Retornado

A função retorna o número de *bytes* vagos no *buffer* de recepção quando sua execução for bem sucedida, caso contrário retorna um código de erro sempre menor do que zero. O significado de cada código de erro retornado é fornecido no item "Códigos de Erro".

Consulte

ComGetnRx, ComGetnTx, ComGetfTx.

Exemplo

```
#include <windows.h>
#include <stdlib.h>
#include "xpcom16.h"

long lInfo;
char cBuffer[10];

int PASCAL WinMain(HANDLE hInstance, HANDLE hPrevInstance, LPSTR lpCmdLine, int nCmdShow)
        {
        if (ComOpen(COM1,9600,8,1,COM_NONEP,COM_NOFLOW,COM_NOFLOW,256,256))
                return FALSE;
        lInfo = ComGetfRx(COM1);
        if (lInfo < 0)
                return FALSE;
        MessageBox((HWND)NULL, ltoa(lInfo, cBuffer, 10), "Numero de bytes vagos no buffer de recepcao", MB_OK);
        if (ComClose(COM1))
                return FALSE;
        return TRUE;
        }
```

ComGetfTx

Sumário

```
long FAR PASCAL ComGetfTx (int nCanal          // Numero do canal de comunicacao
                           );
```

Windows 16	Windows 32	DOS
•	•	•

Descrição

A função ComGetnTx é utilizada para se obter o número de *bytes* vagos no *buffer* de transmissão

O argumento 'nCanal' especifica o canal de comunicação, possibilita o uso de no máximo 16 canais (COM1 a COM16).

Valor Retornado

A função retorna o número de *bytes* vagos no *buffer* de transmissão quando sua execução for bem sucedida, caso contrário retorna um código de erro sempre menor do que zero. O significado de cada código de erro retornado é fornecido no item "Códigos de Erro".

Consulte

ComGetnRx, ComGetnTx, ComGetfRx.

Exemplo

```
#include <windows.h>
#include <stdlib.h>
#include "xpcom16.h"

long lInfo;
char cBuffer[10];

int PASCAL WinMain(HANDLE hInstance, HANDLE hPrevInstance, LPSTR lpCmdLine, int nCmdShow)
    {
    if (ComOpen(COM1,9600,8,1,COM_NONEP,COM_NOFLOW,COM_NOFLOW,256,256))
        return FALSE;
    lInfo = ComGetfTx(COM1);
    if (lInfo < 0)
        return FALSE;
    MessageBox((HWND)NULL, ltoa(lInfo, cBuffer, 10), "Numero de bytes vagos no buffer de transmissao", MB_OK);
    if (ComClose(COM1))
        return FALSE;
    return TRUE;
    }
```

ComGetErrors

Sumário

```
int FAR PASCAL ComGetErrors (int nCanal          // Numero do canal de comunicacao
                             );
```

Windows 16	Windows 32	DOS
•	•	•

Descrição

A função ‘ComGetErrors’ retorna o *byte* de erros de comunicacão ocorridos no canal de comunicação.

O argumento 'nCanal' especifica o canal de comunicação, possibilita o uso de no máximo 16 canais (COM1 a COM16).

A função retorna o *byte* de erros de comunicação quando for bem sucedida. Caso contrário retorna um código de erro sempre menor do que zero. O significado de cada código de erro retornado é fornecido no item “Códigos de Erro”.

O *byte* de erros de comunicação deve ser examinado com as Máscaras de Erros de Comunicação:

COM_OVERUN:	Erro de overrun na recepção,
COM_PARITY:	Erro de paridade na recepção,
COM_FRAME:	Erro de framing na recepção,
COM_BREAK:	Detecção de break,
COM_RXBFUL:	Erro de *buffer* de recepção cheio.

Consulte

ComClearErrors.

Exemplo

```
#include <windows.h>
#include “xpcom16.h”

int nError;

int PASCAL WinMain(HANDLE hInstance, HANDLE hPrevInstance, LPSTR lpCmdLine, int nCmdShow)
    {
    if (ComOpen(COM1,9600,8,1,COM_NONEP,COM_NOFLOW,COM_NOFLOW,256,256))
        return FALSE;
    nError = ComGetErrors(COM1);
    if (nError < 0)
        return FALSE;
    if (nError & COM_OVERUN)
        MessageBox((HWND)NULL, “Erro de overrun na recepcao.”, “Exemplo”, MB_OK);
    if (nError & COM_PARITY)
        MessageBox((HWND)NULL, “Erro de paridade na recepcao.”, “Exemplo”, MB_OK);
    if (nError & COM_FRAME)
        MessageBox((HWND)NULL, “Erro de framing na recepcao.”, “Exemplo”, MB_OK);
    if (nError & COM_BREAK)
        MessageBox((HWND)NULL, “Deteccao de break.”, “Exemplo”, MB_OK);
    if (nError & COM_RXBFUL)
        MessageBox((HWND)NULL, “Erro de buffer de recepcao cheio.”, “Exemplo”, MB_OK);
    if (ComClose(COM1))
        return FALSE;
    return TRUE;
    }
```

ComClearErrors

Sumário

```
int FAR PASCAL ComClearErrors (int nCanal        // Numero do canal de comunicacao
                               );
```

Windows 16	Windows 32	DOS
•	•	•

Descrição

A função ‘ComClearErros’ limpa o *byte* que guarda os erros de comunicacão ocorridos no canal de comunicação especificado.

O argumento 'nCanal' especifica o canal de comunicação, possibilita o uso de no máximo 16 canais (COM1 a COM16).

Valor Retornado

A função retorna zero quando sua execução for bem sucedida, caso contrário retorna um código de erro sempre menor do que zero. O significado de cada código de erro retornado é fornecido no item

Consulte

ComGetErrors.

Exemplo

```
#include <windows.h>
#include “xpcom16.h”

int PASCAL WinMain(HANDLE hInstance, HANDLE hPrevInstance, LPSTR lpCmdLine, int nCmdShow)
        {
        if (ComOpen(COM1,9600,8,1,COM_NONEP,COM_NOFLOW,COM_NOFLOW,256,256))
                return FALSE;
        if (ComClearErrors(COM1))
                return FALSE;
        if (ComClose(COM1))
                return FALSE;
        return TRUE;
        }
```

ComClearRxBuffer

Sumário

```
int FAR PASCAL ComClearRxBuffer (int nCanal     // Numero do canal de comunicacao
                                 );
```

Windows 16	Windows 32	DOS
•	•	•

Descrição

A função ‘ComClearRxBuffer’ limpa o *buffer* de recepção do canal de comunicação especificado.

O argumento 'nCanal' especifica o canal de comunicação, possibilita o uso de no máximo 16 canais (COM1 a COM16).

Valor Retornado

A função retorna zero quando sua execução for bem sucedida, caso contrário retorna um código de erro sempre menor do que zero. O significado de cada código de erro retornado é fornecido no item “Códigos de Erro”.

Consulte

ComClearTxBuffer, ComClearTxRx.

Exemplo

```
#include <windows.h>
#include “xpcom16.h”

int PASCAL WinMain(HANDLE hInstance, HANDLE hPrevInstance, LPSTR lpCmdLine, int nCmdShow)
        {
        if (ComOpen(COM1,9600,8,1,COM_NONEP,COM_NOFLOW,COM_NOFLOW,256,256))
```

```
        return FALSE;
    if (ComClearRxBuffer(COM1))
        return FALSE;
    if (ComClose(COM1))
        return FALSE;
    return TRUE;
    }
```

ComClearTxBuffer

Sumário

```
int FAR PASCAL ComClearTxBuffer (int nCanal     // Numero do canal de comunicacao
                                 );
```

Windows 16	Windows 32	DOS
•	•	•

Descrição

A função 'ComClearTxBuffer' limpa o *buffer* de transmissão do canal de comunicação especificado.

O argumento 'nCanal' especifica o canal de comunicação, possibilita o uso de no máximo 16 canais (COM1 a COM16).

Valor Retornado

A função retorna zero quando sua execução for bem sucedida, caso contrário retorna um código de erro sempre menor do que zero. O significado de cada código de erro retornado é fornecido no item "Códigos de Erro".

Consulte

ComClearRxBuffer, ComClearTxRx.

Exemplo

```
#include <windows.h>
#include "xpcom16.h"

int PASCAL WinMain(HANDLE hInstance, HANDLE hPrevInstance, LPSTR lpCmdLine, int nCmdShow)
    {
    if (ComOpen(COM1,9600,8,1,COM_NONEP,COM_NOFLOW,COM_NOFLOW,256,256))
        return FALSE;
    if (ComClearTxBuffer(COM1))
        return FALSE;
    if (ComClose(COM1))
        return FALSE;
    return TRUE;
    }
```

ComClearTxRx

Sumário

```
int FAR PASCAL ComClearTxRx (int nCanal         // Numero do canal de comunicacao
                             );
```

Windows 16	Windows 32	DOS
•	•	•

Descrição

A função ‘ComClearTxRx’ limpa tanto o *buffer* de transmissão quanto o de recepção para o canal de comunicação especificado.

O argumento 'nCanal' especifica o canal de comunicação, possibilita o uso de no máximo 16 canais (COM1 a COM16).

Valor Retornado

A função retorna zero quando sua execução for bem sucedida, caso contrário retorna um código de erro sempre menor do que zero. O significado de cada código de erro retornado é fornecido no item “Códigos de Erro”.

Consulte

ComClearTxBuffer, ComClearRxBuffer.

Exemplo

```
#include <windows.h>
#include “xpcom16.h”

int PASCAL WinMain(HANDLE hInstance, HANDLE hPrevInstance, LPSTR lpCmdLine, int nCmdShow)
    {
    if (ComOpen(COM1,9600,8,1,COM_NONEP,COM_NOFLOW,COM_NOFLOW,256,256))
        return FALSE;
    if (ComClearTxRx(COM1))
        return FALSE;
    if (ComClose(COM1))
        return FALSE;
    return TRUE;
    }
```

ComRxChar

Sumário

```
int FAR PASCAL ComRxChar (int nCanal          // Numero do canal de comunicacao
                          );
```

Windows 16	Windows 32	DOS
•	•	•

Descrição

A função ‘ComRxChar’ retira um caractere do *buffer* de recepção do canal especificado.

O argumento 'nCanal' especifica o canal de comunicação, possibilita o uso de no máximo 16 canais (COM1 a COM16).

Valor Retornado

A função retorna o código do caractere retirado do *buffer* quando sua execução for bem sucedida, caso contrário retorna um código de erro sempre menor do que zero. O significado de cada código de erro retornado é fornecido no item “Códigos de Erro”.

Consulte

ComRxData, ComTxChar, ComTxData.

Exemplo

```
#include <windows.h>
#include <stdlib.h>
#include “xpcom16.h”
```

```
long lBytes;
int nChar;
char cBuffer[10];

int PASCAL WinMain(HANDLE hInstance, HANDLE hPrevInstance, LPSTR lpCmdLine, int nCmdShow)
        {
        if (ComOpen(COM1,9600,8,1,COM_NONEP,COM_NOFLOW,COM_NOFLOW,256,256))
                return FALSE;
        lBytes = ComGetnRx(COM1);
        if (lBytes < 0)
                return FALSE;
        if (0 == lBytes)
                MessageBox((HWND)NULL, "Nao existem dados disponiveis.", "Exemplo", MB_OK);
        else
                {
                for (;lBytes > 0; lBytes--)
                        {
                        nChar = ComRxChar(COM1);
                        if (nChar < 0)
                                return FALSE;
                        MessageBox((HWND)NULL, itoa(nChar, cBuffer, 10), "Caractere recebido ", MB_OK);
                        }
                }
        if (ComClose(COM1))
                return FALSE;
        return TRUE;
        }
```

ComRxData

Sumário

```
int FAR PASCAL ComRxData (int nCanal,                   // Numero do canal de comunicacao
                          char _far * sBuffer,          // Ponteiro para buffer de destino
                          unsigned int wBytes           // Numero de bytes a serem transferidos
                         );
```

Windows 16	Windows 32	DOS
•	•	•

Descrição

A função 'ComRxData' retira um ou mais *bytes* do *buffer* de recepção do canal especificado e coloca no *buffer* de destino apontado por sBuffer.

O argumento 'nCanal' especifica o canal de comunicação, possibilita o uso de no máximo 16 canais (COM1 a COM16).

O argumento 'sBuffer' é um ponteiro que define a localização do *buffer* de destino.

O argumento 'wBytes' informa o número de caracteres a serem retirados do *buffer* de recepção e guardados no *buffer* de destino.

Valor Retornado

A função retorna zero quando sua execução for bem sucedida, caso contrário retorna um código de erro sempre menor do que zero. O significado de cada código de erro retornado é fornecido no item "Códigos de Erro".

Consulte

ComRxChar, ComTxChar, ComTxData.

Exemplo

```
#include <windows.h>
#include “xpcom16.h”

long lBytes;
char sBuffer[256];

int PASCAL WinMain(HANDLE hInstance, HANDLE hPrevInstance, LPSTR lpCmdLine, int nCmdShow)
        {
        if (ComOpen(COM1,9600,8,1,COM_NONEP,COM_NOFLOW,COM_NOFLOW,256,256))
                return FALSE;
        lBytes = ComGetnRx(COM1);
        if (lBytes < 0)
                return FALSE;
        if (0 == lBytes)
                MessageBox((HWND)NULL, “Nao existem dados disponiveis.”, “Exemplo”, MB_OK);
        else
                {
                if (ComRxData(COM1, sBuffer, (unsigned int)lBytes))
                        return FALSE;
                }
        if (ComClose(COM1))
                return FALSE;
        return TRUE;
        }
```

ComTxChar

Sumário

```
int FAR PASCAL ComTxChar (int nCanal,                   // Numero do canal de comunicacao
                          unsigned char cByte           // Caractere a ser transmitido
                         );
```

Windows 16	Windows 32	DOS
•	•	•

Descrição

A função ‘ComTxChar’ transmite um caractere no canal de comunicação especificado.

O argumento 'nCanal' especifica o canal de comunicação, possibilita o uso de no máximo 16 canais (COM1 a COM16).

O argumento ‘cByte’ define o caractere a ser transmitido no canal de comunicação.

Valor Retornado

A função retorna zero quando sua execução for bem sucedida, caso contrário retorna um código de erro sempre menor do que zero. O significado de cada código de erro retornado é fornecido no item “Códigos de Erro”.

Consulte

ComRxChar, ComRxData, ComTxData.

Exemplo

```
#include <windows.h>
#include “xpcom16.h”
```

```
int PASCAL WinMain(HANDLE hInstance, HANDLE hPrevInstance, LPSTR lpCmdLine, int nCmdShow)
    {
    if (ComOpen(COM1,9600,8,1,COM_NONEP,COM_NOFLOW,COM_NOFLOW,256,256))
        return FALSE;
    if (ComTxChar(COM1,0x0D))
        return FALSE;
    if (ComClose(COM1))
        return FALSE;
    return TRUE;
    }
```

ComTxData

Sumário

```
int FAR PASCAL ComTxData (int nCanal,              // Numero do canal de comunicacao
                          char _far * psBuffer,    // Ponteiro para buffer de origem dos bytes
                          unsigned int wBytes      // Numero de bytes a serem transmitidos
                          );
```

Windows 16	Windows 32	DOS
•	•	•

Descrição

A função ‘ComTxData’ transmite *bytes* do *buffer* de origem apontado por ‘psBuffer’ para o canal de comunicação especificado.

O argumento 'nCanal' especifica o canal de comunicação, possibilita o uso de no máximo 16 canais (COM1 a COM16).

O argumento ‘psBuffer’ é um ponteiro que define a localização do *buffer* onde estão os *bytes* a serem transmitidos.

O argumento ‘wBytes’ informa o número de caracteres a serem transmitidos.

Valor Retornado

A função retorna zero quando sua execução for bem sucedida, caso contrário retorna um código de erro sempre menor do que zero. O significado de cada código de erro retornado é fornecido no item “Códigos de Erro”.

Consulte

ComRxChar, ComRxData, ComTxChar.

Exemplo

```
#include <windows.h>
#include <string.h>
#include “xpcom16.h”

char sBuffer[256];

int PASCAL WinMain(HANDLE hInstance, HANDLE hPrevInstance, LPSTR lpCmdLine, int nCmdShow)
    {
    if (ComOpen(COM1,9600,8,1,COM_NONEP,COM_NOFLOW,COM_NOFLOW,256,256))
        return FALSE;
    strcpy(sBuffer, “Teste”);
    if (ComTxData(COM1, sBuffer, strlen(sBuffer)))
        return FALSE;
    if (ComClose(COM1))
        return FALSE;
    return TRUE;
```

```
}
```

ComGetTxEmpty

Sumário

```
int FAR PASCAL ComGetTxEmpty (int nCanal        // Numero do canal de comunicacao
                              );
```

Windows 16	Windows 32	DOS
•		•

Descrição

A função 'ComGetTxEmpty' verifica o estado dos dois registros da UART, informando se ambos estão vazios ou se ao menos um deles tem dados, para o canal especificado.

O argumento 'nCanal' especifica o canal de comunicação, possibilita o uso de no máximo 16 canais (COM1 a COM16).

Valor Retornado

A função retorna o estado da transmissão quando sua execução for bem sucedida, caso contrário retorna um código de erro sempre menor do que zero. O significado de cada código de erro retornado é fornecido no item "Códigos de Erro".

Os possíveis estados da transmissão são:

COM_ON: transmissão ativa,
COM_OFF: transmissão inativa. (THRE e TSRE setados)

Consulte

ComGetCTS, ComGetDCD, ComGetDSR, ComGetRI, ComGetDTR, ComGetRTS.

Exemplo

```
#include <windows.h>
#include "xpcom16.h"

int nEstado;

int PASCAL WinMain(HANDLE hInstance, HANDLE hPrevInstance, LPSTR lpCmdLine, int nCmdShow)
    {
    if (ComOpen(COM1,9600,8,1,COM_NONEP,COM_NOFLOW,COM_NOFLOW,256,256))
        return FALSE;
    nEstado = ComGetTxEmpty(COM1);
    switch(nEstado)
        {
        case COM_ON :
            MessageBox((HWND)NULL, "A transmissão está ativa.", "Exemplo", MB_OK);
            break;
        case COM_OFF :
            MessageBox((HWND)NULL, "A transmissao nao está ativa.", "Exemplo", MB_OK);
            break;
        default :
        return FALSE;
        }
    if (ComClose(COM1))
        return FALSE;
    return TRUE;
    }
```

ComGetCTS

Sumário

```
int FAR PASCAL ComGetCTS (int nCanal          // Numero do canal de comunicacao
                          );
```

Windows 16	Windows 32	DOS
•	•	•

Descrição

A função ‘ComGetCTS’ retorna o estado do sinal CTS *Clear To Send* no canal de comunicação especificado.

O argumento 'nCanal' especifica o canal de comunicação, possibilita o uso de no máximo 16 canais (COM1 a COM16).

Valor Retornado

A função retorna o estado do sinal CTS quando sua execução for bem sucedida, caso contrário retorna um código de erro sempre menor do que zero. O significado de cada código de erro retornado é fornecido no item “Códigos de Erro”.

Os possíveis estados do sinal CTS são:

COM_ON:	CTS ativo,
COM_OFF:	CTS inativo.

Consulte

ComGetTxEmpty, ComGetDCD, ComGetDSR, ComGetRI, ComGetDTR, ComGetRTS.

Exemplo

```
#include <windows.h>
#include “xpcom16.h”

int nEstado;

int PASCAL WinMain(HANDLE hInstance, HANDLE hPrevInstance, LPSTR lpCmdLine, int nCmdShow)
    {
    if (ComOpen(COM1,9600,8,1,COM_NONEP,COM_NOFLOW,COM_NOFLOW,256,256))
        return FALSE;
    nEstado = ComGetCTS(COM1);
    switch(nEstado)
        {
        case COM_ON :
            MessageBox((HWND)NULL, “CTS está ativo.”, “Exemplo”, MB_OK);
            break;
        case COM_OFF :
            MessageBox((HWND)NULL, “CTS está inativo.”, “Exemplo”, MB_OK);
            break;
        default :
            return FALSE;
        }
    if (ComClose(COM1))
        return FALSE;
    return TRUE;
    }
```

ComGetDCD

Sumário

```
int FAR PASCAL ComGetDCD (int nCanal          // Numero do canal de comunicacao
```

```
                                    );
```

Windows 16	Windows 32	DOS
•	•	•

Descrição

A função ‘ComGetDCD’ retorna o estado do sinal DCD *Data Carrier Detected* no canal de comunicação especificado.

O argumento 'nCanal' especifica o canal de comunicação, possibilita o uso de no máximo 16 canais (COM1 a COM16).

Valor Retornado

A função retorna zero quando sua execução for bem sucedida, caso contrário retorna um código de erro sempre menor do que zero. O significado de cada código de erro retornado é fornecido no item “Códigos de Erro”.

Os possíveis estados do sinal DCD são:

COM_ON:	DCD ativo,
COM_OFF:	DCD inativo.

Consulte

ComGetTxEmpty, ComGetCTS, ComGetDSR, ComGetRI, ComGetDTR, ComGetRTS.

Exemplo

```
#include <windows.h>
#include “xpcom16.h”

int nEstado;

int PASCAL WinMain(HANDLE hInstance, HANDLE hPrevInstance, LPSTR lpCmdLine, int nCmdShow)
        {
        if (ComOpen(COM1,9600,8,1,COM_NONEP,COM_NOFLOW,COM_NOFLOW,256,256))
                return FALSE;
        nEstado = ComGetDCD(COM1);
        switch(nEstado)
                {
                case COM_ON :
                        MessageBox((HWND)NULL, “DCD está ativo.”, “Exemplo”, MB_OK);
                        break;
                case COM_OFF :
                        MessageBox((HWND)NULL, “DCD está inativo.”, “Exemplo”, MB_OK);
                        break;
                default :
                        return FALSE;
                }
        if (ComClose(COM1))
                return FALSE;
        return TRUE;
        }
```

ComGetDSR

Sumário

```
int FAR PASCAL ComGetDSR (int nCanal      // Numero do canal de comunicacao
                                    );
```

Windows 16	Windows 32	DOS
•	•	•

Descrição

A função ‘ComGetDSR’ retorna o estado do sinal DSR *Data Set Ready* no canal de comunicação especificado.

O argumento 'nCanal' especifica o canal de comunicação, possibilita o uso de no máximo 16 canais (COM1 a COM16).

Valor Retornado

A função retorna zero quando sua execução for bem sucedida, caso contrário retorna um código de erro sempre menor do que zero. O significado de cada código de erro retornado é fornecido no item “Códigos de Erro”.

Os possíveis estados do sinal DSR são:

COM_ON:	DSR ativo,
COM_OFF:	DSR inativo.

Consulte

ComGetTxEmpty, ComGetCTS, ComGetDCD, ComGetRI, ComGetDTR, ComGetRTS.

Exemplo

```
#include <windows.h>
#include “xpcom16.h”

int nEstado;

int PASCAL WinMain(HANDLE hInstance, HANDLE hPrevInstance, LPSTR lpCmdLine, int nCmdShow)
    {
    if (ComOpen(COM1,9600,8,1,COM_NONEP,COM_NOFLOW,COM_NOFLOW,256,256))
        return FALSE;
    nEstado = ComGetDSR(COM1);
    switch(nEstado)
        {
        case COM_ON :
            MessageBox((HWND)NULL, “DSR está ativo.”, “Exemplo”, MB_OK);
            break;
        case COM_OFF :
            MessageBox((HWND)NULL, “DSR está inativo.”, “Exemplo”, MB_OK);
            break;
        default :
            return FALSE;
        }
    if (ComClose(COM1))
        return FALSE;
    return TRUE;
    }
```

ComGetRI

Sumário

```
int FAR PASCAL ComGetRI (int nCanal              // Numero do canal de comunicacao
                         );
```

Windows 16	Windows 32	DOS
•	•	•

Descrição

A função ‘ComGetRI’ retorna o estado do sinal RI *Ring Indicator* no canal de comunicação especificado.

O argumento 'nCanal' especifica o canal de comunicação, possibilita o uso de no máximo 16 canais (COM1 a COM16).

Valor Retornado

A função retorna zero quando sua execução for bem sucedida, caso contrário retorna um código de erro sempre menor do que zero. O significado de cada código de erro retornado é fornecido no item “Códigos de Erro”.

Os possíveis estados do sinal RI são:

COM_ON:	RI ativo,
COM_OFF:	RI inativo.

Consulte

ComGetTxEmpty, ComGetCTS, ComGetDCD, ComGetDSR, ComGetDTR, ComGetRTS.

Exemplo

```
#include <windows.h>
#include “xpcom16.h”

int nEstado;

int PASCAL WinMain(HANDLE hInstance, HANDLE hPrevInstance, LPSTR lpCmdLine, int nCmdShow)
    {
    if (ComOpen(COM1,9600,8,1,COM_NONEP,COM_NOFLOW,COM_NOFLOW,256,256))
        return FALSE;
    nEstado = ComGetRI(COM1);
    switch(nEstado)
        {
        case COM_ON :
            MessageBox((HWND)NULL, “RI está ativo.”, “Exemplo”, MB_OK);
            break;
        case COM_OFF :
            MessageBox((HWND)NULL, “RI está inativo.”, “Exemplo”, MB_OK);
            break;
        default :
            return FALSE;
        }
    if (ComClose(COM1))
        return FALSE;
    return TRUE;
    }
```

ComGetDTR

Sumário

```
int FAR PASCAL ComGetDTR (int nCanal          // Numero do canal de comunicacao
                          );
```

Windows 16	Windows 32	DOS
•		•

Descrição

A função ‘ComGetDTR retorna o estado do sinal DTR *Data Terminal Ready* no canal de comunicação especificado.

O argumento 'nCanal' especifica o canal de comunicação, possibilita o uso de no máximo 16 canais (COM1 a COM16).

Valor Retornado

A função retorna zero quando sua execução for bem sucedida, caso contrário retorna um código de erro sempre menor do que zero. O significado de cada código de erro retornado é fornecido no item “Códigos de Erro”.

Os possíveis estados do sinal DTR são:

COM_ON:	DTR ativo,
COM_OFF:	DTR inativo.

Consulte

ComGetTxEmpty, ComGetCTS, ComGetDCD, ComGetDSR, ComGetRI, ComGetRTS.

Exemplo

```
#include <windows.h>
#include “xpcom16.h”

int nEstado;

int PASCAL WinMain(HANDLE hInstance, HANDLE hPrevInstance, LPSTR lpCmdLine, int nCmdShow)
    {
    if (ComOpen(COM1,9600,8,1,COM_NONEP,COM_NOFLOW,COM_NOFLOW,256,256))
        return FALSE;
    nEstado = ComGetDTR(COM1);
    switch(nEstado)
        {
        case COM_ON :
            MessageBox((HWND)NULL, “DTR está ativo.”, “Exemplo”, MB_OK);
            break;
        case COM_OFF :
            MessageBox((HWND)NULL, “DTR está inativo.”, “Exemplo”, MB_OK);
            break;
        default :
            return FALSE;
        }
    if (ComClose(COM1))
        return FALSE;
    return TRUE;
    }
```

ComGetRTS

Sumário

```
int FAR PASCAL ComGetRTS (int nCanal          // Numero do canal de comunicacao
                          );
```

Windows 16	Windows 32	DOS
•		•

Descrição

A função ‘ComGetRTS’ retorna o estado do sinal RTS *Request To Send* no canal de comunicação especificado.

O argumento 'nCanal' especifica o canal de comunicação, possibilita o uso de no máximo 16 canais (COM1 a COM16).

Valor Retornado

A função retorna o estado do sinal RTS quando sua execução for bem sucedida, caso contrário retorna um código de erro sempre menor do que zero. O significado de cada código de erro retornado é fornecido no item “Códigos de Erro”.

Os possíveis estados do sinal RTS são:

COM_ON:	RTS ativo,
COM_OFF:	RTS inativo.

Consulte

ComGetTxEmpty, ComGetCTS, ComGetDCD, ComGetDSR, ComGetRI, ComGetDTR.

Exemplo

```
#include <windows.h>
#include "xpcom16.h"

int nEstado;

int PASCAL WinMain(HANDLE hInstance, HANDLE hPrevInstance, LPSTR lpCmdLine, int nCmdShow)
    {
    if (ComOpen(COM1,9600,8,1,COM_NONEP,COM_NOFLOW,COM_NOFLOW,256,256))
        return FALSE;
    nEstado = ComGetRTS(COM1);
    switch(nEstado)
        {
        case COM_ON :
            MessageBox((HWND)NULL, "RTS está ativo.", "Exemplo", MB_OK);
            break;
        case COM_OFF :
            MessageBox((HWND)NULL, "RTS está inativo.", "Exemplo", MB_OK);
            break;
        default :
            return FALSE;
        }
    if (ComClose(COM1))
        return FALSE;
    return TRUE;
    }
```

ComSetDTR

Sumário

```
int FAR PASCAL ComSetDTR (int nCanal,       // Numero do canal de comunicacao
                          int nState        // Novo estado do sinal DTR
                          );
```

Windows 16	Windows 32	DOS
•	•	•

Descrição

A função 'ComSetDTR controla o estado do sinal DTR *Data Terminal Ready* no canal de comunicação especificado. Em geral é necessário ligar o DTR e o RTS antes de enviar comandos para um modem.

O argumento 'nCanal' especifica o canal de comunicação, possibilita o uso de no máximo 16 canais (COM1 a COM16).

O argumento 'nState' define o novo estado do sinal DTR para o canal especificado:

COM_ON:	Ativa DTR,
COM_OFF:	Desativa DTR.

Valor Retornado

A função retorna zero quando sua execução for bem sucedida, caso contrário retorna um código de erro sempre menor do que zero. O significado de cada código de erro retornado é fornecido no item “Códigos de Erro”.

Consulte

ComSetRTS, ComSetBREAK.

Exemplo

```
#include <windows.h>
#include “xpcom16.h”

int PASCAL WinMain(HANDLE hInstance, HANDLE hPrevInstance, LPSTR lpCmdLine, int nCmdShow)
    {
    if (ComOpen(COM1,9600,8,1,COM_NONEP,COM_NOFLOW,COM_NOFLOW,256,256))
        return FALSE;
    if ( ComSetDTR(COM1,COM_ON))
        return FALSE;
    if (ComClose(COM1))
        return FALSE;
    return TRUE;
    }
```

ComSetRTS

Sumário

```
int FAR PASCAL ComSetRTS (int nCanal,        // Numero do canal de comunicacao
                          int nState         // Novo estado do sinal RTS
                          );
```

Windows 16	Windows 32	DOS
•	•	•

Descrição

A função ‘ComSetRTS controla o estado do sinal RTS *Request To Send* no canal de comunicação especificado. Em geral é necessário ligar o DTR e o RTS antes de enviar comandos para um modem.

O argumento 'nCanal' especifica o canal de comunicação, possibilita o uso de no máximo 16 canais (COM1 a COM16).

O argumento ‘nState’ define o novo estado do sinal RTS para o canal especificado:

COM_ON:	Ativa RTS,
COM_OFF:	Desativa RTS.

Valor Retornado

A função retorna zero quando sua execução for bem sucedida, caso contrário retorna um código de erro sempre menor do que zero. O significado de cada código de erro retornado é fornecido no item “Códigos de Erro”.

Consulte

ComSetDTR, ComSetBREAK.

Exemplo

```
#include <windows.h>
#include “xpcom16.h”

int PASCAL WinMain(HANDLE hInstance, HANDLE hPrevInstance, LPSTR lpCmdLine, int nCmdShow)
    {
    if (ComOpen(COM1,9600,8,1,COM_NONEP,COM_NOFLOW,COM_NOFLOW,256,256))
```

```
        return FALSE;
    if ( ComSetRTS(COM1,COM_ON))
        return FALSE;
    if (ComClose(COM1))
        return FALSE;
    return TRUE;
    }
```

ComSetBREAK

Sumário

```
int FAR PASCAL ComSetBREAK (int nCanal,        // Numero do canal de comunicacao
                            int nState         // Novo estado do sinal BREAK
                            );
```

Windows 16	Windows 32	DOS
•	•	•

Descrição

A função 'ComSetBREAK controla o estado do sinal BREAK no canal de comunicação especificado.

O argumento 'nCanal' especifica o canal de comunicação, possibilita o uso de no máximo 16 canais (COM1 a COM16).

O argumento 'nState' define o novo estado do sinal BREAK para o canal especificado:

COM_ON:	Ativa BREAK,
COM_OFF:	Desativa BREAK.

Valor Retornado

A função retorna zero quando sua execução for bem sucedida, caso contrário retorna um código de erro sempre menor do que zero. O significado de cada código de erro retornado é fornecido no item "Códigos de Erro".

Consulte

ComSetDTR, ComSetRTS.

Exemplo

```
#include <windows.h>
#include "xpcom16.h"

int PASCAL WinMain(HANDLE hInstance, HANDLE hPrevInstance, LPSTR lpCmdLine, int nCmdShow)
    {
    if (ComOpen(COM1,9600,8,1,COM_NONEP,COM_NOFLOW,COM_NOFLOW,256,256))
        return FALSE;
    if ( ComSetBREAK(COM1,COM_ON))
        return FALSE;
    if (ComClose(COM1))
        return FALSE;
    return TRUE;
    }
```

Funções de Comunicação com a Rede XPnet

Estas funções são destinadas unicamente ao controle operacional da rede, e somente podem ser executadas após a inicialização da mesma. As funções de controle operacional iniciam rotinas de transmissão e recepção de dados entre o equipamento concentrador e os dispositivos conectados na rede, e serão executadas por interrupção. Também são fornecidas funções de verificação do estado da comunicação entre o concentrador e os dispositivos conectados na rede.

Funções Básicas de Comunicação com a Rede XPnet

Estas funções são destinadas a:

Inicializar operacionalmente a rede. Sempre que a rede for finalizada operacionalmente e reinicializada, todas as variáveis internas da biblioteca que indicam *status* de comunicação da rede também serão reinicializadas.

As funções de inicialização operacional da rede têm como função principal a definição da interface serial a ser utilizada, a taxa de comunicação em que a rede será operada, a definição de um *buffer* de transmissão no qual os dados transmitidos serão colocados para serem posteriormente retirados pelos terminais, e a definição de um *buffer* de recepção no qual os dados recebidos serão colocados para posteriormente serem retirados pelo aplicativo.

Verificar a comunicação entre o concentrador e os dispositivos conectados na rede.

Comunicação específica com determinado terminal da rede, não sendo necessário que o terminal especificado seja um dos válidos na comunicação automática.

Estabelecer controle da comunicação automática entre o concentrador e os terminais. A comunicação automática consiste em execuções automáticas da função 'XPnetmTxMsgPoll' utilizando como argumentos o canal e os terminais inseridos na varredura. Os dados recebidos são colocados no *buffer* de recepção do concentrador, dessa maneira o aplicativo somente precisa verificar, após iniciar a comunicação automática, se existem dados disponíveis no *buffer* de recepção.

Controlar a transferência dos dados do *buffer* de recepção do concentrador, permitindo que o programa aplicativo tenha acesso aos dados provenientes dos terminais.

XPnetmOpen

Sumário

```
int FAR PASCAL XPnetmOpen (int nCanal,                // Numero do canal de comunicacao
                           int nBaud,                 // Taxa de comunicacao
                           unsigned int wTxBuffer,    // Tamanho do buffer de transmissao
                           unsigned int wRxBuffer,    // Tamanho do buffer de recepcao
                           int nComType,              // Tipo de comunicacao Half / Full duplex
                           int nMaxRetries            // Numero maximo de tentativas quando ocorrer erros
                          );
```

Windows 16	Windows 32	DOS
•	•	•

Descrição

A função 'XPnetmOpen' inicializa a rede operacionalmente, permitindo que as funções da biblioteca possam ser executadas com sucesso. Esta função deve ser executada após 'ComAddress' que inicializa a rede para uma configuração de *hardware* específica.

O argumento 'nCanal' especifica o canal de comunicação, possibilita o uso de no máximo 16 canais (COM1 a COM16).

O argumento 'nBaud' define a taxa de comunicação a ser utilizada com 8 *bits*, 1 *stop-bit* e sem paridade, sendo os valores possíveis para o argumento mostrados abaixo em *baud*:

600 , 1200 , 2400 , 4800 , 9600 , 19200.

O argumento 'wTxBuffer' define o espaço a ser alocado para o *buffer* de transmissão de mensagens, podendo ser um valor de 16 *bytes* a 65500 *bytes*.

O argumento 'wRxBuffer' define o espaço a ser alocado para o *buffer* de recepção de mensagens, podendo ser um valor de 16 *bytes* a 65500 *bytes*.

O argumento ‘nComType’ define o tipo de comunicação usada Full ou Half duplex:

XPN_COMHALF:	Operação half-duplex para interface 485 sob DOS ou Windows em 16 bits;
XPN_COMAUTO:	Operação half-duplex automática com placa MOS-485 revisão E ou superior. Usar para Windows 3.x/95/98/NT. Não usar para comunicação via modem.
XPN_COMFULL:	Operação full-duplex para interface RS232, RS422 ou comunicação via modem.

O argumento ‘nMaxRetries’ informa o número máximo de tentativas de retransmissão quando ocorrer erros. Aumentar na comunicação via modem. Usar 1 ou 2 na comunicação via cabo, quando não se espera ter muitos erros de comunicação. Usar 10 na comunicação via modem quando se espera ter muitos erros na comunicação.

Valor Retornado

A função retorna zero quando sua execução for bem sucedida, caso contrário retorna um código de erro sempre menor do que zero. O significado de cada código de erro retornado é fornecido no item “Códigos de Erro”.

Consulte

ComAddress, XPnetmClose.

Exemplo

```
#include <windows.h>
#include “xpcom16.h”

int PASCAL WinMain(HANDLE hInstance, HANDLE hPrevInstance, LPSTR lpCmdLine, int nCmdShow)
        {
        if (XPnetmOpen(COM1,9600,256,256,XPN_COMAUTO,2))
                return FALSE;
        if (XPnetmClose(COM1))
                return FALSE;
        return TRUE;
        }
```

XPnetmClose

Sumário

```
int FAR PASCAL XPnetmClose (int nCanal              // Numero do canal de comunicacao
                                );
```

Windows 16	Windows 32	DOS
•	•	•

Descrição

A função 'XPnetmClose' finaliza operacionalmente a rede previamente inicializada.

O argumento 'nCanal' especifica o canal de comunicação, possibilita o uso de no máximo 16 canais (COM1 a COM16).

Valor Retornado

A função retorna zero quando sua execução for bem sucedida, caso contrário retorna um código de erro sempre menor do que zero. O significado de cada código de erro retornado é fornecido no item “Códigos de Erro”.

Consulte

ComAddress, XPnetmOpen.

Exemplo

```
#include <windows.h>
#include “xpcom16.h”

int PASCAL WinMain(HANDLE hInstance, HANDLE hPrevInstance, LPSTR lpCmdLine, int nCmdShow)
        {
        if (XPnetmOpen(COM1,9600,256,256,XPN_COMAUTO,2))
```

```
        return FALSE;
    if (XPnetmClose(COM1))
        return FALSE;
    return TRUE;
    }
```

XPnetmTermInfoClear

Sumário

```
int FAR PASCAL XPnetmTermInfoClear (int nCanal,      // Canal de comunicacao
                                    int nTerm        // Terminal
                                   );
```

Windows 16	Windows 32	DOS
•	•	

Descrição

A função 'XPnetmTermInfoClear' inicializa a estrutura do tipo XPNETMSTAT armazenada no PC, com o "*status* de comunicação do terminal", para o terminal especificado nos argumentos 'nCanal' e 'nTerm'. Ver esta estrutura na descrição da função XPnetmTermInfo.

O argumento 'nCanal' especifica o canal de comunicação, possibilita o uso de no máximo 16 canais (COM1 a COM16).

O argumento 'nTerm' informa o número do terminal, que deve ser de 1 a 32.

Valor Retornado

A função retorna zero quando sua execução for bem sucedida, caso contrário retorna um código de erro sempre menor do que zero. O significado de cada código de erro retornado é fornecido no item "Códigos de Erro".

Consulte

XPnetmTermInfo.

Exemplo

```
#include <windows.h>
#include "xpcom16.h"

XPNETMSTAT Stat;

int PASCAL WinMain(HANDLE hInstance, HANDLE hPrevInstance, LPSTR lpCmdLine, int nCmdShow)
    {
    if (XPnetmOpen(COM1,9600,256,256,XPN_COMAUTO,2))
        return FALSE;
    if (XPnetmTermInfoClear(COM1,1))
        return FALSE;
    if (XPnetmClose(COM1))
        return FALSE;
    return TRUE;
    }
```

XPnetmTermInfo - Versão para Windows

Sumário

```
int FAR PASCAL XPnetmTermInfo (int nCanal,                  // Canal de comunicacao
                               int nTerm,                   // Terminal
                               XPNETMSTAT FAR *Stat         // Estrutura com status do terminal
```

```
                                        );

struct XPNETMSTAT
        {
        long lError;                    // Ultimo erro de comunicacao ou zero se recuperado
        long lLastError;                // Ultimo erro de comunicacao, não limpa na recuperação
        long lChipError;                // Ultimo erro do driver de comunicação
        unsigned long dwStatus;         // Ultimo status interno do terminal
        unsigned long dwTxPolls;        // Numero de mensagens de requisicao de dados transmitidos
        unsigned long dwTxMsgs;         // Numero de mensagens de comando transmitidas
        unsigned long dwTxData;         // Numero de bytes de dados transmitidos
        unsigned long dwRxMsgs;         // Numero de mensagens de dados recebidas do terminal
        unsigned long dwRxData;         // Numero de bytes de dados recebidos do terminal
        unsigned long dwErrors;         // Numero de erros ocorridos
        unsigned long dwRetries;        // Numero de tentativas
        } ;
```

Windows 16	Windows 32	DOS
•	•	

Descrição

A função 'XPnetmTermInfo' obtém o "*status* de comunicação do terminal", armazenado no PC, para o terminal especificado nos argumentos 'nCanal' e 'nTerm', atualizando a estrutura de informações do argumento 'Stat'. Para atualizar o campo Stat.dwStatus que é o "*status* interno do terminal", armazenado no PC, é necessário que o polling dos terminais esteja ativo. A cada requisição de dados feita pelo concentrador ao terminal será enviada uma mensagem do terminal ao concentrador. Se o terminal não tiver dados então é enviado o seu status interno que o concentrado usa automaticamente para atualizar na estrutura Stat o campo dwStatus. Use a função 'XPnetmTxMsgPoll' para atualizar este status quando o polling não estiver ativo.

O argumento 'nCanal' especifica o canal de comunicação, possibilita o uso de no máximo 16 canais (COM1 a COM16).

O argumento 'nTerm' informa o número do terminal, que deve ser de 1 a 32.

O argumento 'Stat' é uma estrutura passada por referência, constituída dos elementos abaixo:

- último erro de comunicação ocorrido ou zero após recuperação;
- último erro de comunicação, não é limpado após recuperação;
- último erro do *driver* de comunicação;
- último *status* interno do terminal, 16 bits informando:;

```
V V V D D D R H 0 1 2 D D D B A   Bit = 0                    Bit = 1
                              |__ AC presente                AC ausente
                             |____ Bateria OK                Bateria Não OK
                          |_______ Chave 2 aberta            Chave 2 fechada
                        |_________ Chave 1 aberta            Chave 1 fechada
                      |___________ Chave 0 aberta            Chave 0 fechada
                    |_____________ Habilitação remota OK     Habilitação remota Não OK
                  |_______________ Estado remoto ON          Estado remoto OFF
          |__|__|_____________|__|__|_______ Posição disponível, não usada
       |__|__|_____________________________ Versão do status a partir de 001
```

- número de mensagens de requisição de dados feitas pelo concentrador para o terminal especificado;
- número de mensagens de comando transmitidas pelo concentrador para o terminal especificado;
- número de *bytes* de dados transmitidos pelo concentrador para o terminal especificado;
- número de mensagens de dados recebidas pelo concentrador do terminal especificado;
- número de *bytes* de dados recebidos pelo concentrador do terminal especificado;
- número de erros de comunicação ocorridos;
- número de repetições de tentativas de comunicação.

Esta estrutura é definida no arquivo *header* 'XPCOMLIB.H', com o nome de 'XPNETMSTAT'.

Valor Retornado

A função retorna zero quando sua execução for bem sucedida, caso contrário retorna um código de erro sempre menor do que zero. O significado de cada código de erro retornado é fornecido no item "Códigos de Erro".

Consulte

XPnetmTxMsgStatus.

Exemplo

```
#include <windows.h>
#include <stdlib.h>
#include <string.h>
#include “xpcom16.h”

XPNETMSTAT Stat;
char cBuffer[3000];
cbar cBuf[10];

int PASCAL WinMain(HANDLE hInstance, HANDLE hPrevInstance, LPSTR lpCmdLine, int nCmdShow)
        {
        if (XPnetmOpen(COM1,9600,256,256,XPN_COMAUTO,2))
                return FALSE;
        if (XPnetmTxMsgPoll(COM1,1))
                return FALSE;
        if (XPnetmWaitCom(COM1))
                return FALSE;
        if (XPnetmTermInfo(COM1,1,&Stat))
                return FALSE;
        strcpy(cBuffer, “lError.  .....: ”);
        strcat(cBuffer, ltoa(Stat.lError, cBuf, 10));
        strcat(cBuffer, “\rlLastError..: ”);
        strcat(cBuffer, ltoa(Stat.lLastError, cBuf, 10));
        strcat(cBuffer, “\rlChipError.: ”);
        strcat(cBuffer, ltoa(Stat.lChipError, cBuf, 10));
        strcat(cBuffer, “\rdwStatus....: ”);
        strcat(cBuffer, ltoa(Stat.dwStatus, cBuf, 10));
        strcat(cBuffer, “\rdwTxPolls..: ”);
        strcat(cBuffer, ltoa(Stat.dwTxPolls, cBuf, 10));
        strcat(cBuffer, “\rdwTxMsgs.: ”);
        strcat(cBuffer, ltoa(Stat.dwTxMsgs, cBuf, 10));
        strcat(cBuffer, “\rdwTxData..: ”);
        strcat(cBuffer, ltoa(Stat.dwTxData, cBuf, 10));
        strcat(cBuffer, “\rdwRxMsgs.: ”);
        strcat(cBuffer, ltoa(Stat.dwRxMsgs, cBuf, 10));
        strcat(cBuffer, “\rdwRxData..: ”);
        strcat(cBuffer, ltoa(Stat.dwRxData, cBuf, 10));
        strcat(cBuffer, “\rdwErrors....: ”);
        strcat(cBuffer, ltoa(Stat.dwErrors, cBuf, 10));
        strcat(cBuffer, “\rdwRetries...: ”);
        strcat(cBuffer, ltoa(Stat.dwRetries, cBuf, 10));
        MessageBox((HWND)NULL, cBuffer, “Status do Terminal”, MB_OK);
        if (XPnetmClose(COM1))
                return FALSE;
        return TRUE;
        }
```

XPnetmTermInfo - Versão para DOS

Sumário

```
int XPnetmTermInfo (int nCanal,                 // Canal de comunicacao
                    int nTerm,                  // Terminal
                    XPNETMSTAT *Stat            // Estrutura com status do terminal
```

```
                        );

struct XPNETMSTAT
        {
        int nError;                     // Ultimo erro de comunicacao ou zero se recuperado
        int nLastError;                 // Ultimo erro de comunicacao, não limpa na recuperação
        int nChipError;                 // Ultimo erro do chip UART
        unsigned int wStatus;           // Ultimo status interno do terminal
        unsigned long dwTxPolls;        // Numero de mensagens de requisicao de dados transmitidos
        unsigned long dwTxMsgs;         // Numero de mensagens de comando transmitidas
        unsigned long dwTxData;         // Numero de bytes de dados transmitidos
        unsigned long dwRxMsgs;         // Numero de mensagens de dados recebidas do terminal
        unsigned long dwRxData;         // Numero de bytes de dados recebidos do terminal
        unsigned long dwErrors;         // Numero de erros ocorridos
        unsigned long dwRetries;        // Numero de tentativas
        } Stat;
```

Windows 16	Windows 32	DOS
		•

Descrição

A função 'XPnetmTermInfo' obtém o "*status* de comunicação do terminal", armazenado no PC, para o terminal especificado nos argumentos 'nCanal' e 'nTerm', atualizando a estrutura de informações do argumento 'Stat'. Para atualizar o campo Stat.dwStatus que é o "*status* interno do terminal", armazenado no PC, é necessário que o polling dos terminais esteja ativo. A cada requisição de dados feita pelo concentrador ao terminal será enviada uma mensagem do terminal ao concentrador. Se o terminal não tiver dados então é enviado o seu status interno que o concentrado usa automaticamente para atualizar na estrutura Stat o campo dwStatus. Use a função 'XPnetmTxMsgPoll' para atualizar este status quando o polling não estiver ativo.

O argumento 'nCanal' especifica o canal de comunicação, possibilita o uso de no máximo 16 canais (COM1 a COM16).

O argumento 'nTerm' informa o número do terminal, que deve ser de 1 a 32.

O argumento 'Stat' é uma estrutura passada por referência, constituída dos elementos abaixo:

- último erro de comunicação ocorrido ou zero após recuperação;
- último erro de comunicação, não é limpado após recuperação;
- último erro do *chip* UART ver Máscara de Erros de Comunicação em XPCOMDEF.H;

COM_OVERUN	2	- Erro de *overrun* na recepção
COM_PARITY	4	- Erro de paridade na recepção
COM_FRAME	8	- Erro de *framing* na recepção
COM_BREAK	16	- Detecção de *break*
COM_RXBFUL	128	- Erro de buffer de recepção cheio

- último *status* interno do terminal, 16 bits informando:;

V V V D D D R H 0 1 2 D D D B A	Bit = 0	Bit = 1
|__	AC ausente	AC presente
|____	Bateria OK	Bateria Não OK
|_____	Chave 2 aberta	Chave 2 fechada
|_____	Chave 1 aberta	Chave 1 fechada
|_____	Chave 0 aberta	Chave 0 fechada
|_____	Habilitação remota OK	Habilitação remota Não OK
|_____	Estado remoto ON	Estado remoto OFF
|__|__|__|_____|__|__|_____	Posição disponível, não usada	
|__|__|__|_____	Versão do status a partir de 001	

- número de mensagens de requisição de dados feitas pelo concentrador para o terminal especificado;
- número de mensagens de comando transmitidas pelo concentrador para o terminal especificado;
- número de *bytes* de dados transmitidos pelo concentrador para o terminal especificado;
- número de mensagens de dados recebidas pelo concentrador do terminal especificado;
- número de *bytes* de dados recebidos pelo concentrador do terminal especificado;
- número de erros de comunicação ocorridos;
- número de repetições de tentativas de comunicação.

Esta estrutura é definida no arquivo *header* 'XPCOMLIB.H', com o nome de 'XPNETMSTAT'.

Valor Retornado

A função retorna zero quando sua execução for bem sucedida, caso contrário retorna um código de erro sempre menor do que zero. O significado de cada código de erro retornado é fornecido no item “Códigos de Erro”.

Consulte

XPnetmTxMsgStatus, XPnetmTermStat.

Exemplo

```
#include <stdlib.h>
#include “xpcomlib.h”
#include “xpcomdef.h”

XPNETMSTAT Stat;

void main(void)
    {
    if (ComAddress(COM1,0x238,3))
        exit(1);
    if (XPnetmOpen(COM1,9600,256,256,XPN_COMAUTO,2))
        exit(1);
    if (XPnetmTxMsgPoll(COM1,1))
        return FALSE;
    if (XPnetmWaitTx(COM1))
        return FALSE;
    if (XPnetmTermInfo(COM1,1,&Stat))
        exit(1);
    printf(“ Status do Terminal:\n”);
    printf(“nError:          %d  \n”, Stat.nError);
    printf(“nLastError:      %d  \n”, Stat.nLastError);
    printf(“nChipError:      %d  \n”, Stat.nChipError);
    printf(“ wStatus:        %u  \n”, Stat.wStatus);
    printf(“dwTxPolls:       %lu \n”, Stat.dwTxPools);
    printf(“dwTxMsgs:        %lu \n”, Stat.dwTxMsgs);
    printf(“dwTxData:        %lu \n”, Stat.dwTxData);
    printf(“dwRxMsgs:        %lu \n”, Stat.dwRxMsgs);
    printf(“dwRxData:        %lu \n”, Stat.dwRxData);
    printf(“dwErrors:        %lu \n”, Stat.dwErrors);
    printf(“dwRetries:       %lu \n”, Stat.dwRetries);
    if (XPnetmClose(COM1))
        exit(1);
    }
```

XPnetmTermStat

Sumário

```
long FAR PASCAL XPnetmTermStat (int nCanal,        // Canal de comunicacao
                                int nTerm,         // Terminal
                                int nStatType      // Identificação da estatística
                               );
```

Windows 16	Windows 32	DOS
•	•	

Descrição

A função 'XPnetmTermStat' obtém a estatística solicitada do “*status* interno do terminal”, armazenado no PC, para o terminal especificado nos argumentos 'nCanal' e 'nTerm', retornando este valor. Deve-se utilizar esta função no lugar de ‘XPnetmTermInfo’ quando se deseja saber o valor de uma única estatística ou com linguagens que não permitam utilizar ponteiro para estruturas. É

necessário executar XPnetmTxMsgPoll ou estar com o polling ativo antes de executar esta função para obter o último status interno do terminal XPNSTAT_STATUS atualizado.

O argumento 'nCanal' especifica o canal de comunicação, possibilita o uso de no máximo 16 canais (COM1 a COM16).

O argumento 'nTerm' informa o número do terminal, que deve ser de 1 a 32.

O argumento 'nStatType' é a identificação da estatística que desejamos seja retornada. Pode assumir os valores abaixo:

XPNSTAT_ERROR	- último erro de comunicação ocorrido ou zero se recuperado;
XPNSTAT_LASTERROR	- último erro de comunicação, não é limpado após recuperação;
XPNSTAT_UARTERROR	- último erro do *driver* de comunicação;
XPNSTAT_STATUS	- último *status* interno do terminal;
XPNSTAT_POLLS	- número de mensagens de requisição de dados feitas pelo concentrador para o terminal especificado;
XPNSTAT_TXMSGS	- número de mensagens de comando transmitidas pelo concentrador para o terminal especificado;
XPNSTAT_TXDATA	- número de *bytes* de dados transmitidos pelo concentrador para o terminal especificado;
XPNSTAT_RXMSGS	- número de mensagens de dados recebidas pelo concentrador do terminal especificado;
XPNSTAT_RXDATA	- número de *bytes* de dados recebidos pelo concentrador do terminal especificado;
XPNSTAT_ERRORS	- número de erros de comunicação ocorridos;
XPNSTAT_RETRIES	- número de repetições de tentativas de comunicação.

Estas constantes são definidas no arquivo *header* 'XPCOMLIB.H'.

Valor Retornado

A função retorna zero quando sua execução for bem sucedida, caso contrário retorna um código de erro sempre menor do que zero. O significado de cada código de erro retornado é fornecido no item “Códigos de Erro”.

Consulte

XPnetmTxMsgStatus, XPnetmTermInfo.

Exemplo

```
#include <windows.h>
#include <stdlib.h>
#include “xpcom16.h”

long lErro;
char cBuffer[10];

int PASCAL WinMain(HANDLE hInstance, HANDLE hPrevInstance, LPSTR lpCmdLine, int nCmdShow)
    {
    if (XPnetmOpen(COM1,9600,256,256,XPN_COMAUTO,2))
        return FALSE;
    if (XPnetmTxMsgPoll(COM1,1))
        return FALSE;
    if (XPnetmWaitCom(COM1))
        return FALSE;
    lErro = XPnetmTermStat(COM1,1, XPNSTAT_LASTERROR)
    if (0 > lErro)
        return FALSE;
    MessageBox((HWND)NULL, ltoa(lErro, cBuffer, 10), “Ultimo erro”, MB_OK);
    if (XPnetmClose(COM1))
        return FALSE;
    return TRUE;
    }
```

XPnetmComStatus

Sumário

```
int FAR PASCAL XPnetmComStatus (int nCanal      // Canal de comunicacao
                                );
```

Windows 16	Windows 32	DOS
•	•	•

Descrição

A função 'XPnetmComStatus' informa ao aplicativo o estado do canal de comunicação especificado. No ambiente DOS pode ser utilizada para aguardar que uma mensagem seja transmitida ao coletor antes de prosseguir o programa. No ambiente Windows este loop irá impedir que outros programas sejam executados, inclusive a DLL de comunicação causando um loop infinito. Use a função XPnetmWaitCom, no ambiente Windows, e XPnetmWaitTx no ambiente DOS para fazer loops de espera.

O argumento 'nCanal' especifica o canal de comunicação, possibilita o uso de no máximo 16 canais (COM1 a COM16).

Valor Retornado

A função retorna as constantes abaixo, definidas no arquivo 'XPCOMDEF.H', quando termina sua execução:

XPN_STATIDLE	- Sem transmissão (Comunicação inativa),
XPN_STATACTIVE	- Mensagem em transmissão (Comunicação ativa),
XPN_STATOK	- Mensagem transmitida (Comunicação inativa - OK),
XPN_STATERROR	- Mensagem não transmitida (Comunicação inativa com erro).

Caso contrário retorna um código de erro sempre menor do que zero. O significado de cada código de erro retornado é fornecido no item “Códigos de Erro”.

Consulte

XPnetmWaitCom, XPnetmWaitTx, XPnetmTxMsg, XPnetmTxMsgAttention, XPnetmTxMsgStatus, XPnetmTxMsgPoll.

Exemplo

```
#include <windows.h>
#include “xpcom16.h”

int nStatus;
int bLoop;

int PASCAL WinMain(HANDLE hInstance, HANDLE hPrevInstance, LPSTR lpCmdLine, int nCmdShow)
    {
    if (XPnetmOpen(COM1,9600,256,256,XPN_COMAUTO,2))
        return FALSE;
    if (XPnetmTxMsgAttention(COM1,1))
        return FALSE;
    bLoop = 1;
    while (bLoop)
        {
        nStatus = XPnetmComStatus(COM1);
        switch(nStatus)
                {
                case XPN_STATIDLE :
                    MessageBox((HWND)NULL, “Comunicacao inativa”, "Exemplo", MB_OK);
                    bLoop = 0;
                    break;
                case XPN_STATACTIVE :
                    break;
                case XPN_STATOK :
                    MessageBox((HWND)NULL, “Requisicao de atencao transmitida”, "Exemplo", MB_OK);
```

```
                    bLoop = 0;
                    break;
                case XPN_STATERROR :
                    MessageBox((HWND)NULL, “Comunicacao inativa com erro”, "Exemplo", MB_OK);
                    bLoop = 0;
                    break;
                default :
                    return FALSE;
                }
        }
    if (XPnetmClose(COM1))
        return FALSE;
    return TRUE;
    }
```

XPnetmGetConfig

Sumário

```
long FAR PASCAL XPnetmGetConfig (int nCanal,      // Canal de comunicacao
                                 int nInfo,       // Tipo de informacao desejada
                                 int nParam       // Parametro adicional para especificar informacao
                                 );
```

Windows 16	Windows 32	DOS
•	•	•

Descrição

A função 'XPnetmGetConfig' é utilizada para se obter informações diversas sobre a configuração da rede XPnet.

O argumento 'nCanal' especifica o canal de comunicação, possibilita o uso de no máximo 16 canais (COM1 a COM16).

O argumento 'nInfo' determina a informação desejada, é definido conforme as constantes abaixo, que já estão declaradas no arquivo *header* ‘XPCOMDEF.H’ da biblioteca:

XPNCFG_OPEN:	Informa se o canal de comunicação está aberto;
XPNCFG_RATE:	Informa o valor da taxa de comunicação, na qual a rede está operando;
XPNCFG_BUFRXMSG:	Informa o tamanho do *buffer* de recepção de mensagens;
XPNCFG_BUFTXMSG:	Informa o tamanho do *buffer* de transmissão de mensagens;
XPNCFG_POLLSTART:	Informa se a varredura automática de terminais está ativa;
XPNCFG_POLLINT:	Informa o intervalo, entre varreduras;
XPNCFG_POLLTERM:	Informa o número de requisições de dados a ser feita em uma varredura a um mesmo terminal, sendo que deve-se especificar o número do terminal (1 a 32) no argumento 'nParam'.

O argumento 'nParam', como mostrado acima, é utilizado em apenas um caso, nos demais o valor passado não é levado em consideração.

Valor Retornado

O valor retornado pela função varia de acordo com o especificado no argumento 'nInfo', relacionados abaixo:

XPNCFG_OPEN:	Retorna 'TRUE' ou 1 para o canal aberto, 'FALSE' ou 0 para canal fechado, caso contrário retorna um código de erro;
XPNCFG_RATE:	Retorna o valor da taxa de comunicação, caso contrário retorna um código de erro;
XPNCFG_BUFRXMSG:	Retorna o tamanho do *buffer* de recepção, caso contrário retorna um código de erro;
XPNCFG_BUFTXMSG:	Retorna o tamanho do *buffer* de transmissão, caso contrário retorna um código de erro;
XPNCFG_POLLSTART:	Retorna 'TRUE' ou 1 se a varredura automática de terminais está ativa, 'FALSE' ou 0 quando inativa, caso contrário retorna um código de erro;
XPNCFG_POLLINT:	Retorna o valor do intervalo entre varreduras, caso contrário retorna um código de erro;
XPNCFG_POLLTERM:	Retorna o número de requisições feitas em uma varredura a um mesmo terminal, caso contrário retorna um código de erro;

O valor do código de erro retornado é sempre menor do que zero. Seu significado é fornecido no item “Códigos de Erro”.

Obs.: 'TRUE' e 'FALSE' são constantes já definidas no arquivo *header* para a “Biblioteca de Funções de Comunicação para Linguagem C”.

Consulte

XPnetmTermInfo.

Exemplo

```
#include <windows.h>
#include <stdlib.h>
#include “xpcom16.h”

long lInfo;
char cBuffer[10];

int PASCAL WinMain(HANDLE hInstance, HANDLE hPrevInstance, LPSTR lpCmdLine, int nCmdShow)
        {
        if (XPnetmOpen(COM1,9600,256,256,XPN_COMAUTO,2))
                return FALSE;
        lInfo = XPnetmGetConfig(COM1,XPNCFG_OPEN,0);
        switch (lInfo)
                        {
                        case TRUE:
                                {
                                MessageBox((HWND)NULL, “Canal de comunicacao esta aberto”, "Exemplo", MB_OK);
                                break;
                                }
                        case FALSE:
                                {
                                MessageBox((HWND)NULL, “Canal de comunicacao nao esta aberto”, "Exemplo", MB_OK);
                                break;
                                }
                        default:
                                return FALSE;
                        }
        lInfo = XPnetmGetConfig(COM1,XPNCFG_BUFRXMSG,0);
        if (lInfo < 0)
                return FALSE;
        MessageBox((HWND)NULL, ltoa(lInfo, cBuffer, 10), "Tamanho do buffer de recepcao", MB_OK);
        if (XPnetmClose(COM1))
                return FALSE;
        return TRUE;
        }
```

XPnetmComError

Sumário

```
int FAR PASCAL XPnetmComError (int nCanal               // Canal de comunicacao
                                        );
```

Windows 16	Windows 32	DOS
•	•	•

Descrição

A função 'XPnetmComError' informa o último erro de comunicação ocorrido durante a execução de uma das funções da biblioteca XPnet.

O argumento 'nCanal' especifica o canal de comunicação, possibilita o uso de no máximo 16 canais (COM1 a COM16).

Valor Retornado

A função retorna zero quando nenhum erro foi encontrado, caso contrário retorna um código de erro sempre menor do que zero. O significado de cada código de erro retornado é fornecido no item “Códigos de Erro”.

Consulte

XPnetmComStatus.

Exemplo

```
#include <windows.h>
#include <stdlib.h>
#include “xpcom16.h”

int nError;
char cBuffer[10];

int PASCAL WinMain(HANDLE hInstance, HANDLE hPrevInstance, LPSTR lpCmdLine, int nCmdShow)
        {
        if (XPnetmOpen(COM1,9600,256,256,XPN_COMAUTO,2))
                return FALSE;
        nError = XPnetmComError(COM1);
        MessageBox((HWND)NULL, itoa(nError, cBuffer, 10), "Codigo de erro", MB_OK);
        if (XPnetmClose(COM1))
                return FALSE;
        return TRUE;
        }
```

XPComLibErrorMsg

Sumário

```
char* XPComLibErrorMsg (int nError                    // Numero do erro
                        );
```

Windows 16	Windows 32	DOS
		•

Descrição

A função 'XPComLibErrorMsg' recebe um código de erro e retorna um ponteiro para o string com a respectiva mensagem de erro.

O argumento 'nError' especifica o código de erro para o qual queremos o ponteiro de sua mensagem.

Valor Retornado

A função retorna um ponteiro para o string com a mensagem de erro correspondente ao código de erro enviado. O significado de cada código de erro retornado é fornecido no item “Códigos de Erro”.

Exemplo

```
#include <stdlib.h>
#include <stdio.h>
#include “xpcomlib.h”
#include “xpcomdef.h”
```

```
int nError;
unsigned char *pMsg;

void main(void)
        {
        if (nError = ComAddress(COM1,0x238,3))
                pMsg = XPComLibErrorMsg(nErro);
                printf("\n\n%s.\n", pMsg);
                return 1;
        if (nError = XPnetmOpen(COM1,9600,256,256,XPN_COMAUTO,2))
                pMsg = XPComLibErrorMsg(nErro);
                printf("\n\n%s.\n", pMsg);
                return 1;
        if (nError = XPnetmClose(COM1))
                pMsg = XPComLibErrorMsg(nErro);
                printf("\n\n%s.\n", pMsg);
                return 1;
        }
```

XPnetmTxMsg

Sumário

```
int FAR PASCAL XPnetmTxMsg (int nCanal,                 // Canal de comunicacao
                            int nTerm,                  // Terminal
                            const char FAR *pbMsg,      // Ponteiro para mensagem
                            unsigned int wTamMsg        // Tamanho da mensagem
                            );
```

Windows 16	Windows 32	DOS
•	•	•

Descrição

A função 'XPnetmTxMsg' transmite uma mensagem ao coletor, conforme o destino indicado pelos argumentos 'nCanal' e 'nTerm'. Esta mensagem pode ser retirada, do buffer de recepção de mensagens, pelo programa do coletor. Para que o coletor interprete a mensagem como um comando em XPBASIC, coloque o coletor em modo remoto antes de enviar a mensagem.

O argumento 'nCanal' especifica o canal de comunicação, possibilita o uso de no máximo 16 canais (COM1 a COM16).

O argumento 'nTerm' informa o número do terminal, que deve ser de 1 a 32.

O argumentos 'pbMsg' e 'wTamMsg' são, respectivamente, o ponteiro para mensagem a ser transmitida e seu tamanho.

Como a transmissão é feita por interrupção, a função retorna imediatamente após ter sido chamada, sendo necessário utilizar a função 'XPnetmWaitCom' (Windows) ou 'XPnetmWaitTx' (DOS) para verificar se a mensagem foi corretamente transmitida. A execução desta função é mal sucedida quando uma outra comunicação já estiver em andamento.

Valor Retornado

A função retorna zero quando sua execução for bem sucedida, caso contrário retorna um código de erro sempre menor do que zero. O significado de cada código de erro retornado é fornecido no item "Códigos de Erro".

Consulte

XPnetmTxCmd, XPnetmWaitCom, XPnetmWaitTx, XPnetmComStatus, XPnetmTxMsgPoll, XPnetmTxMsgAttention, XPnetmTxMsgStatus, XPnetmTxBufCount, XPnetmTxBufBytes, XPnetmTxBufFree, XPnetmTxBufClear.

Exemplo

```
#include <windows.h>
#include <string.h>
```

```
#include “xpcom16.h”

char sMsg[10];
int nStatus;

int PASCAL WinMain(HANDLE hInstance, HANDLE hPrevInstance, LPSTR lpCmdLine, int nCmdShow)
        {
        if (XPnetmOpen(COM1,9600,256,256,XPN_COMAUTO,2))
                return FALSE;
        strcpy(sMsg, “Teste”);
        if (XPnetmTxMsg(COM1,1,sMsg,strlen(sMsg)))
                return FALSE;
        if (XPnetmWaitCom(COM1))
                return FALSE;
        if (XPnetmClose(COM1))
                return FALSE;
        return TRUE;
        }
```

XPnetmTxMsgPoll

Sumário

```
int FAR PASCAL XPnetmTxMsgPoll (int nCanal,    // Canal de comunicacao
                                int nTerm      // Terminal
                                );
```

Windows 16	Windows 32	DOS
•	•	•

Descrição

A função 'XPnetmTxMsgPoll' transmite mensagem de requisição de dados para um terminal. Sendo que os dados recebidos serão armazenados no *buffer* de recepção. Se o terminal não tiver dados enviará o seu status que será removido automaticamente do buffer de recepção do PC e será guardado no campo ‘status interno do terminal’ da estrutura que guarda o ‘status interno do terminal’.

O argumento 'nCanal' especifica o canal de comunicação, possibilita o uso de no máximo 16 canais (COM1 a COM16).

O argumento 'nTerm' informa o número do terminal, que deve ser de 1 a 32.

Como a transmissão é feita por interrupção, a função retorna imediatamente após ter sido chamada, sendo necessário utilizar a função 'XPnetmWaitCom' (Windows) ou 'XPnetmWaitTx' (DOS) para verificar se a mensagem foi corretamente transmitida. A execução desta função é mal sucedida quando uma outra comunicação já estiver em andamento.

Valor Retornado

A função retorna zero quando sua execução for bem sucedida, caso contrário retorna um código de erro sempre menor do que zero. O significado de cada código de erro retornado é fornecido no item “Códigos de Erro”.

Consulte

XPnetmWaitCom, XPnetmWaitTx, XPnetmTermInfo, XPnetmTxMsg, XPnetmTxMsgAttention, XPnetmTxMsgStatus, XPnetmTxBufCount, XPnetmTxBufBytes, XPnetmTxBufFree, XPnetmTxBufClear, XPnetmRxMsg.

Exemplo

```
#include <windows.h>
#include “xpcom16.h”

int nStatus;

int PASCAL WinMain(HANDLE hInstance, HANDLE hPrevInstance, LPSTR lpCmdLine, int nCmdShow)
```

```
{
if (XPnetmOpen(COM1,9600,256,256,XPN_COMAUTO,2))
        return FALSE;
if (XPnetmTxMsgPoll(COM1,1))
        return FALSE;
if (XPnetmWaitCom(COM1))
        return FALSE;
if (XPnetmClose(COM1))
        return FALSE;
return TRUE;
}
```

XPnetmTxMsgStatus

Sumário

```
int FAR PASCAL XPnetmTxMsgStatus (int nCanal,      // Canal de comunicacao
                                  int nTerm        // Terminal
                                  );
```

Windows 16	Windows 32	DOS
•	•	•

Descrição

A função 'XPnetmTxMsgStatus' requisita o *status* interno de um terminal especificado pelos argumentos 'nCanal' e 'nTerm'. O status fica no buffer de recepção do PC até ser retirado. Não é atualizado o campo "*status* interno do terminal" na estrutura interna de controle da rede lida pela função 'XPnetmTermInfo'.

O argumento 'nCanal' especifica o canal de comunicação, possibilita o uso de no máximo 16 canais (COM1 a COM16).

O argumento 'nTerm' informa o número do terminal, que deve ser de 1 a 32.

Como a transmissão é feita por interrupção, a função retorna imediatamente após ter sido chamada, sendo necessário utilizar a função 'XPnetmWaitCom' (Windows) ou 'XPnetmWaitTx' (DOS) para verificar se a mensagem foi corretamente transmitida. A execução desta função é mal sucedida quando uma outra comunicação já estiver em andamento.

Valor Retornado

A função retorna zero quando sua execução for bem sucedida, caso contrário retorna um código de erro sempre menor do que zero. O significado de cada código de erro retornado é fornecido no item "Códigos de Erro".

Consulte

XPnetmWaitTx, XPnetmWaitCom, XPnetmTxMsg, XPnetmTxMsgPoll, XPnetmTxMsgAttention, XPnetmTxBufCount, XPnetmTxBufBytes, XPnetmTxBufFree, XPnetmTxBufClear.

Exemplo

```
#include <windows.h>
#include <string.h>
#include "xpcom16.h"

char cBuffer[256];
char cMsg[256];
char cBuf[20];

int PASCAL WinMain(HANDLE hInstance, HANDLE hPrevInstance, LPSTR lpCmdLine, int nCmdShow)
        {
        if (XPnetmOpen(COM1,9600,256,256,XPN_COMAUTO,2))
                return FALSE;
        if (XPnetmTxMsgStatus(COM1,1))
```

```
        return FALSE;
    if (XPnetmWaitCom(COM1))
        return FALSE;
    if (XPnetmRxMsg(COM1, cMsg))
        return FALSE;
    strcpy(cBuffer, "Terminal: ");
    strncat(cBuffer, cMsg, 2);
    strcat(cBuffer, "\rStatus: ");
    cBuf[ 0] = (cMsg[2] & 128) ? '1' : '0';
    cBuf[ 1] = (cMsg[2] &  64) ? '1' : '0';
    cBuf[ 2] = (cMsg[2] &  32) ? '1' : '0';
    cBuf[ 3] = (cMsg[2] &  16) ? '1' : '0';
    cBuf[ 4] = ' ';
    cBuf[ 5] = (cMsg[2] &   8) ? '1' : '0';
    cBuf[ 6] = (cMsg[2] &   4) ? '1' : '0';
    cBuf[ 7] = (cMsg[2] &   2) ? '1' : '0';
    cBuf[ 8] = (cMsg[2] &   1) ? '1' : '0';
    cBuf[ 9] = ' ';
    cBuf[10] = (cMsg[3] & 128) ? '1' : '0';
    cBuf[11] = (cMsg[3] &  64) ? '1' : '0';
    cBuf[12] = (cMsg[3] &  32) ? '1' : '0';
    cBuf[13] = (cMsg[3] &  16) ? '1' : '0';
    cBuf[14] = ' ';
    cBuf[15] = (cMsg[3] &   8) ? '1' : '0';
    cBuf[16] = (cMsg[3] &   4) ? '1' : '0';
    cBuf[17] = (cMsg[3] &   2) ? '1' : '0';
    cBuf[18] = (cMsg[3] &   1) ? '1' : '0';
    cBuf[19] = 0;
    strcat(cBuffer, cBuf);
    MessageBox((HWND)NULL, cBuffer, "Status do Terminal", MB_OK);
    if (XPnetmClose(COM1))
        return FALSE;
    return TRUE;
    }
```

XPnetmTxMsgAttention

Sumário

```
int FAR PASCAL XPnetmTxMsgAttention (int nCanal,        // Canal de comunicacao
                                     int nTerm          // Terminal
                                     );
```

Windows 16	Windows 32	DOS
•	•	•

Descrição

A função 'XPnetmTxMsgAttention' coloca o terminal especificado pelos argumentos 'nCanal' e 'nTerm' em MODO REMOTO e retorna imediatamente. Dai em diante todas mensagens para o terminal serão interpretadas como comandos XPBASIC até que um comando de REMOTE “OFF” seja enviado. A função ‘XPnetmRemote’ chama 'XPnetmTxMsgAttention' ao colocar o coletor em modo remoto e aguarda a transmissão para coletor, garantindo assim que o coletor recebeu o comando de atenção e teve tempo de entrar em modo remoto.

O argumento 'nCanal' especifica o canal de comunicação, possibilita o uso de no máximo 16 canais (COM1 a COM16).

O argumento 'nTerm' informa o número do terminal, que deve ser de 1 a 32.

Como a transmissão é feita por interrupção, a função retorna imediatamente após ter sido chamada, sendo necessário utilizar a função 'XPnetmWaitCom' (Windows) ou 'XPnetmWaitTx' (DOS) para verificar se a mensagem foi corretamente transmitida. A execução desta função é mal sucedida quando uma outra comunicação já estiver em andamento.

Valor Retornado

A função retorna zero quando sua execução for bem sucedida, caso contrário retorna um código de erro sempre menor do que zero. O significado de cada código de erro retornado é fornecido no item “Códigos de Erro”.

Consulte

XPnetmRemote, XPnetmWaitCom, XPnetmWaitTx, XPnetmComStatus, XPnetmTxMsg, XPnetmTxMsgPoll, XPnetmTxMsgStatus, XPnetmTxBufCount, XPnetmTxBufBytes, XPnetmTxBufFree, XPnetmTxBufClear.

Exemplo

```
#include <windows.h>
#include “xpcom16.h”

int nStatus;

int PASCAL WinMain(HANDLE hInstance, HANDLE hPrevInstance, LPSTR lpCmdLine, int nCmdShow)
    {
    if (XPnetmOpen(COM1,9600,256,256,XPN_COMAUTO,2))
        return FALSE;
    if (XPnetmTxMsgAttention(COM1,1))
        return FALSE;
    if (XPnetmWaitCom(COM1))
        return FALSE;
    if (XPnetmClose(COM1))
        return FALSE;
    return TRUE;
    }
```

XPnetmTxBufCount

Sumário

```
int FAR PASCAL XPnetmTxBufCount (int nCanal          // Canal de comunicacao
                                 );
```

Windows 16	Windows 32	DOS
•	•	•

Descrição

A função 'XPnetmTxBufCount', retorna o número de mensagens a transmitir na área reservada para o *buffer* de transmissão.

O argumento 'nCanal' especifica o canal de comunicação, possibilita o uso de no máximo 16 canais (COM1 a COM16).

Valor Retornado

A função retorna valor positivo (número de mensagens) ou igual a zero (*buffer* vazio) quando sua execução for bem sucedida, caso contrário retorna um código de erro sempre menor do que zero. O significado de cada código de erro retornado é fornecido no item “Códigos de Erro”.

Consulte

XPnetmComStatus, XPnetmTxMsg, XPnetmTxMsgPoll, XPnetmTxMsgAttention, XPnetmTxMsgStatus, XPnetmTxBufBytes, XPnetmTxBufFree, XPnetmTxBufClear.

Exemplo

```
#include <windows.h>
```

```
#include <stdlib.h>
#include “xpcom16.h”

int nCount;
char cBuffer[10];

int PASCAL WinMain(HANDLE hInstance, HANDLE hPrevInstance, LPSTR lpCmdLine, int nCmdShow)
        {
        if (XPnetmOpen(COM1,9600,256,256,XPN_COMAUTO,2))
                return FALSE;
        nCount = XPnetmTxBufCount(COM1);
        if (nCount < 0)
                return FALSE;
        MessageBox((HWND)NULL, itoa(nCount, cBuffer, 10), “Mensagens a transmitir”, MB_OK);
        if (XPnetmClose(COM1))
                return FALSE;
        return TRUE;
        }
```

XPnetmTxBufBytes

Sumário

```
int FAR PASCAL XPnetmTxBufBytes (int nCanal          // Canal de comunicacao
                                 );
```

Windows 16	Windows 32	DOS
•	•	•

Descrição

A função 'XPnetmTxBufBytes', retorna o número de *bytes* armazenados na área reservada para o *buffer* de transmissão.

O argumento 'nCanal' especifica o canal de comunicação, possibilita o uso de no máximo 16 canais (COM1 a COM16).

Valor Retornado

A função retorna valor positivo (*bytes* armazenados) ou igual a zero (*buffer* vazio) quando sua execução for bem sucedida, caso contrário retorna um código de erro sempre menor do que zero. O significado de cada código de erro retornado é fornecido no item “Códigos de Erro”.

Consulte

XPnetmComStatus, XPnetmTxMsg, XPnetmTxMsgPoll, XPnetmTxMsgAttention, XPnetmTxMsgStatus, XPnetmTxBufCount, XPnetmTxBufClear.

Exemplo

```
#include <windows.h>
#include <stdlib.h>
#include “xpcom16.h”

int nBytes;
char cBuffer[10];

int PASCAL WinMain(HANDLE hInstance, HANDLE hPrevInstance, LPSTR lpCmdLine, int nCmdShow)
        {
        if (XPnetmOpen(COM1,9600,256,256,XPN_COMAUTO,2))
                return FALSE;
        nBytes = XPnetmTxBufBytes(COM1);
        if (nBytes < 0)
```

```
        return FALSE;
    MessageBox((HWND)NULL, itoa(nBytes, cBuffer, 10), "Area ocupada", MB_OK);
    if (XPnetmClose(COM1))
        return FALSE;
    return TRUE;
    }
```

XPnetmTxBufFree

Sumário

```
int FAR PASCAL XPnetmTxBufFree (int nCanal          // Canal de comunicacao
                                );
```

Windows 16	Windows 32	DOS
•	•	•

Descrição

A função 'XPnetmTxBufFree', retorna o espaço (em *bytes*) disponível na área reservada para o *buffer* de transmissão.

O argumento 'nCanal' especifica o canal de comunicação, possibilita o uso de no máximo 16 canais (COM1 a COM16).

Valor Retornado

A função retorna valor positivo (*bytes* disponíveis) ou igual a zero (*buffer* cheio) quando sua execução for bem sucedida, caso contrário retorna um código de erro sempre menor do que zero. O significado de cada código de erro retornado é fornecido no item "Códigos de Erro".

Consulte

XPnetmComStatus, XPnetmTxMsg, XPnetmTxMsgPoll, XPnetmTxMsgAttention, XPnetmTxMsgStatus, XPnetmTxBufCount, XPnetmTxBufBytes, XPnetmTxBufClear.

Exemplo

```
#include <windows.h>
#include <stdlib.h>
#include "xpcom16.h"

int nBytes;
char cBuffer[10];

int PASCAL WinMain(HANDLE hInstance, HANDLE hPrevInstance, LPSTR lpCmdLine, int nCmdShow)
    {
    if (XPnetmOpen(COM1,9600,256,256,XPN_COMAUTO,2))
        return FALSE;
    nBytes = XPnetmTxBufFree(COM1);
    if (nBytes < 0)
        return FALSE;
    MessageBox((HWND)NULL, itoa(nBytes, cBuffer, 10), "Area disponivel", MB_OK);
    if (XPnetmClose(COM1))
        return FALSE;
    return TRUE;
    }
```

XPnetmTxBufClear

Sumário

```
int FAR PASCAL XPnetmTxBufClear (int nCanal          // Canal de comunicacao
```

);

Windows 16	Windows 32	DOS
•	•	•

Descrição

A função 'XPnetmTxBufClear' limpa toda a área reservada para o *buffer* de transmissão, descartando todas as mensagens anteriormente armazenadas.

O argumento 'nCanal' especifica o canal de comunicação, possibilita o uso de no máximo 16 canais (COM1 a COM16).

Valor Retornado

A função retorna zero quando sua execução for bem sucedida, caso contrário retorna um código de erro sempre menor do que zero. O significado de cada código de erro retornado é fornecido no item “Códigos de Erro”.

Consulte

XPnetmComStatus, XPnetmTxMsg, XPnetmTxMsgPoll, XPnetmTxMsgAttention, XPnetmTxMsgStatus, XPnetmTxBufCount, XPnetmTxBufBytes, XPnetmTxBufFree.

Exemplo

```
#include <windows.h>
#include “xpcom16.h”

int PASCAL WinMain(HANDLE hInstance, HANDLE hPrevInstance, LPSTR lpCmdLine, int nCmdShow)
        {
        if (XPnetmOpen(COM1,9600,256,256,XPN_COMAUTO,2))
                return FALSE;
        if (XPnetmTxBufClear(COM1))
                return FALSE;
        if (XPnetmClose(COM1))
                return FALSE;
        return TRUE;
        }
```

XPnetmPollStart

Sumário

```
int FAR PASCAL XPnetmPollStart (int nCanal,                 // Canal de comunicacao
                                unsigned int wIntervalo     // Intervalo entre requisicoes
                                );
```

Windows 16	Windows 32	DOS
•	•	•

Descrição

A função 'XPnetmPollStart' inicia a varredura automática, que faz a requisição de dados e recebe o status interno do terminal caso não haja dados, de todos os terminais inseridos na mesma. Após ser executada esta função, a inserção de um terminal na varredura automática deve ser feita pela função 'XPnetmPollTerm'.

O argumento 'nCanal' especifica o canal de comunicação, possibilita o uso de no máximo 16 canais (COM1 a COM16).

O argumento 'wIntervalo' especifica o tempo entre o término de uma requisição de dados de um terminal e o início da requisição de dados do próximo terminal na varredura, determinando assim a freqüência de varredura da rede. Este intervalo de tempo é fornecido em *ticks* (unidades de 0,055 segundos), e deve ser maior que 0.

Valor Retornado

A função retorna zero quando sua execução for bem sucedida, caso contrário retorna um código de erro sempre menor do que zero. O significado de cada código de erro retornado é fornecido no item “Códigos de Erro”.

Consulte

XPnetmPollStop, XPnetmPollTerm, XPnetmTermInfo.

Exemplo

```
#include <windows.h>
#include “xpcom16.h”

int PASCAL WinMain(HANDLE hInstance, HANDLE hPrevInstance, LPSTR lpCmdLine, int nCmdShow)
    {
    if (XPnetmOpen(COM1,9600,256,256,XPN_COMAUTO,2))
        return FALSE;
    if (XPnetmPollStart(COM1, 2))
        return FALSE;
    if (XPnetmPollTerm(COM1,1,2))
        return FALSE;
    if (XPnetmPollTerm(COM1,2,1))
        return FALSE;
    if (XPnetmPollStop(COM1))
        return FALSE;
    if (XPnetmClose(COM1))
        return FALSE;
    return TRUE;
    }
```

XPnetmPollStop

Sumário

```
int FAR PASCAL XPnetmPollStop (int nCanal                // Canal de comunicacao
                               );
```

Windows 16	Windows 32	DOS
•	•	•

Descrição

A função 'XPnetmPollStop' termina a varredura automática de requisição de dados previamente inicializada. Após a execução desta função, os dados obtidos através da varredura automática e que ainda não foram retirados do *buffer* de recepção não são descartados.

O argumento 'nCanal' especifica o canal de comunicação, possibilita o uso de no máximo 16 canais (COM1 a COM16).

Valor Retornado

A função retorna zero quando sua execução for bem sucedida, caso contrário retorna um código de erro sempre menor do que zero. O significado de cada código de erro retornado é fornecido no item “Códigos de Erro”.

Consulte

XPnetmPollStart, XPnetmPollTerm.

Exemplo

```
#include <windows.h>
#include “xpcom16.h”
```

```
int PASCAL WinMain(HANDLE hInstance, HANDLE hPrevInstance, LPSTR lpCmdLine, int nCmdShow)
        {
        if (XPnetmOpen(COM1,9600,256,256,XPN_COMAUTO,2))
                return FALSE;
        if (XPnetmPollStart(COM1, 2))
                return FALSE;
        if (XPnetmPollTerm(COM1,1,2))
                return FALSE;
        if (XPnetmPollTerm(COM1,2,1))
                return FALSE;
        if (XPnetmPollStop(COM1))
                return FALSE;
        if (XPnetmClose(COM1))
                return FALSE;
        return TRUE;
        }
```

XPnetmPollTerm

Sumário

```
int FAR PASCAL XPnetmPollTerm (int nCanal,          // Canal de comunicacao
                               int nTerm,           // Terminal
                               int nRepet           // Repeticao de requisicoes
                              );
```

Windows 16	Windows 32	DOS
•	•	•

Descrição

A função 'XPnetmPollTerm' insere um terminal na varredura automática de requisição de dados da rede XPnet, conforme os argumentos 'nCanal' e 'nTerm'.

O argumento 'nCanal' especifica o canal de comunicação, possibilita o uso de no máximo 16 canais (COM1 a COM16).

O argumento 'nTerm' informa o número do terminal, que deve ser de 1 a 32.

O argumento 'nRepet' informa o número de requisições de dados que a rede XPnet deve fazer a um mesmo terminal em uma mesma varredura, este número varia de 0 a 10. Sendo que para se retirar um terminal da varredura, deve-se executar a função 'XPnetmPollTerm', com o argumento para número de requisições igual a zero.

Valor Retornado

A função retorna zero quando sua execução for bem sucedida, caso contrário retorna um código de erro sempre menor do que zero. O significado de cada código de erro retornado é fornecido no item “Códigos de Erro”.

Consulte

XPnetmPollStart, XPnetmPollStop.

Exemplo

```
#include <windows.h>
#include “xpcom16.h”

int PASCAL WinMain(HANDLE hInstance, HANDLE hPrevInstance, LPSTR lpCmdLine, int nCmdShow)
        {
        if (XPnetmOpen(COM1,9600,256,256,XPN_COMAUTO,2))
                return FALSE;
```

```
if (XPnetmPollStart(COM1, 2))
        return FALSE;
if (XPnetmPollTerm(COM1,1,2))
        return FALSE;
if (XPnetmPollTerm(COM1,2,1))
        return FALSE;
if (XPnetmPollStop(COM1))
        return FALSE;
if (XPnetmClose(COM1))
        return FALSE;
return TRUE;
}
```

XPnetmRxMsgBytes

Sumário

```
int FAR PASCAL XPnetmRxMsgBytes (int nCanal          // Canal de comunicacao
                                 );
```

Windows 16	Windows 32	DOS
•	•	•

Descrição

A função 'XPnetmRxMsgBytes' informa o número de *bytes* da primeira mensagem que está armazenada e ainda não foi retirada da área reservada para o *buffer* de recepção.

O argumento 'nCanal' especifica o canal de comunicação, possibilita o uso de no máximo 16 canais (COM1 a COM16).

Valor Retornado

A função retorna valor positivo ou igual a zero (*buffer* vazio) quando sua execução for bem sucedida, caso contrário retorna um código de erro sempre menor do que zero. O significado de cada código de erro retornado é fornecido no item “Códigos de Erro”.

Consulte

XPnetmRxMsg, XPnetmRxBufCount, XPnetmRxBufBytes, XPnetmRxBufFree, XPnetmRxBufClear.

Exemplo

```
#include <windows.h>
#include “xpcom16.h”

int nBytes;
int nStatus;

int PASCAL WinMain(HANDLE hInstance, HANDLE hPrevInstance, LPSTR lpCmdLine, int nCmdShow)
        {
        if (XPnetmOpen(COM1,9600,256,256,XPN_COMAUTO,2))
                return FALSE;
        if (XPnetmTxMsgPoll(COM1, 1))
                return FALSE;
        if (XPnetmWaitCom(COM1))
                return FALSE;
        if (XPnetmWaitCom(COM1))
                return FALSE;
        nBytes = XPnetmRxMsgBytes(COM1);
        if (nBytes <= 0)
                {
                if (XPN_RXBEMPTY == nBytes || 0 == nBytes)
```

```
        {
        MessageBox((HWND)NULL, "Nao existem dados disponiveis", "Exemplo", MB_OK);
        return TRUE;
        }
    return FALSE;
    }
MessageBox((HWND)NULL, "Existem dados disponiveis", "Exemplo", MB_OK);
if (XPnetmClose(COM1))
    return FALSE;
return TRUE;
}
```

XPnetmRxMsg

Sumário

```
int FAR PASCAL XPnetmRxMsg (int nCanal,                 // Canal de comunicacao
                            char FAR *pbDest            // Ponteiro para area de retorno da mensagem
                            );
```

Windows 16	Windows 32	DOS
•	•	•

Descrição

A função 'XPnetmRxMsg' retira do *buffer* de recepção do PC uma mensagem, e a coloca a partir da área de retorno apontada pelo argumento 'pbDest'. As mensagens provenientes de um terminal são agrupadas no *buffer* de recepção, sendo cada uma destas uma seqüência de caracteres, onde os primeiros dois caracteres são o número do terminal em ASCII e os demais são a mensagem recebida.

Para que exista mensagem no *buffer* de recepção do PC é necessário executar a função XPnetmTxMsgPoll que retira dados do *buffer* de transmissão do coletor e transfere para o PC.

O argumento 'nCanal' especifica o canal de comunicação, possibilita o uso de no máximo 16 canais (COM1 a COM16).

O argumento ‘*pbDest’ aponta para a área de retorno onde a mensagem será armazenada.

Valor Retornado

A função retorna zero quando sua execução for bem sucedida, caso contrário retorna um código de erro sempre menor do que zero. O significado de cada código de erro retornado é fornecido no item “Códigos de Erro”.

Consulte

XPnetmRxMsgBytes, XPnetmRxBufCount, XPnetmRxBufBytes, XPnetmRxBufFree, XPnetmRxBufClear, XPnetmTxMsgPoll.

Exemplo

```
#include <windows.h>
#include <string.h>
#include “xpcom16.h”

int nBytes;
char sMsg[256];
char sBuffer[300];

int PASCAL WinMain(HANDLE hInstance, HANDLE hPrevInstance, LPSTR lpCmdLine, int nCmdShow)
    {
    if (XPnetmOpen(COM1,9600,256,256,XPN_COMAUTO,2))
        return FALSE;
    if (XPnetmTxMsgPoll(COM1, 1))
        return FALSE;
```

```
if (XPnetmWaitCom(COM1))
        return FALSE;
nBytes = XPnetmRxMsgBytes(COM1);
if (nBytes <= 0)
        {
        if (XPN_RXBEMPTY == nBytes || 0 == nBytes)
                {
                MessageBox((HWND)NULL, "Nao existem dados disponiveis", "Exemplo", MB_OK);
                return TRUE;
                }
        return FALSE;
        }
if (XPnetmRxMsg(COM1, sMsg))
        return FALSE;
strcpy(sBuffer, “Terminal: ”);
strncat(sBuffer, sMsg, 2);
strcat(sBuffer, "\rMensagem: \r");
strcat(sBuffer, &sMsg[2]);
MessageBox((HWND)NULL, sBuffer, "Exemplo", MB_OK);
if (XPnetmClose(COM1))
        return FALSE;
return TRUE;
}
```

XPnetmRxBufCount

Sumário

```
int FAR PASCAL XPnetmRxBufCount (int nCanal          // Canal de comunicacao
                                 );
```

Windows 16	Windows 32	DOS
•	•	•

Descrição

A função 'XPnetmRxBufCount' informa o número de mensagens armazenadas na área reservada para o *buffer* de recepção.

O argumento 'nCanal' especifica o canal de comunicação, possibilita o uso de no máximo 16 canais (COM1 a COM16).

Valor Retornado

A função retorna valor positivo (número de mensagens) ou igual a zero (*buffer* vazio) quando sua execução for bem sucedida, caso contrário retorna um código de erro sempre menor do que zero. O significado de cada código de erro retornado é fornecido no item “Códigos de Erro”.

Consulte

XPnetmRxMsg, XPnetmRxMsgBytes, XPnetmRxBufBytes, XPnetmRxBufFree, XPnetmRxBufClear.

Exemplo

```
#include <windows.h>
#include <stdlib.h>
#include “xpcom16.h”

int nNumMsg;
int nStatus;
char cBuffer[10];

int PASCAL WinMain(HANDLE hInstance, HANDLE hPrevInstance, LPSTR lpCmdLine, int nCmdShow)
```

```
{
if (XPnetmOpen(COM1,9600,256,256,XPN_COMAUTO,2))
        return FALSE;
if (XPnetmTxMsgPoll(COM1, 1))
        return FALSE;
if (XPnetmWaitCom(COM1))
        return FALSE;
nNumMsg = XPnetmRxBufCount(COM1);
if (nNumMsg < 0)
        return FALSE;
if (0 == nNumMsg)
        MessageBox((HWND)NULL, “Nao existem dados disponiveis”, "Exemplo", MB_OK);
else
        MessageBox((HWND)NULL, itoa(nNumMsg, cBuffer, 10), " Mensagens armazenadas", MB_OK);
if (XPnetmClose(COM1))
        return FALSE;
return TRUE;
}
```

XPnetmRxBufBytes

Sumário

```
int FAR PASCAL XPnetmRxBufBytes (int nCanal          // Canal de comunicacao
                                 );
```

Windows 16	Windows 32	DOS
•	•	•

Descrição

A função 'XPnetmRxBufBytes' informa o número de *bytes* armazenados na área reservada do *buffer* de recepção.

O argumento 'nCanal' especifica o canal de comunicação, possibilita o uso de no máximo 16 canais (COM1 a COM16).

Valor Retornado

A função retorna valor positivo (*bytes* armazenados) ou igual a zero (*buffer* vazio) quando sua execução for bem sucedida, caso contrário retorna um código de erro sempre menor do que zero. O significado de cada código de erro retornado é fornecido no item “Códigos de Erro”.

Consulte

XPnetmRxMsg, XPnetmRxMsgBytes, XPnetmRxBufCount, XPnetmRxBufFree, XPnetmRxBufClear.

Exemplo

```
#include <windows.h>
#include <stdlib.h>
#include “xpcom16.h”

int nBytes;
int nStatus;
char cBuffer[10];

int PASCAL WinMain(HANDLE hInstance, HANDLE hPrevInstance, LPSTR lpCmdLine, int nCmdShow)
        {
        if (XPnetmOpen(COM1,9600,256,256,XPN_COMAUTO,2))
                return FALSE;
        if (XPnetmTxMsgPoll(COM1, 1))
                return FALSE;
```

```
if (XPnetmWaitCom(COM1))
        return FALSE;
nBytes = XPnetmRxBufBytes(COM1);
if (nBytes < 0)
        return FALSE;
if (0 == nBytes)
        MessageBox((HWND)NULL, “Nao existem dados disponiveis”, "Exemplo", MB_OK);
else
        MessageBox((HWND)NULL, itoa(nBytes, cBuffer, 10), “Area ocupada”, MB_OK);
if (XPnetmClose(COM1))
        return FALSE;
return TRUE;
}
```

XPnetmRxBufFree

Sumário

```
int FAR PASCAL XPnetmRxBufFree (int nCanal              // Canal de comunicacao
                                );
```

Windows 16	Windows 32	DOS
•	•	•

Descrição

A função 'XPnetmRxBufFree' informa o numero de *bytes* livres na área reservada para o *buffer* de recepção.

O argumento 'nCanal' especifica o canal de comunicação, possibilita o uso de no máximo 16 canais (COM1 a COM16).

Valor Retornado

A função retorna valor positivo (*bytes* disponíveis) ou igual a zero (*buffer* cheio) quando sua execução for bem sucedida, caso contrário retorna um código de erro sempre menor do que zero. O significado de cada código de erro retornado é fornecido no item “Códigos de Erro”.

Consulte

XPnetmRxMsgBytes, XPnetmRxMsg, XPnetmRxBufCount, XPnetmRxBufBytes, XPnetmRxBufClear.

Exemplo

```
#include <windows.h>
#include <stdlib.h>
#include “xpcom16.h”

int nBytes;
int nStatus;
char cBuffer[10];

int PASCAL WinMain(HANDLE hInstance, HANDLE hPrevInstance, LPSTR lpCmdLine, int nCmdShow)
        {
        if (XPnetmOpen(COM1,9600,256,256,XPN_COMAUTO,2))
                return FALSE;
        if (XPnetmTxMsgPoll(COM1,1))
                return FALSE;
        if (XPnetmWaitCom(COM1))
                return FALSE;
        nBytes = XPnetmRxBufFree(COM1);
        if (nBytes < 0)
                return FALSE;
```

```
MessageBox((HWND)NULL, itoa(nBytes, cBuffer, 10), “Area disponivel”, MB_OK);
if (XPnetmClose(COM1))
        return FALSE;
return TRUE;
}
```

XPnetmRxBufClear

Sumário

```
int FAR PASCAL XPnetmRxBufClear (int nCanal          // Canal de comunicacao
                                 );
```

Windows 16	Windows 32	DOS
•	•	•

Descrição

A função 'XPnetmRxBufClear' descarta os dados do *buffer* de recepção da rede XPnet.

O argumento 'nCanal' especifica o canal de comunicação, possibilita o uso de no máximo 16 canais (COM1 a COM16).

Valor Retornado

A função retorna zero quando sua execução for bem sucedida, caso contrário retorna um código de erro sempre menor do que zero. O significado de cada código de erro retornado é fornecido no item “Códigos de Erro”.

Consulte

XPnetmRxMsgBytes, XPnetmRxMsg, XPnetmRxBufCount, XPnetmRxBufBytes, XPnetmRxBufFree.

Exemplo

```
#include <windows.h>
#include “xpcom16.h”

int PASCAL WinMain(HANDLE hInstance, HANDLE hPrevInstance, LPSTR lpCmdLine, int nCmdShow)
        {
        if (XPnetmOpen(COM1,9600,256,256,XPN_COMAUTO,2))
                return FALSE;
        if (XPnetmRxBufClear(COM1))
                return FALSE;
        if (XPnetmClose(COM1))
                return FALSE;
        return TRUE;
        }
```

Funções Auxiliares de Comunicação com a Rede XPnet

Estas funções são destinadas à transmissão de arquivos e programas do concentrador para os terminais, assim como à recepção de arquivos e programas dos terminais para o concentrador. Para que estas funções sejam executadas, é necessário que a rede esteja inicializada operacionalmente e a varredura automática não esteja ativada.

XPnetmRemote - Versão para Windows

Sumário

```
int FAR PASCAL XPnetmRemote(HWND hWnd,          // Handle da janela pai
                            int nCanal,         // Canal de comunicacao
                            int nTerm,          // Terminal
```

```
                    BOOL bAtiva          // Ativação, modo remoto ou local
                    );
```

Windows 16	Windows 32	DOS
•	•	

Descrição

A função 'XPnetmRemote' transmite comando para ativar ou desativar o modo remoto no terminal da rede XPnet e aguarda sua transmissão. Quando ativado, o terminal passa a ser comandado remotamente através de seu canal de comunicação. Desativando-se o modo remoto o terminal retorna ao *prompt*, podendo assim ser operado normalmente. Assim, sempre que o modo remoto não for mais necessário, aconselha-se desativá-lo.

O argumento 'hWnd' especifica o manipulador (handle) da janela pai utilizado para desabilitá-a durante a operação na rede XPnet.

O argumento 'nCanal' especifica o canal de comunicação, possibilita o uso de no máximo 16 canais (COM1 a COM16).

O argumento 'nTerm' informa o número do terminal, que deve ser de 1 a 32.

O argumento ‘bAtiva’ define o código da operação a ser relizada pela função:

Argumento	Descrição
TRUE ou 1	Estabelece modo remoto no coletor;
FALSE ou 0	Estabelece modo local no coletor e executa programa.

Para estabelecer modo remoto sem executar o programa do coletor envie o comando “REMOTE OFF”.

Valor Retornado

A função retorna zero quando sua execução for bem sucedida, caso contrário retorna um código de erro sempre menor do que zero. O significado de cada código de erro retornado é fornecido no item “Códigos de Erro”.

Consulte

XPnetmTxMsgAttention, XPnetmTxClock, XPnetmTxCmd, XPnetmWaitCom, XPnetmRxFile, XPnetmTxFile, XPnetmRxProg, XPnetmTxProg.

Exemplo

```
#include <windows.h>
#include “xpcom16.h”

int PASCAL WinMain(HANDLE hInstance, HANDLE hPrevInstance, LPSTR lpCmdLine, int nCmdShow)
    {
    if (XPnetmOpen(COM1,9600,256,256,XPN_COMAUTO,2))
        return FALSE;
    if (XPnetmPollStop(COM1))
        return FALSE;
    if (XPnetmRemote((HWND)NULL, COM1, 31, TRUE))
        return FALSE;
    if (XPnetmRemote((HWND)NULL, COM1, 31, FALSE))
        return FALSE;
    if (XPnetmPollStart(COM1, 1))
        return FALSE;
    if (XPnetmClose(COM1))
        return FALSE;
    return TRUE;
    }
```

XPnetmRemote - Versão para DOS

Sumário

```
int XPnetmRemote(unsigned int nCanal,          // Canal de comunicacao
                 unsigned int nTerm,           // Terminal
                 unsigned char cRemote         // Código de operação
                 );
```

Windows 16	Windows 32	DOS
		•

Descrição

A função 'XPnetmRemote' transmite comando para ativar ou desativar o modo remoto no terminal da rede XPnet e aguarda sua transmissão. Quando ativado, o terminal passa a ser comandado remotamente através de seu canal de comunicação. Desativando-se o modo remoto o terminal retorna ao *prompt*, podendo assim ser operado normalmente. Assim, sempre que o modo remoto não for mais necessário, aconselha-se desativá-lo.

O argumento 'nCanal' especifica o canal de comunicação, possibilita o uso de no máximo 16 canais (COM1 a COM16).

O argumento 'nTerm' informa o número do terminal, que deve ser de 1 a 32.

O argumento ‘cRemote’ define o código da operação a ser relizada pela função:

```
XPN_REMOTE          // Estabelece modo remoto no coletor,
XPN_LOCAL           // Estabelece modo local no coletor,
XPN_LOCALRUN        // Estabelece modo local no coletor e executa programa.
```

Valor Retornado

A função retorna zero quando sua execução for bem sucedida, caso contrário retorna um código de erro sempre menor do que zero. O significado de cada código de erro retornado é fornecido no item “Códigos de Erro”.

Consulte

XPnetmTxMsgAttention, XPnetmTxClock, XPnetmWaitTx, XPnetmRxFile, XPnetmTxFile, XPnetmRxProg, XPnetmTxProg.

Exemplo

```
#include <stdlib.h>
#include <stdio.h>
#include “xpcomlib.h”
#include “xpcomdef.h”

void main(void)
    {
    if (ComAddress(COM1, 0x238, 3))
        exit(1);
    if (XPnetmPollStop(COM1))
        return FALSE;
    if (XPnetmOpen(COM1,9600,256,256,XPN_COMAUTO,2))
        exit(1);
    if (XPnetmRemote(COM1, 31, XPN_REMOTE))
        exit(1);
    if (XPnetmRemote(COM1, 31, XPN_LOCAL))
        exit(1);
    if (XPnetmPollStart(COM1, 1))
        return FALSE;
    if (XPnetmClose(COM1))
        exit(1);
    }
```

XPnetmWaitCom

Sumário

```
int FAR PASCAL XPnetmWaitCom (int nCanal              // Canal de comunicacao
                              );
```

Windows 16	Windows 32	DOS
•	•	

Descrição

A função ‘XPnetmWaitCom’ aguarda o término da última comunicação realizada no canal antes de retornar ao mesmo tempo que permite a execução de programas em outras janelas do Windows enquanto espera. Deve-se utilizar só quando é necessário aguardar o fim da comunicação antes de prosseguir.

O argumento 'nCanal' especifica o canal de comunicação, possibilita o uso de no máximo 16 canais (COM1 a COM16).

Valor Retornado

A função retorna zero quando sua execução for bem sucedida, caso contrário retorna um código de erro sempre menor do que zero. O significado de cada código de erro retornado é fornecido no item “Códigos de Erro”.

Consulte

XPnetmRemote, XPnetmTxClock, XPnetmTxCmd, XPnetmRxFile, XPnetmTxFile, XPnetmRxProg, XPnetmTxProg.

Exemplo

```
#include <windows.h>
#include “xpcom16.h”

int PASCAL WinMain(HANDLE hInstance, HANDLE hPrevInstance, LPSTR lpCmdLine, int nCmdShow)
    {
    if (XPnetmOpen(COM1,9600,256,256,XPN_COMAUTO,2))
        return FALSE;
    if (XPnetmRemote((HWND)NULL, COM1, 31, TRUE))
        return FALSE;
    if (XPnetmTxClock((HWND)NULL, COM1, 31))
        return FALSE;
    if (XPnetmWaitCom(COM1))
        return FALSE;
    if (XPnetmPollStop(COM1))
        return FALSE;
    if (XPnetmRemote((HWND)NULL, COM1, 31, FALSE))
        return FALSE;
    if (XPnetmPollStart(COM1, 1))
        return FALSE;
    if (XPnetmClose(COM1))
        return FALSE;
    return TRUE;
    }
```

XPnetmWaitTx

Sumário

```
int XPnetmWaitTx (int nCanal                    // Canal de comunicacao
                  );
```

Windows 16	Windows 32	DOS
		•

Descrição

A função ‘XPnetmWaitTx’ aguarda o término da última transmissão realizada no canal antes de retornar. Deve-se utilizar só quando é necessário aguardar o fim da transmissão antes de prosseguir.

O argumento 'nCanal' especifica o canal de comunicação, possibilita o uso de no máximo 16 canais (COM1 a COM16).

Valor Retornado

A função retorna zero quando sua execução for bem sucedida, caso contrário retorna um código de erro sempre menor do que zero. O significado de cada código de erro retornado é fornecido no item “Códigos de Erro”.

Consulte

XPnetmRemote, XPnetmTxClock, XPnetmRxFile, XPnetmTxFile, XPnetmRxProg, XPnetmTxProg.

Exemplo

```
#include <stdlib.h>
#include “xpcomlib.h”
#include “xpcomdef.h”

void main(void)
    {
    if (ComAddress(COM1, 0x238, 3))
        exit(1);
    if (XPnetmOpen(COM1,9600,256,256,XPN_COMAUTO,2))
        exit(1);
    if (XPnetmRemote(COM1, 31, XPN_REMOTE))
        exit(1);
    if (XPnetmTxClock(COM1, 31))
        exit(1);
    if (XPnetmWaitTx(COM1))
        exit(1);
    if (XPnetmRemote(COM1, 31, XPN_LOCAL))
        exit(1);
    if (XPnetmClose(COM1))
        exit(1);
    }
```

XPnetmTxClock - Versão para Windows

Sumário

```
int FAR PASCAL XPnetmTxClock (HWND hWnd,        // Handle da janela pai
                              int nCanal,       // Canal de comunicacao
                              int nTerm         // Terminal
                              );
```

Windows 16	Windows 32	DOS
•	•	

Descrição

A função 'XPnetmTxClock' atualiza relógio e data do terminal especificado. A atualização é feita conforme o relógio e a data do micro concentrador da rede XPnet. Para a execução desta função exige-se que o terminal esteja em modo remoto, isto pode ser feito executando-se previamente a função 'XPnetmRemote'.

O argumento 'hWnd' especifica o manipulador (handle) da janela pai utilizado para desabilitá-a durante a operação na rede XPnet.

O argumento 'nCanal' especifica o canal de comunicação, possibilita o uso de no máximo 16 canais (COM1 a COM16).

O argumento 'nTerm' informa o número do terminal, que deve ser de 1 a 32.

Valor Retornado

A função retorna zero quando sua execução for bem sucedida, caso contrário retorna um código de erro sempre menor do que zero. O significado de cada código de erro retornado é fornecido no item “Códigos de Erro”.

Consulte

XPnetmTxCmd, XPnetmRemote, XPnetmWaitCom, XPnetmRxFile, XPnetmTxFile, XPnetmRxProg, XPnetmTxProg.

Exemplo

```
#include <windows.h>
#include “xpcom16.h”

int PASCAL WinMain(HANDLE hInstance, HANDLE hPrevInstance, LPSTR lpCmdLine, int nCmdShow)
    {
    if (XPnetmOpen(COM1,9600,256,256,XPN_COMAUTO,2))
        return FALSE;
    if (XPnetmRemote((HWND)NULL,COM1, 31, TRUE))
        return FALSE;
    if (XPnetmTxClock((HWND)NULL, COM1, 31))
        return FALSE;
    if (XPnetmRemote((HWND)NULL,COM1, 31, FALSE))
        return FALSE;
    if (XPnetmClose(COM1))
        return FALSE;
    return TRUE;
    }
```

XPnetmTxClock - Versão para DOS

Sumário

```
int XPnetmTxClock (int nCanal,          // Canal de comunicacao
                   int nTerm            // Terminal
                  );
```

Windows 16	Windows 32	DOS
		•

Descrição

A função 'XPnetmTxClock' atualiza relógio e data do terminal especificado. A atualização é feita conforme o relógio e a data do micro concentrador da rede XPnet. Para a execução desta função exige-se que o terminal esteja em modo remoto, isto pode ser feito executando-se previamente a função 'XPnetmRemote'.

O argumento 'nCanal' especifica o canal de comunicação, possibilita o uso de no máximo 16 canais (COM1 a COM16).

O argumento 'nTerm' informa o número do terminal, que deve ser de 1 a 32.

Valor Retornado

A função retorna zero quando sua execução for bem sucedida, caso contrário retorna um código de erro sempre menor do que zero. O significado de cada código de erro retornado é fornecido no item “Códigos de Erro”.

Consulte

XPnetmRemote, XPnetmWaitTx, XPnetmRxFile, XPnetmTxFile, XPnetmRxProg, XPnetmTxProg.

Exemplo

```
#include <stdlib.h>
#include “xpcomlib.h”
#include “xpcomdef.h”

void main(void)
        {
        if (ComAddress(COM1, 0x238, 3))
                exit(1);
        if (XPnetmOpen(COM1,9600,256,256,XPN_COMAUTO,2))
                exit(1);
        if (XPnetmRemote(COM1, 31, XPN_REMOTE))
                exit(1);
        if (XPnetmTxClock(COM1, 31))
                exit(1);
        if (XPnetmRemote(COM1, 31, XPN_LOCAL))
                exit(1);
        if (XPnetmClose(COM1))
                exit(1);
        }
```

XPnetmTxCmd

Sumário

```
int FAR PASCAL XPnetmTxCmd (HWND hWnd,                  // Handle da janela pai
                            int nCanal,                 // Canal de comunicacao
                            int nTerm,                  // Terminal
                            const char FAR *cCmd        // Ponteiro para string com comando
                            );
```

Windows 16	Windows 32	DOS
•	•	

Descrição

A função 'XPnetmTxCmd' transmite um string com o comando a ser executado do concentrador para um terminal da rede XPnet. Esta função insere automaticamente um “CR LF” no final da string o que faz com que o coletor execute o comando enviado. Para a execução desta função exige-se que o terminal esteja em modo remoto, isto pode ser feito executando-se previamente a função 'XPnetmRemote'. Sob ambiente DOS use a função XPnetmTxMsg inserindo o caractere CR (retorno de carro) no fim do comando.

O argumento 'hWnd' especifica o manipulador (handle) da janela pai utilizado para desabilitá-a durante a operação na rede XPnet.

O argumento 'nCanal' especifica o canal de comunicação, possibilita o uso de no máximo 16 canais (COM1 a COM16).

O argumento 'nTerm' informa o número do terminal, que deve ser de 1 a 32.

O argumento tipo caractere '*cCmd' informa qual é o string terminado por zero que contém o comando a ser enviado ao terminal.

Valor Retornado

A função retorna zero quando sua execução for bem sucedida, caso contrário retorna um código de erro sempre menor do que zero. O significado de cada código de erro retornado é fornecido no item “Códigos de Erro”.

Consulte

XPnetmRemote, XPnetmWaitCom, XPnetmTxClock, XPnetmTxCmd, XPnetmRxFile, XPnetmRxProg, XPnetmTxProg.

Exemplo

```
#include <windows.h>
#include “xpcom16.h”

int PASCAL WinMain(HANDLE hInstance, HANDLE hPrevInstance, LPSTR lpCmdLine, int nCmdShow)
        {
        if (XPnetmOpen(COM1,9600,256,256,XPN_COMAUTO,2))
                return FALSE;
        if (XPnetmRemote((HWND)NULL, COM1, 31, TRUE))
                return FALSE;
        if (XPnetmTxCmd((HWND)NULL, COM1, 31, "CLOSE 1:KILL 1:CLD:DISP 1,1,\"APAGOU ARQ 1\""))
                return FALSE;
        if (XPnetmRemote((HWND)NULL, COM1, 31, FALSE))
                return FALSE;
        if (XPnetmClose(COM1))
                return FALSE;
        return TRUE;
        }
```

XPnetmRxFile - Versão para Windows

Sumário

```
int FAR PASCAL XPnetmRxFile (HWND hWnd,                 // Handle da janela pai
                             int nCanal,                // Canal de comunicacao
                             int nTerm,                 // Terminal
                             const char FAR *cArq,      // Ponteiro para nome do arquivo
                             int nNumArqTerm            // Numero do arquivo no terminal
                             );
```

Windows 16	Windows 32	DOS
•	•	

Descrição

A função 'XPnetmRxFile' permite a recepção de um arquivo de dados em formato XPbasic de um terminal para o concentrador da rede XPnet. Para a execução desta função exige-se que o terminal esteja em modo remoto, isto pode ser feito executando-se previamente a função 'XPnetmRemote'.

O argumento 'hWnd' especifica o manipulador (handle) da janela pai utilizado para desabilitá-a durante a operação na rede XPnet.

O argumento 'nCanal' especifica o canal de comunicação, possibilita o uso de no máximo 16 canais (COM1 a COM16).

O argumento 'nTerm' informa o número do terminal, que deve ser de 1 a 32.

O argumento tipo caractere '*cArq' informa o nome e a localização com o qual o arquivo de dados proveniente do terminal deverá ser gravado no concentrador.

O argumento 'nNumArqTerm' informa o número do arquivo no terminal que se deseja receber.

Valor Retornado

A função retorna zero quando sua execução for bem sucedida, caso contrário retorna um código de erro sempre menor do que zero. O significado de cada código de erro retornado é fornecido no item “Códigos de Erro”.

Consulte

XPnetmRemote, XPnetmWaitCom, XPnetmTxClock, XPnetmTxCmd, XPnetmTxFile, XPnetmRxProg, XPnetmTxProg.

Exemplo

```
#include <windows.h>
```

```
#include “xpcom16.h”

char cNomeArq[20];

int PASCAL WinMain(HANDLE hInstance, HANDLE hPrevInstance, LPSTR lpCmdLine, int nCmdShow)
        {
        if (XPnetmOpen(COM1,9600,256,256,XPN_COMAUTO,2))
                return FALSE;
        if (XPnetmRemote((HWND)NULL, COM1, 31, TRUE))
                return FALSE;
        if (XPnetmRxFile((HWND)NULL, COM1, 31, “C:\\TRIX\\RXARQ.BIN”, 3))
                return FALSE;
        if (XPnetmRemote((HWND)NULL, COM1, 31, FALSE))
                return FALSE;
        if (XPnetmClose(COM1))
                return FALSE;
        return TRUE;
        }
```

XPnetmRxFile - Versão para DOS

Sumário

```
int XPnetmRxFile (int nCanal,                                   // Canal de comunicacao
                  int nTerm,                                    // Terminal
                  unsigned char *cArq,                          // Ponteiro para nome do arquivo
                  unsigned int wNumArqTerm,                     // Numero do arquivo no terminal
                  int (*pfShowMsg)(unsigned int wNblocos)       // Ponteiro para função que exibe numero de blocos recebidos
                 );
```

Windows 16	Windows 32	DOS
		•

Descrição

A função 'XPnetmRxFile' permite a recepção de um arquivo de dados em formato XPbasic de um terminal para o concentrador da rede XPnet. Para a execução desta função exige-se que o terminal esteja em modo remoto, isto pode ser feito executando-se previamente a função 'XPnetmRemote'. Durante o processo de recepção de arquivo pode-se mostrar na tela o número de blocos recebidos usando uma função local chamada a cada fim de bloco. Esta função pode testar o teclado e retornar um valor diferente de zero para abortar a operação.

O argumento 'nCanal' especifica o canal de comunicação, possibilita o uso de no máximo 16 canais (COM1 a COM16).

O argumento 'nTerm' informa o número do terminal, que deve ser de 1 a 32.

O argumento tipo caractere '*cArq' informa o nome e a localização com o qual o arquivo de dados proveniente do terminal deverá ser gravado no concentrador.

O argumento 'wNumArqTerm' informa o número do arquivo no terminal que se deseja receber.

O argumento ‘(*pfShowMsg)(wNblocos)’ é um ponteiro para uma função local que recebe o número de blocos já recebidos e pode ser usada escrever este número na tela. Esta função pode também abortar a recepção se testar o teclado e retornar um número diferente de zero quando determinada tecla for pressionada, retornando zero a recepção continua. O código de erro retornado é XPN_USERABORT. Caso não se deseje usar esta função fazer pfShowMsg = NULL.

Valor Retornado

A função retorna zero quando sua execução for bem sucedida, caso contrário retorna um código de erro sempre menor do que zero. O significado de cada código de erro retornado é fornecido no item “Códigos de Erro”.

Consulte

XPnetmRemote, XPnetmWaitTx, XPnetmTxClock, XPnetmTxFile, XPnetmRxProg, XPnetmTxProg.

Exemplo

```
#include <stdlib.h>
#include <stdio.h>
#include “xpcomlib.h”
#include “xpcomdef.h”

int ExibeMsg(unsigned int wNblocos);
char cNomeArq[20];

void main(void)
        {
        int (*pfExibeMsg)(unsigned int);
        if (ComAddress(COM1, 0x238, 3))
                exit(1);
        if (XPnetmOpen(COM1,9600,256,256,XPN_COMAUTO,2))
                exit(1);
        if (XPnetmRemote(COM1, 31, XPN_REMOTE))
                exit(1);
        strcpy(cNomeArq, “C:\\TRIX\\RXARQ.BIN”);
        pfExibeMsg = ExibeMsg;
        if (XPnetmRxFile(COM1, 31, cNomeArq, 3, pfExibeMsg))
                exit(1);
        if (XPnetmRemote(COM1, 31, XPN_LOCAL))
                exit(1);
        if (XPnetmClose(COM1))
                exit(1);
        }

int ExibeMsg(unsigned int wNblocos)
        {
        printf(“\nNumero de blocos recebidos: %03hu”, wNblocos);
        return 0;
        }
```

XPnetmRxFileStart

Sumário

```
int FAR PASCAL XPnetmRxFileStart (int nCanal,                   // Canal de comunicacao
                                  int nTerm,                    // Terminal
                                  const char *cArq,             // Ponteiro para nome do arquivo
                                  int nNumArqTerm,                  // Numero do arquivo no terminal
                                  const char *cText);               // Ponteiro para texto a ser mostrado no terminal
```

Windows 16	Windows 32	DOS
	•	

Descrição

A função 'XPnetmRxFileStart' permite iniciar a recepção de um arquivo de dados em formato XPbasic de um terminal para o concentrador da rede XPnet. Esta função retorna imediatamente e a comunicação continua a ser realizada por interrupção e deve ser monitorada dentro de um timer utilizando a função XPnetmTransferStatus. Somente após o término da comunicação ou após o aborto com a função XPnetmRxFileAbort é que o canal de comunicação deve ser fechado. Para a execução desta função exige-se que o terminal esteja em modo remoto, isto pode ser feito executando-se previamente a função 'XPnetmRemote'.

O argumento 'nCanal' especifica o canal de comunicação, possibilita o uso de no máximo 16 canais (COM1 a COM16).

O argumento 'nTerm' informa o número do terminal, que deve ser de 1 a 32.

O argumento tipo caractere '*cArq' informa o nome e a localização com o qual o arquivo de dados proveniente do terminal deverá ser gravado no concentrador.

O argumento 'nNumArqTerm' informa o número do arquivo no terminal que se deseja receber.

O argumento tipo caractere '*cText' informa a mensagem que deve ser exibida no terminal durante a transferência.

Valor Retornado

A função retorna zero quando sua execução for bem sucedida, caso contrário retorna um código de erro sempre menor do que zero. O significado de cada código de erro retornado é fornecido no item “Códigos de Erro”.

Consulte

XPnetmTransferStatus, XPnetmRxFileAbort, XPnetmTxFileStart, XPnetmRxProgStart, XPnetmTxProgStart, XPnetmRemote.

Exemplo

```
#include <windows.h>
#include "xpcom32.h"

XPNET_TRANSFER_STAT         pS;
int nAguardando = 1;
int nErro;
int nId;
HINSTANCE hInst;
void CALLSEQ WTimerDispatch (HWND hWnd, WORD wMsg, int nId, DWORD dwTime);

int PASCAL WinMain(HANDLE hInstance, HANDLE hPrevInstance, LPSTR lpCmdLine, int nCmdShow)
        {
        FARPROC lpFunc;

        hInst = hInstance;
        lpFunc = MakeProcInstance((FARPROC)WTimerDispatch, hInst);
        nId = SetTimer(NULL, 0, 1000, lpFunc);
        if (XPnetmOpen(COM1, 9600, 256, 256, XPN_COMAUTO, 2))
                return FALSE;
        if (XPnetmRemote((HWND)NULL, COM1, 31, TRUE))
                return FALSE;
        if (nErro = XPnetmRxFileStart(COM1, 31, "C:\\TRIX\\ARQ.BIN", 1, "TX ARQUIVO"))
                return FALSE;
        while (nAguardando)
                {
                GetMessage(&Msg, NULL, 0, 0);
                DispatchMessage(&Msg);
                }
        return TRUE;
        }

void CALLSEQ WTimerDispatch (HWND hWnd, WORD wMsg, int nId, DWORD dwTime)
        {
        XPnetTransferStatus(COM1, &pS);
        if (TRUE != pS.nActive)
                {
                nAguardando = 0;
                XPnetmRemote((HWND)NULL, COM1, 31, FALSE);
                XPnetmClose(COM1);
                }
        }
```

XPnetmRxFileAbort

Sumário

```
int FAR PASCAL XPnetmRxFileAbort(int nChannel);                    // Canal de comunicacao
```

Windows 16	Windows 32	DOS
	•	

Descrição

A função XPnetmRxFileAbort aborta a transmissão iniciada pela função XPnetmRxFileStart no canal especificado.

O argumento 'nCanal' especifica o canal de comunicação, possibilita o uso de no máximo 16 canais (COM1 a COM16).

Valor Retornado

A função retorna zero quando sua execução for bem sucedida, caso contrário retorna um código de erro sempre menor do que zero. O significado de cada código de erro retornado é fornecido no item “Códigos de Erro”.

Consulte

XPnetmRxFileStart.

XPnetmTxFile - Versão para Windows

Sumário

```
int FAR PASCAL XPnetmTxFile (HWND hWnd,                    // Handle da janela pai
                             int nCanal,                   // Canal de comunicacao
                             int nTerm,                    // Terminal
                             const char FAR *cArq,         // Ponteiro para nome do arquivo
                             int nNumArqTerm               // Numero do arquivo no terminal
                             );
```

Windows 16	Windows 32	DOS
•	•	

Descrição

A função 'XPnetmTxFile' transmite um arquivo de dados em formato XPbasic do concentrador para um terminal da rede XPnet. Para a execução desta função exige-se que o terminal esteja em modo remoto, isto pode ser feito executando-se previamente a função 'XPnetmRemote'.

O argumento 'hWnd' especifica o manipulador (handle) da janela pai utilizado para desabilitá-a durante a operação na rede XPnet.

O argumento 'nCanal' especifica o canal de comunicação, possibilita o uso de no máximo 16 canais (COM1 a COM16).

O argumento 'nTerm' informa o número do terminal, que deve ser de 1 a 32.

O argumento tipo caractere '*cArq' informa o nome e a localização do o arquivo de dados no concentrador que deverá ser enviado ao terminal.

O argumento 'nNumArqTerm' informa o número do arquivo no terminal para o qual se deseja transmitir.

Valor Retornado

A função retorna zero quando sua execução for bem sucedida, caso contrário retorna um código de erro sempre menor do que zero. O significado de cada código de erro retornado é fornecido no item “Códigos de Erro”.

Consulte

XPnetmRemote, XPnetmWaitCom, XPnetmTxClock, XPnetmTxCmd, XPnetmRxFile, XPnetmRxProg, XPnetmTxProg.

Exemplo

```
#include <windows.h>
#include “xpcom16.h”

char cNomeArq[20];

int PASCAL WinMain(HANDLE hInstance, HANDLE hPrevInstance, LPSTR lpCmdLine, int nCmdShow)
    {
    if (XPnetmOpen(COM1,9600,256,256,XPN_COMAUTO,2))
        return FALSE;
    if (XPnetmRemote((HWND)NULL, COM1, 31, TRUE))
        return FALSE;
    if (XPnetmTxFile((HWND)NULL, COM1, 31, “C:\\TRIX\\TXARQ.BIN”, 3))
        return FALSE;
    if (XPnetmRemote((HWND)NULL, COM1, 31, FALSE))
        return FALSE;
    if (XPnetmClose(COM1))
        return FALSE;
    return TRUE;
    }
```

XPnetmTxFile - Versão para DOS

Sumário

```
int XPnetmTxFile (int nCanal,                                                   // Canal de comunicacao
                  int nTerm,                                                    // Terminal
                  unsigned char *cArq,                                          // Ponteiro para nome do arquivo
                  unsigned int wNumArqTerm,                                     // Numero do arquivo no terminal
                  int (*pfShowMsg)(unsigned int wNblocos, unsigned int wNtotal) // Ponteiro para função que exibe nº de
                                                                                // blocos transmitidos e nº total de blocos
                 );
```

Windows 16	Windows 32	DOS
		•

Descrição

A função 'XPnetmTxFile' transmite um arquivo de dados em formato XPbasic do concentrador para um terminal da rede XPnet. Para a execução desta função exige-se que o terminal esteja em modo remoto, isto pode ser feito executando-se previamente a função 'XPnetmRemote'. Durante o processo de transmissão de arquivo pode-se mostrar na tela o número de blocos transmitidos e o número de blocos total usando uma função local chamada a cada fim de bloco. Esta função pode testar o teclado e retornar um valor diferente de zero para abortar a operação.

O argumento 'nCanal' especifica o canal de comunicação, possibilita o uso de no máximo 16 canais (COM1 a COM16).

O argumento 'nTerm' informa o número do terminal, que deve ser de 1 a 32.

O argumento tipo caractere '*cArq' informa o nome e a localização do o arquivo de dados no concentrador que deverá ser enviado ao terminal.

O argumento 'wNumArqTerm' informa o número do arquivo no terminal para o qual se deseja transmitir.

O argumento ‘(*pfShowMsg)(wNblocos, wNtotal)’ é um ponteiro para uma função local que recebe o número de blocos já transmitidos e o número total de blocos e pode ser usada para escrever estes números na tela. Esta função pode também abortar a transmissão se testar o teclado e retornar um número diferente de zero quando determinada tecla for pressionada. O código de erro retornado é XPN_USERABORT. Caso não se deseje usar esta função fazer pfShowMsg = NULL.

Valor Retornado

A função retorna zero quando sua execução for bem sucedida, caso contrário retorna um código de erro sempre menor do que zero. O significado de cada código de erro retornado é fornecido no item “Códigos de Erro”.

Consulte

XPnetmRemote, XPnetmWaitTx, XPnetmTxClock, XPnetmRxFile, XPnetmRxProg, XPnetmTxProg.

Exemplo

```
#include <stdlib.h>
#include <stdio.h>
#include “xpcomlib.h”
#include “xpcomdef.h”

int ExibeMsg(unsigned int wNblocos, unsigned int wNtotal);
char cNomeArq[20];

void main(void)
        {
        int (*pfExibeMsg)(unsigned int, unsigned int);
        if (ComAddress(COM1, 0x238, 3))
                exit(1);
        if (XPnetmOpen(COM1,9600,256,256,XPN_COMAUTO,2))
                exit(1);
        if (XPnetmRemote(COM1, 31, XPN_REMOTE))
                exit(1);
        strcpy(cNomeArq, “C:\\TRIX\\TXARQ.BIN”);
        pfExibeMsg = ExibeMsg;
        if (XPnetmTxFile(COM1, 31, cNomeArq, 3, pfExibeMsg))
                exit(1);
        if (XPnetmRemote(COM1, 31, XPN_LOCAL))
                exit(1);
        if (XPnetmClose(COM1))
                exit(1);
        }

int ExibeMsg(unsigned int wNblocos, unsigned int wNtotal)
        {
        printf(“\nNumero de blocos transmitidos: %03hu/%03hu”, wNblocos, wNtotal);
        return 0;
        }
```

XPnetmTxFileStart

Sumário

```
int FAR PASCAL XPnetmTxFileStart (int nCanal,                   // Canal de comunicacao
                                  int nTerm,                    // Terminal
                                  const char *cArq,             // Ponteiro para nome do arquivo
                                  int nNumArqTerm,                    // Numero do arquivo no terminal
                                  const char *cText);                 // Ponteiro para texto a ser mostrado no terminal
```

Windows 16	Windows 32	DOS
	•	

Descrição

A função 'XPnetmTxFileStart' permite iniciar a transmissão de um arquivo de dados em formato XPbasic do concentrador para um terminal da rede XPnet. Esta função retorna imediatamente e a comunicação continua a ser realizada por interrupção e deve

ser monitorada dentro de um timer utilizando a função XPnetmTransferStatus. Somente após o término da comunicação ou após o aborto com a função XPnetmTxFileAbort é que o canal de comunicação deve ser fechado. Para a execução desta função exige-se que o terminal esteja em modo remoto, isto pode ser feito executando-se previamente a função 'XPnetmRemote'.

O argumento 'nCanal' especifica o canal de comunicação, possibilita o uso de no máximo 16 canais (COM1 a COM16).

O argumento 'nTerm' informa o número do terminal, que deve ser de 1 a 32.

O argumento tipo caractere '*cArq' informa o nome e a localização do arquivo de dados, no concentrador, que será enviado ao terminal.

O argumento 'nNumArqTerm' informa o número do arquivo no terminal.

O argumento tipo caractere '*cText' informa a mensagem que deve ser exibida no terminal durante a transferência.

Valor Retornado

A função retorna zero quando sua execução for bem sucedida, caso contrário retorna um código de erro sempre menor do que zero. O significado de cada código de erro retornado é fornecido no item “Códigos de Erro”.

Consulte

XPnetmTransferStatus, XPnetmTxFileAbort, XPnetmRxFileStart, XPnetmRxProgStart, XPnetmTxProgStart, XPnetmRemote.

Exemplo

```
#include <windows.h>
#include "xpcom32.h"

XPNET_TRANSFER_STAT         pS;
int nAguardando = 1;
int nErro;
int nId;
HINSTANCE hInst;
void CALLSEQ WTimerDispatch (HWND hWnd, WORD wMsg, int nId, DWORD dwTime);

int PASCAL WinMain(HANDLE hInstance, HANDLE hPrevInstance, LPSTR lpCmdLine, int nCmdShow)
        {
        FARPROC lpFunc;

        hInst = hInstance;
        lpFunc = MakeProcInstance((FARPROC)WTimerDispatch, hInst);
        nId = SetTimer(NULL, 0, 1000, lpFunc);
        if (XPnetmOpen(COM1,9600,256,256,XPN_COMAUTO,2))
                return FALSE;
        if (XPnetmRemote((HWND)NULL, COM1, 31, TRUE))
                return FALSE;
        if (nErro = XPnetmTxFileStart(COM1, 31, "C:\\TRIX\\ARQ.BIN", 1, "RX ARQUIVO"))
                return FALSE;
        while (nAguardando)
                {
                GetMessage(&Msg, NULL, 0, 0);
                DispatchMessage(&Msg);
                }
        return TRUE;
        }

void CALLSEQ WTimerDispatch (HWND hWnd, WORD wMsg, int nId, DWORD dwTime)
        {
        XPnetTransferStatus(COM1, &pS);
        if (TRUE != pS.nActive)
                {
```

```
        nAguardando = 0;
        XPnetmRemote((HWND)NULL, COM1, 31, FALSE);
        XPnetmClose(COM1);
        }
}
```

XPnetmTxFileAbort

Sumário

```
int FAR PASCAL XPnetmTxFileAbort(int nChannel);          // Canal de comunicacao
```

Windows 16	Windows 32	DOS
	•	

Descrição

A função XPnetmTxFileAbort aborta a transmissão iniciada pela função XPnetmTxFileStart no canal especificado.

O argumento 'nCanal' especifica o canal de comunicação, possibilita o uso de no máximo 16 canais (COM1 a COM16).

Valor Retornado

A função retorna zero quando sua execução for bem sucedida, caso contrário retorna um código de erro sempre menor do que zero. O significado de cada código de erro retornado é fornecido no item “Códigos de Erro”.

Consulte

XPnetmTxFileStart.

XPnetmRxProg - Versão para Windows

Sumário

```
int FAR PASCAL XPnetmRxProg (HWND hWnd,          // Handle da janela pai
                             int nCanal,         // Canal de comunicacao
                             int nTerm,          // Terminal
                             const char FAR *cArq // Ponteiro para nome do arquivo
                             );
```

Windows 16	Windows 32	DOS
•	•	

Descrição

A função 'XPnetmRxProg' permite a recepção de um arquivo de programas em formato XPbasic de um terminal para o concentrador da rede XPnet. Para a execução desta função exige-se que o terminal esteja em modo remoto, isto pode ser feito executando-se previamente a função 'XPnetmRemote'

O argumento 'hWnd' especifica o manipulador (handle) da janela pai utilizado para desabilitá-a durante a operação na rede XPnet.

O argumento 'nCanal' especifica o canal de comunicação, possibilita o uso de no máximo 16 canais (COM1 a COM16).

O argumento 'nTerm' informa o número do terminal, que deve ser de 1 a 32.

O argumento tipo caractere '*cArq' informa o nome e a localização com o qual o arquivo de programa proveniente do terminal deverá ser gravado no concentrador.

Valor Retornado

A função retorna zero quando sua execução for bem sucedida, caso contrário retorna um código de erro sempre menor do que zero. O significado de cada código de erro retornado é fornecido no item “Códigos de Erro”.

Consulte

XPnetmRemote, XPnetmWaitCom, XPnetmTxClock, XPnetmTxCmd, XPnetmRxFile, XPnetmTxFile, XPnetmTxProg.

Exemplo

```
#include <windows.h>
#include “xpcom16.h”

char cNomeArq[20];

int PASCAL WinMain(HANDLE hInstance, HANDLE hPrevInstance, LPSTR lpCmdLine, int nCmdShow)
    {
    if (XPnetmOpen(COM1,9600,256,256,XPN_COMAUTO,2))
        return FALSE;
    if (XPnetmRemote((HWND)NULL, COM1, 31, TRUE))
        return FALSE;
    if (XPnetmRxProg((HWND)NULL, COM1, 31, “C:\\TRIX\\RXPROG.BAS”))
        return FALSE;
    if (XPnetmRemote((HWND)NULL, COM1, 31, FALSE))
        return FALSE;
    if (XPnetmClose(COM1))
        return FALSE;
    return TRUE;
    }
```

XPnetmRxProg - Versão para DOS

Sumário

```
int XPnetmRxProg (int nCanal,                          // Canal de comunicacao
                  int nTerm,                           // Terminal
                  unsigned char *cArq,                 // Ponteiro para nome do arquivo
                  int (*pfShowMsg)(unsigned int wNblocos)  // Ponteiro para função que exibe numero de blocos recebidos
                 );
```

Windows 16	Windows 32	DOS
		•

Descrição

A função 'XPnetmRxProg' permite a recepção de um arquivo de programas em formato XPbasic de um terminal para o concentrador da rede XPnet. Para a execução desta função exige-se que o terminal esteja em modo remoto, isto pode ser feito executando-se previamente a função 'XPnetmRemote'. Durante o processo de recepção de programa pode-se mostrar na tela o número de blocos recebidos usando uma função local chamada a cada fim de bloco. Esta função pode testar o teclado e retornar um valor diferente de zero para abortar a operação.

O argumento 'nCanal' especifica o canal de comunicação, possibilita o uso de no máximo 16 canais (COM1 a COM16).

O argumento 'nTerm' informa o número do terminal, que deve ser de 1 a 32.

O argumento tipo caractere '*cArq' informa o nome e a localização com o qual o arquivo de programa proveniente do terminal deverá ser gravado no concentrador.

O argumento ‘(*pfShowMsg)(wNblocos)’ é um ponteiro para uma função local que recebe o número de blocos já recebidos e pode ser usada escrever este número na tela. Esta função pode também abortar a recepção se testar o teclado e retornar um número diferente de zero quando determinada tecla for pressionada. O código de erro retornado é XPN_USERABORT. Caso não se deseje usar esta função fazer pfShowMsg = NULL.

Valor Retornado

A função retorna zero quando sua execução for bem sucedida, caso contrário retorna um código de erro sempre menor do que zero. O significado de cada código de erro retornado é fornecido no item “Códigos de Erro”.

Consulte

XPnetmRemote, XPnetmWaitTx, XPnetmTxClock, XPnetmRxFile, XPnetmTxFile, XPnetmTxProg.

Exemplo

```
#include <stdlib.h>
#include <stdio.h>
#include “xpcomlib.h”
#include “xpcomdef.h”

int ExibeMsg(unsigned int wNblocos);
char cNomeArq[20];

void main(void)
        {
        int (*pfExibeMsg)(unsigned int);
        if (ComAddress(COM1, 0x238, 3))
                exit(1);
        if (XPnetmOpen(COM1,9600,256,256,XPN_COMAUTO,2))
                exit(1);
        if (XPnetmRemote(COM1, 31, XPN_REMOTE))
                exit(1);
        strcpy(cNomeArq, “C:\\TRIX\\RXPROG.BAS”);
        pfExibeMsg = ExibeMsg;
        if (XPnetmRxProg(COM1, 31, cNomeArq, pfExibeMsg))
                exit(1);
        if (XPnetmRemote(COM1, 31, XPN_LOCAL))
                exit(1);
        if (XPnetmClose(COM1))
                exit(1);
        }

int ExibeMsg(unsigned int wNblocos)
        {
        printf(“Numero de blocos recebidos: %03hu”, wNblocos);
        return 0;
        }
```

XPnetmRxProgStart

Sumário

```
int FAR PASCAL XPnetmRxProgStart (int nCanal,                   // Canal de comunicacao
                                  int nTerm,                    // Terminal
                                  const char *cArq,             // Ponteiro para nome do arquivo
                                  const char *cText);                   // Ponteiro para texto a ser mostrado no terminal
```

Windows 16	Windows 32	DOS
	•	

Descrição

A função 'XPnetmRxProgStart' permite iniciar a recepção do arquivo de programa de um terminal para o concentrador da rede XPnet. Esta função retorna imediatamente e a comunicação continua a ser realizada por interrupção e deve ser monitorada dentro de um timer utilizando a função XPnetmTransferStatus. Somente após o término da comunicação ou após o aborto com a função

XPnetmRxProgAbort é que o canal de comunicação deve ser fechado. Para a execução desta função exige-se que o terminal esteja em modo remoto, isto pode ser feito executando-se previamente a função 'XPnetmRemote'.

O argumento 'nCanal' especifica o canal de comunicação, possibilita o uso de no máximo 16 canais (COM1 a COM16).

O argumento 'nTerm' informa o número do terminal, que deve ser de 1 a 32.

O argumento tipo caractere '*cArq' informa o nome e a localização com o qual o arquivo proveniente do terminal deverá ser gravado no concentrador.

O argumento tipo caractere '*cText' informa a mensagem que deve ser exibida no terminal durante a transferência.

Valor Retornado

A função retorna zero quando sua execução for bem sucedida, caso contrário retorna um código de erro sempre menor do que zero. O significado de cada código de erro retornado é fornecido no item “Códigos de Erro”.

Consulte

XPnetmTransferStatus, XPnetmRxProgAbort, XPnetmTxProgStart, XPnetmRxFileStart, XPnetmTxFileStart, XPnetmRemote.

Exemplo

```
#include <windows.h>
#include "xpcom32.h"

XPNET_TRANSFER_STAT         pS;
int nAguardando = 1;
int nErro;
int nId;
HINSTANCE hInst;
void CALLSEQ WTimerDispatch (HWND hWnd, WORD wMsg, int nId, DWORD dwTime);

int PASCAL WinMain(HANDLE hInstance, HANDLE hPrevInstance, LPSTR lpCmdLine, int nCmdShow)
        {
        FARPROC lpFunc;

        hInst = hInstance;
        lpFunc = MakeProcInstance((FARPROC)WTimerDispatch, hInst);
        nId = SetTimer(NULL, 0, 1000, lpFunc);
        if (XPnetmOpen(COM1,9600,256,256,XPN_COMAUTO,2))
                return FALSE;
        if (XPnetmRemote((HWND)NULL, COM1, 31, TRUE))
                return FALSE;
        if (nErro = XPnetmRxProgStart(COM1, 31, "C:\\TRIX\\PROG.BAS", "TX PROGRAMA"))
                return FALSE;
        while (nAguardando)
                {
                GetMessage(&Msg, NULL, 0, 0);
                DispatchMessage(&Msg);
                }
        return TRUE;
        }

void CALLSEQ WTimerDispatch (HWND hWnd, WORD wMsg, int nId, DWORD dwTime)
        {
        XPnetTransferStatus(COM1, &pS);
        if (TRUE != pS.nActive)
                {
                nAguardando = 0;
                XPnetmRemote((HWND)NULL, COM1, 31, FALSE);
                XPnetmClose(COM1);
```

```
        }
    }
```

XPnetmRxProgAbort

Sumário

```
int FAR PASCAL XPnetmRxProgAbort(int nChannel);          // Canal de comunicacao
```

Windows 16	Windows 32	DOS
	•	

Descrição

A função XPnetmRxProgAbort aborta a transmissão iniciada pela função XPnetmRxProgStart no canal especificado.

O argumento 'nCanal' especifica o canal de comunicação, possibilita o uso de no máximo 16 canais (COM1 a COM16).

Valor Retornado

A função retorna zero quando sua execução for bem sucedida, caso contrário retorna um código de erro sempre menor do que zero. O significado de cada código de erro retornado é fornecido no item “Códigos de Erro”.

Consulte

XPnetmRxProgStart.

XPnetmTxProg - Versão para Windows

Sumário

```
int FAR PASCAL XPnetmTxProg (HWND hWnd,          // Handle da janela pai
                             int nCanal,         // Canal de comunicacao
                             int nTerm,          // Terminal
                             const char FAR *cArq // Ponteiro para nome do arquivo
                             );
```

Windows 16	Windows 32	DOS
•	•	

Descrição

A função 'XPnetmTxFile' transmite um arquivo de programa em formato XPbasic do concentrador para um terminal da rede XPnet. Para a execução desta função exige-se que o terminal esteja em modo remoto, isto pode ser feito executando-se previamente a função 'XPnetmRemote'.

O argumento 'hWnd' especifica o manipulador (handle) da janela pai utilizado para desabilitá-a durante a operação na rede XPnet.

O argumento 'nCanal' especifica o canal de comunicação, possibilita o uso de no máximo 16 canais (COM1 a COM16).

O argumento 'nTerm' informa o número do terminal, que deve ser de 1 a 32.

O argumento tipo caractere '*cArq' informa o nome e a localização do o arquivo de programa no concentrador que deverá ser enviado ao terminal.

Valor Retornado

A função retorna zero quando sua execução for bem suce+dida, caso contrário retorna um código de erro sempre menor do que zero. O significado de cada código de erro retornado é fornecido no item “Códigos de Erro”.

Consulte

XPnetmRemote, XPnetmWaitCom, XPnetmTxClock, XPnetmTxCmd, XPnetmRxFile, XPnetmTxFile, XPnetmRxProg.

Exemplo

```
#include <windows.h>
#include “xpcom16.h”

char cNomeArq[20];

int PASCAL WinMain(HANDLE hInstance, HANDLE hPrevInstance, LPSTR lpCmdLine, int nCmdShow)
    {
    if (XPnetmOpen(COM1,9600,256,256,XPN_COMAUTO,2))
        return FALSE;
    if (XPnetmRemote((HWND)NULL, COM1, 31, TRUE))
        return FALSE;
    if (XPnetmTxProg((HWND)NULL, COM1, 31, “C:\\TRIX\\TXPROG.BAS”))
        return FALSE;
    if (XPnetmRemote((HWND)NULL, COM1, 31, FALSE))
        return FALSE;
    if (XPnetmClose(COM1))
        return FALSE;
    return TRUE;
    }
```

XPnetmTxProg - Versão para DOS

Sumário

```
int XPnetmTxProg (int nCanal,                                                  // Canal de comunicacao
                  int  nTerm,                                                  // Terminal
                  unsigned char *cArq,                                         // Ponteiro para nome do arquivo
                  int (*pfShowMsg)(unsigned int wNblocos, unsigned int wNtotal)// Ponteiro para função que exibe nº de
                                                                               // blocos transmitidos e nº total de blocos
                 );
```

Windows 16	Windows 32	DOS
		•

Descrição

A função 'XPnetmTxFile' transmite um arquivo de programa em formato XPbasic do concentrador para um terminal da rede XPnet. Para a execução desta função exige-se que o terminal esteja em modo remoto, isto pode ser feito executando-se previamente a função 'XPnetmRemote'. Durante o processo de transmissão de programa pode-se mostrar na tela o número de blocos transmitidos e o número de blocos total usando uma função local chamada a cada fim de bloco. Esta função pode testar o teclado e retornar um valor diferente de zero para abortar a operação.

O argumento 'nCanal' especifica o canal de comunicação, possibilita o uso de no máximo 16 canais (COM1 a COM16).

O argumento 'nTerm' informa o número do terminal, que deve ser de 1 a 32.

O argumento tipo caractere '*cArq' informa o nome e a localização do o arquivo de programa no concentrador que deverá ser enviado ao terminal.

O argumento ‘(*pfShowMsg)(wNblocos, wNtotal)’ é um ponteiro para uma função local que recebe o número de blocos já transmitidos e o número total de blocos e pode ser usada para escrever estes números na tela. Esta função pode também abortar a transmissão se testar o teclado e retornar um número diferente de zero quando determinada tecla for pressionada. O código de erro retornado é XPN_USERABORT. Caso não se deseje usar esta função fazer pfShowMsg = NULL.

Valor Retornado

A função retorna zero quando sua execução for bem suce+dida, caso contrário retorna um código de erro sempre menor do que zero. O significado de cada código de erro retornado é fornecido no item “Códigos de Erro”.

Consulte

XPnetmRemote, XPnetmWaitTx, XPnetmTxClock, XPnetmRxFile, XPnetmTxFile, XPnetmRxProg.

Exemplo

```
#include <stdlib.h>
#include <stdio.h>
#include “xpcomlib.h”
#include “xpcomdef.h”

int ExibeMsg(unsigned int wNblocos, unsigned int wNtotal);
char cNomeArq[20];

void main(void)
        {
        int (*pfExibeMsg)(unsigned int, unsigned int);
        if (ComAddress(COM1, 0x238, 3))
                exit(1);
        if (XPnetmOpen(COM1,9600,256,256,XPN_COMAUTO,2))
                exit(1);
        if (XPnetmRemote(COM1, 31, XPN_REMOTE))
                exit(1);
        strcpy(cNomeArq, “C:\\TRIX\\TXPROG.BAS”);
        pfExibeMsg = ExibeMsg;
        if (XPnetmTxProg(COM1, 31, cNomeArq, pfExibeMsg))
                exit(1);
        if (XPnetmRemote(COM1, 31, XPN_LOCALRUN))
                exit(1);
        if (XPnetmClose(COM1))
                exit(1);
        }

int ExibeMsg(unsigned int wNblocos, unsigned int wNtotal)
        {
        printf(“Numero de blocos transmitidos: %03hu/%03hu “, wNblocos, wNtotal);
        return 0;
        }
```

XPnetmTxProgStart

Sumário

```
int FAR PASCAL XPnetmTxProgStart (int nCanal,                   // Canal de comunicacao
                                  int nTerm,                    // Terminal
                                  const char *cArq,             // Ponteiro para nome do arquivo
                                  const char *cText);                   // Ponteiro para texto a ser mostrado no terminal
```

Windows 16	Windows 32	DOS
	•	

Descrição

A função 'XPnetmTxProgStart' permite iniciar a transmissão de um arquivo de programa do concentrador para um terminal da rede XPnet. Esta função retorna imediatamente e a comunicação continua a ser realizada por interrupção e deve ser monitorada dentro de um timer utilizando a função XPnetmTransferStatus. Somente após o término da comunicação ou após o aborto com a função

XPnetmTxProgAbort é que o canal de comunicação deve ser fechado. Para a execução desta função exige-se que o terminal esteja em modo remoto, isto pode ser feito executando-se previamente a função 'XPnetmRemote'.

O argumento 'nCanal' especifica o canal de comunicação, possibilita o uso de no máximo 16 canais (COM1 a COM16).

O argumento 'nTerm' informa o número do terminal, que deve ser de 1 a 32.

O argumento tipo caractere '*cArq' informa o nome e a localização do arquivo, no concentrador, que será enviado ao terminal.

O argumento tipo caractere '*cText' informa a mensagem que deve ser exibida no terminal durante a transferência.

Valor Retornado

A função retorna zero quando sua execução for bem sucedida, caso contrário retorna um código de erro sempre menor do que zero. O significado de cada código de erro retornado é fornecido no item “Códigos de Erro”.

Consulte

XPnetmTransferStatus, XPnetmTxProgAbort, XPnetmRxProgStart, XPnetmRxFileStart, XPnetmTxFileStart, XPnetmRemote.

Exemplo

```
#include <windows.h>
#include "xpcom32.h"

XPNET_TRANSFER_STAT         pS;
int nAguardando = 1;
int nErro;
int nId;
HINSTANCE hInst;
void CALLSEQ WTimerDispatch (HWND hWnd, WORD wMsg, int nId, DWORD dwTime);

int PASCAL WinMain(HANDLE hInstance, HANDLE hPrevInstance, LPSTR lpCmdLine, int nCmdShow)
        {
        FARPROC lpFunc;

        hInst = hInstance;
        lpFunc = MakeProcInstance((FARPROC)WTimerDispatch, hInst);
        nId = SetTimer(NULL, 0, 1000, lpFunc);
        if (XPnetmOpen(COM1,9600,256,256,XPN_COMAUTO,2))
                return FALSE;
        if (XPnetmRemote((HWND)NULL, COM1, 31, TRUE))
                return FALSE;
        if (nErro = XPnetmTxProgStart(COM1, 31, "C:\\TRIX\\PROG.BAS", "RX PROGRAMA"))
                return FALSE;
        while (nAguardando)
                {
                GetMessage(&Msg, NULL, 0, 0);
                DispatchMessage(&Msg);
                }
        return TRUE;
        }

void CALLSEQ WTimerDispatch (HWND hWnd, WORD wMsg, int nId, DWORD dwTime)
        {
        XPnetTransferStatus(COM1, &pS);
        if (TRUE != pS.nActive)
                {
                nAguardando = 0;
                XPnetmRemote((HWND)NULL, COM1, 31, FALSE);
                XPnetmClose(COM1);
                }
```

}

XPnetmTxProgAbort

Sumário

```
int FAR PASCAL XPnetmTxProgAbort(int nChannel);          // Canal de comunicacao
```

Windows 16	Windows 32	DOS
	•	

Descrição

A função XPnetmTxProgAbort aborta a transmissão iniciada pela função XPnetmTxProgStart no canal especificado.

O argumento 'nCanal' especifica o canal de comunicação, possibilita o uso de no máximo 16 canais (COM1 a COM16).

Valor Retornado

A função retorna zero quando sua execução for bem sucedida, caso contrário retorna um código de erro sempre menor do que zero. O significado de cada código de erro retornado é fornecido no item “Códigos de Erro”.

Consulte

XPnetmTxProgStart.

XPnetTransferStatus

Sumário

```
int FAR PASCAL XPnetTransferStatus(int nChannel,                    // Canal de comunicacao
                                   XPNET_TRANSFER_STAT* pS);        // Ponteiro para estrutura de status da
                                                                    // transferencia
```

Windows 16	Windows 32	DOS
	•	

Descrição

A função XPnetTransferStatus atualiza a estrutura de status da transmissão que foi iniciada pela função XPnetTxFileStart, XPnetRxFileStart, XPnetTxProgStart ou XPnetRxProgStart no canal especificado. Deve ser executada dentro de um timer para monitorar a comunicação.

O argumento 'nCanal' especifica o canal de comunicação, possibilita o uso de no máximo 16 canais (COM1 a COM16).

O argumento pS é o endereço da estrutura do tipo XPNET_TRANSFER_STAT que vai receber o status da Transferência de Arquivo em Rede XPnet. Esta estrutura tem os seguintes campos:

int	nActive	TRUE ou 1 se ativa, FALSE ou 0 caso contrário
int	nType	Tipo de transferência: XPNXFER_TXPROG, XPNXFER_RXPROG, XPNXFER_TXFILE ou XPNXFER_RXFILE
int	nTerm	Terminal sendo utilizado
int	nError	Código do último erro ocorrido
DWORD	dwTxTotal	Número total de blocos a transmitir
DWORD	dwTxBlocks	Número total de blocos transmitidos
DWORD	dwRxTotal	Número total de blocos a receber
DWORD	dwRxBlocks	Número de blocos recebidos

Valor Retornado

A função retorna zero quando sua execução for bem sucedida, caso contrário retorna um código de erro sempre menor do que zero. O significado de cada código de erro retornado é fornecido no item “Códigos de Erro”.

Consulte

XPnetTxFileStart, XPnetRxFileStart, XPnetTxProgStart ou XPnetRxProgStart.

Exemplo

Veja exemplo nas funções XPnetTxFileStart, XPnetRxFileStart, XPnetTxProgStart ou XPnetRxProgStart.

Funções de Comunicação para Protocolo XMODEM

Estas funções são destinadas a fornecer funções adicionais às funções básicas de comunicação, para que se possa realizar comunicação com os coletores de dados TRIX utilizando o protocolo XMODEM com *check sum*. Para utilizar estas funções é necessário que o canal esteja inicializado operacionalmente com as funções ComAddress e ComOpen. Em cada canal de comunicação só pode ser conectado um coletor, mas pode-se ter mais do que um coletor se estiverem conectados a canais diferentes.

Sob ambiente Windows:

As funções de transmissão e recepção do protocolo XMODEM sob ambiente Windows, XModemTxFileStart e XModemRxFileStart iniciam a comunicação e retornam imediatamente, permitindo que seja iniciada a comunicação em diversos canais simultaneamente. Posteriormente deve-se executar as funções XModemTxStatus e XModemRxStatus dentro de um timer da aplicação para monitorar a transferência. Pode-se abortar a comunicação utilizando as funções XModemTxAbort e XModemRxAbort.

Pode-se alterar os parâmetros de comunicação do protocolo XMODEM utilizando as funções XModemTxConfig e XModemRxConfig.

Sob ambiente DOS:

Para enviar ou receber arquivos de um único coletor usar XmodemTxFile e XmodemRxFile.

Para enviar ou receber arquivos de vários coletores conectados a canais diferentes, simultaneamente, utilizar as funções XmodemRxOpen ou XmodemTxOpen. Abrir um canal para transmissão ou recepção e processar a transferência por XmodemTxProcess ou XmodemRxProcess. Estas funções processam os caracteres disponíveis num determinado canal e retornam a seguir, podendo assim ser usadas para processar a comunicação seqüencialmente em diversos canais diferentes. Após processada a comunicação em determinado canal deve-se fechá-lo com XmodemTxClose ou XmodemRxClose.

XModemTxConfig

Sumário

```
int FAR PASCAL XModemTxConfig (int nChannel,        // Canal de comunicacao
                               int nConfig,         // Item da configuracao a ser alterado
                               long lValue);        // Novo valor do item da configuração
```

Windows 16	Windows 32	DOS
•	•	

Descrição

A função XModemTxConfig é opcional e serve para modificar os valores da configuração padrão utilizada na transmissão com o protocolo XMODEM.

O argumento 'nCanal' especifica o canal de comunicação, possibilita o uso de no máximo 16 canais (COM1 a COM16).

O argumento nConfig especifica o item da configuração que se deseja alterar. Os itens possíveis são:

XPM_TXINIT_TIMEOUT	time-out de espera inicial da transmissão, padrão 60000 (60 s)
XPM_TXACK_TIMEOUT	time-out de espera de ACK após transmissão de bloco, padrão 1000 (1 s)
XPM_TXACKEOT_TIMEOUT	time-out de espera de ACK após EOT, padrão 1000 (1 s)

O argumento lValue é o novo valor que será atribuido ao item da configuração que está sendo alterado.

Valor Retornado

A função retorna zero quando sua execução for bem sucedida, caso contrário retorna um código de erro sempre menor do que zero. O significado de cada código de erro retornado é fornecido no item “Códigos de Erro”.

Consulte

XModemTxFileStart, XModemTxAbort, XModemTxStatus.

Exemplo

```
#include <windows.h>
#include “xpcom16.h”

int PASCAL WinMain(HANDLE hInstance, HANDLE hPrevInstance, LPSTR lpCmdLine, int nCmdShow)
    {
    if (ComOpen(COM1, 9600, 8, 1, COM_NONEP, COM_NOFLOW, COM_NOFLOW, 256, 256))
        return FALSE;
    if (XModemTxConfig(COM1, XPM_TXINIT_TIMEOUT, 180000L))          // aguarda inicio por 3 minutos
        return FALSE;
    if (ComClose(COM1))
        return FALSE;
    return TRUE;
    }
```

XModemTxFileStart

Sumário

```
int FAR PASCAL XModemTxFileStart(int nChannel,          // Canal de comunicacao
                                 LPCSTR pszFile);       // Ponteiro para nome do arquivo
```

Windows 16	Windows 32	DOS
•	•	

Descrição

A função XModemTxFileStart inicia a transmissão de um arquivo utilizando o protocolo XMODEM com *check sum*. Antes de executar esta função é necessário abrir o canal de comunicação, reservando espaço de pelo menos 132 caracteres para os blocos de dados. Esta função retorna imediatamente e a comunicação continua a ser realizada por interrupção e deve ser monitorada dentro de um timer utilizando a função XModemTxStatus. Somente após o término da comunicação ou após o aborto com a função XModemTxAbort é que o canal de comunicação deve ser fechado.

O argumento 'nCanal' especifica o canal de comunicação, possibilita o uso de no máximo 16 canais (COM1 a COM16).

O argumento tipo caractere 'pszFile' informa o nome e a localização do arquivo, do concentrador, que será enviado.

Valor Retornado

A função retorna zero quando sua execução for bem sucedida, caso contrário retorna um código de erro sempre menor do que zero. O significado de cada código de erro retornado é fornecido no item “Códigos de Erro”.

Consulte

XModemTxConfig, XModemTxAbort, XModemTxStatus.

Exemplo

```
#include <windows.h>
#include "xpcom32.h"
```

```
XMODEMSTATS       pS;
int nAguardando = 1;
int nErro;
int nId;
HINSTANCE hInst;
void CALLSEQ WTimerDispatch (HWND hWnd, WORD wMsg, int nId, DWORD dwTime);

int PASCAL WinMain(HANDLE hInstance, HANDLE hPrevInstance, LPSTR lpCmdLine, int nCmdShow)
        {
        FARPROC lpFunc;

        hInst = hInstance;
        lpFunc = MakeProcInstance((FARPROC)WTimerDispatch, hInst);
        nId = SetTimer(NULL, 0, 1000, lpFunc);
        if (ComOpen(COM1, 9600, 8, 1, COM_NONEP, COM_NOFLOW, COM_NOFLOW, 256, 256))
                return FALSE;
        if (nErro = XModemTxFileStart(COM1, "C:\\TRIX\\PROG.BAS"))
                return FALSE;
        while (nAguardando)
                {
                GetMessage(&Msg, NULL, 0, 0);
                DispatchMessage(&Msg);
                }
        return TRUE;
        }

void CALLSEQ WTimerDispatch (HWND hWnd, WORD wMsg, int nId, DWORD dwTime)
        {
        XModemTxStatus(COM1, &pS);
        if (XPM_ACTIVE != pS.nStatus)
                {
                ComClose(COM1);
                nAguardando = 0;
                }
        }
```

XModemTxAbort

Sumário

```
int FAR PASCAL XModemTxAbort(int nChannel);                    // Canal de comunicacao
```

Windows 16	Windows 32	DOS
•	•	

Descrição

A função XModemTxAbort aborta a transmissão iniciada pela função XModemTxStart no canal especificado.

O argumento 'nCanal' especifica o canal de comunicação, possibilita o uso de no máximo 16 canais (COM1 a COM16).

Valor Retornado

A função retorna zero quando sua execução for bem sucedida, caso contrário retorna um código de erro sempre menor do que zero. O significado de cada código de erro retornado é fornecido no item “Códigos de Erro”.

Consulte

XModemTxConfig, XModemTxFileStart, XModemTxStatus.

XModemTxStatus

Sumário

```
int FAR PASCAL XModemTxStatus  (int nChannel,                // Canal de comunicacao
                                XMODEMSTATS* pS);            // Ponteiro para estrutura de status da transmissao
```

Windows 16	Windows 32	DOS
•	•	

Descrição

A função XModemTxStatus atualiza a estrutura de status da transmissão que foi iniciada pela função XModemTxStart no canal especificado. Deve ser executada dentro de um timer para monitorar a comunicação.

O argumento 'nCanal' especifica o canal de comunicação, possibilita o uso de no máximo 16 canais (COM1 a COM16).

O argumento pS é o endereço da estrutura do tipo XMODEMSTATS que vai receber o status da comunicacão. Esta estrutura tem os seguintes campos:

int	nStatus	Status: XPM_ACTIVE se ativa, 0 se terminou ou último erro se menor que 0
DWORD	dwTxTotal	Número total de blocos a transmitir
DWORD	dwTxBlocks	Número total de blocos transmitidos
DWORD	dwRxBlocks	Número de blocos recebidos

Valor Retornado

A função retorna zero quando sua execução for bem sucedida, caso contrário retorna um código de erro sempre menor do que zero. O significado de cada código de erro retornado é fornecido no item “Códigos de Erro”.

Consulte

XModemTxConfig, XModemTxFileStart, XModemTxAbort.

Exemplo

Ver exemplo na função XModemTxFileStart.

XModemRxConfig

Sumário

```
int FAR PASCAL XModemRxConfig (int nChannel,                 // Canal de comunicacao
                               int nConfig,                  // Item da configuracao a ser alterado
                               long lValue);                 // Novo valor do item da configuração
```

Windows 16	Windows 32	DOS
•	•	

Descrição

A função XModemRxConfig é opcional e serve para modificar os valores da configuração padrão utilizada na recepção com o protocolo XMODEM.

O argumento 'nCanal' especifica o canal de comunicação, possibilita o uso de no máximo 16 canais (COM1 a COM16).

O argumento nConfig especifica o item da configuração que se deseja alterar. Os itens possíveis são:

XPM_RXNAKINIT_TIMEOUT	time-out entre envio de NAKs iniciais, padrão 10000 (10 s)
XPM_RXSOH_TIMEOUT	time-out de espera de SOH após envio de ACK ou NAK, padrão 10000 (10 s)
XPM_RXERRSOH_TIMEOUT	time-out de espera de SOH após erro na recepção de SOH, padrão 500 (0,5 s)
XPM_RXBLOCK_TIMEOUT	time-out de espera de bloco após SOH, padrão 10000 (10 s)

XPM_RXMAXERRORS numero máximo de erros de recepção antes de abortar comunicação, padrão 10

O argumento lValue é o novo valor que será atribuido ao item da configuração que está sendo alterado.

Valor Retornado

A função retorna zero quando sua execução for bem sucedida, caso contrário retorna um código de erro sempre menor do que zero. O significado de cada código de erro retornado é fornecido no item “Códigos de Erro”.

Consulte

XModemRxFileStart, XModemRxAbort, XModemRxStatus.

Exemplo

```
#include <windows.h>
#include “xpcom16.h”

int PASCAL WinMain(HANDLE hInstance, HANDLE hPrevInstance, LPSTR lpCmdLine, int nCmdShow)
    {
    if (ComOpen(COM1, 9600, 8, 1, COM_NONEP, COM_NOFLOW, COM_NOFLOW, 256, 256))
        return FALSE;
    if (XModemTxConfig(COM1, XPM_RXNAKINIT_TIMEOUT, 3000))          // envie 1 NAK a cada 3 s
        return FALSE;
    if (ComClose(COM1))
        return FALSE;
    return TRUE;
    }
```

XModemRxFileStart

Sumário

```
int FAR PASCAL XModemRxFileStart(int nChannel,          // Canal de comunicacao
                                 LPCSTR pszFile);       // Ponteiro para nome do arquivo
```

Windows 16	Windows 32	DOS
•	•	

Descrição

A função XModemRxFileStart inicia a recepção de um arquivo utilizando o protocolo XMODEM com *check sum*. Antes de executar esta função é necessário abrir o canal de comunicação, reservando espaço de pelo menos 132 caracteres para os blocos de dados. Esta função retorna imediatamente e a comunicação continua a ser realizada por interrupção e deve ser monitorada dentro de um timer utilizando a função XModemRxStatus. Somente após o término da comunicação ou após o aborto com a função XModemRxAbort é que o canal de comunicação deve ser fechado.

O argumento 'nCanal' especifica o canal de comunicação, possibilita o uso de no máximo 16 canais (COM1 a COM16).

O argumento tipo caractere 'pszFile' informa o nome e a localização, no concentrador, onde será guardado o arquivo recebido.

Valor Retornado

A função retorna zero quando sua execução for bem sucedida, caso contrário retorna um código de erro sempre menor do que zero. O significado de cada código de erro retornado é fornecido no item “Códigos de Erro”.

Consulte

XModemRxConfig, XModemRxAbort, XModemRxStatus.

Exemplo

Ver exemplo da função XModemTxFileStart.

XModemRxAbort

Sumário

```
int FAR PASCAL XModemRxAbort(int nChannel);                    // Canal de comunicacao
```

Windows 16	Windows 32	DOS
•	•	

Descrição

A função XModemRxAbort aborta a transmissão iniciada pela função XModemRxStart no canal especificado.

O argumento 'nCanal' especifica o canal de comunicação, possibilita o uso de no máximo 16 canais (COM1 a COM16).

Valor Retornado

A função retorna zero quando sua execução for bem sucedida, caso contrário retorna um código de erro sempre menor do que zero. O significado de cada código de erro retornado é fornecido no item “Códigos de Erro”.

Consulte

XModemRxConfig, XModemRxFileStart, XModemRxStatus.

XModemRxStatus

Sumário

```
int FAR PASCAL XModemRxStatus  (int nChannel,                  // Canal de comunicacao
                                XMODEMSTATS* pS);              // Ponteiro para estrutura de status da recepcao
```

Windows 16	Windows 32	DOS
•	•	

Descrição

A função XModemRxStatus atualiza a estrutura de status da recepção que foi iniciada pela função XModemRxStart no canal especificado. Deve ser executada dentro de um timer para monitorar a comunicação.

O argumento 'nCanal' especifica o canal de comunicação, possibilita o uso de no máximo 16 canais (COM1 a COM16).

O argumento pS é o endereço da estrutura do tipo XMODEMSTATS que vai receber o status da comunicacão. Esta estrutura tem os seguintes campos:

int	nStatus	Status: XPM_ACTIVE se ativa, 0 se terminou ou último erro se menor que 0
DWORD	dwTxTotal	Número total de blocos a transmitir
DWORD	dwTxBlocks	Número total de blocos transmitidos
DWORD	dwRxBlocks	Número de blocos recebidos

Valor Retornado

A função retorna zero quando sua execução for bem sucedida, caso contrário retorna um código de erro sempre menor do que zero. O significado de cada código de erro retornado é fornecido no item “Códigos de Erro”.

Consulte

XModemRxConfig, XModemRxFileStart, XModemRxAbort.

Exemplo

Ver exemplo da função XModemTxFileStart.

XmodemRxOpen

Sumário

```
int XmodemRxOpen(int nCanal,                                  // Canal de comunicacao
                 char* cArq,                                  // Ponteiro para nome do arquivo
                 int (* pfShowMsg)(unsigned int wNblocks)     // Ponteiro para função que exibe numero de blocos recebidos
                );
```

Windows 16	Windows 32	DOS
		•

Descrição

A função 'XmodemRxOpen' abre uma estrutura de controle para o canal especificado e o arquivo a ser recebido. Sinaliza início de transmissão e inicializa estado de espera por início de bloco utilizando o protocolo XMODEM com *check sum*. O canal de comunicação deve ter sido previamente inicializado operacionalmente.

O argumento 'nCanal' especifica o canal de comunicação, possibilita o uso de no máximo 16 canais (COM1 a COM16).

O argumento tipo caractere '*cArq' é um ponteiro que informa o nome e a localização do arquivo a ser recebido do terminal.

O argumento ‘(*pfShowMsg)(wNblocos)’ é um ponteiro para a função local que recebe o número de blocos já recebidos e pode ser usada escrever este número na tela. Esta função pode também abortar a recepção se testar o teclado e retornar um número diferente de zero quando determinada tecla for pressionada, retornando zero a recepção continua. O código de erro retornado é XPN_USERABORT. Caso não se deseje usar esta função fazer pfShowMsg = NULL.

Valor Retornado

A função retorna zero quando a execução for bem sucedida, caso contrário retorna um código de erro sempre menor do que zero. O significado de cada código de erro retornado é fornecido no item “Códigos de Erro”.

Consulte

XmodemRxProcess, XmodemRxClose.

Exemplo

```
#include <stdlib.h>
#include <stdio.h>
#include “xpcomlib.h”
#include “xpcomdef.h”

int ExibeMsg (unsigned int wNblocos);
char cNomeArq[20];
int nError;

void main(void)
    {
    int (*pfExibeMsg)(unsigned int);
    if (ComAddress(COM1, 0x238, 3))
        exit(1);
    if (ComOpen(COM1,9600,8,1,COM_NONEP,COM_NOFLOW,COM_NOFLOW,256,256, FALSE))
        exit(1);
    strcpy(cNomeArq, “C:\\TRIX\\RXARQ.BIN”);
    pfExibeMsg = ExibeMsg;
    if (XmodemRxOpen(COM1,cNomeArq,pfExibeMsg))
        exit(1);
    while (TRUE)
```

```
        {
        nError = XmodemRxProcess(COM1);
        if (nError != XM_ACTIVE)
            exit(1);
        }
    if (XmodemRxClose(COM1)
        exit(1);
    if (ComClose(COM1))
        exit(1);
    }

int ExibeMsg(unsigned int wNblocos)
    {
    printf("\nNumero de blocos recebidos: %03hu", wNblocos);
    return 0;
    }
```

XmodemTxOpen

Sumário

```
int XmodemTxOpen(int nCanal,                                                    // Canal de comunicacao
                 unsigned char* cArq,                                           // Ponteiro para nome do arquivo
                 int (* pfShowMsg)(unsigned int wNblocks, unsigned int wNtotal) // Ponteiro para função que exibe nº de blocos
                                                                                // transmitidos e numero total de blocos
                );
```

Windows 16	Windows 32	DOS
		•

Descrição

A função 'XmodemTxOpen' abre uma estrutura de controle para o canal especificado e o arquivo a ser transmitido. Inicializa estado de espera por sinalização de início de transmissão utilizando o protocolo XMODEM com *check sum*. O canal de comunicação deve ter sido previamente inicializado operacionalmente.

O argumento 'nCanal' especifica o canal de comunicação, possibilita o uso de no máximo 16 canais (COM1 a COM16).

O argumento tipo caractere '*cArq' é um ponteiro que informa o nome e a localização do arquivo a ser enviado ao terminal.

O argumento '(*pfShowMsg)(wNblocos, wNtotal)' é um ponteiro para a função local que recebe o número de blocos já transmitidos e o número total de blocos e pode ser usada para escrever estes números na tela. Esta função pode também abortar a transmissão se testar o teclado e retornar um número diferente de zero quando determinada tecla for pressionada, retornando zero a transmissão continua. O código de erro retornado é XPN_USERABORT. Caso não se deseje usar esta função fazer pfShowMsg = NULL.

Valor Retornado

A função retorna zero quando a execução for bem sucedida, caso contrário retorna um código de erro sempre menor do que zero. O significado de cada código de erro retornado é fornecido no item "Códigos de Erro".

Consulte

XmodemTxProcess, XmodemTxClose.

Exemplo

```
#include <stdlib.h>
#include <stdio.h>
#include "xpcomlib.h"
#include "xpcomdef.h"
```

```
int ExibeMsg(unsigned int wNblocos, unsigned int wNtotal);
char cNomeArq[20];
int nError;

void main(void)
        {
        int (*pfExibeMsg)(unsigned int, unsigned int);
        if (ComAddress(COM1, 0x238, 3))
                exit(1);
        if (ComOpen(COM1,9600,8,1,COM_NONEP,COM_NOFLOW,COM_NOFLOW,256,256, FALSE))
                exit(1);
        strcpy(cNomeArq, "C:\\TRIX\\RXARQ.BIN");
        pfExibeMsg = ExibeMsg;
        if (XmodemTxOpen(COM1,cNomeArq,pfExibeMsg))
                exit(1);
        while (TRUE)
                {
                nError = XmodemTxProcess(COM1);
                if (nError != XM_ACTIVE)
                        exit(1);
                }
        if (XmodemTxClose(COM1)
                exit(1);
        if (ComClose(COM1))
                exit(1);
        }

int ExibeMsg(unsigned int wNblocos, unsigned int wNtotal)
        {
        printf("\nNumero de blocos transmitidos: %03hu/%03hu", wNblocos, wNtotal);
        return 0;
        }
```

XmodemRxProcess

Sumário

```
int XmodemRxProcess(int nCanal                    // Canal de comunicacao
                    );
```

Windows 16	Windows 32	DOS
		•

Descrição

A função 'XmodemRxProcess' processa recepção de caracteres de controle e dados do terminal e envia para rotinas que controlam a recepção utilizando o protocolo XMODEM com *check sum*. Retorna quando não houver mais caracteres do terminal a processar. O canal de comunicação deve ter sido previamente aberto com a função XmodemRxOpen.

O argumento 'nCanal' especifica o canal de comunicação, possibilita o uso de no máximo 16 canais (COM1 a COM16).

Valor Retornado

A função retorna zero no fim da comunicação ou XM_ACTIVE quando não houver mais caracteres recebidos a processar e não houverem excesso de erros ou estouro de tempo, caso contrário retorna um código de erro sempre menor do que zero. O significado de cada código de erro retornado é fornecido no item "Códigos de Erro".

Consulte

XmodemRxOpen, XmodemRxClose.

Exemplo

Ver exemplo da função XmodemRxOpen.

XmodemTxProcess

Sumário

```
int XmodemTxProcess(int nCanal                    // Canal de comunicacao
                    );
```

Windows 16	Windows 32	DOS
		•

Descrição

A função 'XmodemTxProcess' processa recepção de caracteres de controle do terminal e envia para rotinas que controlam a transmissão utilizando o protocolo XMODEM com *check sum*. Retorna quando não houver mais caracteres de controle do terminal a processar. O canal de comunicação deve ter sido previamente aberto com a função XmodemTxOpen.

O argumento 'nCanal' especifica o canal de comunicação, possibilita o uso de no máximo 16 canais (COM1 a COM16).

Valor Retornado

A função retorna zero no fim da comunicação ou XM_ACTIVE quando não houver mais caracteres de controle recebidos a processar e não houverem excesso de erros ou estouro de tempo, caso contrário retorna um código de erro sempre menor do que zero. O significado de cada código de erro retornado é fornecido no item “Códigos de Erro”.

Consulte

XmodemTxOpen, XmodemTxClose.

Exemplo

Ver exemplo da função XmodemTxOpen.

XmodemRxClose

Sumário

```
int XmodemRxClose(int nCanal                      // Canal de comunicacao
                  );
```

Windows 16	Windows 32	DOS
		•

Descrição

A função 'XmodemRxClose' fecha a estrutura de controle para o canal especificado e o arquivo recebido, finalizando a recepção com o protocolo XMODEM com *check sum*. O canal de comunicação deve ter sido previamente aberto com a função XmodemRxOpen.

O argumento 'nCanal' especifica o canal de comunicação, possibilita o uso de no máximo 16 canais (COM1 a COM16).

Valor Retornado

A função retorna zero quando a execução for bem sucedida, caso contrário retorna um código de erro sempre menor do que zero. O significado de cada código de erro retornado é fornecido no item “Códigos de Erro”.

Consulte

XmodemRxOpen, XmodemRxProcess.

Exemplo

Ver exemplo da função XmodemRxOpen.

XmodemTxClose

Sumário

```
int XmodemTxClose(int nCanal                    // Canal de comunicacao
                  );
```

Windows 16	Windows 32	DOS
		•

Descrição

A função 'XmodemTxClose' fecha a estrutura de controle para o canal especificado e o arquivo transmitido, finalizando a transmissão como protocolo XMODEM com *check sum*. O canal de comunicação deve ter sido previamente aberto com a função XmodemTxOpen.

O argumento 'nCanal' especifica o canal de comunicação, possibilita o uso de no máximo 16 canais (COM1 a COM16).

Valor Retornado

A função retorna zero quando a execução for bem sucedida, caso contrário retorna um código de erro sempre menor do que zero. O significado de cada código de erro retornado é fornecido no item “Códigos de Erro”.

Consulte

XmodemTxOpen, XmodemTxProcess.

Exemplo

Ver exemplo da função XmodemTxOpen.

XmodemRxFile

Sumário

```
int XmodemRxFile (int nCanal,                          // Canal de comunicacao
                  unsigned char *cArq,                 // Ponteiro para nome do arquivo
                  int (*pfShowMsg)(unsigned int wNblocos) // Ponteiro para função que exibe numero de blocos recebidos
                 );
```

Windows 16	Windows 32	DOS
		•

Descrição

A função 'XmodemRxFile' recebe um arquivo, de dados ou programa, do coletor de dados TRIX para um concentrador utilizando protocolo XMODEM com *check sum*. O canal de comunicação deve ter sido previamente inicializado operacionalmente.

O argumento 'nCanal' especifica o canal de comunicação, possibilita o uso de no máximo 16 canais (COM1 a COM16).

O argumento tipo caractere '*cArq' é um ponteiro que informa o nome e a localização com o qual o arquivo proveniente do terminal deverá ser gravado no concentrador.

O argumento ‘(*pfShowMsg)(wNblocos)’ é um ponteiro para a função local que recebe o número de blocos já recebidos e pode ser usada escrever este número na tela. Esta função pode também abortar a recepção se testar o teclado e retornar um número diferente de zero quando determinada tecla for pressionada, retornando zero a recepção continua. O código de erro retornado é XPN_USERABORT. Caso não se deseje usar esta função fazer pfShowMsg = NULL.

Valor Retornado

A função retorna zero quando a execução for bem sucedida, caso contrário retorna um código de erro sempre menor do que zero. O significado de cada código de erro retornado é fornecido no item “Códigos de Erro”.

Consulte

XmodemTxFile.

Exemplo

```
#include <stdlib.h>
#include <stdio.h>
#include “xpcomlib.h”
#include “xpcomdef.h”

int ExibeMsg(unsigned int wNblocos);
char cNomeArq[20];

void main(void)
        {
        int (*pfExibeMsg)(unsigned int);
        if (ComAddress(COM1, 0x238, 3))
                exit(1);
        if (ComOpen(COM1,9600,8,1,COM_NONEP,COM_NOFLOW,COM_NOFLOW,256,256, FALSE))
                exit(1);
        strcpy(cNomeArq, “C:\\TRIX\\RXARQ.BIN”);
        pfExibeMsg = ExibeMsg;
        if (XmodemRxFile(COM1,cNomeArq,pfExibeMsg))
                exit(1);
        if (ComClose(COM1))
                exit(1);
        }

int ExibeMsg(unsigned int wNblocos)
        {
        printf(“\nNumero de blocos recebidos: %03hu”, wNblocos);
        return 0;
        }
```

XmodemTxFile

Sumário

```
int XmodemTxFile (int nCanal,                                                   // Canal de comunicacao
                  unsignet char *cArq,                                          // Ponteiro para nome do arquivo
                  int (*pfShowMsg)(unsigned int wNblocos, unsigned int wTotal)  // Ponteiro para função que exibe numero
                                                                                // de blocos transmitidos
                  );
```

Windows 16	Windows 32	DOS
		•

Descrição

A função 'XmodemTxFile' transmite um arquivo, de dados ou programa, do coletor de dados TRIX para um concentrador utilizando o protocolo XMODEM com *check sum*. O canal de comunicação deve ter sido previamente inicializado operacionalmente.

O argumento 'nCanal' especifica o canal de comunicação, possibilita o uso de no máximo 16 canais (COM1 a COM16).

O argumento tipo caractere 'cArq' é um ponteiro que informa o nome e a localização com o qual o arquivo proveniente do terminal deverá ser gravado no concentrador.

O argumento ‘(*pfShowMsg)(wNblocos)’ é um ponteiro para a função local que recebe o número de blocos já transmitidos e o número total de blocos e pode ser usada para escrever estes números na tela. Esta função pode também abortar a transmissão se testar o teclado e retornar um número diferente de zero quando determinada tecla for pressionada, retornando zero a transmissão continua. O código de erro retornado é XPN_USERABORT. Caso não se deseje usar esta função fazer pfShowMsg = NULL.

Valor Retornado

A função retorna zero quando a execução for bem sucedida, caso contrário retorna um código de erro sempre menor do que zero. O significado de cada código de erro retornado é fornecido no item “Códigos de Erro”.

Consulte

XmodemRxFile.

Exemplo

```
#include <stdlib.h>
#include <stdio.h>
#include “xpcomlib.h”
#include “xpcomdef.h”

int ExibeMsg(unsigned int wNblocos, unsigned int wNtotal);
char cNomeArq[20];

void main(void)
        {
        int (*pfExibeMsg)(unsigned int, unsigned int);
        if (ComAddress(COM1, 0x238, 3))
                exit(1);
        if (ComOpen(COM1,9600,8,1,COM_NONEP,COM_NOFLOW,COM_NOFLOW,256,256, FALSE))
                exit(1);
        strcpy(cNomeArq, “C:\\TRIX\\RXARQ.BIN”);
        pfExibeMsg = ExibeMsg;
        if (XmodemTxFile(COM1,cNomeArq,pfExibeMsg))
                exit(1);
        if (ComClose(COM1))
                exit(1);
        }

int ExibeMsg(unsigned int wNblocos, unsigned int wNtotal)
        {
        printf(“\nNumero de blocos transmitidos: %03hu/%03hu”, wNblocos, wNtotal);
        return 0;
        }
```

Funções de Comunicação com a Rede de Rádio Freqüência

Estas funções são destinadas ao controle operacional da rede de rádio freqüência. Somente podem ser executadas após a inicialização da mesma.

As funções de controle operacional iniciam e executam rotinas de transmissão e recepção de dados entre o Host, a Base de RF e os Terminais conectados na rede. Elas terminam imediatamente após serem chamadas, sua execução prossegue, independentemente do programa aplicativo, por interrupção.

Também são fornecidas funções de verificação do estado da comunicação entre os dispositivos.

Funções para inicializar operacionalmente a rede

RFnetOpen
RFnetClose

Sempre que a rede for inicializada ou finalizada e reinicializada, todas as variáveis internas da biblioteca que indicam *status* de comunicação da rede também serão inicializadas.

Estas funções têm como função principal a definição da interface serial a ser utilizada, a taxa de comunicação em que a rede será operada, a definição de um *buffer* de transmissão no qual os dados transmitidos serão colocados para serem posteriormente retirados pela Base de RF e a definição de dois *bufferes* de recepção nos quais os dados recebidos da Base de RF serão colocados para posteriormente serem retirados pelo programa aplicativo. Um desses *bufferes* de recepção armazena mensagens de RF recebidas e eventos ocorridos na comunicação com os terminais. O Outro *buffer* de recepção armazena respostas a requisições feitas pelo Host para a Base de RF.

Verificar a comunicação entre o Host e a Base de RF. Chame as funções com número do terminal igual a zero.

RFnetTermInfoClear
RFnetTermInfo

Verificar a comunicação entre Base e Repetidores.

RFnetRepInfoClear
RFnetRepInfo

Verificar a comunicação entre Base e Terminais.

RFnetTermInfoClear
RFnetTermInfo

Configurar e verificar os diversos parâmetros envolvidos na comunicação automática entre o Host, a Base de RF, os Repetidores e os Terminais.

RFnetGetConfig
RFnetSetConfig
RFnetGetRadioConfig
RFnetSetRadioConfig
RFnetGetBaseVersion
A comunicação automática é inteiramente controlada pela Base de RF. Os dados recebidos são colocados em um dos dois *bufferes de* recepção do Host. Desta maneira o aplicativo somente precisa verificar se existem dados disponíveis no *buffer* de recepção, ou respostas a requisições no *buffer* de respostas.

Controlar a leitura do *buffer* de recepção de dados do concentrador, permitindo que o programa aplicativo tenha acesso às mensagens provenientes dos terminais.

RFnetRxMsg
RFnetRxMsgBytes
RFnetRxBufCount
RFnetRxBufBytes
RFnetRxBufFree
RFnetRxBufClear

Controlar a leitura do *buffer* de recepção de resultados do concentrador, permitindo que o programa aplicativo tenha acesso às respostas a requisições feitas pelo Host para a Base de RF.

RFnetWaitResult
RFnetGetResult

Controlar a escrita no *buffer* de transmissão de dados do concentrador, permitindo que o programa aplicativo envie mensagens aos terminais.

RFnetTxMsg
RFnetTxPing
RFnetTxBufCount
RFnetTxBufBytes
RFnetTxBufFree
RFnetTxBufClear

RFnetOpen

Sumário

```
int FAR PASCAL RFnetOpen (int nCanal,          // Numero do canal de comunicação
                          int nBaud,           // Taxa de comunicação
```

```
                    unsigned int wTxBuffer,     // Tamanho do buffer de transmissão
                    unsigned int wRxBuffer,     // Tamanho do buffer de recepção
                    int nMaxRetries             // Numero maximo de tentativas quando ocorrer erros
                    );
```

Windows 16	Windows 32	DOS
	•	

Descrição

A função 'RFnetOpen' inicializa a rede operacionalmente, permitindo que as funções da biblioteca possam ser executadas com sucesso.

O argumento 'nCanal' especifica o canal de comunicação, possibilita o uso de no máximo 16 canais (COM1 a COM16).

O argumento 'nBaud' define a taxa de comunicação a ser utilizada com 8 *bits*, 1 *stop-bit* e sem paridade, sendo os valores possíveis para o argumento mostrados abaixo em *bauds*:

1200 , 2400 , 4800 , 9600 , 19200, 38400.

O argumento 'wTxBuffer' define o espaço a ser alocado para o *buffer* de transmissão de mensagens ou comandos, podendo ser um valor de 16 *bytes* a 65500 *bytes.*

O argumento 'wRxBuffer' define o espaço a ser alocado para o *buffer* de recepção de mensagens, podendo ser um valor de 16 *bytes* a 65500 *bytes*.

O argumento ‘nMaxRetries’ informa o número máximo de tentativas de retransmissão para a base, quando ocorrer erros.

Ao executar esta função é alocada memória para o *buffer* de recepção de respostas a requisições que possui tamanho fixo, não podendo ser configurado pelo usuário. Ele armazena apenas a resposta à última requisição feita para a Base de RF.

Valor Retornado

A função retorna zero quando sua execução for bem sucedida, caso contrário retorna um código de erro sempre menor do que zero. O significado de cada código de erro retornado é fornecido no item “Códigos de Erro”.

Consulte

RFnetClose.

Exemplo

```
#include <windows.h>
#include "xpcom32.h"

int PASCAL WinMain(HANDLE hInstance, HANDLE hPrevInstance, LPSTR lpCmdLine, int nCmdShow)
        {
        if (RFnetOpen(COM1,9600,256,256,2))
                return FALSE;
        if (RFnetClose(COM1))
                return FALSE;
        return TRUE;
        }
```

RFnetClose

Sumário

```
int FAR PASCAL RFnetClose (int nCanal           // Numero do canal de comunicação
                    );
```

Windows 16	Windows 32	DOS
	•	

Descrição

A função 'RFnetClose' finaliza operacionalmente a rede previamente inicializada.

O argumento 'nCanal' especifica o canal de comunicação, possibilita o uso de no máximo 16 canais (COM1 a COM16).

Valor Retornado

A função retorna zero quando sua execução for bem sucedida, caso contrário retorna um código de erro sempre menor do que zero. O significado de cada código de erro retornado é fornecido no item “Códigos de Erro”.

Consulte

RFnetOpen.

Exemplo

Ver exemplo da função RFnetOpen.

RFnetTermInfoClear

Sumário

```
int FAR PASCAL RFnetTermInfoClear (int nCanal,          // Canal de comunicação
                                   int nTerm            // Terminal
                                   );
```

Windows 16	Windows 32	DOS
	•	

Descrição

A função 'RFnetTermInfoClear' inicializa a estrutura de estatísticas de comunicação do terminal armazenada na Base de RF, para o terminal especificado.

O argumento 'nCanal' especifica o canal de comunicação, possibilita o uso de no máximo 16 canais (COM1 a COM16).

O argumento 'nTerm' informa o número do terminal, que deve ser de 0 a 63. Se nTerm = 0, as estatísticas são as consolidadas pela Base de RF, para todos os terminais.

Valor Retornado

A função retorna zero quando sua execução for bem sucedida, caso contrário retorna um código de erro sempre menor do que zero. O significado de cada código de erro retornado é fornecido no item “Códigos de Erro”. Caso seja necessário verificar se a Base de RF executou corretamente a operação, deve-se executar a função “RFnetWaitResult” ou “RFnetGetResult”, conforme exemplo abaixo.

Consulte

RFnetTermInfo, RFnetWaitResult, RFnetGetResult.

Exemplo

```
#include <windows.h>
#include "xpcom32.h"

int PASCAL WinMain(HANDLE hInstance, HANDLE hPrevInstance, LPSTR lpCmdLine, int nCmdShow)
        {

        if (RFnetOpen(COM1,9600,256,256,2))
                return FALSE;
        if (RFnetTermInfoClear(COM1,1))
```

```
        return FALSE;
if (RFN_ITS != RFnetWaitResult(COM1))
        return FALSE;
if (RFnetClose(COM1))
        return FALSE;
return TRUE;
}
```

RFnetTermInfo

Sumário

```
int FAR PASCAL RFnetTermInfo (int nCanal,          // Canal de comunicação
                              int nTerm            // Terminal
                              );
```

Windows 16	Windows 32	DOS
	•	

```
struct RFNETSTAT
    {
    int         nTerm;                  // Numero do terminal
    unsigned long dwTxMsgs;             // Numero de mensagens de dados transmitidas com sucesso
    unsigned long dwTxBytes;            // Numero de bytes de dados transmitidos com sucesso
    unsigned long dwRetries;            // Numero de tentativas de retransmissão da base para o coletor
    unsigned long dwErrors;             // Numero de erros de transmissão ocorridos
    unsigned long dwRxMsgs;             // Numero de mensagens de dados recebidas do terminal
    unsigned long dwRxBytes;            // Numero de bytes de dados recebidos do terminal
    unsigned long dwRepeat;             // Estrutura de bits indicando os repetidores no alcance do terminal.
    unsigned long dwRfErrors;           // Numero de mensagens recebidas com erro pelo radio da base.
    unsigned long dwNetIDErrors;        // Numero de msgs recebidas com NetID errado pelo radio da base.
    unsigned long dwTxHostMsg;          // Numero de mensagens Transmitidas pela base ao Host.
    unsigned long dwRxHostMsg;          // Numero de mensagens Recebidas pela base vindas do Host.
    }
```

Descrição

A função 'RFnetTermInfo' obtém as estatísticas de comunicação com o terminal, armazenadas na Base de RF, para o terminal especificado. As estatísticas são depois lidas utilizando-se a função ‘RFnetGetResult’ onde se passa como parametro um ponteiro para uma estrutura do tipo RFNETSTAT descrita acima.

Nesta estrutura, o elemento dwRepeat só é válido para nTerm diferente de 0, enquanto os elementos dwRfErrors, dwNetIdErrors, dwTxHostMsg e dwRxHostMsg somente são válidos se nTerm=0, ou seja se forem Estatísticas consolidadas na Base.

O argumento 'nCanal' especifica o canal de comunicação, possibilita o uso de no máximo 16 canais (COM1 a COM16).

O argumento 'nTerm' informa o número do terminal, que deve ser de 0 a 63. Se nTerm = 0, são requisitadas as estatísticas consolidadas da Base de RF.

A Estrutura RFNETSTAT é definida no arquivo *header* 'XPCOMLIB.H'.

Valor Retornado

A função retorna zero quando sua execução for bem sucedida, caso contrário retorna um código de erro sempre menor do que zero. O significado de cada código de erro retornado é fornecido no item “Códigos de Erro”.

Consulte

RFnetTermInfoClear, RFnetWaitResult, RFnetGetResult.

Exemplo

```
#include <windows.h>
#include <stdlib.h>
#include <string.h>
#include "xpcom32.h"

RFNETSTAT Stat;
char cBuffer[3000];
char cBuf[10];

int PASCAL WinMain(HANDLE hInstance, HANDLE hPrevInstance, LPSTR lpCmdLine, int nCmdShow)
        {
        if (RFnetOpen(COM1,9600,256,256,2))
                return FALSE;
        if (RFnetTermInfo(COM1,1))
                return FALSE;
        if (RFnetWaitResult(COM1) != RFN_RTS)
                return FALSE;
        if (RFnetGetResult(COM1, &Stat) != RFN_RTS)
                return FALSE;

        strcpy(cBuffer, "\rdwTxMsgs...: ");
        strcat(cBuffer, ltoa(Stat.dwTxMsgs, cBuf, 10));
        strcat(cBuffer, "\rdwTxBytes..: ");
        strcat(cBuffer, ltoa(Stat.dwTxBytes, cBuf, 10));
        strcat(cBuffer, "\rdwRetries....: ");
        strcat(cBuffer, ltoa(Stat.dwRetries, cBuf, 10));
        strcat(cBuffer, "\rdwErrors.....: ");
        strcat(cBuffer, ltoa(Stat.dwErrors, cBuf, 10));
        strcat(cBuffer, "\rdwRxMsgs..: ");
        strcat(cBuffer, ltoa(Stat.dwRxMsgs, cBuf, 10));
        strcat(cBuffer, "\rdwRxBytes..: ");
        strcat(cBuffer, ltoa(Stat.dwRxBytes, cBuf, 10));
        strcat(cBuffer, "\rdwRepeat...: ");
        strcat(cBuffer, ltoa(Stat.dwRepeat, cBuf, 2));

        MessageBox((HWND)NULL, cBuffer, "Status do Terminal", MB_OK);
        if (RFnetClose(COM1))
                return FALSE;
        return TRUE;
        }
```

RFnetRepInfoClear

Sumário

```
int FAR PASCAL RFnetRepInfoClear (int nCanal,      // Canal de comunicação
                                  int nRep         // Repetidor
                                  );
```

Windows 16	Windows 32	DOS
	•	

Descrição

A função 'RFnetRepInfoClear' inicializa a estrutura de estatísticas de comunicação de repetidor armazenada na base RF, para o repetidor especificado.

O argumento 'nCanal' especifica o canal de comunicação, possibilita o uso de no máximo 16 canais (COM1 a COM16).

O argumento 'nRep' informa o número do repetidor, que deve ser de 0 a 31. Se nRep=0, as estatísticas são as consolidadas pela base, para todos os repetidores.

Valor Retornado

A função retorna zero quando sua execução for bem sucedida, caso contrário retorna um código de erro sempre menor do que zero. O significado de cada código de erro retornado é fornecido no item “Códigos de Erro”. Caso seja necessário verificar se a Base executou corretamente a operação, deve-se executar a função “RFnetWaitResult” ou “RFnetGetResult” , conforme exemplo abaixo.

Consulte

RFnetRepInfo, RFnetWaitResult, RFnetGetResult.

Exemplo

```
#include <windows.h>
#include "xpcom32.h"

int PASCAL WinMain(HANDLE hInstance, HANDLE hPrevInstance, LPSTR lpCmdLine, int nCmdShow)
        {

        if (RFnetOpen(COM1,9600,256,256,2))
                return FALSE;
        if (RFnetRepInfoClear(COM1,1))
                return FALSE;
        if (RFnetWaitResult(COM1) != RFN_IRS)
                return FALSE;
        if (RFnetClose(COM1))
                return FALSE;
        return TRUE;
        }
```

RFnetRepInfo

Sumário

```
int FAR PASCAL RFnetRepInfo (int nCanal,                    // Canal de comunicação
                             int nRep               // Repetidor
                             );
```

Windows 16	Windows 32	DOS
	•	

```
struct RFNETREPSTAT
        {
        int nRep;                       // Numero do repetidor
        unsigned long dwTxPolls;        // Numero de mensagens de requisição de dados transmitidos
        unsigned long dwTxMsgs;         // Numero de mensagens de dados transmitidas com sucesso
        unsigned long dwTxBytes;        // Numero de bytes de dados transmitidos com sucesso
        unsigned long dwRetries;        // Numero de tentativas de retransmissão da base para o repetidor
        unsigned long dwErrors;         // Numero de erros de transmissão ocorridos
        unsigned long dwRxMsgs;         // Numero de mensagens de dados recebidas pela base do repetidor
        unsigned long dwRxBytes;        // Numero de bytes de dados recebidos pela base do repetidor
        BOOL bStat;                     // Estado do repetidor: TRUE = ativo. FALSE = inativo / fora do ar.
        }
```

Descrição

A função 'RFnetRepInfo' obtém as estatísticas de comunicação com o repetidor armazenadas Base de RF, para o repetidor especificado. As estatísticas são depois lidas utilizando-se a função ‘RFnetGetResult’ onde se passa como parâmetro um ponteiro para uma estrutura do tipo RFNETREPSTAT descrita acima.

O argumento 'nCanal' especifica o canal de comunicação, possibilita o uso de no máximo 16 canais (COM1 a COM16).

O argumento 'nRep' informa o número do repetidor, que deve ser de 0 a 31. Se nRep =0, são requisitadas as estatísticas consolidadas dos repetidores na base.

A Estrutura RFNETREPSTAT é definida no arquivo *header* 'XPCOMLIB.H'.

Valor Retornado

A função retorna zero quando sua execução for bem sucedida, caso contrário retorna um código de erro sempre menor do que zero. O significado de cada código de erro retornado é fornecido no item “Códigos de Erro”.

Consulte

RFnetRepInfoClear, RFnetWaitResult, RFnetGetResult.

Exemplo

```
#include <windows.h>
#include <stdlib.h>
#include <string.h>
#include "xpcom32.h"

RFNETREPSTAT Stat;
char cBuffer[3000];
char cBuf[10];

int PASCAL WinMain(HANDLE hInstance, HANDLE hPrevInstance, LPSTR lpCmdLine, int nCmdShow)
	{
	if (RFnetOpen(COM1,9600,256,256,2))
		return FALSE;
	if (RFnetRepInfo(COM1,1))
		return FALSE;
	if (RFnetWaitResult(COM1) != RFN_RRS)
		return FALSE;
	if (RFnetGetResult(COM1, &Stat) != RFN_RRS)
		return FALSE;

	strcpy(cBuffer, "\rdwTxMsgs...: ");
	strcat(cBuffer, ltoa(Stat.dwTxMsgs, cBuf, 10));
	strcat(cBuffer, "\rdwTxBytes..: ");
	strcat(cBuffer, ltoa(Stat.dwTxBytes, cBuf, 10));
	strcat(cBuffer, "\rdwRetries....: ");
	strcat(cBuffer, ltoa(Stat.dwRetries, cBuf, 10));
	strcat(cBuffer, "\rdwErrors......: ");
	strcat(cBuffer, ltoa(Stat.dwErrors, cBuf, 10));
	strcat(cBuffer, "\rdwRxMsgs..: ");
	strcat(cBuffer, ltoa(Stat.dwRxMsgs, cBuf, 10));
	strcat(cBuffer, "\rdwRxBytes..: ");
	strcat(cBuffer, ltoa(Stat.dwRxBytes, cBuf, 10));
	strcat(cBuffer, "\rbStat...: ");
	if (Stat.bStat)
		strcat(cBuffer, " = TRUE");
	else
		strcat(cBuffer, " = FALSE");

	MessageBox((HWND)NULL, cBuffer, "Status do Repetidor", MB_OK);
	if (RFnetClose(COM1))
		return FALSE;
	return TRUE;
	}
```

RFnetGetConfig

Sumário

```
int FAR PASCAL RFnetGetConfig (int nCanal                    // Canal de comunicação
                               );
```

Windows 16	Windows 32	DOS

```
struct RFNETCONFIG{
   DWORD dwRate;              // valor da taxa de comunicação, na qual a rede está operando;
   DWORD dwBufRxMsg;          // tamanho do buffer de recepção de mensagens do Host;
   DWORD dwBufTxMsg;          // tamanho do buffer de transmissão de mensagens do Host;
   DWORD dwNetID;             // número de identificação desta  rede de Radio.
   DWORD dwRFRetries;         // numero de tentativas de retransmissão do rádio da Base
   DWORD dwRFTimeoutAck;      // intervalo de retransmissão do radio da Base, p/ mensagens
   DWORD dwRFTimeoutPong;     // intervalo de retransmissão do rádio da Base, para Ping.
   DWORD dwRadio0;            // status de funcionamento do Radio 0 da Base.
   DWORD dwRadio1;            // status de funcionamento do Radio 1 da base.
   DWORD dwHostMode;          // modo de comunicação com Host setado na Base.
   DWORD dwHostEcho;// se o canal de comunicação com Host está em modo ECHO
   DWORD dwHostProt;          // se o protocolo de comunicação é com correção de erros.
   DWORD dwHostRate;          // taxa de comunicação com Host setada na Base.
   DWORD dwHostTimeout;       // intervalo de tentativas de retransmissão da base para o Host.
   DWORD dwHostRetries;       // numero de tentativas de retransmissão da Base para o Host.
   DWORD dwNetRetries;        // número de tentativas de retransmissão da Base para Repetidores
   DWORD dwNetTimeout;        // intervalo de retransmissões de msgs da Base para Repetidores.
   DWORD dwNetRate;           // taxa de comunicação entre Base e Repetidores.
   DWORD dwRepeat;            // quais repetidores estão ativos para transmissão de RF (estrut.de bits)
   DWORD dwDebugFlags;        // estado dos flags de debug na Base (estrutura de bits)
   DWORD  dwSwitches;         // estado da dip-switch e jumpers internos.
   };
```

Descrição

A função 'RFnetGetConfig' é utilizada para se obter informações diversas sobre a configuração da rede de RF, tanto armazenadas na Base quanto armazenadas no próprio Host. As configurações são depois lidas utilizando-se a função ‘RFnetGetResult’onde se passa como parâmetro um ponteiro para uma estrutura do tipo RFNETCONFIG descrita acima.

O argumento 'nCanal' especifica o canal de comunicação, possibilita o uso de no máximo 16 canais (COM1 a COM16).

A Estrutura RFNETCONFIG é definida no arquivo *header* 'XPCOMLIB.H'.

Valor Retornado

A função retorna zero quando sua execução for bem sucedida, caso contrário retorna um código de erro sempre menor do que zero. O significado de cada código de erro retornado é fornecido no item “Códigos de Erro”.

Consulte

RFnetSetConfig, RFnetWaitResult, RFnetGetResult.

Exemplo

```
#include <windows.h>
#include <stdlib.h>
#include "xpcom32.h"

RFNETCONFIG Cfg;
char cBuffer[10];

int PASCAL WinMain(HANDLE hInstance, HANDLE hPrevInstance, LPSTR lpCmdLine, int nCmdShow)
        {
        if (RFnetOpen(COM1,9600,256,256,2))
                return FALSE;
        if (RFnetGetConfig(COM1))
                return FALSE;
        if (RFnetWaitResult(COM1) != RFN_VP)
                return FALSE;
        if (RFnetGetResult(COM1, &Cfg) != RFN_VP)
                return FALSE;

        MessageBox((HWND)NULL, ltoa(Cfg.dwRFTimeoutAck, cBuffer, 10), "Timeout de Ack (x 10ms)", MB_OK);
        if (RFnetClose(COM1))
```

```
        return FALSE;
return TRUE;
}
```

RFnetSetConfig

Sumário

```
int FAR PASCAL RFnetSetConfig (int nCanal,              // Canal de comunicação
                               int nParam,              // Tipo de parêmetro que se quer configurar
                               unsigned long lValue     // valor
                               );
```

Windows 16	Windows 32	DOS
	•	

Descrição

A função 'RFnetSetConfig' é utilizada para se modificar o valor dos diversos parâmetros de configuração da rede de RF.

O argumento 'nCanal' especifica o canal de comunicação, possibilita o uso de no máximo 16 canais (COM1 a COM16).

O argumento 'nParam' determina qual parâmetro de configuração que se quer alterar. Os parâmetros possíveis são os definidos pelas constantes abaixo, que já estão declaradas no arquivo *header* 'XPCOMDEF.H' da biblioteca XPComLib:

RFNCFG_NETID	Configura número de identificação desta rede de rádio (0 a 255).
RFNCFG_RFRETRIES	Configura número de tentativas de retransmissão do rádio da Base (0 a 255)
RFNCFG_RFTIMEOUTACK	Configura tempo para retransmissão de mensagens do rádio da Base (0 a 255)
RFNCFG_RFTIMEOUTPONG	Configura tempo de espera de Pong do rádio da Base.
RFNCFG_RADIO0	Ativa/Desativa o rádio 0 da Base.
RFNCFG_RADIO1	Ativa/Desativa o rádio 1 da base.
RFNCFG_HOSTMODE	Ativa/desativa modo de comunicação com Host, na Base.
RFNCFG_HOSTECHO	Ativa/desativa modo ECHO da Base.
RFNCFG_HOSTPROT	Ativa/desativa o protocolo de comunicação na base com correção de erros.
RFNCFG_HOSTRATE	Configura a taxa de comunicação com Host setada na Base.
RFNCFG_HOSTTIMEOUT	Configura intervalo de tentativas de retransmissão da base para o Host.
RFNCFG_HOSTRETRIES	Configura número de tentativas de retransmissão da Base para o Host.
RFNCFG_NETRETRIES	Configura número de tentativas de retransmissão da Base para os Repetidores.
RFNCFG_NETTIMEOUT	Configura o intervalo de retransmissões de mensagens da Base para os Repetidores.
RFNCFG_NETRATE	Configura a taxa de comunicação entre a Base e os Repetidores.
RFNCFG_REPEAT	Ativa/desativa repetidores para conversação com a Base (estrutura de bits)
RFNCFG_DEBUGFLAGS	Configura estado dos flags de debug na Base (estrutura de bits)
RFNCFG_SWITCHES:	Usado para ler dip-switches da Base de RF

Valor Retornado

A função retorna zero quando sua execução for bem sucedida, caso contrário retorna um código de erro sempre menor do que zero. O significado de cada código de erro retornado é fornecido no item "Códigos de Erro". Caso seja necessário verificar se a Base executou corretamente a operação, deve-se executar a função "RFnetWaitResult" ou "RFnetGetResult" , conforme exemplo abaixo.

Consulte

RFnetWaitResult, RFnetGetResult, RFnetGetConfig.

Exemplo

```
#include <windows.h>
#include <stdlib.h>
#include "xpcom32.h"

int PASCAL WinMain(HANDLE hInstance, HANDLE hPrevInstance, LPSTR lpCmdLine, int nCmdShow)
        {
        if (RFnetOpen(COM1,9600,256,256,2))
                return FALSE;
```

```
if (RFnetSetConfig(COM1, RFNCFG_NETRATE, 19200))
        return FALSE;
if (RFnetWaitResult(COM1) != RFN_CP)
        return FALSE;

if (RFnetClose(COM1))
        return FALSE;
return TRUE;
}
```

RFnetGetRadioConfig

Sumário

```
int FAR PASCAL RFnetGetRadioConfig (int nCanal,          // Canal de comunicação
                                    int nRadio           // número do radio
                                   );
```

Windows 16	Windows 32	DOS
	•	

```
struct RFNETRADIOCONFIG{
   int nRadio;                  // se o canal de comunicação está aberto.
   int nPreamble;               // ciclos de preambulo.
   int nWakeup;                 // numero de preambulos em modo power-save.
   int nSleepTime;              // tempo desligado no modo power save.
   int nTxRx;                   // tempo de troca TX-RX.
   int nPwr;                    // tempo de estabilizacao apos power on.
   int nTxBackoff;              // Tx Back-Off maximo.
   int nTxSlot;                 // Numero de slot.
   int nResetState;             // Estado apos reset.
   };
```

Descrição

A função 'RFnetGetRadioConfig' é utilizada para se obter informações sobre a configuração dos módulos de rádio da Base. As configurações são depois lidas utilizando-se a função ‘RFnetGetResult’onde se passa como parâmetro um ponteiro para uma estrutura do tipo RFNETRADIOCONFIG descrita acima.

O argumento 'nCanal' especifica o canal de comunicação, possibilita o uso de no máximo 16 canais (COM1 a COM16).

O argumento ‘nRadio’especifica o número do rádio, de 0 a 1.

A Estrutura RFNETRADIOCONFIG é definida no arquivo *header* 'XPCOMLIB.H'.

Valor Retornado

A função retorna zero quando sua execução for bem sucedida, caso contrário retorna um código de erro sempre menor do que zero. O significado de cada código de erro retornado é fornecido no item “Códigos de Erro”.

Consulte

RFnetSetRadioConfig, RFnetWaitResult, RFnetGetResult.

Exemplo

```
#include <windows.h>
#include <stdlib.h>
#include "xpcom32.h"

RFNETRADIOCONFIG Cfg;
char cBuffer[10];

int PASCAL WinMain(HANDLE hInstance, HANDLE hPrevInstance, LPSTR lpCmdLine, int nCmdShow)
```

```
{
if (RFnetOpen(COM1,9600,256,256,2))
        return FALSE;
if (RFnetGetRadioConfig(COM1,0))
        return FALSE;
if (RFnetWaitResult(COM1) != RFN_VR)
        return FALSE;
if (RFnetGetResult(COM1, &Cfg) != RFN_VR)
        return FALSE;

MessageBox((HWND)NULL, ltoa(Cfg.nPreamble, cBuffer, 10), "Ciclos de preambulo do Radio 0", MB_OK);
if (RFnetClose(COM1))
        return FALSE;
return TRUE;
}
```

RFnetSetRadioConfig

Sumário

```
int FAR PASCAL RFnetSetRadioConfig (int nCanal,        // Canal de comunicação
                                    int nRadio,        // Numero do radio
                                    int nParam,        // Tipo de parêmetro que se quer configurar
                                    int nValue         // valor
                                     );
```

Windows 16	Windows 32	DOS
	•	

Descrição

A função 'RFnetSetRadioConfig' é utilizada para se modificar o valor dos diversos parâmetros de configuração de um rádio da Base.

O argumento 'nCanal' especifica o canal de comunicação, possibilita o uso de no máximo 16 canais (COM1 a COM16).

O argumento ‘nRadio’especifica o número do rádio, de 0 a 1.

O argumento 'nParam' determina qual parâmetro de configuração que se quer alterar. Os parâmetros possíveis são os definidos pelas constantes abaixo, que já estão declaradas no arquivo *header* ‘XPCOMDEF.H’ da biblioteca XPComLib:

RFNRAD_PREAMBLE	Configura nPreamble, ciclos de preâmbulo.
RFNRAD_WAKEUP	Configura nWakeup, número de preâmbulos em modo power save.
RFNRAD_SLEEP	Configura nSleepTime, tempo desligado no modo power save do rádio.
RFNRAD_TXRX	Configura nTxRx, tempo de trtoca entre Tx e Rx do rádio
RFNRAD_PWR	Configura nPwr, tempo de estabilização após power up do rádio
RFNRAD_TXBACKOFF	Configura nTxBackoff, tempo máximo de Tx Back-Off do rádio
RFNRAD_TXSLOT	Configura nTxSlot, número de slot do rádio
RFNRAD_RESSTATE	ConfiguranResetState, estado do switches register do rádio após reset
RFNRAD_DIAG	Configura nDiag;

Importante: Esses parametros somente devem ser alterados por pessoal habilitado.

Valor Retornado

A função retorna zero quando sua execução for bem sucedida, caso contrário retorna um código de erro sempre menor do que zero. O significado de cada código de erro retornado é fornecido no item “Códigos de Erro”. Caso seja necessário verificar se a Base executou corretamente a operação, deve-se executar a função “RFnetWaitResult” ou “RFnetGetResult” , conforme exemplo abaixo.

Consulte

RFnetGetRadioConfig, RFnetWaitResult, RFnetGetResult.

Exemplo

```
#include <windows.h>
#include <stdlib.h>
#include "xpcom32.h"

int PASCAL WinMain(HANDLE hInstance, HANDLE hPrevInstance, LPSTR lpCmdLine, int nCmdShow)
        {
        if (RFnetOpen(COM1,9600,256,256,2))
                return FALSE;
        if (RFnetSetRadioConfig(COM1,0, RFNRAD_TXSLOT, 4))
                return FALSE;
        if (RFnetWaitResult(COM1) != RFN_CR)
                return FALSE;

        if (RFnetClose(COM1))
                return FALSE;
        return TRUE;
        }
```

RFnetGetBaseVersion

Sumário

```
int FAR PASCAL RFnetGetBaseVersion(int nCanal                   // Canal de comunicação
                                   );
```

Windows 16	Windows 32	DOS
	•	

Descrição

A função 'RFnetGetBaseVersion' é utilizada para se obter a versão de hardware e de firmware da Base de RF. As configurações são depois lidas utilizando-se a função ‘RFnetGetResult’onde se passa como parâmetro um ponteiro para um string. O string vem no formato: “X.XX,Y.YY” , onde:

X.XX = versão de Hardware
Y.YY = versão de Firmware

O argumento 'nCanal' especifica o canal de comunicação, possibilita o uso de no máximo 16 canais (COM1 a COM16).

Valor Retornado

A função retorna zero quando sua execução for bem sucedida, caso contrário retorna um código de erro sempre menor do que zero. O significado de cada código de erro retornado é fornecido no item “Códigos de Erro”.

Consulte

RFnetWaitResult, RFnetGetResult.

Exemplo

```
#include <windows.h>
#include <stdlib.h>
#include "xpcom32.h"

char cBuffer[256];
char cBuf[20];

int PASCAL WinMain(HANDLE hInstance, HANDLE hPrevInstance, LPSTR lpCmdLine, int nCmdShow)
        {
        if (RFnetOpen(COM1,9600,256,256,2))
                return FALSE;
        if (RFnetGetBaseVersion(COM1))
                return FALSE;
        if (RFnetWaitResult(COM1) != RFN_RV)
                return FALSE;
```

```
if (RFnetGetResult(COM1, &cBuf) != RFN_RV)
        return FALSE;

strcpy(cBuffer,"Versão de Hardware: ");
strncat(cBuffer, & cBuf[0],4);
strcat(cBuffer, "\rVersão de Firmware: ");
strncat(cBuffer, & cBuf[5],4);

MessageBox((HWND)NULL, cBuffer, "Versão de Firmware da Base",MB_OK);
if (RFnetClose(COM1))
        return FALSE;
return TRUE;
}
```

RFnetTxMsg

Sumário

```
int FAR PASCAL RFnetTxMsg(int nCanal,                   // Canal de comunicação
                          int nTerm,                    // Terminal
                          const char FAR *pbMsg,        // Ponteiro para mensagem
                          unsigned int wTamMsg          // Tamanho da mensagem
                          );
```

Windows 16	Windows 32	DOS
	•	

```
struct RFNETMSGRES
        {
        int     nTerm;          // numero do terminal terminal para o qual a mensagem sera enviada
        BYTE    cMsg;           // numero da mensagem anviada
        };
```

Descrição

A função 'RFnetTxMsg' transmite uma mensagem, conforme o destino indicado pelos argumentos 'nCanal' e 'nTerm'.

O argumento 'nCanal' especifica o canal de comunicação, possibilita o uso de no máximo 16 canais (COM1 a COM16).

O argumento 'nTerm' informa o número do terminal, que deve ser de 1 a 63.

O argumentos 'pbMsg' e 'wTamMsg' são, respectivamente, o ponteiro para a mensagem a ser transmitida e o seu tamanho.

Como a transmissão é feita pela Base, através do repetidor que ela mesma escolheu, esta função apenas coloca a mensagem no *buffer* de transmissão e retorna. Caso se queira saber se o terminal recebeu a mensagem corretamente, é necessário fazer dois procedimentos em seguida: O primeiro procedimento é esperar que a Base responda que recebeu a mensagem do Host, e pegar o número que a Base associou a esta mensagem. Isso é feito com a função RFnetGetResult, que retorna a resposta em uma estrutura do tipo RFNETMSGRES descrita acima. O segundo procedimento é esperar uma mensagem, com a função RFnetRxMsg, indicativa de mensagem transmitida com sucesso ou erro de transmissão de mensagem, para o mesmo nTerm e numero de mensagem. Vide descrição da função RFnetGetResult.

A Estrutura RFNETMSGRES é definida no arquivo *header* 'XPCOMLIB.H'.

Valor Retornado

A função retorna zero quando sua execução for bem sucedida, caso contrário retorna um código de erro sempre menor do que zero. O significado de cada código de erro retornado é fornecido no item “Códigos de Erro”.

Consulte

RFnetWaitResult, RFnetGetResult, RFnetRxMsg, RFnetRxMsgBytes, RFnetTxBufCount, RFnetTxBufBytes, RFnetTxBufFree, RFnetTxBufClear.

Exemplo

```
#include <windows.h>
#include <string.h>
#include <stdlib.h>
#include "xpcom32.h"

char sMsg[10];
RFNETMSGRES sBuf;
RFNETMESSAGE Resp;
int nCnt;

int PASCAL WinMain(HANDLE hInstance, HANDLE hPrevInstance, LPSTR lpCmdLine, int nCmdShow)
        {
        if (RFnetOpen(COM1,9600,256,256,2))
                return FALSE;
        strcpy(sMsg, "Teste");
        if (RFnetTxMsg(COM1,1,sMsg,(WORD)strlen(sMsg)))          // transmite mensagem para Base
                return FALSE;
        if (RFnetWaitResult(COM1) != RFN_TM)                     // espera resposta da transmissão para Base
                return FALSE;
        if (RFnetGetResult(COM1, &sBuf) != RFN_TM)                       // lê o numero da msg que a Base
associou
                return FALSE;

        nCnt =0;
        while (TRUE)                                             // espera Base avisar sucesso de transmissão
                {
                if (RFN_RXBEMPTY != RFnetRxMsg(COM1, &Resp))     // se recebeu mensagem da Base
                        {
                        // se mensagem enviada com sucesso ou erro de transmissao:
                        if (RFN_MT == Resp.nType || RFN_ETM == Resp.nType)
                                {
                                if (sBuf.nTerm == Resp.nTerm  &&  // despreza mensagens de outro terminal
                                        sBuf.cMsg  == Resp.nMsg)  // despreza mensagens com numero diferente
                                        break;                    // veio resposta esperada
                                }
                        }
                        Sleep(100);                              // espera 100 ms
                        if (++nCnt == 20)                        //se passou 2 segundos, sai com erro
                                return FALSE;
                }

        if (RFN_MT != Resp.nType)                                // se erro na transmissao da mensagem:
                return FALSE;

        if (RFnetClose(COM1))
                return FALSE;
        return TRUE;
        }
```

RFnetTxPing

Sumário

```
int FAR PASCAL RFnetTxPing(int nCanal,                  // Canal de comunicação
                           int nTerm,                   // Terminal
                           const char FAR *pbMsg,       // Ponteiro para mensagem
                           unsigned int wTamMsg         // Tamanho da mensagem
                           );
```

Windows 16	Windows 32	DOS
	•	

Descrição

A função 'RFnetTxPing' transmite uma mensagem de ping, conforme o destino indicado pelos argumentos 'nCanal' e 'nTerm'.

O argumento 'nCanal' especifica o canal de comunicação, possibilita o uso de no máximo 16 canais (COM1 a COM16).

O argumento 'nTerm' informa o número do terminal, que deve ser de 1 a 63.

O argumentos 'pbMsg' e 'wTamMsg' são, respectivamente, o ponteiro para mensagem a ser transmitida e seu tamanho.

Como a transmissão é feita pela Base, através do repetidor que ela mesma escolheu, esta função apenas coloca a mensagem no *buffer* de transmissão e retorna. Caso se queira saber se o terminal recebeu a mensagem corretamente, é necessário fazer dois procedimentos em seguida: O primeiro procedimento é esperar que a Base responda que recebeu a mensagem do Host, e pegar o número que a Base associou a esta mensagem. Isto é feito com a função RFnetGetResult, que passa a resposta em uma estrutura do tipo RFNETMSGRES conforme descrito na função RFneTxMsg. O segundo procedimento é esperar a mensagem, com a função RFnetRxMsg, indicativa de recebimento de Pong ou erro de transmissão de mensagem, para o mesmo nTerm e número de mensagem. Vide descrição da função RFnetGetResult.

A Estrutura RFNETCONFIG é definida no arquivo *header* 'XPCOMLIB.H'.

Valor Retornado

A função retorna zero quando sua execução for bem sucedida, caso contrário retorna um código de erro sempre menor do que zero. O significado de cada código de erro retornado é fornecido no item “Códigos de Erro”.

Consulte

RFnetWaitResult, RFnetGetResult, RFnetRxMsg, RFnetTxBufCount, RFnetTxBufBytes, RFnetTxBufFree, RFnetTxBufClear.

Exemplo

```
#include <windows.h>
#include <string.h>
#include <stdlib.h>
#include "xpcom32.h"

char sMsg[10];
RFNETMSGRES sBuf;
RFNETMESSAGE Resp;
int i;
int nRxTerm;
BYTE cRxMsg;

int PASCAL WinMain(HANDLE hInstance, HANDLE hPrevInstance, LPSTR lpCmdLine, int nCmdShow)
        {
        if (RFnetOpen(COM1,9600,256,256,2))
                return FALSE;
        strcpy(sMsg, "Teste");
        if (RFnetTxPing(COM1, 1, sMsg, (WORD)strlen(sMsg)))          // transmite mensagem para Base
                return FALSE;
        if (RFnetWaitResult(COM1) != RFN_TP)                         // espera resposta da transmissão do Ping
                return FALSE;

        if (RFnetGetResult(COM1, &sBuf) != RFN_TP)                   // lê numero da mensagem que a Base associou
                return FALSE;

        i =0;
        while (TRUE)                                                 // espera Base avisar sucesso de transmissão
                {
                if (RFnetRxMsg(COM1, &Resp))                         // se recebeu mensagem da Base
                        {
                        // se Ping foi respondido ou houve erro na resposta do Ping:
                        if (RFN_PR == Resp.nType || RFN_ETP == Resp.nType)
                                {
                                if (sBuf.nTerm == Resp.nTerm  &&     // despreza mensagens de outro terminal
                                sBuf.cMsg  == Resp.nMsg)             // despreza mensagens com numero diferente
                                break;                               // veio resposta esperada
                                }
```

```
                }
            Sleep(100);                                     // espera 100 ms
            if (++i == 20)                                  //se passou 2 segundos, sai com erro
                return FALSE;
            }

        if (RFN_PR != Resp.nType)                           // se Ping nao foi respondido:
            return FALSE;

        if (RFnetClose(COM1))
            return FALSE;
        return TRUE;
        }
```

RFnetRxMsg

Sumário

```
int FAR PASCAL RFnetRxMsg (int nCanal,                  // Canal de comunicação
                           RFNETMESSAGE  *pbDest        // Ponteiro para area de retorno da mensagem
                           );
```

Windows 16	Windows 32	DOS
	•	

```
struct RFNETMESSAGE
    {
    int     nType;      // tipo de msg: RFN_ERRMSG, RFN_RM,RFN_MT,RFN_ETM,RFN_PR,RFN_ETP,
                        // RFN_EV.
    int     nTerm;      // numero do terminal ou do repetidor.
    int     nMsg;       // numero da mensagem.
    int     nStatus;    // estado do repetidor: 1= inacessível, 2 = acessível.
    int     nTimeout;   // tempo de timeout, quando erro de Tx. Tempo de resposta ao Ping (Pong).
    int     nBytes;     // tamanho da mensagem recebida em sMsg.
    BYTE    sMsg[129];  // mensagem recebida.
    };
```

Descrição

A função 'RFnetRxMsg' retira do *buffer* de recepção do PC uma mensagem, e a coloca a partir da área de retorno apontada pelo argumento 'pbDest'. A mensagem retornada vem no formato da estrutura de dados RFNETMESSAGE descrita acima, onde os campos retornam valores ou não, dependendo do tipo de mensagem.

O argumento 'nCanal' especifica o canal de comunicação, possibilita o uso de no máximo 16 canais (COM1 a COM16).

O argumento '*pbDest' aponta para a área de retorno onde a mensagem será armazenada.

Os possíveis tipos de mensagem estão na tabela abaixo:

Tipo da mensagem recebida:	Campos da estrutura retornados:
Chegou mensagem com formatação errada	RFN_ERRMSG
Mensagem recebida	RFN_RM, nTerm, nMsg, nBytes, sMsg
Mensagem transmitida com sucesso	RFN_MT, nTerm, nMsg
Erro na transmissao de mensagem.	RFN_ETM, nTerm, nMsg
Mensagem de Pong recebida.	RFN_PR, nTerm, nMsg, nTimeout
Erro na transmissao de Ping.	RFN_ETP, nTerm, nMsg, nTimeout
Chegou mensagem de evento ocorrido	RFN_EV, nTerm=repetidor, nStatus (1=inacessível, 2=acessível)

Valor Retornado

A função retorna o código com o tipo de mensagem recebida. Caso contrário retorna um código de erro sempre menor do que zero. O significado de cada código de erro retornado é fornecido no item "Códigos de Erro". Retorna o erro RFN_RXBEMPTY quando o buffer de recepção estiver vazio.

Consulte

RFnetRxMsgBytes, RFnetRxBufCount, RFnetRxBufBytes, RFnetRxBufFree, RFnetRxBufClear.

Exemplo

```
return TRUE;
}
```

RFnetGetResult

Sumário

```
int FAR PASCAL RFnetGetResult (int nCanal,          // Canal de comunicação
                               void FAR *pbDest     // ponteiro para area de retorno da resposta
                               );
```

Windows 16	Windows 32	DOS
	•	

Descrição

A função 'RFnetGetResult' retira do *buffer* de recepção de resultados do PC a ultima resposta dada pela Base a um comando enviado, e a coloca a partir da área de retorno apontada pelo argumento 'pbDest'. A mensagem retornada vem no formato da estrutura de dados compatível com o comando enviado. A cada envio de comando, o *buffer* de recepção é esvaziado e possíveis respostas da Base que sejam incompatíveis com o comando pedido são descartadas.

O argumento 'nCanal' especifica o canal de comunicação, possibilita o uso de no máximo 16 canais (COM1 a COM16).

O argumento '*pbDest' aponta para a área de retorno onde a mensagem será armazenada.

Valor Retornado

A função retorna o código do comando executado, sempre maior que zero se sua execução for bem sucedida, senão retorna um código de erro sempre menor do que zero. O significado de cada código de erro retornado é fornecido no item "Códigos de Erro".

Os tipos de resultado possíveis estão definidos na tabela abaixo:

Função executada	Constante	Descrição	Tipo de estrutura apontada por pbDest
	RFN_ACTIVE	Erro: ainda esperando resposta.	
	RFN_RESULTERR	Retornou erro de comando ou parâmetro inválido.	
	RFN_RESULTIDLE	Não tem pedido pendente.	
RFnetGetConfig	RFN_VP	Retornou visualização de parâmetro da Base de RF.	RFNETCONFIG
RFnetGetRadioConfig	RFN_VR	Retornou visualizalização de parâmetro do rádio.	RFNETRADIOCONFIG
RFnetTermInfo	RFN_RTS	Retornou Estatística do coletor.	RFNETSTAT
RFnetRepInfo	RFN_RRS	Retornou Estatística do repetidor.	RFNETREPSTAT
RFnetTxMsg	RFN_TM	Retornou mensagem aceita.	RFNETMSGRES
RFnetTxPing	RFN_TP	Retornou Ping aceito.	RFNETMSGRES
RFnetTermInfoClear	RFN_ITS	Retornou reset do status do coletor OK.	NULL(não retorna dados na estrutura)
RFnetGetBaseVersion	RFN_RV	Retornou número da versão.	string: "X.XX,Y.YY"
RFnetRepInfoClear	RFN_IRS	Retornou reset do status do repetidor OK.	NULL
RFnetSetConfig	RFN_CP	Retornou alteração de parâmetro de configuração OK.	NULL
RFnetSetRadioConfig	RFN_CR	Retornou configuração de parâmetros do rádio OK.	NULL

Consulte

RFnetWaitResult

Exemplo

Vide exemplo específico de cada comando na descrição do mesmo.

RFnetWaitResult

Sumário

```
int FAR PASCAL RFnetWaitResult (int nCanal          // Canal de comunicação
```

);

Windows 16	Windows 32	DOS
	•	

Descrição

A função 'RFnetWaitResult' espera até que esteja disponível no *buffer* de resultados a resposta da Base a comando solicitado. A operação da função chamadora é suspensa até vir resposta ou até terminar o tempo pré-estabelecido de timeout.

O argumento 'nCanal' especifica o canal de comunicação, possibilita o uso de no máximo 16 canais (COM1 a COM16).

Valor Retornado

A função retorna o tipo de resultado, sempre maior que zero, quando sua execução for bem sucedida ou caso contrário retorna um código de erro sempre menor do que zero. O significado de cada código de erro retornado é fornecido no item "Códigos de Erro". Os Valores do tipo de resultado retornado são os listados em RFnetGetResult. Retorna RFN_RESULTIDLE se não tem pedido de resultado pendente.

Ver os tipos de resultados possíveis na função RFnetGetResult.

Consulte

RFnetGetResult.

Exemplo

Vide exemplo específico de cada função, na descrição da mesma.

RFnetTxBufCount

Sumário

int FAR PASCAL RFnetTxBufCount (int nCanal // Canal de comunicação
);

Windows 16	Windows 32	DOS
	•	

Descrição

A função 'RFnetTxBufCount', retorna o número de mensagens a transmitir na área reservada para o *buffer* de transmissão.

O argumento 'nCanal' especifica o canal de comunicação, possibilita o uso de no máximo 16 canais (COM1 a COM16).

Valor Retornado

A função retorna valor positivo (número de mensagens) ou igual a zero (*buffer* vazio) quando sua execução for bem sucedida, caso contrário retorna um código de erro sempre menor do que zero. O significado de cada código de erro retornado é fornecido no item "Códigos de Erro".

Consulte

RFnetTxMsg, RFnetTxBufBytes, RFnetTxBufFree, RFnetTxBufClear.

Exemplo

```
#include <windows.h>
#include <stdlib.h>
#include "xpcom32.h"

int nCount;
```

```
char cBuffer[10];

int PASCAL WinMain(HANDLE hInstance, HANDLE hPrevInstance, LPSTR lpCmdLine, int nCmdShow)
        {
        if (RFnetOpen(COM1,9600,256,256,2))
                return FALSE;
        nCount = RFnetTxBufCount(COM1);
        if (nCount < 0)
                return FALSE;
        MessageBox((HWND)NULL, itoa(nCount, cBuffer, 10), "Mensagens a transmitir", MB_OK);
        if (RFnetClose(COM1))
                return FALSE;
        return TRUE;
        }
```

RFnetTxBufBytes

Sumário

```
int FAR PASCAL RFnetTxBufBytes (int nCanal                    // Canal de comunicação
                                );
```

Windows 16	Windows 32	DOS
	•	

Descrição

A função 'RFnetTxBufBytes', retorna o número de *bytes* utilizados na área reservada para o *buffer* de transmissão.

O argumento 'nCanal' especifica o canal de comunicação, possibilita o uso de no máximo 16 canais (COM1 a COM16).

Valor Retornado

A função retorna valor positivo (*bytes* utilizados) ou igual a zero (*buffer* vazio) quando sua execução for bem sucedida, caso contrário retorna um código de erro sempre menor do que zero. O significado de cada código de erro retornado é fornecido no item “Códigos de Erro”.

Consulte

RFnetTxMsg, RFnetTxBufCount, RFnetTxBufClear.

Exemplo

```
#include <windows.h>
#include <stdlib.h>
#include "xpcom32.h"

int nBytes;
char cBuffer[10];

int PASCAL WinMain(HANDLE hInstance, HANDLE hPrevInstance, LPSTR lpCmdLine, int nCmdShow)
        {
        if (RFnetOpen(COM1,9600,256,256,2))
                return FALSE;
        nBytes = RFnetTxBufBytes(COM1);
        if (nBytes < 0)
                return FALSE;
        MessageBox((HWND)NULL, itoa(nBytes, cBuffer, 10), "Area ocupada", MB_OK);
        if (RFnetClose(COM1))
                return FALSE;
        return TRUE;
        }
```

RFnetTxBufFree

Sumário

```
int FAR PASCAL RFnetTxBufFree (int nCanal                    // Canal de comunicação
                               );
```

Windows 16	Windows 32	DOS
	•	

Descrição

A função 'RFnetTxBufFree', retorna o espaço (em *bytes*) disponível na área reservada para o *buffer* de transmissão.

O argumento 'nCanal' especifica o canal de comunicação, possibilita o uso de no máximo 16 canais (COM1 a COM16).

Valor Retornado

A função retorna valor positivo (*bytes* disponíveis) ou igual a zero (*buffer* cheio) quando sua execução for bem sucedida, caso contrário retorna um código de erro sempre menor do que zero. O significado de cada código de erro retornado é fornecido no item “Códigos de Erro”.

Consulte

RFnetTxMsg, RFnetTxBufCount, RFnetTxBufBytes, RFnetTxBufClear.

Exemplo

```
#include <windows.h>
#include <stdlib.h>
#include "xpcom32.h"

char cBuffer[10];
int  nBytes;

int PASCAL WinMain(HANDLE hInstance, HANDLE hPrevInstance, LPSTR lpCmdLine, int nCmdShow)
        {
        if (RFnetOpen(COM1,9600,256,256,2))
                return FALSE;
        nBytes = RFnetTxBufFree(COM1);
        if (nBytes < 0)
                return FALSE;
        MessageBox((HWND)NULL, itoa(nBytes, cBuffer, 10), "Area disponivel", MB_OK);
        if (RFnetClose(COM1))
                return FALSE;
        return TRUE;
        }
```

RFnetTxBufClear

Sumário

```
int FAR PASCAL RFnetTxBufClear (int nCanal                   // Canal de comunicação
                                );
```

Windows 16	Windows 32	DOS
	•	

Descrição

A função 'RFnetTxBufClear' limpa toda a área reservada para o *buffer* de transmissão, descartando todas as mensagens anteriormente armazenadas.

O argumento 'nCanal' especifica o canal de comunicação, possibilita o uso de no máximo 16 canais (COM1 a COM16).

Valor Retornado

A função retorna zero quando sua execução for bem sucedida, caso contrário retorna um código de erro sempre menor do que zero. O significado de cada código de erro retornado é fornecido no item “Códigos de Erro”.

Consulte

RFnetTxMsg, RFnetTxBufCount, RFnetTxBufBytes, RFnetTxBufFree.

Exemplo

```
#include <windows.h>
#include "xpcom32.h"

int PASCAL WinMain(HANDLE hInstance, HANDLE hPrevInstance, LPSTR lpCmdLine, int nCmdShow)
        {
        if (RFnetOpen(COM1,9600,256,256,2))
                return FALSE;
        if (RFnetTxBufClear(COM1))
                return FALSE;
        if (RFnetClose(COM1))
                return FALSE;
        return TRUE;
        }
```

RFnetRxMsgBytes

Sumário

```
int FAR PASCAL RFnetRxMsgBytes(int nCanal                    // Canal de comunicação
                               );
```

Windows 16	Windows 32	DOS
	•	

Descrição

A função 'RFnetRxMsgBytes' informa o número de *bytes* da primeira mensagem que está armazenada e ainda não foi retirada da área reservada para o *buffer* de recepção.

O argumento 'nCanal' especifica o canal de comunicação, possibilita o uso de no máximo 16 canais (COM1 a COM16).

Valor Retornado

A função retorna valor positivo ou igual a zero (*buffer* vazio) quando sua execução for bem sucedida, caso contrário retorna um código de erro sempre menor do que zero. O significado de cada código de erro retornado é fornecido no item “Códigos de Erro”.

Consulte

RFnetRxMsg, RFnetRxBufCount, RFnetRxBufBytes, RFnetRxBufFree, RFnetRxBufClear.

Exemplo

```
#include <windows.h>
#include "xpcom32.h"

char cBuffer[10];
int  nBytes;

int PASCAL WinMain(HANDLE hInstance, HANDLE hPrevInstance, LPSTR lpCmdLine, int nCmdShow)
        {
        if (RFnetOpen(COM1,9600,256,256,2))
```

```
        return FALSE;
nBytes = RFnetRxMsgBytes(COM1);
if (nBytes <= 0)
        {
        if (RFN_RXBEMPTY == nBytes || 0 == nBytes)
                {
                MessageBox((HWND)NULL, "Não existem dados disponiveis", "Exemplo", MB_OK);
                return TRUE;
                }
        return FALSE;
        }
MessageBox((HWND)NULL, itoa(nBytes, cBuffer, 10), "Bytes de dados disponivel", MB_OK);
if (RFnetClose(COM1))
        return FALSE;
return TRUE;
}
```

RFnetRxBufCount

Sumário

```
int FAR PASCAL RFnetRxBufCount (int nCanal                    // Canal de comunicação
                                );
```

Windows 16	Windows 32	DOS
	•	

Descrição

A função 'RFnetRxBufCount' informa o número de mensagens armazenadas na área reservada para o *buffer* de recepção.

O argumento 'nCanal' especifica o canal de comunicação, possibilita o uso de no máximo 16 canais (COM1 a COM16).

Valor Retornado

A função retorna valor positivo (número de mensagens) ou igual a zero (*buffer* vazio) quando sua execução for bem sucedida, caso contrário retorna um código de erro sempre menor do que zero. O significado de cada código de erro retornado é fornecido no item “Códigos de Erro”.

Consulte

RFnetRxMsg, RFnetRxMsgBytes, RFnetRxBufBytes, RFnetRxBufFree, RFnetRxBufClear.

Exemplo

```
#include <windows.h>
#include <stdlib.h>
#include "xpcom32.h"

int nNumMsg;
char cBuffer[10];

int PASCAL WinMain(HANDLE hInstance, HANDLE hPrevInstance, LPSTR lpCmdLine, int nCmdShow)
        {
        if (RFnetOpen(COM1,9600,256,256,2))
                return FALSE;
        nNumMsg = RFnetRxBufCount(COM1);
        if (nNumMsg < 0)
                return FALSE;
        if (0 == nNumMsg)
                MessageBox((HWND)NULL, "Não existem mensagens disponiveis", "Exemplo", MB_OK);
        else
                MessageBox((HWND)NULL, itoa(nNumMsg, cBuffer, 10), " Mensagens armazenadas", MB_OK);
        if (RFnetClose(COM1))
```

```
        return FALSE;
    return TRUE;
    }
```

RFnetRxBufBytes

Sumário

```
int FAR PASCAL RFnetRxBufBytes (int nCanal          // Canal de comunicação
                                );
```

Windows 16	Windows 32	DOS
	•	

Descrição

A função 'RFnetRxBufBytes' informa o número de *bytes* utilizados na área reservada do *buffer* de recepção.

O argumento 'nCanal' especifica o canal de comunicação, possibilita o uso de no máximo 16 canais (COM1 a COM16).

Valor Retornado

A função retorna valor positivo (*bytes* utilizados) ou igual a zero (*buffer* vazio) quando sua execução for bem sucedida, caso contrário retorna um código de erro sempre menor do que zero. O significado de cada código de erro retornado é fornecido no item “Códigos de Erro”.

Consulte

RFnetRxMsg, RFnetRxMsgBytes, RFnetRxBufCount, RFnetRxBufFree, RFnetRxBufClear.

Exemplo

```
#include <windows.h>
#include <stdlib.h>
#include "xpcom32.h"

int nBytes;
char cBuffer[10];

int PASCAL WinMain(HANDLE hInstance, HANDLE hPrevInstance, LPSTR lpCmdLine, int nCmdShow)
    {
    if (RFnetOpen(COM1,9600,256,256,2))
        return FALSE;
    nBytes = RFnetRxBufBytes(COM1);
    if (nBytes < 0)
        return FALSE;
    if (0 == nBytes)
        MessageBox((HWND)NULL, "Não existem dados disponiveis", "Exemplo", MB_OK);
    else
        MessageBox((HWND)NULL, itoa(nBytes, cBuffer, 10), "Area ocupada", MB_OK);
    if (RFnetClose(COM1))
        return FALSE;
    return TRUE;
    }
```

RFnetRxBufFree

Sumário

```
int FAR PASCAL RFnetRxBufFree (int nCanal          // Canal de comunicação
                               );
```

Windows 16	Windows 32	DOS

	•	

Descrição

A função 'RFnetRxBufFree' informa o numero de *bytes* livres na área reservada para o *buffer* de recepção.

O argumento 'nCanal' especifica o canal de comunicação, possibilita o uso de no máximo 16 canais (COM1 a COM16).

Valor Retornado

A função retorna valor positivo (*bytes* disponíveis) ou igual a zero (*buffer* cheio) quando sua execução for bem sucedida, caso contrário retorna um código de erro sempre menor do que zero. O significado de cada código de erro retornado é fornecido no item “Códigos de Erro”.

Consulte

RFnetRxMsgBytes, RFnetRxMsg, RFnetRxBufCount, RFnetRxBufBytes, RFnetRxBufClear.

Exemplo

```
#include <windows.h>
#include <stdlib.h>
#include "xpcom32.h"

int nBytes;
int nStatus;
char cBuffer[10];

int PASCAL WinMain(HANDLE hInstance, HANDLE hPrevInstance, LPSTR lpCmdLine, int nCmdShow)
        {
        if (RFnetOpen(COM1,9600,256,256,2))
                return FALSE;
        nBytes = RFnetRxBufFree(COM1);
        if (nBytes < 0)
                return FALSE;
        MessageBox((HWND)NULL, itoa(nBytes, cBuffer, 10), "Area disponivel", MB_OK);
        if (RFnetClose(COM1))
                return FALSE;
        return TRUE;
        }
```

RFnetRxBufClear

Sumário

```
int FAR PASCAL RFnetRxBufClear (int nCanal                    // Canal de comunicação
                                );
```

Windows 16	Windows 32	DOS
	•	

Descrição

A função 'RFnetRxBufClear' descarta os dados do *buffer* de recepção da rede de RF.

O argumento 'nCanal' especifica o canal de comunicação, possibilita o uso de no máximo 16 canais (COM1 a COM16).

Valor Retornado

A função retorna zero quando sua execução for bem sucedida, caso contrário retorna um código de erro sempre menor do que zero. O significado de cada código de erro retornado é fornecido no item “Códigos de Erro”.

Consulte

RFnetRxMsgBytes, RFnetRxMsg, RFnetRxBufCount, RFnetRxBufBytes, RFnetRxBufFree.

Exemplo

```
#include <windows.h>
#include "xpcom32.h"

int PASCAL WinMain(HANDLE hInstance, HANDLE hPrevInstance, LPSTR lpCmdLine, int nCmdShow)
        {
        if (RFnetOpen(COM1,9600,256,256,2))
                return FALSE;
        if (RFnetRxBufClear(COM1))
                return FALSE;
        if (RFnetClose(COM1))
                return FALSE;
        return TRUE;
        }
```

Funções de Conversão de Formato

Estas funções complementam as funções de comunicação propriamente ditas, fornecendo facilidades de conversão do formato XPbasic para ASCII, TEXTO ou DBASE (e LOTUS sob DOS) e vice versa.

Descrição detalhada do formato XPbasic pode ser encontrada no Manual de Referência Técnica dos coletores de dados. Este formato admite registros com até 10 campos. São aceitos três tipos de campo: Campo de valor inteiro com valores de -32768 até 32767, campo de valor em ponto flutuante com valores de -1.0E-154 até 5.2E+151 e campo alfanumérico contendo de 1 a 80 caracteres. São admitidos um máximo de 10 campos por registro.

Arquivo no formato ASCII admite os três tipos de campo, separados por vírgula. Os campos podem ter tamanhos variáveis mas precisam estar dentro do tamanho aceito pelo XPbasic. Campos alfanuméricos devem iniciar com aspas e terminar com aspas seguida de vírgula. São admitidas aspas dentro do campo contanto que não sejam seguidas de vírgula. A última aspas no último campo de um registro não precisa estar seguida de vírgula. Para separar registros são aceitos os caracteres CR, LF ou CR LF.

Arquivo no formato TEXTO só admite um campo alfanumérico, de tamanho fixo, por registro. Para separar registros são aceitos os caracteres CR, LF ou CR LF.

Arquivo no formato DBASE e LOTUS devem conter campos dentro dos limites aceitos pelo formato XPbasic.

A biblioteca para Windows não suporta conversão de/para formato LOTUS.

FXPBasicConvFromAsc

Sumário

```
int FAR PASCAL FXPBasicConvFromAsc(const char FAR* szFileOrig,      // Nome do arquivo origem a ser convertido
                                   const char FAR* szFileDest,      // Nome do arquivo destino a ser gerado
                                   CONVFILEFORMAT FAR* pF           // Ponteiro p/ estrutura com formato arquivo origem
                                   );
```

Windows 16	Windows 32	DOS
•	•	

Descrição

A função 'FXPBasicConvFromAsc' converte o arquivo fornecido no formato ASCII para formato XPbasic.

O argumento 'szFileOrig' é uma cadeia de caracteres com o nome do arquivo origem.

O argumento 'szFileDest' é uma cadeia de caracteres com o nome do arquivo destino.

O argumento 'pF' é um ponteiro para uma estrutura do tipo CONVFILEFORMAT que define o formato do arquivo origem. Esta estrutura está definida em XPCOMxx.H e contém os campos:

cFields — Número de campos do arquivo origem, deve ser entre 1 e 10.

Para cada um dos ‘cFields’ campos acima temos a estrutura ‘Field’ definida por:

```
struct  {
        cName[11]       Nome do campo só para conversões com arquivo destino tipo DBF
        cType           Tipo do campo XPFLD_INT, XPFLD_FLO, XPFLD_STR. Ver XPCOMxx.H.
        cLength         Comprimento do campo
        cDec            Número de casas decimais usado para ponto flutuante na conversão de arquivos
                        formato XPbasic para formato DBASE
        } Field[10];
```

Valor Retornado

A função retorna zero quando a execução for bem sucedida, caso contrário retorna um código de erro sempre menor do que zero. O significado de cada código de erro retornado é fornecido no item “Códigos de Erro”.

Consulte

FXPBasicConvFromDBF, FXPBasicConvFromText, FXPBasicConvToAsc, FXPBasicConvToDBF, FXPBasicConvToText.

Exemplo

```
#include <windows.h>
#include <string.h>
#include “xpcom16.h”

char szInFile[20];
char szOutFile[20];
CONVFILEFORMAT fF;

int PASCAL WinMain(HANDLE hInstance, HANDLE hPrevInstance, LPSTR lpCmdLine, int nCmdShow)
        {
        fF.cFields = 3;
        fF.Field[0].cType = XPFLD_STR;
        fF.Field[0].cLength = 80;
        fF.Field[0].cDec = 0;
        fF.Field[1].cType = XPFLD_INT;
        fF.Field[1].cLength = 2;
        fF.Field[1].cDec = 0;
        fF.Field[2].cType = XPFLD_FLO;
        fF.Field[2].cLength = 8;
        fF.Field[2].cDec = 0;
        strcpy(szInFile, “C:\\TRIX\\ARQASC.DAT”);
        strcpy(szOutFile, “C:\\TRIX\\ARQXPB.DAT”);
        if (FXPBasicConvFromAsc(szInFile, szOutFile, &fF))
                return FALSE;
        return TRUE;
        }
```

ConvASC2XPbasic

Sumário

```
int ConvASC2XPbasic(char* szFileOrig,           // Nome do arquivo origem a ser convertido
                    char* szFileDest,           // Nome do arquivo destino a ser gerado
                    struct FileFormat_t* pF     // Ponteiro para estrutura com formato do arquivo origem
                   );
```

Windows 16	Windows 32	DOS
		•

Descrição

A função 'ConvASC2XPbasic' converte o arquivo fornecido no formato ASCII para formato XPbasic.

O argumento 'szFileOrig' é uma cadeia de caracteres com o nome do arquivo origem.

O argumento 'szFileDest' é uma cadeia de caracteres com o nome do arquivo destino.

O argumento 'pF' é um ponteiro para uma estrutura do tipo FileFormat_t que define o formato do arquivo origem. Esta estrutura está definida em XPCOMLIB.H e contém os campos:

cFields Número de campos do arquivo origem, deve ser entre 1 e 10.

Para cada um dos ‘cFields’ campos acima temos a estrutura ‘Field’ definida por:

```
struct  {
        cTipo           Tipo do campo XPFLD_INT, XPFLD_FLO, XPFLD_STR. Ver XPCOMDEF.H.
        cLength         Comprimento do campo
        cDec            Número de casas decimais usado para ponto flutuante na conversão de arquivos
                        formato XPbasic para formato DBASE
        } Field[10];
```

Valor Retornado

A função retorna zero quando a execução for bem sucedida, caso contrário retorna um código de erro sempre menor do que zero. O significado de cada código de erro retornado é fornecido no item “Códigos de Erro”.

Consulte

ConvDB2XPbasic, ConvLOT2XPbasic, ConvTXT2XPbasic, ConvXPbasic2ASC, ConvXPbasic2DB, ConvXPbasic2LOT, ConvXPbasic2TXT.

Exemplo

```
#include <stdlib.h>
#include <string.h>
#include “xpcomlib.h”
#include “xpcomdef.h”

char szInFile[20];
char szOutFile[20];
struct FileFormat_t fF;

void main(void)
        {
        fF.cFields = 3;
        fF.Field[0].cTipo = XPFLD_STR;
        fF.Field[0].cLength = 80;
        fF.Field[0].cDec = 0;
        fF.Field[1].cTipo = XPFLD_INT;
        fF.Field[1].cLength = 2;
        fF.Field[1].cDec = 0;
        fF.Field[2].cTipo = XPFLD_FLO;
        fF.Field[2].cLength = 8;
        fF.Field[2].cDec = 0;
        strcpy(szInFile, “C:\\TRIX\\ARQASC.DAT”);
        strcpy(szOutFile, “C:\\TRIX\\ARQXPB.DAT”);
        if (ConvASC2XPbasic(szInFile, szOutFile, &fF))
                exit(1);
        }
```

FXPBasicConvFromDBF

Sumário

```
int FAR PASCAL FXPBasicConvFromDBF(const char FAR* szFileOrig,  // Nome do arquivo origem a ser convertido
                                   const char FAR* szFileDest,  // Nome do arquivo destino a ser gerado
```

```
    CONVFILEFORMAT FAR* pF          // Ponteiro para estrutura com formato do arquivo origem
);
```

Windows 16	Windows 32	DOS
•	•	

Descrição

A função 'FXPBasicConvFromDBF' converte o arquivo fornecido no formato DBASE para formato XPbasic.

O argumento 'szFileOrig' é uma cadeia de caracteres com o nome do arquivo origem.

O argumento 'szFileDest' é uma cadeia de caracteres com o nome do arquivo destino.

O argumento 'pF' é um ponteiro para uma estrutura do tipo CONVFILEFORMAT que define o formato do arquivo origem. Esta estrutura está definida em XPCOMxx.H e contém os campos:

cFields — Número de campos do arquivo origem, deve ser entre 1 e 10.

Para cada um dos ‘cFields’ campos acima temos a estrutura ‘Field’ definida por:

```
struct  {
        cName[11]       Nome do campo só para conversões com arquivo destino tipo DBF
        cType           Tipo do campo XPFLD_INT, XPFLD_FLO, XPFLD_STR. Ver XPCOMDEF.H.
        cLength         Comprimento do campo
        cDec            Número de casas decimais usado para ponto flutuante na conversão de arquivos
                        formato XPbasic para formato DBASE
        } Field[10];
```

Valor Retornado

A função retorna zero quando a execução for bem sucedida, caso contrário retorna um código de erro sempre menor do que zero. O significado de cada código de erro retornado é fornecido no item “Códigos de Erro”.

Consulte

FXPBasicConvFromAsc, FXPBasicConvFromText, FXPBasicConvToAsc, FXPBasicConvToDBF, FXPBasicConvToText.

Exemplo

```
#include <windows.h>
#include <string.h>
#include “xpcom16.h”

char szInFile[20];
char szOutFile[20];
CONVFILEFORMAT fF;

int PASCAL WinMain(HANDLE hInstance, HANDLE hPrevInstance, LPSTR lpCmdLine, int nCmdShow)
    {
    fF.cFields = 3;
    fF.Field[0].cType = XPFLD_STR;
    fF.Field[0].cLength = 80;
    fF.Field[0].cDec = 0;
    fF.Field[1].cType = XPFLD_INT;
    fF.Field[1].cLength = 2;
    fF.Field[1].cDec = 0;
    fF.Field[2].cType = XPFLD_FLO;
    fF.Field[2].cLength = 8;
    fF.Field[2].cDec = 0;
    strcpy(szInFile, “C:\\TRIX\\ARQDB.DBF”);
    strcpy(szOutFile, “C:\\TRIX\\ARQXPB.DAT”);
    if (FXPBasicConvFromDBF(szInFile, szOutFile, &fF))
        return FALSE;
    return TRUE;
```

```
}
```

ConvDB2XPbasic

Sumário

```
int ConvDB2XPbasic(char* szFileOrig,                    // Nome do arquivo origem a ser convertido
                   char* szFileDest,                        // Nome do arquivo destino a ser gerado
                   struct FileFormat_t* pF              // Ponteiro para estrutura com formato do arquivo origem
                   );
```

Windows 16	Windows 32	DOS
		•

Descrição

A função 'ConvDB2XPbasic' converte o arquivo fornecido no formato DBASE para formato XPbasic.

O argumento 'szFileOrig' é uma cadeia de caracteres com o nome do arquivo origem.

O argumento 'szFileDest' é uma cadeia de caracteres com o nome do arquivo destino.

O argumento 'pF' é um ponteiro para uma estrutura do tipo FileFormat_t que define o formato do arquivo origem. Esta estrutura está definida em XPCOMLIB.H e contém os campos:

cFields — Número de campos do arquivo origem, deve ser entre 1 e 10.

Para cada um dos ‘cFields’ campos acima temos a estrutura ‘Field’ definida por:

```
struct  {
        cTipo       Tipo do campo XPFLD_INT, XPFLD_FLO, XPFLD_STR. Ver XPCOMDEF.H.
        cLength     Comprimento do campo
        cDec        Número de casas decimais usado para ponto flutuante na conversão de arquivos
                    formato XPbasic para formato DBASE
        } Field[10];
```

Valor Retornado

A função retorna zero quando a execução for bem sucedida, caso contrário retorna um código de erro sempre menor do que zero. O significado de cada código de erro retornado é fornecido no item “Códigos de Erro”.

Consulte

ConvASC2XPbasic, ConvLOT2XPbasic, ConvTXT2XPbasic, ConvXPbasic2ASC, ConvXPbasic2DB, ConvXPbasic2LOT, ConvXPbasic2TXT.

Exemplo

```
#include <stdlib.h>
#include <string.h>
#include “xpcomlib.h”
#include “xpcomdef.h”

char szInFile[20];
char szOutFile[20];
struct FileFormat_t fF;

void main(void)
    {
    fF.cFields = 3;
    fF.Field[0].cTipo = XPFLD_STR;
    fF.Field[0].cLength = 80;
    fF.Field[0].cDec = 0;
    fF.Field[1].cTipo = XPFLD_INT;
    fF.Field[1].cLength = 2;
```

```
fF.Field[1].cDec = 0;
fF.Field[2].cTipo = XPFLD_FLO;
fF.Field[2].cLength = 8;
fF.Field[2].cDec = 0;
strcpy(szInFile, "C:\\TRIX\\ARQDB.DBF");
strcpy(szOutFile, "C:\\TRIX\\ARQXPB.DAT");
if (ConvDB2XPbasic(szInFile, szOutFile, &fF))
    exit(1);
}
```

FXPBasicConvFromText

Sumário

```
int FAR PASCAL FXPBasicConvFromText(const char FAR* szFileOrig,      // Nome do arquivo origem a ser convertido
                                    const char FAR* szFileDest,      // Nome do arquivo destino a ser gerado
                                    CONVFILEFORMAT FAR* pF           // Ponteiro p/ estrutura com formato arquivo origem
                                    );
```

Windows 16	Windows 32	DOS
•	•	

Descrição

A função 'FXPBasicConvFromText' converte o arquivo fornecido no formato TEXTO para formato XPbasic.

O argumento 'szFileOrig' é uma cadeia de caracteres com o nome do arquivo origem.

O argumento 'szFileDest' é uma cadeia de caracteres com o nome do arquivo destino.

O argumento 'pF' é um ponteiro para uma estrutura do tipo CONVFILEFORMAT que define o formato do arquivo origem. Esta estrutura está definida em XPCOMxx.H e contém os campos:

cFields — Número de campos do arquivo origem, deve ser entre 1 e 10.

Para cada um dos 'cFields' campos acima temos a estrutura 'Field' definida por:

```
struct  {
        cName[11]       Nome do campo só para conversões com arquivo destino tipo DBF
        cType           Tipo do campo XPFLD_INT, XPFLD_FLO, XPFLD_STR. Ver XPCOMxx.H.
        cLength         Comprimento do campo
        cDec            Número de casas decimais usado para ponto flutuante na conversão de arquivos
                        formato XPbasic para formato DBASE
        } Field[10];
```

Valor Retornado

A função retorna zero quando a execução for bem sucedida, caso contrário retorna um código de erro sempre menor do que zero. O significado de cada código de erro retornado é fornecido no item "Códigos de Erro".

Consulte

FXPBasicConvFromAsc, FXPBasicConvFromDBF, FXPBasicConvToAsc, FXPBasicConvToDBF, FXPBasicConvToText.

Exemplo

```
#include <windows.h>
#include <string.h>
#include "xpcom16.h"

char szInFile[20];
char szOutFile[20];
CONVFILEFORMAT fF;

int PASCAL WinMain(HANDLE hInstance, HANDLE hPrevInstance, LPSTR lpCmdLine, int nCmdShow)
```

```
{
fF.cFields = 3;
fF.Field[0].cType = XPFLD_STR;
fF.Field[0].cLength = 80;
fF.Field[0].cDec = 0;
fF.Field[1].cType = XPFLD_INT;
fF.Field[1].cLength = 2;
fF.Field[1].cDec = 0;
fF.Field[2].cType = XPFLD_FLO;
fF.Field[2].cLength = 8;
fF.Field[2].cDec = 0;
strcpy(szInFile, "C:\\TRIX\\ARQTXT.DAT");
strcpy(szOutFile, "C:\\TRIX\\ARQXPB.DAT");
if (FXPBasicConvFromText(szInFile, szOutFile, &fF))
      return FALSE;
return TRUE;
}
```

ConvTXT2XPbasic

Sumário

```
int ConvTXT2XPbasic(char* szFileOrig,          // Nome do arquivo origem a ser convertido
                    char*  szFileDest,         // Nome do arquivo destino a ser gerado
                    struct FileFormat_t* pF    // Ponteiro para estrutura com formato do arquivo origem
                    );
```

Windows 16	Windows 32	DOS
		•

Descrição

A função 'ConvTXT2XPbasic' converte o arquivo fornecido no formato TEXTO para formato XPbasic.

O argumento 'szFileOrig' é uma cadeia de caracteres com o nome do arquivo origem.

O argumento 'szFileDest' é uma cadeia de caracteres com o nome do arquivo destino.

O argumento 'pF' é um ponteiro para uma estrutura do tipo FileFormat_t que define o formato do arquivo origem. Esta estrutura está definida em XPCOMLIB.H e contém os campos:

cFields — Número de campos do arquivo origem, deve ser entre 1 e 10.

Para cada um dos 'cFields' campos acima temos a estrutura 'Field' definida por:

```
struct    {
          cTipo          Tipo do campo XPFLD_INT, XPFLD_FLO, XPFLD_STR. Ver XPCOMDEF.H.
          cLength        Comprimento do campo
          cDec           Número de casas decimais usado para ponto flutuante na conversão de arquivos
                         formato XPbasic para formato DBASE
          } Field[10];
```

Valor Retornado

A função retorna zero quando a execução for bem sucedida, caso contrário retorna um código de erro sempre menor do que zero. O significado de cada código de erro retornado é fornecido no item "Códigos de Erro".

Consulte

ConvDB2XPbasic, ConvLOT2XPbasic, ConvASC2XPbasic, ConvXPbasic2ASC, ConvXPbasic2DB, ConvXPbasic2LOT, ConvXPbasic2TXT.

Exemplo

```
#include <stdlib.h>
```

```
#include <string.h>
#include “xpcomlib.h”
#include “xpcomdef.h”

char szInFile[20];
char szOutFile[20];
struct FileFormat_t fF;

void main(void)
        {
        fF.cFields = 3;
        fF.Field[0].cTipo = XPFLD_STR;
        fF.Field[0].cLength = 80;
        fF.Field[0].cDec = 0;
        fF.Field[1].cTipo = XPFLD_INT;
        fF.Field[1].cLength = 2;
        fF.Field[1].cDec = 0;
        fF.Field[2].cTipo = XPFLD_FLO;
        fF.Field[2].cLength = 8;
        fF.Field[2].cDec = 0;
        strcpy(szInFile, “C:\\TRIX\\ARQTXT.DAT”);
        strcpy(szOutFile, “C:\\TRIX\\ARQXPB.DAT”);
        if (ConvTXT2XPbasic(szInFile, szOutFile, &fF))
                exit(1);
        }
```

ConvLOT2XPbasic

Sumário

```
int ConvLOT2XPbasic(char* szFileOrig,          // Nome do arquivo origem a ser convertido
                    char*  szFileDest,         // Nome do arquivo destino a ser gerado
                    struct FileFormat_t* pF    // Ponteiro para estrutura com formato do arquivo origem
                    );
```

Windows 16	Windows 32	DOS
		•

Descrição

A função 'ConvLOT2XPbasic' converte o arquivo fornecido no formato LOTUS para formato XPbasic.

O argumento 'szFileOrig' é uma cadeia de caracteres com o nome do arquivo origem.

O argumento 'szFileDest' é uma cadeia de caracteres com o nome do arquivo destino.

O argumento 'pF' é um ponteiro para uma estrutura do tipo FileFormat_t que define o formato do arquivo origem. Esta estrutura está definida em XPCOMLIB.H e contém os campos:

cFields — Número de campos do arquivo origem, deve ser entre 1 e 10.

Para cada um dos ‘cFields’ campos acima temos a estrutura ‘Field’ definida por:

```
struct  {
        cTipo       Tipo do campo XPFLD_INT, XPFLD_FLO, XPFLD_STR. Ver XPCOMDEF.H.
        cLength     Comprimento do campo
        cDec        Número de casas decimais usado para ponto flutuante na conversão de arquivos
                    formato XPbasic para formato DBASE
        } Field[10];
```

Valor Retornado

A função retorna zero quando a execução for bem sucedida, caso contrário retorna um código de erro sempre menor do que zero. O significado de cada código de erro retornado é fornecido no item “Códigos de Erro”.

Consulte

ConvDB2XPbasic, ConvASC2XPbasic, ConvTXT2XPbasic, ConvXPbasic2ASC, ConvXPbasic2DB, ConvXPbasic2LOT, ConvXPbasic2TXT.

Exemplo

```
#include <stdlib.h>
#include <string.h>
#include “xpcomlib.h”
#include “xpcomdef.h”

char szInFile[20];
char szOutFile[20];
struct FileFormat_t fF;

void main(void)
    {
    fF.cFields = 3;
    fF.Field[0].cTipo = XPFLD_STR;
    fF.Field[0].cLength = 80;
    fF.Field[0].cDec = 0;
    fF.Field[1].cTipo = XPFLD_INT;
    fF.Field[1].cLength = 2;
    fF.Field[1].cDec = 0;
    fF.Field[2].cTipo = XPFLD_FLO;
    fF.Field[2].cLength = 8;
    fF.Field[2].cDec = 0;
    strcpy(szInFile, “C:\\TRIX\\ARQLOT.WK1”);
    strcpy(szOutFile, “C:\\TRIX\\ARQXPB.DAT”);
    if (ConvLOT2XPbasic(szInFile, szOutFile, &fF))
        exit(1);
    }
```

FXPBasicConvToAsc

Sumário

```
int FAR PASCAL FXPBasicConvToAsc(const char FAR* szFileOrig,     // Nome do arquivo origem a ser convertido
                                 const char FAR* szFileDest,     // Nome do arquivo destino a ser gerado
                                 int nLineTerm                   // Especifica qual terminador de linha deve ser usado
                                 );
```

Windows 16	Windows 32	DOS
•	•	

Descrição

A função 'FXPBasicConvToAsc' converte o arquivo fornecido no formato XPbasic para formato ASCII.

O argumento 'szFileOrig' é uma cadeia de caracteres com o nome do arquivo origem.

O argumento 'szFileDest' é uma cadeia de caracteres com o nome do arquivo destino.

O argumento ‘nLineTerm’ especifica qual terminador de linha deve ser usado. O padrão é usar CR seguido de LF, usar as constantes definidas em XPCOMxx.H:

XPLTERM_CRLF	Terminador de linha usado será caractere CR (0x0D) seguido do caractere LF (0x0A).
XPLTERM_CR	Terminador de linha usado será caractere CR (0x0D).
XPLTERM_LF	Terminador de linha usado será caractere LF (0x0A).

Valor Retornado

A função retorna zero quando a execução for bem sucedida, caso contrário retorna um código de erro sempre menor do que zero. O significado de cada código de erro retornado é fornecido no item “Códigos de Erro”.

Consulte

FXPBasicConvFromAsc, FXPBasicConvFromDBF, FXPBasicConvFromText, FXPBasicConvToDBF, FXPBasicConvToText.

Exemplo

```
#include <windows.h>
#include <string.h>
#include “xpcom16.h”

char szInFile[20];
char szOutFile[20];

int PASCAL WinMain(HANDLE hInstance, HANDLE hPrevInstance, LPSTR lpCmdLine, int nCmdShow)
    {
    strcpy(szInFile, “C:\\TRIX\\ARQXPB.DAT”);
    strcpy(szOutFile, “C:\\TRIX\\ARQASC.DAT”);
    if (FXPBasicConvToAsc(szInFile, szOutFile,XPLTERM_CRLF))
        return FALSE;
    return TRUE;
    }
```

ConvXPbasic2ASC

Sumário

```
int ConvXPbasic2ASC(char* szFileOrig,          // Nome do arquivo origem a ser convertido
                    char*  szFileDest,         // Nome do arquivo destino a ser gerado
                    struct FileFormat_t* pF,   // Ponteiro para estrutura com formato do arquivo destino
                    int nTerminador,           // Especifica qual terminador de linha deve ser usado
                    unsigned int bWrite        // Escreve a estrutura do arquivo no início quando bWrite diferente de 0
                    );
```

Windows 16	Windows 32	DOS
		•

Descrição

A função 'ConvXPbasic2ASC' converte o arquivo fornecido no formato XPbasic para formato ASCII.

O argumento 'szFileOrig' é uma cadeia de caracteres com o nome do arquivo origem.

O argumento 'szFileDest' é uma cadeia de caracteres com o nome do arquivo destino.

O argumento 'pF' é um ponteiro para uma estrutura do tipo FileFormat_t que define o formato do arquivo destino. Esta estrutura está definida em XPCOMLIB.H e contém os campos:

cFields — Número de campos do arquivo destino, deve ser entre 1 e 10.

Para cada um dos ‘cFields’ campos acima temos a estrutura ‘Field’ definida por:

```
struct  {
        cTipo       Tipo do campo XPFLD_INT, XPFLD_FLO, XPFLD_STR. Ver XPCOMDEF.H.
        cLength     Comprimento do campo
        cDec        Número de casas decimais usado para ponto flutuante na conversão de arquivos
                    formato XPbasic para formato DBASE
        } Field[10];
```

O argumento ‘nTerminador’ especifica qual terminador de linha deve ser usado. O padrão é usar CR seguido de LF, usar as constantes definidas em XPCOMDEF.H:

CF_CRLF — Terminador de linha usado será caractere CR (0x0D) seguido do caractere LF (0x0A).

CF_CR	Terminador de linha usado será caractere CR (0x0D).
CF_LF	Terminador de linha usado será caractere LF (0x0A).

O argumento ‘bWrite’ quando diferente de zero faz com que seja escrito no início do arquivo destino a estrutura do arquivo convertido.

Valor Retornado

A função retorna zero quando a execução for bem sucedida, caso contrário retorna um código de erro sempre menor do que zero. O significado de cada código de erro retornado é fornecido no item “Códigos de Erro”.

Consulte

ConvASC2XPbasic, ConvDB2XPbasic, ConvLOT2XPbasic, ConvTXT2XPbasic, ConvXPbasic2DB, ConvXPbasic2LOT, ConvXPbasic2TXT.

Exemplo

```
#include <stdlib.h>
#include <string.h>
#include “xpcomlib.h”
#include “xpcomdef.h”

char szInFile[20];
char szOutFile[20];
struct FileFormat_t fF;

void main(void)
        {
        fF.cFields = 3;
        fF.Field[0].cTipo = XPFLD_STR;
        fF.Field[0].cLength = 80;
        fF.Field[0].cDec = 0;
        fF.Field[1].cTipo = XPFLD_INT;
        fF.Field[1].cLength = 2;
        fF.Field[1].cDec = 0;
        fF.Field[2].cTipo = XPFLD_FLO;
        fF.Field[2].cLength = 8;
        fF.Field[2].cDec = 0;
        strcpy(szInFile, “C:\\TRIX\\ARQXPB.DAT”);
        strcpy(szOutFile, “C:\\TRIX\\ARQASC.DAT”);
        if (ConvXPbasic2ASC(szInFile, szOutFile, &fF , CF_CRLF, 0))
                exit(1);
        }
```

FXPBasicConvToDBF

Sumário

```
int FAR PASCAL FXPBasicConvToDBF(const char FAR* szFileOrig,       // Nome do arquivo origem a ser convertido
                                 const char FAR* szFileDest,       // Nome do arquivo destino a ser gerado
                                 CONVFILEFORMAT FAR* pF  // Ponteiro para estrutura com formato do arquivo origem
                                 );
```

Windows 16	Windows 32	DOS
•	•	

Descrição

A função 'FXPBasicConvToDBF' converte o arquivo fornecido no formato XPbasic para formato DBASE.

O argumento 'szFileOrig' é uma cadeia de caracteres com o nome do arquivo origem.

O argumento 'szFileDest' é uma cadeia de caracteres com o nome do arquivo destino.

O argumento 'pF' é um ponteiro para uma estrutura do tipo CONVFILEFORMAT que define o formato do arquivo destino. Esta estrutura está definida em XPCOMxx.H e contém os campos:

cFields	Número de campos do arquivo destino, deve ser entre 1 e 10.

Para cada um dos ‘cFields’ campos acima temos a estrutura ‘Field’ definida por:

struct {

cName[11]	Nome do campo só para conversões com arquivo destino tipo DBF
cType	Tipo do campo ‘C’ ou ‘N’ para campo tipo caractere ou tipo numérico.
cLength	Comprimento do campo
cDec	Número de casas decimais usado para ponto flutuante na conversão de arquivos formato XPbasic para formato DBASE

} Field[10];

Valor Retornado

A função retorna zero quando a execução for bem sucedida, caso contrário retorna um código de erro sempre menor do que zero. O significado de cada código de erro retornado é fornecido no item “Códigos de Erro”.

Consulte

FXPBasicConvFromAsc, FXPBasicConvFromDBF, FXPBasicConvFromText, FXPBasicConvToAsc, FXPBasicConvToText.

Exemplo

```
#include <windows.h>
#include <string.h>
#include “xpcom16.h”

char szInFile[20];
char szOutFile[20];
CONVFILEFORMAT fF;

int PASCAL WinMain(HANDLE hInstance, HANDLE hPrevInstance, LPSTR lpCmdLine, int nCmdShow)
    {
    fF.cFields = 3;
    fF.Field[0].cType = ‘C’;
    fF.Field[0].cLength = 80;
    fF.Field[0].cDec = 0;
    fF.Field[1].cType = ‘N’;
    fF.Field[1].cLength = 2;
    fF.Field[1].cDec = 0;
    fF.Field[2].cType = ‘N’;
    fF.Field[2].cLength = 8;
    fF.Field[2].cDec = 3;
    strcpy(szInFile, “C:\\TRIX\\ARQXPB.DAT”);
    strcpy(szOutFile, “C:\\TRIX\\ARQDB.DBF”);
    if (FXPBasicConvToDBF(szInFile, szOutFile, &fF))
        return FALSE;
    return TRUE;
    }
```

ConvXPbasic2DB

Sumário

```
int ConvXPbasic2DB(char* szFileOrig,          // Nome do arquivo origem a ser convertido
                   char* szFileDest,          // Nome do arquivo destino a ser gerado
                   struct FileFormat_t* pF    // Ponteiro para estrutura com formato do arquivo destino
                   );
```

Windows 16	Windows 32	DOS
		•

Descrição

A função 'ConvXPbasic2DB' converte o arquivo fornecido no formato XPbasic para formato DBASE.

O argumento 'szFileOrig' é uma cadeia de caracteres com o nome do arquivo origem.

O argumento 'szFileDest' é uma cadeia de caracteres com o nome do arquivo destino.

O argumento 'pF' é um ponteiro para uma estrutura do tipo FileFormat_t que define o formato do arquivo destino. Esta estrutura está definida em XPCOMLIB.H e contém os campos:

cFields Número de campos do arquivo destino, deve ser entre 1 e 10.

Para cada um dos ‘cFields’ campos acima temos a estrutura ‘Field’ definida por:

```
struct  {
        cTipo       Tipo do campo ‘C’ ou ‘N’ para campo tipo caractere ou tipo numérico.
        cLength     Comprimento do campo
        cDec        Número de casas decimais usado para ponto flutuante na conversão de arquivos
                    formato XPbasic para formato DBASE
        } Field[10];
```

Valor Retornado

A função retorna zero quando a execução for bem sucedida, caso contrário retorna um código de erro sempre menor do que zero. O significado de cada código de erro retornado é fornecido no item “Códigos de Erro”.

Consulte

ConvASC2XPbasic, ConvDB2XPbasic, ConvLOT2XPbasic, ConvTXT2XPbasic, ConvXPbasic2ASC, ConvXPbasic2LOT, ConvXPbasic2TXT.

Exemplo

```
#include <stdlib.h>
#include <string.h>
#include “xpcomlib.h”
#include “xpcomdef.h”

char szInFile[20];
char szOutFile[20];
struct FileFormat_t fF;

void main(void)
    {
    fF.cFields = 3;
    fF.Field[0].cTipo = ‘C’;
    fF.Field[0].cLength = 80;
    fF.Field[0].cDec = 0;
    fF.Field[1].cTipo = ‘N;
    fF.Field[1].cLength = 2;
    fF.Field[1].cDec = 0;
    fF.Field[2].cTipo = ‘N’;
    fF.Field[2].cLength = 8;
    fF.Field[2].cDec = 3;
    strcpy(szInFile, “C:\\TRIX\\ARQXPB.DAT”);
    strcpy(szOutFile, “C:\\TRIX\\ARQDB.DBF”);
    if (ConvXPbasic2DB(szInFile, szOutFile, &fF))
        exit(1);
    }
```

ConvXPbasic2LOT

Sumário

```
int ConvXPbasic2LOT(char* szFileOrig,          // Nome do arquivo origem a ser convertido
                    char* szFileDest,          // Nome do arquivo destino a ser gerado
                    struct FileFormat_t* pF    // Ponteiro para estrutura com formato do arquivo destino
                    );
```

Windows 16	Windows 32	DOS
		•

Descrição

A função 'ConvXPbasic2LOT' converte o arquivo fornecido no formato XPbasic para formato LOTUS.

O argumento 'szFileOrig' é uma cadeia de caracteres com o nome do arquivo origem.

O argumento 'szFileDest' é uma cadeia de caracteres com o nome do arquivo destino.

O argumento 'pF' é um ponteiro para uma estrutura do tipo FileFormat_t que define o formato do arquivo destino. Esta estrutura está definida em XPCOMLIB.H e contém os campos:

cFields — Número de campos do arquivo destino, deve ser entre 1 e 10.

Para cada um dos ‘cFields’ campos acima temos a estrutura ‘Field’ definida por:

```
struct  {
        cTipo       Tipo do campo XPFLD_INT, XPFLD_FLO, XPFLD_STR. Ver XPCOMDEF.H.
        cLength     Comprimento do campo
        cDec        Número de casas decimais usado para ponto flutuante na conversão de arquivos
                    formato XPbasic para formato DBASE
        } Field[10];
```

Valor Retornado

A função retorna zero quando a execução for bem sucedida, caso contrário retorna um código de erro sempre menor do que zero. O significado de cada código de erro retornado é fornecido no item “Códigos de Erro”.

Consulte

ConvASC2XPbasic, ConvDB2XPbasic, ConvLOT2XPbasic, ConvTXT2XPbasic, ConvXPbasic2DB, ConvXPbasic2ASC, ConvXPbasic2TXT.

Exemplo

```
#include <stdlib.h>
#include <string.h>
#include “xpcomlib.h”
#include “xpcomdef.h”

char szInFile[20];
char szOutFile[20];
struct FileFormat_t fF;

void main(void)
    {
    fF.cFields = 3;
    fF.Field[0].cTipo = XPFLD_STR;
    fF.Field[0].cLength = 80;
    fF.Field[0].cDec = 0;
    fF.Field[1].cTipo = XPFLD_INT;
    fF.Field[1].cLength = 2;
    fF.Field[1].cDec = 0;
    fF.Field[2].cTipo = XPFLD_FLO;
```

```
fF.Field[2].cLength = 8;
fF.Field[2].cDec = 0;
strcpy(szInFile, "C:\\TRIX\\ARQXPB.DAT");
strcpy(szOutFile, "C:\\TRIX\\ARQLOT.WK1");
if (ConvXPbasic2LOT(szInFile, szOutFile, &fF))
    exit(1);
}
```

FXPBasicConvToText

Sumário

```
int FAR PASCAL FXPBasicConvToText(const char FAR* szFileOrig,    // Nome do arquivo origem a ser convertido
                                  const char FAR* szFileDest,    // Nome do arquivo destino a ser gerado
                                  int nLineTerm                  // Especifica qual terminador de linha deve ser usado
                                  );
```

Windows 16	Windows 32	DOS
•	•	

Descrição

A função 'FXPBasicConvToText' converte o arquivo fornecido no formato XPbasic para formato TEXTO.

O argumento 'szFileOrig' é uma cadeia de caracteres com o nome do arquivo origem.

O argumento 'szFileDest' é uma cadeia de caracteres com o nome do arquivo destino.

O argumento 'nLineTerm' especifica qual terminador de linha deve ser usado. O padrão é usar CR seguido de LF, usar as constantes definidas em XPCOMxx.H:

XPLTERM_CRLF	Terminador de linha usado será caractere CR (0x0D) seguido do caractere LF (0x0A).
XPLTERM_CR	Terminador de linha usado será caractere CR (0x0D).
XPLTERM_LF	Terminador de linha usado será caractere LF (0x0A).

Valor Retornado

A função retorna zero quando a execução for bem sucedida, caso contrário retorna um código de erro sempre menor do que zero. O significado de cada código de erro retornado é fornecido no item "Códigos de Erro".

Consulte

FXPBasicConvFromAsc, FXPBasicConvFromDBF, FXPBasicConvFromText, FXPBasicConvToAsc, FXPBasicConvToDBF.

Exemplo

```
#include <windows.h>
#include <string.h>
#include "xpcom16.h"

char szInFile[20];
char szOutFile[20];

int PASCAL WinMain(HANDLE hInstance, HANDLE hPrevInstance, LPSTR lpCmdLine, int nCmdShow)
    {
    strcpy(szInFile, "C:\\TRIX\\ARQXPB.DAT");
    strcpy(szOutFile, "C:\\TRIX\\ARQTXT.DAT");
    if (FXPBasicConvToText(szInFile, szOutFile,XPLTERM_CRLF))
        return FALSE;
    return TRUE;
    }
```

ConvXPbasic2TXT

Sumário

```
int ConvXPbasic2TXT(char* szFileOrig,          // Nome do arquivo origem a ser convertido
                    char* szFileDest,          // Nome do arquivo destino a ser gerado
                    struct FileFormat_t* pF,   // Ponteiro para estrutura com formato do arquivo destino
                    int nTerminador            // Especifica qual terminador de linha deve ser usado
                   );
```

Windows 16	Windows 32	DOS
		•

Descrição

A função 'ConvXPbasic2TXT' converte o arquivo fornecido no formato XPbasic para formato TEXTO.

O argumento 'szFileOrig' é uma cadeia de caracteres com o nome do arquivo origem.

O argumento 'szFileDest' é uma cadeia de caracteres com o nome do arquivo destino.

O argumento 'pF' é um ponteiro para uma estrutura do tipo FileFormat_t que define o formato do arquivo destino. Esta estrutura está definida em XPCOMLIB.H e contém os campos:

cFields — Número de campos do arquivo destino, deve ser entre 1 e 10.

Para cada um dos ‘cFields’ campos acima temos a estrutura ‘Field’ definida por:

```
struct  {
        cTipo       Tipo do campo XPFLD_INT, XPFLD_FLO, XPFLD_STR. Ver XPCOMDEF.H.
        cLength     Comprimento do campo
        cDec        Número de casas decimais usado para ponto flutuante na conversão de arquivos
                    formato XPbasic para formato DBASE
        } Field[10];
```

O argumento ‘nTerminador’ especifica qual terminador de linha deve ser usado. O padrão é usar CR seguido de LF, usar as constantes definidas em XPCOMDEF.H:

CF_CRLF	Terminador de linha usado será caractere CR (0x0D) seguido do caractere LF (0x0A).
CF_CR	Terminador de linha usado será caractere CR (0x0D).
CF_LF	Terminador de linha usado será caractere LF (0x0A).

Valor Retornado

A função retorna zero quando a execução for bem sucedida, caso contrário retorna um código de erro sempre menor do que zero. O significado de cada código de erro retornado é fornecido no item “Códigos de Erro”.

Consulte

ConvASC2XPbasic, ConvDB2XPbasic, ConvLOT2XPbasic, ConvTXT2XPbasic, ConvXPbasic2DB, ConvXPbasic2LOT, ConvXPbasic2ASC.

Exemplo

```
#include <stdlib.h>
#include <string.h>
#include “xpcomlib.h”
#include “xpcomdef.h”

char szInFile[20];
char szOutFile[20];
struct FileFormat_t fF;

void main(void)
    {
    fF.cFields = 3;
```

```
fF.Field[0].cTipo = XPFLD_STR;
fF.Field[0].cLength = 80;
fF.Field[0].cDec = 0;
fF.Field[1].cTipo = XPFLD_INT;
fF.Field[1].cLength = 2;
fF.Field[1].cDec = 0;
fF.Field[2].cTipo = XPFLD_FLO;
fF.Field[2].cLength = 8;
fF.Field[2].cDec = 0;
strcpy(szInFile, "C:\\TRIX\\ARQXPB.DAT");
strcpy(szOutFile, "C:\\TRIX\\ARQTXT.DAT");
if (ConvXPbasic2TXT(szInFile, szOutFile, &fF , CF_CRLF))
        exit(1);
}
```

Biblioteca de Funções de Comunicação para Linguagem Clipper ®

Apresentação

A “Biblioteca de Funções de Comunicação para Linguagem Clipper ®” destina-se a aplicativos que serão executados em sistema operacional MS-DOS 5.0 ou superior, ou sistema operacional compatível para fornecer interface de comunicação com os coletores de dados TRIX. O EXOSPACE é suportado somente para canais de comunicação com interrupção de hardware abaixo de 8.

Como a “Biblioteca de Funções de Comunicação para Linguagem Clipper ®” utiliza as funções da “Biblioteca de Funções de Comunicação para Linguagem C”, ela possui as mesmas características gerais com exceção das características particulares da linguagem. Portanto informações não contidas nesta seção são fornecidas na documentação da “Biblioteca de Funções de Comunicação para Linguagem C”.

É fornecido com a “Biblioteca de Funções de Comunicação para Linguagem Clipper ®” um arquivo *header* a ser incluído nos arquivos fonte codificados em Clipper ®. Este arquivos é o 'XPCOMDEF.H', que contém a definição de constantes auxiliares para as chamadas de função da rede XPnet.

Faz-se necessário lembrar que a biblioteca fornecida utiliza recursos de interrupção de *hardware* do microcomputador, sendo portanto imprevisível o comportamento de um aplicativo que faz uso dos mesmos recursos.

Compilação e *Linkagem*

Esta biblioteca é compatível com código compilado pelo Clipper ® 5.01 ou superior e a *linkagem* deve ser feita utilizando o Rtlink. Outros produtos podem ser parcialmente compatíveis.

Os arquivos fonte codificados em Clipper ® que utilizam esta biblioteca podem ser compilados utilizando o Clipper ® 5.01 ou superior.

Um exemplo de compilação do arquivo ROTINA.PRG é o seguinte:

```
CLIPPER ROTINA.PRG
```

A “Biblioteca de Funções de Comunicação para Linguagem Clipper ®” deve ser *linkada* com os módulos de um aplicativo e com outras bibliotecas utilizando-se o Rtlink. Deve-se ressaltar que os modelos de memória utilizados pelos módulos e pelas bibliotecas devem ser os mesmos.

Um exemplo de *link-edição* do arquivo ROTINA.PRG é o seguinte:

```
RTLINK  @L.RSP
```

O conteúdo de L.RSP é dado logo abaixo

```
/NOE
FILE    ROTINA
LIB     CLIPPER , EXTEND , XPCOMCLI
```

Programa de demontração DEMOXTM

O programa exemplo DEMOXTM fornecido com a biblioteca é um projeto, desenvolvido em Clipper 5.20, utilizando praticamente todas as funções da biblioteca, permite a transmissão e recepção de mensagens dos terminais, ativação ou desativação de polling, transferência de arquivos ou programa e atualização de data e hora dos terminais. O programa DEMOXTM trabalha em conjunto com os programas DEMOXTM.BAS (para coletores da linha XTM e Smart-Block) e DEMOCD10.BAS (para coletores da linha CD10 e JOBBY), escritos em XPBASIC, que devem ser transferidos para os coletores.

Este programa demostra a utilização de Coletores de Dados da linha XTM e Smart-Block da TRIX em aplicacoes ON-LINE. Neste tipo de aplicação o PC passa a ser um Servidor de Coletores e fica se comunicando constantemente com eles, esperando requisições (transações) deles.

Neste demonstrativo, existem apenas dois tipos de transações:

Transação de Coleta, na qual um conjunto de caracteres proveniente de qualquer coletor da rede é enviado para o PC que simplesmente armazena os dados recebidos em um arquivo tipo DBF e retorna para o coletor uma indicação de que os dados foram gravados.

Transacao de Consulta, na qual o coletor envia um código de produto para o PC, que por sua vez consulta um banco de dados utilizando este código como chave. Caso o código seja encontrado, é enviado para o coletor de origem da transação, a descrição e a quantidade do produto correspondente ao código. Caso o código não seja encontrado é enviada uma indicação para o coletor indicando este problema.

O Programa DEMOXTM.BAS, desenvolvido em XPbasic, foi feito para os Coletores da linha XTM e Smart-Block e apresenta um menu que permite a execução dos dois tipos de transações descritos anteriormente.

OPERACAO DO DEMONSTRATIVO

Apos executar o demonstrativo, selecionar usuário através do menu de Operações (Alt-O) opção Seleção de Usuário (U) e selecionar usuário TRIX com senha TRIX.

Executar o Menu de Configurações (Alt-C) opção Ambiente de Hardware (H) e configurar o endereço, interrupção e taxa do canal de comunição a ser utilizado para a Porta 1. Por exemplo:

Se você utilizar o COM1, fornecer:
endereço = 3F8
interrupção = 4
taxa = 9600
intervalo = 1

Se você utilizar o COM2, fornecer:
endereço = 2F8
interrupção = 3
taxa = 9600
intervalo = 1

Se você utilizar a placa MOS-485 da TRIX, fornecer:
endereço = 238 (ou outra, conforme estiver configurado na placa)
interrupção = 10 (ou outra, conforme estiver configurado na placa)
taxa = 9600
intervalo = 1
Verificar se a placa MOS-485 está configurada como neste exemplo.

Após preencher os campos de configuração do canal, pressionar PgDn ou diversos Enters até passar por todos os campos e a janela de configuração ser fechada. Esta configuracao será então salva em disco.

Após a configuração do Ambiente de Hardware execute a opção Coletores para cadastrar os Coletores de Dados ligados ao Servidor. Para cada coletor ligado (se você utilizar interface serial RS-232 apenas um coletor pode ser ligado simultaneamente) indicar que está ativo e o nome do programa em XPBASIC (DEMOXTM).

Para ativar o processo de comunicação deve ser executado o menu de Operações (Alt-O) seguido da opção de Modo Ativo. Verifique através do menu de Comunicações, opção de Estatísticas de Comunicação se a comunicação com os coletores está operacional.

Em seguida, transmitir o programa para os coletores através de opção do Menu de Comunicações. Após estas etapas você pode experimentar executando as opções de Coleta e Consulta do menu do Coletor de Dados.

COMO REGERAR O SISTEMA

Este projeto é fornecido completo com todos os fontes para servir de modelo a aplicações semelhantes.

Executar o arquivo DEMOZIP.EXE e ele expandirá automaticamente todos os fontes e o programa executável que pode ser usado para testar a comunicação com coletores da TRIX.

No módulo TRANSAC.PRG está a recepção das mensagens do coletor e a resposta do PC. Nele são executadas todas as transações entre o coletor e o PC. Em seu arquivo de inclusão TRANSAC.CH estão definidas as constantes utilizadas para especificar o tamanho e posição dos campos nas mensagens.

Toda consulta a banco de dados do PC está sendo feita no módulo CONSULTA.PRG que deve ser alterado para que uma outra base de dados posssa ser acessada.

As principais funções de comunicação com os coletores estão nos módulos:

XPNET.PRG	Controla operação dos coletores no modo on-line.
XPREMOTE.PRG	Muda o estado do coletor de remoto para local e vice versa para operação off-line.
RXDATA.PRG	Recolhe arquivos do coletor.
TXDATA.PRG	Envia arquivos para coletor.
STATS.PRG	Mostra estatística de comunicação.
ERROR.PRG	Fecha canais na saída por erro.

Para compilação e lincagem automática é fornecido o arquivo DEMOXTM.RMK para ser utilizado com o RMAKE.EXE do Clipper 5.20 e o arquivo MAKEFILE para ser usado com o MAKE ou NMAKE.EXE da Microsoft.

Alterar os arquivos RMAKE.BAT, DEMOXTM.RMK e MAKEFILE que fazem parte do projeto DEMOXTM para que o caminho do compilador e do lincador estejam corretos. Criar o subdiretório OBJ dentro do diretório DEMOXTM. Executar RMAKE.BAT, o NMAKE ou MAKE para compilar e lincar todos os módulos.

Funções por categoria

A descrição das funções da “Biblioteca de Funções de Comunicação para Linguagem Clipper ®” é feita utilizando o seguinte formato:

- nome da função em negrito;
- sumário mostrando o modelo sintático a ser utilizado;
- descrição contendo as ações da mesma e argumentos de entrada;
- o valor retornado pela função e sua descrição;
- sugestão de consulta a funções similares ou relacionadas;
- exemplo de utilização da função descrita.

Como a biblioteca fornecida utiliza recursos de interrupção de *hardware* do microcomputador, se alguma ocorrência causar o término do aplicativo antes da finalização operacional da rede, o sistema terá um comportamento errático e imprevisível, sendo necessário desligá-lo. Dessa forma, faz-se necessário que o aplicativo não permita intervenções para seu término através da digitação de “CTRL-BREAK” ou “CTRL-C”.

É necessário também que o aplicativo não permita ser interrompido através de erros críticos do DOS, como por exemplo o erro gerado na leitura de um disquete quando a porta de seu *drive* estiver aberta, o que pode ser feito interceptando o *handler* das interrupções que geram esses erros.

Não é recomendável que o aplicativo execute “shells” do DOS permitindo a execução de outros aplicativos com a rede em operação. Aconselha-se neste caso finalizar operacionalmente a rede antes da execução do *shell* e ao seu término reinicializá-la com os argumentos originais.

Funções Básicas de Comunicação

Estas funções são destinadas à inicialização, transferência de dados, verificação da comunicação e finalização através do canal selecionado. São utilizadas pelas funções que implementam o protocolo XPnet e podem ser utilizadas para implementar outros protocolos de comunicação.

Com_Addr

Sumário

Com_Addr (nCanal, wPort, nIrq)

Descrição

A função 'Com_Addr' associa um número de canal ao endereço de hardware do canal de comunicação e à interrupção associada a ele. Antes desta associação é feito um teste para verificar que no endereço de hardware especificado está instalado um canal de comunicação. Todas as demais funções de comunicação recebem o número do canal como parâmetro. Os canais COM1, COM2, COM3 e COM4 já estão previamente definidos com os endereços padrões do IBM-PC, mas podem ser reconfigurados se desejado.

O argumento 'nCanal' especifica o canal de comunicação, possibilita o uso de no máximo 16 canais (0 a 15), cada qual permitindo conectar até 32 terminais. As constantes, já definidas para cada canal, são apresentadas abaixo:

COM1, COM2, COM3, COM4, . . . , COM16.

O argumento 'wPort' define o *port* base de “I/O” da “UART” da placa de comunicação utilizada, sendo alguns dos valores possíveis para o argumento mostrados abaixo:

552 para *port* 0228H,
568 para *port* 0238H,
616 para *port* 0268H,
632 para *port* 0278H (PRINTER),
680 para *port* 02A8H,
696 para *port* 02B8H,
744 para *port* 02E8H,
760 para *port* 02F8H (COM2),
808 para *port* 0328H,
824 para *port* 0338H,
872 para *port* 0368H,
888 para *port* 0378H (PRINTER),
936 para *port* 03A8H,
952 para *port* 03B8H (MONOCHROME ADAPTER),
1000 para *port* 03E8H,
1016 para *port* 03F8H (COM1).

O argumento 'nIrq' define o nível de interrupção da placa de comunicação utilizada, sendo alguns dos valores possíveis para o argumento:

2,
3 (COM2),
4 (COM1),
5 (LPT2),
6 (FLOPPY DISK),
7 (LPT1),
10,
11,
12,
15.

Valor Retornado

A função retorna zero quando a execução for bem sucedida, caso contrário retorna um código de erro sempre menor do que zero. O significado de cada código de erro retornado é fornecido no item “Códigos de Erro”.

Consulte

Com_Open, Com_Close.

Exemplo

```
#include “xpcomdef.h”

nError:= 0

nError:= Com_Addr(COM1,568,3)
if nError <> 0
      return 1
endif
nError:= Com_Open(COM1,9600,8,1,COM_NONEP,COM_NOFLOW,COM_NOFLOW,256,256, UART8250)
if nError <> 0
      return 1
endif
nError:= Com_Close(COM1)
if nError <> 0
      return 1
endif
quit
```

Com_Test

Sumário

Com_Test (nCanal)

Descrição

A função ‘Com_Test’ testa o *hardware* do canal de comunicação e identifica o tipo de UART. O canal não pode estar aberto.

O argumento 'nCanal' especifica o canal de comunicação, possibilita o uso de no máximo 16 canais (COM1 a COM16).

Valor Retornado

A função retorna o tipo de UART, conforme definido no arquivo *header* ‘XPCOMDEF.H’ quando sua execução for bem sucedida:

UART8250:	UART 8250;
UART16450:	UART 16450 com scratchpad;
UART16550:	UART 16550 com FIFOs;

Caso contrário retorna um código de erro sempre menor do que zero. O significado de cada código de erro retornado é fornecido no item “Códigos de Erro”.

Consulte

Com_Addr.

Exemplo

```
#include “xpcomdef.h”

nError:= 0
nUart:= 0

nError:= Com_Addr(COM1,568,3)
if nError <> 0
    return 1
endif
nUart:= Com_Test(COM1)
do case
    case nUart = UART8250
        ? “UART 8250 instalada.”
    case nUart = UART16450
        ? “UART 16450 instalada.”
    case nUart = UART16550
        ? “UART 16550 instalada.”
    otherwise
        return 1
endcase
quit
```

Com_Open

Sumário

Com_Open (nCanal, nBaud, nBits, nStop, nParid, nTxFlow, nRxFlow, wTxBuffer, wRxBuffer, nUART)

Descrição

A função 'Com_Open' abre o canal de comunicação, definindo os parâmetros para sua operação. Esta função deve ser executada após 'Com_Addr' que inicializa a rede para uma configuração de *hardware* específica.

O argumento 'nCanal' especifica o canal de comunicação, possibilita o uso de no máximo 16 canais (COM1 a COM16).

O argumento 'nBaud' define a taxa de comunicação a ser utilizada com 8 *bits*, 1 *stop-bit* e sem paridade, sendo os valores possíveis para o argumento mostrados abaixo em *baud*:

600, 1200, 2400, 4800, 9600, 19200.

O argumento ‘nBits’ informa o número de *bits* por caractere: 7 ou 8.

O argumento ‘nStop’ informa o número de *stop-bits* 1 ou 2.

O argumento ‘nParid’ informa o tipo de paridade, conforme as constantes definidas em ‘XPCOMDEF.H’:

COM_NONEP	- Sem paridade,
COM_EVENP	- Paridade par,
COM_ODDP	- Paridade ímpar.

O argumento ‘nTxFlow’ informa o tipo de controle de fluxo na transmissão, conforme as constantes definidas em ‘XPCOMDEF.H’:

COM_NOFLOW	- Sem controle de fluxo,
COM_RTSCTS	- Controle de fluxo via RTS/CTS,
COM_XONXOFF	- Controle de fluxo via XON/XOFF.

O argumento ‘nRxFlow’ informa o tipo de controle de fluxo na recepção, conforme as constantes definidas em ‘XPCOMDEF.H’:

COM_NOFLOW	- Sem controle de fluxo,
COM_RTSCTS	- Controle de fluxo via RTS/CTS,
COM_XONXOFF	- Controle de fluxo via XON/XOFF.

O argumento 'wTxBuffer' define o espaço a ser alocado para o *buffer* de transmissão de mensagens, podendo ser um valor de 16 *bytes* a 65500 *bytes*.

O argumento 'wRxBuffer' define o espaço a ser alocado para o *buffer* de recepção de mensagens, podendo ser um valor de 16 *bytes* a 65500 *bytes*.

O argumento 'nUART' pode ser UART8250, UART16450 ou UART16550, se for este último valor especifica que iremos usar o FIFO da UART 16550 caso seja detectada no canal de comunicação.

Valor Retornado

A função retorna zero quando sua execução for bem sucedida, caso contrário retorna um código de erro sempre menor do que zero. O significado de cada código de erro retornado é fornecido no item “Códigos de Erro”.

Consulte

Com_Addr, Com_Close, Com_Config.

Exemplo

```
#include “xpcomdef.h”

nError:= 0

nError:= Com_Addr(COM1,568,3)
if nError <> 0
      return 1
endif
nError:= Com_Open(COM1,9600,8,1,COM_NONEP,COM_NOFLOW,COM_NOFLOW,256,256, UART8250)
if nError <> 0
      return 1
endif
nError:= Com_Close(COM1)
if nError <> 0
      return 1
```

```
endif
quit
```

Com_Config

Sumário

Com_Config (nCanal, nBaud, nBits, nStop, nParid, nTxFlow, nRxFlow)

Descrição

A função ’Com_Config’ reconfigura um canal de comunicação já aberto, definindo os parâmetros para sua operação. Esta função deve ser executada quando se deseja alterar a configuração previamente estabelecida para um determinado canal de comunuicação.

O argumento 'nCanal' especifica o canal de comunicação, possibilita o uso de no máximo 16 canais (COM1 a COM16).

O argumento 'nBaud' define a taxa de comunicação a ser utilizada com 8 *bits*, 1 *stop-bit* e sem paridade, sendo os valores possíveis para o argumento mostrados abaixo em *baud*:

600, 1200, 2400, 4800, 9600, 19200.

O argumento ‘nBits’ informa o número de *bits* por caractere: 7 ou 8.

O argumento ‘nStop’ informa o número de *stop-bits* 1 ou 2.

O argumento ‘nParid’ informa o tipo de paridade, conforme as constantes definidas em ‘XPCOMDEF.H’:

COM_NONEP	- Sem paridade,
COM_EVENP	- Paridade par,
COM_ODDP	- Paridade ímpar.

O argumento ‘nTxFlow’ informa o tipo de controle de fluxo na transmissão, conforme as constantes definidas em ‘XPCOMDEF.H’:

COM_NOFLOW	- Sem controle de fluxo,
COM_RTSCTS	- Controle de fluxo via RTS/CTS,
COM_XONXOFF	- Controle de fluxo via XON/XOFF.

O argumento ‘nRxFlow’ informa o tipo de controle de fluxo na recepção, conforme as constantes definidas em ‘XPCOMDEF.H’:

COM_NOFLOW	- Sem controle de fluxo,
COM_RTSCTS	- Controle de fluxo via RTS/CTS,
COM_XONXOFF	- Controle de fluxo via XON/XOFF.

Valor Retornado

A função retorna zero quando sua execução for bem sucedida, caso contrário retorna um código de erro sempre menor do que zero. O significado de cada código de erro retornado é fornecido no item “Códigos de Erro”.

Consulte

Com_Addr, Com_Open.

Exemplo

```
#include “xpcomdef.h”

nError:= 0

nError:= Com_Addr(COM1,568,3)
```

```
if nError <> 0
        return 1
endif
nError:= Com_Open(COM1,9600,8,1,COM_NONEP,COM_NOFLOW,COM_NOFLOW,256,256, UART8250)
if nError <> 0
        return 1
endif
nError:= Com_Config(COM1,2400,8,1,COM_NONEP,COM_NOFLOW,COM_NOFLOW)
if nError <> 0
        return 1
endif
nError:= Com_Close(COM1)
if nError <> 0
        return 1
endif
quit
```

Com_Close

Sumário

Com_Close (nCanal)

Descrição

A função ‘Com_Close’ fecha o canal de comunicação aberto anteriormente.

O argumento 'nCanal' especifica o canal de comunicação, possibilita o uso de no máximo 16 canais (COM1 a COM16).

Valor Retornado

A função retorna zero quando sua execução for bem sucedida, caso contrário retorna um código de erro sempre menor do que zero. O significado de cada código de erro retornado é fornecido no item “Códigos de Erro”.

Consulte

Com_Addr, Com_Open.

Exemplo

```
#include “xpcomdef.h”

nError:= 0

nError:= Com_Addr(COM1,568,3)
if nError <> 0
        return 1
endif
nError:= Com_Open(COM1,9600,8,1,COM_NONEP,COM_NOFLOW,COM_NOFLOW,256,256, UART8250)
if nError <> 0
        return 1
endif
nError:= Com_Close(COM1)
if nError <> 0
        return 1
endif
quit
```

Com_GetCfg

Sumário

Com_GetCfg (nCanal, nInfo)

Descrição

A função ‘Com_GetCfg’ é utilizada para se obter informações diversas sobre a configuração do canal

O argumento 'nCanal' especifica o canal de comunicação, possibilita o uso de no máximo 16 canais (COM1 a COM16).

O argumento 'nInfo' determina a informação desejada, é definido conforme as constantes abaixo, que já estão declaradas no arquivo *header* ‘XPCOMDEF.H’ da biblioteca:

COMCFG_RATE:	Informa o valor da taxa de comunicação;
COMCFG_NBITS:	Informa o número de *bits* por caractere 7 ou 8;
COMCFG_NSTOP:	Informa o número de *stop bits* 1 ou 2;
COMCFG_PAR:	Informa o tipo de paridade;
COMCFG_TXFLOW:	Informa o tipo do controle de fluxo na transmissão;
COMCFG_RXFLOW:	Informa o tipo do controle de fluxo na recepção;
COMCFG_BUFRX:	Informa o tamanho do *buffer* de recepção;
COMCFG_BUFTX:	Informa o tamanho do *buffer* de transmissão;
COMCFG_UART:	Informa o tipo de UART detectada;

Valor Retornado

O valor retornado pela função varia de acordo com o especificado no argumento 'nInfo', relacionados abaixo:

COMCFG_RATE:	Retorna o valor da taxa de comunicação, caso contrário retorna um código de erro;
COMCFG_NBITS:	Retorna o número de *bits* por caractere 7 ou 8, caso contrário retorna um código de erro;
COMCFG_NSTOP:	Retorna o número de *stop bits* 1 ou 2, caso contrário retorna um código de erro;
COMCFG_PAR:	Retorna o tipo de paridade conforme as constantes definidas em ‘XPCOMDEF.H’: COM_NONEP - Sem paridade, COM_EVENP - Paridade par, COM_ODDP - Paridade ímpar, Caso contrário retorna um código de erro;
COMCFG_TXFLOW:	Retorna o tipo do controle de fluxo na transmissão conforme as constantes definidas em ‘XPCOMDEF.H’: COM_NOFLOW - Sem controle de fluxo, COM_RTSCTS - Controle de fluxo via RTS/CTS, COM_XONXOFF - Controle de fluxo via XON/XOFF, Caso contrário retorna um código de erro;
COMCFG_RXFLOW:	Retorna o tipo do controle de fluxo na recepção conforme as constantes definidas em ‘XPCOMDEF.H’: COM_NOFLOW - Sem controle de fluxo, COM_RTSCTS - Controle de fluxo via RTS/CTS, COM_XONXOFF - Controle de fluxo via XON/XOFF, Caso contrário retorna um código de erro;
COMCFG_BUFRX:	Retorna o tamanho do *buffer* de recepção, caso contrário retorna um código de erro;
COMCFG_BUFTX:	Retorna o tamanho do *buffer* de transmissão, caso contrário retorna um código de erro;
COMCFG_UART:	Retorna o tipo de UART detectada conforme as constantes definidas em ‘XPCOMDEF.H’: UART8250: UART 8250, UART16450: UART 16450 com scratchpad, UART16550: UART 16550 com FIFOs, Caso contrário retorna um código de erro;

O valor do código de erro retornado é sempre menor do que zero. Seu significado é fornecido no item “Códigos de Erro”.

Consulte

Com_Test, Com_Open, Com_Config.

Exemplo

```
#include “xpcomdef.h”

lInfo:= 0
```

```
nError:= 0

nError:= Com_Addr(COM1,568,3)
if nError <> 0
        return 1
endif
nError:= Com_Open(COM1,9600,8,1,COM_NONEP,COM_NOFLOW,COM_NOFLOW,256,256, UART8250)
if nError <> 0
        return 1
endif
lInfo:= Com_GetCfg(COM1,COMCFG_RXFLOW)
do case
        case lInfo = COM_NOFLOW
                ? "Sem controle de fluxo."
        case lInfo = COM_RTSCTS
                ? "Controle de fluxo via RTS/CTS."
        case lInfo = COM_XONXOFF
                ? "Controle de fluxo via XON/XOFF."
        otherwise
                return 1
endcase
lInfo:= Com_GetCfg(COM1,COMCFG_RATE)
if lInfo < 0
        return 1
else
        ? "Taxa de transmissao: ", lInfo
endif
nError:= Com_Close(COM1)
if nError <> 0
        return 1
endif
quit
```

Com_GetnRx

Sumário

Com_GetnRx (nCanal)

Descrição

A função Com_GetnRx é utilizada para se obter o número de *bytes* no *buffer* de recepção.

O argumento 'nCanal' especifica o canal de comunicação, possibilita o uso de no máximo 16 canais (COM1 a COM16).

Valor Retornado

A função retorna o número de *bytes* no *buffer* de recepção quando sua execução for bem sucedida, caso contrário retorna um código de erro sempre menor do que zero. O significado de cada código de erro retornado é fornecido no item "Códigos de Erro".

Consulte

Com_GetnTx, Com_GetfRx, Com_GetfTx.

Exemplo

```
#include "xpcomdef.h"

lInfo:= 0
nError:= 0
```

```
nError:= Com_Addr(COM1,568,3)
if nError <> 0
        return 1
endif
nError:= Com_Open(COM1,9600,8,1,COM_NONEP,COM_NOFLOW,COM_NOFLOW,256,256, UART8250)
if nError <> 0
        return 1
endif
lInfo:= Com_GetnRx(COM1)
if lInfo < 0
        return 1
else
        ? "Numero de bytes no buffer de recepcao: ", lInfo
endif
nError:= Com_Close(COM1)
if nError <> 0
        return 1
endif
quit
```

Com_GetnTx

Sumário

Com_GetnTx (nCanal)

Descrição

A função Com_GetnTx é utilizada para se obter o número de *bytes* no *buffer* de transmissão

O argumento 'nCanal' especifica o canal de comunicação, possibilita o uso de no máximo 16 canais (COM1 a COM16).

Valor Retornado

A função retorna o número de *bytes* no *buffer* de transmissão quando sua execução for bem sucedida, caso contrário retorna um código de erro sempre menor do que zero. O significado de cada código de erro retornado é fornecido no item "Códigos de Erro".

Consulte

Com_GetnRx, Com_GetfRx, Com_GetfTx.

Exemplo

```
#include "xpcomdef.h"

lInfo:= 0
nError:= 0

nError:= Com_Addr(COM1,568,3)
if nError <> 0
        return 1
endif
nError:= Com_Open(COM1,9600,8,1,COM_NONEP,COM_NOFLOW,COM_NOFLOW,256,256, UART8250)
if nError <> 0
        return 1
endif
lInfo:= Com_GetnTx(COM1)
if lInfo < 0
        return 1
else
        ? "Numero de bytes no buffer de transmissao: ", lInfo
```

```
endif
nError:= Com_Close(COM1)
if nError <> 0
    return 1
endif
quit
```

Com_GetfRx

Sumário

Com_GetfRx (nCanal)

Descrição

A função Com_GetfRx é utilizada para se obter o número de *bytes* vagos no *buffer* de recepção.

O argumento 'nCanal' especifica o canal de comunicação, possibilita o uso de no máximo 16 canais (COM1 a COM16).

Valor Retornado

A função retorna o número de *bytes* vagos no *buffer* de recepção quando sua execução for bem sucedida, caso contrário retorna um código de erro sempre menor do que zero. O significado de cada código de erro retornado é fornecido no item “Códigos de Erro”.

Consulte

Com_GetnRx, Com_GetnTx, Com_GetfTx.

Exemplo

```
#include “xpcomdef.h”

lInfo:= 0
nError:= 0

nError:= Com_Addr(COM1,568,3)
if nError <> 0
    return 1
endif
nError:= Com_Open(COM1,9600,8,1,COM_NONEP,COM_NOFLOW,COM_NOFLOW,256,256, UART8250)
if nError <> 0
    return 1
endif
lInfo:= Com_GetfRx(COM1)
if lInfo < 0
    return 1
endif
    ? “Numero de bytes vagos no buffer de recepcao: ”, lInfo
nError:= Com_Close(COM1)
if nError <> 0
    return 1
endif
quit
```

Com_GetfTx

Sumário

Com_GetfTx (nCanal)

Descrição

A função Com_GetnTx é utilizada para se obter o número de *bytes* vagos no *buffer* de transmissão

O argumento 'nCanal' especifica o canal de comunicação, possibilita o uso de no máximo 16 canais (COM1 a COM16).

Valor Retornado

A função retorna o número de *bytes* vagos no *buffer* de transmissão quando sua execução for bem sucedida, caso contrário retorna um código de erro sempre menor do que zero. O significado de cada código de erro retornado é fornecido no item "Códigos de Erro".

Consulte

Com_GetnRx, Com_GetnTx, Com_GetfRx.

Exemplo

```
#include "xpcomdef.h"

lInfo:= 0
nError:= 0

nError:= Com_Addr(COM1,568,3)
if nError <> 0
      return 1
endif
nError:= Com_Open(COM1,9600,8,1,COM_NONEP,COM_NOFLOW,COM_NOFLOW,256,256, UART8250)
if nError <> 0
      return 1
endif
lInfo:= Com_GetfTx(COM1)
if lInfo < 0
      return 1
else
      ? "Numero de bytes vagos no buffer de transmissao: ", lInfo
endif
nError:= Com_Close(COM1)
if nError <> 0
      return 1
endif
quit
```

Com_GetErr

Sumário

Com_GetErr (nCanal, nTipoErr)

Descrição

A função 'Com_GetErr' retorna COM_ON ou COM_OFF indicando se o erro de comunicação especificado ocorreu ou não no canal de comunicação.

O argumento 'nCanal' especifica o canal de comunicação, possibilita o uso de no máximo 16 canais (COM1 a COM16).

O argumento 'nTipoErr' especifica o tipo do erro que se deseja saber a ocorrência ou não no canal de comunicação. Os erros possíveis estão definidos em XPCOMDEF.H como 'Máscaras de Erros de Comunicação':

COM_OVERUN:	Erro de overrun na recepção,
COM_PARITY:	Erro de paridade na recepção,

COM_FRAME:	Erro de framing na recepção,
COM_BREAK:	Detecção de break,
COM_RXBFUL:	Erro de *buffer* de recepção cheio.

A função quando bem sucedida retorna COM_ON se houve a ocorrência do erro de comunicação especificado e COM_OFF se o erro não ocorreu. Caso contrário retorna um código de erro sempre menor do que zero. O significado de cada código de erro retornado é fornecido no item “Códigos de Erro”.

Consulte

Com_ClrErr.

Exemplo

```
#include “xpcomdef.h”

nError:= 0

nError:= Com_Addr(COM1,568,3)
if nError <> 0
      return 1
endif
nError:= Com_Open(COM1,9600,8,1,COM_NONEP,COM_NOFLOW,COM_NOFLOW,256,256, UART8250)
if nError <> 0
      return 1
endif
nError:= ComGetErr(COM1,COM_PARITY)
if nError < 0
      return 1
endif
if nError = COM_ON
      ? “Houve erro de paridade na recepcao.”
if nError = COM_OFF
      ? “Nao houve erro de paridade na recepcao.”
nError:= Com_Close(COM1)
if nError <> 0
      return 1
endif
quit
```

Com_ClrErr

Sumário

Com_ClrErr (nCanal)

Descrição

A função ‘Com_ClrErr’ limpa o *byte* que guarda os erros de comunicacão ocorridos no canal de comunicação especificado.

O argumento 'nCanal' especifica o canal de comunicação, possibilita o uso de no máximo 16 canais (COM1 a COM16).

Valor Retornado

A função retorna zero quando sua execução for bem sucedida, caso contrário retorna um código de erro sempre menor do que zero. O significado de cada código de erro retornado é fornecido no item

Consulte

Com_GetErr.

Exemplo

```
#include “xpcomdef.h”

nError:= 0

nError:= Com_Addr(COM1,568,3)
if nError <> 0
	return 1
endif
nError:= Com_Open(COM1,9600,8,1,COM_NONEP,COM_NOFLOW,COM_NOFLOW,256,256, UART8250)
if nError <> 0
	return 1
endif
nError:= Com_ClrErr(COM1)
if nError <> 0
	return 1
endif
nError:= Com_Close(COM1)
if nError <> 0
	return 1
endif
quit
```

Com_ClrRx

Sumário

Com_ClrRx (nCanal)

Descrição

A função ‘Com_ClrRx’ limpa o *buffer* de recepção do canal de comunicação especificado.

O argumento 'nCanal' especifica o canal de comunicação, possibilita o uso de no máximo 16 canais (COM1 a COM16).

Valor Retornado

A função retorna zero quando sua execução for bem sucedida, caso contrário retorna um código de erro sempre menor do que zero. O significado de cada código de erro retornado é fornecido no item “Códigos de Erro”.

Consulte

Com_ClrTx, Com_ClrTR.

Exemplo

```
#include “xpcomdef.h”

nError:= 0

nError:= Com_Addr(COM1,568,3)
if nError <> 0
	return 1
endif
nError:= Com_Open(COM1,9600,8,1,COM_NONEP,COM_NOFLOW,COM_NOFLOW,256,256, UART8250)
if nError <> 0
	return 1
endif
nError:= Com_ClrRx(COM1)
if nError <> 0
```

```
        return 1
endif
nError:= Com_Close(COM1)
if nError <> 0
        return 1
endif
quit
```

Com_ClrTx

Sumário

Com_ClrTx (nCanal)

Descrição

A função ‘Com_ClrTx’ limpa o *buffer* de transmissão do canal de comunicação especificado.

O argumento 'nCanal' especifica o canal de comunicação, possibilita o uso de no máximo 16 canais (COM1 a COM16).

Valor Retornado

A função retorna zero quando sua execução for bem sucedida, caso contrário retorna um código de erro sempre menor do que zero. O significado de cada código de erro retornado é fornecido no item “Códigos de Erro”.

Consulte

Com_ClrRx, Com_ClrTR.

Exemplo

```
#include “xpcomdef.h”

nError:= 0

nError:= Com_Addr(COM1,568,3)
if nError <> 0
        return 1
endif
nError:= Com_Open(COM1,9600,8,1,COM_NONEP,COM_NOFLOW,COM_NOFLOW,256,256, UART8250)
if nError <> 0
        return 1
endif
nError:= Com_ClrTx(COM1)
if nError <> 0
        return 1
endif
nError:= Com_Close(COM1)
if nError <> 0
        return 1
endif
quit
```

Com_ClrTR

Sumário

Com_ClrTR (nCanal)

Descrição

A função ‘Com_ClrTR’ limpa tanto o *buffer* de transmissão quanto o de recepção para o canal de comunicação especificado.

O argumento 'nCanal' especifica o canal de comunicação, possibilita o uso de no máximo 16 canais (COM1 a COM16).

Valor Retornado

A função retorna zero quando sua execução for bem sucedida, caso contrário retorna um código de erro sempre menor do que zero. O significado de cada código de erro retornado é fornecido no item “Códigos de Erro”.

Consulte

Com_ClrTx, Com_ClrRx.

Exemplo

```
#include “xpcomdef.h”

nError:= 0

nError:= Com_Addr(COM1,568,3)
if nError <> 0
	return 1
endif
nError:= Com_Open(COM1,9600,8,1,COM_NONEP,COM_NOFLOW,COM_NOFLOW,256,256, UART8250)
if nError <> 0
	return 1
endif
nError:= Com_ClrTR(COM1)
if nError <> 0
	return 1
endif
nError:= Com_Close(COM1)
if nError <> 0
	return 1
endif
quit
```

Com_RxChar

Sumário

Com_RxChar (nCanal)

Descrição

A função ‘Com_RxChar’ retira um caractere do *buffer* de recepção do canal especificado.

O argumento 'nCanal' especifica o canal de comunicação, possibilita o uso de no máximo 16 canais (COM1 a COM16).

Valor Retornado

A função retorna o código do caractere retirado do *buffer* quando sua execução for bem sucedida, caso contrário retorna um código de erro sempre menor do que zero. O significado de cada código de erro retornado é fornecido no item “Códigos de Erro”.

Consulte

Com_RxData, Com_TxData.

Exemplo

```
#include “xpcomdef.h”

nBytes:= 0
nChar:= 0
nError:= 0

nError:= Com_Addr(COM1,568,3)
if nError <> 0
        return 1
endif
nError:= Com_Open(COM1,9600,8,1,COM_NONEP,COM_NOFLOW,COM_NOFLOW,256,256, UART8250)
if nError <> 0
        return 1
endif
nBytes:= Com_GetnRx(COM1)
if nBytes < 0
        return 1
endif
if nBytes = 0
        ? “Nao existem dados disponiveis”
else
        do while (nBytes > 0)
                nChar:= Com_RxChar(COM1)
                if nChar < 0
                        return 1
                endif
                ? “Caractere recebido: ”, nChar
                nBytes:= nBytes - 1
        enddo
endif
nError:= Com_Close(COM1)
if nError <> 0
        return 1
endif
quit
```

Com_RxData

Sumário

Com_RxData (nCanal, @cBuffer, wBytes)

Descrição

A função ‘Com_RxData’ retira um ou mais *bytes* do *buffer* de recepção do canal especificado e coloca no *buffer* de destino apontado por ‘cBuffer’.

O argumento 'nCanal' especifica o canal de comunicação, possibilita o uso de no máximo 16 canais (COM1 a COM16).

O argumento tipo caractere ‘cBuffer’ é passado por referência, para retirar a mensagem que está armazenada no *buffer* de recepção.

O argumento ‘wBytes’ informa o número de caracteres a serem retirados do *buffer* de recepção e guardados no *buffer* de destino ‘cBuffer’.

Valor Retornado

A função retorna zero quando sua execução for bem sucedida, caso contrário retorna um código de erro sempre menor do que zero. O significado de cada código de erro retornado é fornecido no item “Códigos de Erro”.

Consulte

Com_RxChar, Com_TxData.

Exemplo

```
#include “xpcomdef.h”

wBytes:= 0
cBuffer:= ““
nError:= 0

nError:= Com_Addr(COM1,568,3)
if nError <> 0
	return 1
endif
nError:= Com_Open(COM1,9600,8,1,COM_NONEP,COM_NOFLOW,COM_NOFLOW,256,256, UART8250)
if nError <> 0
	return 1
endif
wBytes:= Com_GetnRx(COM1)
if wBytes < 0
	return 1
endif
if wBytes = 0
	? “Nao existem dados disponiveis.”
else
	nError:= Com_RxData(COM1, @cBuffer, wBytes)
	if nError <> 0
		return 1
	endif
endif
nError:= Com_Close(COM1)
if nError <> 0
	return 1
endif
quit
```

Com_TxData

Sumário

Com_TxData (nCanal, cBuffer)

Descrição

A função ‘Com_TxData’ transmite a mensagem em ‘cBuffer’ para o canal de comunicação especificado.

O argumento 'nCanal' especifica o canal de comunicação, possibilita o uso de no máximo 16 canais (COM1 a COM16).

O argumento tipo caractere ‘cBuffer’ contém a mensagem a ser transmitida.

Valor Retornado

A função retorna zero quando sua execução for bem sucedida, caso contrário retorna um código de erro sempre menor do que zero. O significado de cada código de erro retornado é fornecido no item “Códigos de Erro”.

Consulte

Com_RxChar, Com_RxData.

Exemplo

```
#include “xpcomdef.h”

nError:= 0

nError:= Com_Addr(COM1,568,3)
if nError <> 0
        return 1
endif
nError:= Com_Open(COM1,9600,8,1,COM_NONEP,COM_NOFLOW,COM_NOFLOW,256,256, UART8250)
if nError <> 0
        return 1
endif
nError:= Com_TxData(COM1,”Teste”)
if nError <> 0
        return 1
endif
nError:= Com_Close(COM1)
if nError <> 0
        return 1
endif
quit
```

Com_GetTxE

Sumário

Com_GetTxE (nCanal)

Descrição

A função ‘Com_GetTxE’ verifica o estado dos dois registros da UART, informando se ambos estão vazios ou se ao menos um deles tem dados, para o canal especificado.

O argumento 'nCanal' especifica o canal de comunicação, possibilita o uso de no máximo 16 canais (COM1 a COM16).

Valor Retornado

A função retorna o estado da transmissão quando sua execução for bem sucedida, caso contrário retorna um código de erro sempre menor do que zero. O significado de cada código de erro retornado é fornecido no item “Códigos de Erro”.

Os possíveis estados da transmissão são:

COM_ON:	transmissão ativa,
COM_OFF:	transmissão inativa.

Consulte

Com_GetCTS, Com_GetDCD, Com_GetDSR, Com_GetRI, Com_GetDTR, Com_GetRTS.

Exemplo

```
#include “xpcomdef.h”

nEstado:= 0
nError:= 0

nError:= Com_Addr(COM1,568,3)
if nError <> 0
        return 1
endif
nError:= Com_Open(COM1,9600,8,1,COM_NONEP,COM_NOFLOW,COM_NOFLOW,256,256, UART8250)
if nError <> 0
```

```
        return 1
endif
nEstado:= Com_GetTxE(COM1)
do case
        case nEstado = COM_ON
                ? “A transmissão está ativa.”
        case nEstado = COM_OFF
                ? “A transmissão não está inativa.”
        otherwise
                return 1
encase
nError:= Com_Close(COM1)
if nError <> 0
        return 1
endif
quit
```

Com_GetCTS

Sumário

Com_GetCTS (nCanal)

Descrição

A função ‘Com_GetCTS’ retorna o estado do sinal CTS *Clear To Send* no canal de comunicação especificado.

O argumento 'nCanal' especifica o canal de comunicação, possibilita o uso de no máximo 16 canais (COM1 a COM16).

Valor Retornado

A função retorna o estado do sinal CTS quando sua execução for bem sucedida, caso contrário retorna um código de erro sempre menor do que zero. O significado de cada código de erro retornado é fornecido no item “Códigos de Erro”.

Os possíveis estados do sinal CTS são:

COM_ON:	CTS ativo,
COM_OFF:	CTS inativo.

Consulte

Com_GetTxE, Com_GetDCD, Com_GetDSR, Com_GetRI, Com_GetDTR, Com_GetRTS.

Exemplo

```
#include “xpcomdef.h”

nEstado:= 0
nError:= 0

nError:= Com_Addr(COM1,568,3)
if nError <> 0
        return 1
endif
nError:= Com_Open(COM1,9600,8,1,COM_NONEP,COM_NOFLOW,COM_NOFLOW,256,256, UART8250)
if nError <> 0
        return 1
endif
nEstado:= Com_GetCTS(COM1)
do case
        case nEstado = COM_ON
                ? “CTS está ativo.”
```

```
        case nEstado = COM_OFF
            ? "CTS inativo."
        otherwise
            return 1
    encase
    nError:= Com_Close(COM1)
    if nError <> 0
        return 1
    endif
    quit
```

Com_GetDCD

Sumário

Com_GetDCD (nCanal)

Descrição

A função 'Com_GetDCD' retorna o estado do sinal DCD *Data Carrier Detected* no canal de comunicação especificado.

O argumento 'nCanal' especifica o canal de comunicação, possibilita o uso de no máximo 16 canais (COM1 a COM16).

Valor Retornado

A função retorna zero quando sua execução for bem sucedida, caso contrário retorna um código de erro sempre menor do que zero. O significado de cada código de erro retornado é fornecido no item "Códigos de Erro".

Os possíveis estados do sinal DCD são:

COM_ON:	DCD ativo,
COM_OFF:	DCD inativo.

Consulte

Com_GetTxE, Com_GetCTS, Com_GetDSR, Com_GetRI, Com_GetDTR, Com_GetRTS.

Exemplo

```
#include "xpcomdef.h"

nEstado:= 0
nError:= 0

nError:= Com_Addr(COM1,568,3)
if nError <> 0
    return 1
endif
nError:= Com_Open(COM1,9600,8,1,COM_NONEP,COM_NOFLOW,COM_NOFLOW,256,256, UART8250)
if nError <> 0
    return 1
endif
nEstado:= Com_GetDCD(COM1)
do case
    case nEstado = COM_ON
        ? "DCD está ativo."
    case nEstado = COM_OFF
        ? "DCD inativo."
    otherwise
        return 1
endcase
nError:= Com_Close(COM1)
```

```
if nError <> 0
        return 1
endif
quit
```

Com_GetDSR

Sumário

Com_GetDSR (nCanal)

Descrição

A função ‘Com_GetDSR’ retorna o estado do sinal DSR *Data Set Ready* no canal de comunicação especificado.

O argumento 'nCanal' especifica o canal de comunicação, possibilita o uso de no máximo 16 canais (COM1 a COM16).

Valor Retornado

A função retorna zero quando sua execução for bem sucedida, caso contrário retorna um código de erro sempre menor do que zero. O significado de cada código de erro retornado é fornecido no item “Códigos de Erro”.

Os possíveis estados do sinal DSR são:

COM_ON:	DSR ativo,
COM_OFF:	DSR inativo.

Consulte

Com_GetTxE, Com_GetCTS, Com_GetDCD, Com_GetRI, Com_GetDTR, Com_GetRTS.

Exemplo

```
#include “xpcomdef.h”

nEstado:= 0
nError:= 0

nError:= Com_Addr(COM1,568,3)
if nError <> 0
        return 1
endif
nError:= Com_Open(COM1,9600,8,1,COM_NONEP,COM_NOFLOW,COM_NOFLOW,256,256, UART8250)
if nError <> 0
        return 1
endif
nEstado:= Com_GetDSR(COM1)
do case
        case nEstado = COM_ON
                ? “DSR está ativo.”
        case nEstado = COM_OFF
                ? “DSR inativo.”
        otherwise
                return 1
endcase
nError:= Com_Close(COM1)
if nError <> 0
        return 1
endif
quit
```

Com_GetRI

Sumário

Com_GetRI (nCanal)

Descrição

A função ‘Com_GetRI’ retorna o estado do sinal RI *Ring Indicator* no canal de comunicação especificado.

O argumento 'nCanal' especifica o canal de comunicação, possibilita o uso de no máximo 16 canais (COM1 a COM16).

Valor Retornado

A função retorna zero quando sua execução for bem sucedida, caso contrário retorna um código de erro sempre menor do que zero. O significado de cada código de erro retornado é fornecido no item “Códigos de Erro”.

Os possíveis estados do sinal RI são:

COM_ON:	RI ativo,
COM_OFF:	RI inativo.

Consulte

Com_GetTxE, Com_GetCTS, Com_GetDCD, Com_GetDSR, Com_GetDTR, Com_GetRTS.

Exemplo

```
#include “xpcomdef.h”

nEstado:= 0
nError:= 0

nError:= Com_Addr(COM1,568,3)
if nError <> 0
      return 1
endif
nError:= Com_Open(COM1,9600,8,1,COM_NONEP,COM_NOFLOW,COM_NOFLOW,256,256, UART8250)
if nError <> 0
      return 1
endif
nEstado:= Com_GetRI(COM1)
do case
      case nEstado = COM_ON
            ? “RI está ativo.”
      case nEstado = COM_OFF
            ? “RI inativo.”
      otherwise
            return 1
endcase
nError:= Com_Close(COM1)
if nError <> 0
      return 1
endif
quit
```

Com_GetDTR

Sumário

Com_GetDTR (nCanal)

Descrição

A função ‘Com_GetDTR retorna o estado do sinal DTR *Data Terminal Ready* no canal de comunicação especificado.

O argumento 'nCanal' especifica o canal de comunicação, possibilita o uso de no máximo 16 canais (COM1 a COM16).

Valor Retornado

A função retorna zero quando sua execução for bem sucedida, caso contrário retorna um código de erro sempre menor do que zero. O significado de cada código de erro retornado é fornecido no item “Códigos de Erro”.

Os possíveis estados do sinal DTR são:

COM_ON:	DTR ativo,
COM_OFF:	DTR inativo.

Consulte

Com_GetTxE, Com_GetCTS, Com_GetDCD, Com_GetDSR, Com_GetRI, Com_GetRTS.

Exemplo

```
#include “xpcomdef.h”

nEstado:= 0
nError:= 0

nError:= Com_Addr(COM1,568,3)
if nError <> 0
      return 1
endif
nError:= Com_Open(COM1,9600,8,1,COM_NONEP,COM_NOFLOW,COM_NOFLOW,256,256, UART8250)
if nError <> 0
      return 1
endif
nEstado:= Com_GetDTR(COM1)
do case
      case nEstado = COM_ON
            ? “DTR está ativo.”
      case nEstado = COM_OFF
            ? “DTR inativo.”
      otherwise
            return 1
endcase
nError:= Com_Close(COM1)
if nError <> 0
      return 1
endif
quit
```

Com_GetRTS

Sumário

Com_GetRTS (nCanal)

Descrição

A função ‘Com_GetRTS’ retorna o estado do sinal RTS *Request To Send* no canal de comunicação especificado.

O argumento 'nCanal' especifica o canal de comunicação, possibilita o uso de no máximo 16 canais (COM1 a COM16).

Valor Retornado

A função retorna o estado do sinal RTS quando sua execução for bem sucedida, caso contrário retorna um código de erro sempre menor do que zero. O significado de cada código de erro retornado é fornecido no item “Códigos de Erro”.

Os possíveis estados do sinal RTS são:

COM_ON:	RTS ativo,
COM_OFF:	RTS inativo.

Consulte

Com_GetTxE, Com_GetCTS, Com_GetDCD, Com_GetDSR, Com_GetRI, Com_GetDTR.

Exemplo

```
#include “xpcomdef.h”

nEstado:= 0
nError:= 0

nError:= Com_Addr(COM1,568,3)
if nError <> 0
      return 1
endif
nError:= Com_Open(COM1,9600,8,1,COM_NONEP,COM_NOFLOW,COM_NOFLOW,256,256, UART8250)
if nError <> 0
      return 1
endif
nEstado:= Com_GetRTS(COM1)
do case
      case nEstado = COM_ON
            ? “RTS está ativo.”
      case nEstado = COM_OFF
            ? “RTS inativo.”
      otherwise
            return 1
endcase
nError:= Com_Close(COM1)
if nError <> 0
      return 1
endif
quit
```

Com_SetDTR

Sumário

Com_SetDTR (nCanal, nState)

Descrição

A função ‘Com_SetDTR controla o estado do sinal DTR *Data Terminal Ready* no canal de comunicação especificado.

O argumento 'nCanal' especifica o canal de comunicação, possibilita o uso de no máximo 16 canais (COM1 a COM16).

O argumento ‘nState’ define o novo estado do sinal DTR para o canal especificado:

COM_ON:	Ativa DTR,
COM_OFF:	Desativa DTR.

Valor Retornado

A função retorna zero quando sua execução for bem sucedida, caso contrário retorna um código de erro sempre menor do que zero. O significado de cada código de erro retornado é fornecido no item “Códigos de Erro”.

Consulte

Com_SetRTS, Com_SetBRK.

Exemplo

```
#include “xpcomdef.h”

nError:= 0

nError:= Com_Addr(COM1,568,3)
if nError <> 0
      return 1
endif
nError:= Com_Open(COM1,9600,8,1,COM_NONEP,COM_NOFLOW,COM_NOFLOW,256,256, UART8250)
if nError <> 0
      return 1
endif
nError:= Com_SetDTR(COM1,COM_ON)
if nError <> 0
      return 1
endif
nError:= Com_Close(COM1)
if nError <> 0
      return 1
endif
quit
```

Com_SetRTS

Sumário

Com_SetRTS (nCanal, nState)

Descrição

A função ‘Com_SetRTS controla o estado do sinal RTS *Request To Send* no canal de comunicação especificado.

O argumento 'nCanal' especifica o canal de comunicação, possibilita o uso de no máximo 16 canais (COM1 a COM16).

O argumento ‘nState’ define o novo estado do sinal RTS para o canal especificado:

COM_ON:	Ativa RTS,
COM_OFF:	Desativa RTS.

Valor Retornado

A função retorna zero quando sua execução for bem sucedida, caso contrário retorna um código de erro sempre menor do que zero. O significado de cada código de erro retornado é fornecido no item “Códigos de Erro”.

Consulte

Com_SetDTR, Com_SetBRK.

Exemplo

```
#include “xpcomdef.h”

nError:= 0
```

```
nError:= Com_Addr(COM1,568,3)
if nError <> 0
	return 1
endif
nError:= Com_Open(COM1,9600,8,1,COM_NONEP,COM_NOFLOW,COM_NOFLOW,256,256, UART8250)
if nError <> 0
	return 1
endif
nError:= Com_SetRTS(COM1,COM_ON)
if nError <> 0
	return 1
endif
nError:= Com_Close(COM1)
if nError <> 0
	return 1
endif
quit
```

Com_SetBRK

Sumário

Com_SetBRK (nCanal, nState)

Descrição

A função ‘Com_SetBRK controla o estado do sinal BREAK no canal de comunicação especificado.

O argumento 'nCanal' especifica o canal de comunicação, possibilita o uso de no máximo 16 canais (COM1 a COM16).

O argumento ‘nState’ define o novo estado do sinal BREAK para o canal especificado:

COM_ON:	Ativa BREAK,
COM_OFF:	Desativa BREAK.

Valor Retornado

A função retorna zero quando sua execução for bem sucedida, caso contrário retorna um código de erro sempre menor do que zero. O significado de cada código de erro retornado é fornecido no item “Códigos de Erro”.

Consulte

Com_SetDTR, Com_SetRTS.

Exemplo

```
#include “xpcomdef.h”

nError:= 0

nError:= Com_Addr(COM1,568,3)
if nError <> 0
	return 1
endif
nError:= Com_Open(COM1,9600,8,1,COM_NONEP,COM_NOFLOW,COM_NOFLOW,256,256, UART8250)
if nError <> 0
	return 1
endif
nError:= Com_SetBRK(COM1,COM_ON)
if nError <> 0
	return 1
endif
```

```
nError:= Com_Close(COM1)
if nError <> 0
    return 1
endif
quit
```

Funções de Comunicação com a Rede XPnet

Estas funções são destinadas unicamente ao controle operacional da rede e somente podem ser executadas após a inicialização da mesma. As funções de controle operacional iniciam rotinas de transmissão e recepção de dados entre o equipamento concentrador e os dispositivos conectados na rede e que serão executadas por interrupção. Também são fornecidas funções de verificação do estado da comunicação entre o concentrador e os dispositivos conectados na rede.

Funções Básicas de Comunicação com a Rede XPnet

Estas funções são destinadas a:

Inicializar operacionalmente a rede. Sempre que a rede for finalizada operacionalmente e reinicializada, todas as variáveis internas da biblioteca que indicam *status* de comunicação da rede também serão reinicializadas.

As funções de inicialização operacional da rede tem como função principal a definição da interface serial a ser utilizada, a taxa de comunicação em que a rede será operada, a definição de um *buffer* de transmissão no qual os dados transmitidos serão colocados para serem posteriormente retirados pelos terminais, e a definição de um *buffer* de recepção no qual os dados recebidos serão colocados para posteriormente serem retirados pelo aplicativo.

Verificar a comunicação entre o concentrador e os dispositivos conectados na rede.

Comunicação específica com determinado terminal da rede, não sendo necessário que o terminal especificado seja um dos válidos na comunicação automática.

Estabelecer controle da comunicação automática entre o concentrador e os terminais. A comunicação automática consiste em execuções automáticas da função 'XPnmTxPoll' utilizando como argumentos o canal e os terminais inseridos na varredura. Os dados recebidos são colocados no *buffer* de recepção do concentrador, dessa maneira o aplicativo somente precisa verificar, após iniciar a comunicação automática, se existem dados disponíveis no *buffer* de recepção.

Controlar a transferência dos dados do *buffer* de recepção do concentrador, permitindo que o programa aplicativo tenha acesso aos dados provenientes dos terminais.

XPnmOpen

Sumário

XPnmOpen (nCanal, nBaud, wTxbuffer, wRxbuffer, nComType, nMaxRetries)

Descrição

A função 'XPnmOpen' inicializa a rede operacionamente, permitindo que as funções da biblioteca possam ser executadas com sucesso. Esta função deve ser executada após 'Com_Addr', que inicializa a rede para uma configuração de *hardware* específica.

O argumento 'nCanal' especifica o canal de comunicação, possibilita o uso de no máximo 16 canais (COM1 a COM16).

O argumento 'nBaud' define a taxa de comunicação a ser utilizada com 8 *bits*, 1 *stop-bit* e sem paridade, sendo os valores possíveis para o argumento mostrados abaixo em *baud*:

600 , 1200 , 2400 , 4800 , 9600 , 19200.

O argumento 'wTxbuffer' define o espaço a ser alocado para o *buffer* de transmissão de mensagens, podendo ser um valor de 16 *bytes* a 65500 *bytes*.

O argumento 'wRxbuffer' define o espaço a ser alocado para o *buffer* de recepção de mensagens, podendo ser um valor de 16 *bytes* a 65500 *bytes*.

O argumento ‘nComType’ define o tipo de comunicação usada Full ou Half duplex:

XPN_COMHALF: Operação half-duplex,

XPN_COMFULL: Operação full-duplex.

O argumento ‘nMaxRetries’ informa o número máximo de tentativas de retransmissão quando ocorrer erros.

Valor Retornado

A função retorna zero quando a execução for bem sucedida, e caso contrário retorna um código de erro sempre menor do que zero. O significado de cada código de erro retornado é fornecido no item “Códigos de Erro”.

Consulte

Com_Addr, XPnmClose.

Exemplo

```
#include “xpcomdef.h”

nError:= 0

nError:= Com_Addr(COM1,568,3)
if nError <> 0
      return 1
endif
nError:= XPnmOpen(COM1,9600,1024,1024,XPN_COMHALF,2)
if nError <> 0
      return 1
endif
nError:= XPnmClose(COM1)
if nError <> 0
      return 1
endif
quit
```

XPnmClose

Sumário

XPnmClose (nCanal)

Descrição

A função 'XPnmClose' finaliza operacionalmente a rede previamente inicializada.

O argumento 'nCanal' especifica o canal de comunicação, possibilita o uso de no máximo 16 canais (COM1 a COM16).

Valor Retornado

A função retorna zero quando a execução for bem sucedida, caso contrário retorna um código de erro sempre menor do que zero. O significado de cada código de erro retornado é fornecido no item “Códigos de Erro”.

Consulte

Com_Addr, XPnmOpen.

Exemplo

```
#include “xpcomdef.h”

nError:= 0

nError:= Com_Addr(COM1,568,3)
```

```
if nError <> 0
        return 1
endif
nError:= XPnmOpen(COM1,9600,1024,1024,XPN_COMHALF,2)
if nError <> 0
        return 1
endif
nError:= XPnmClose(COM1)
if nError <> 0
        return 1
endif
quit
```

XPnmTinfo

Sumário

XPnmTinfo (nCanal, nTerm, @anStat)

Descrição

A função 'XPnmTinfo' obtém o "*status* de comunicação do terminal", armazenado no PC, para o terminal especificado nos argumentos 'nCanal' e 'nTerm', atualizando a estrutura de informações do argumento 'anStat'. Para atualizar o campo do "*status* interno do terminal", armazenado no PC, é necessário que o polling dos terminais esteja ativo. A cada requisição de dados feita pelo concentrador ao terminal será enviada uma mensagem do terminal ao concentrador. Se o terminal não tiver dados então é enviado o seu status interno que o concentrado usa automaticamente para atualizar na estrutura o campo "status interno do terminal". Use a função 'XPnmTxPoll' para atualizar este status quando o polling não estiver ativo.

O argumento 'nCanal' especifica o canal de comunicação, possibilita o uso de no máximo 16 canais (COM1 a COM16).

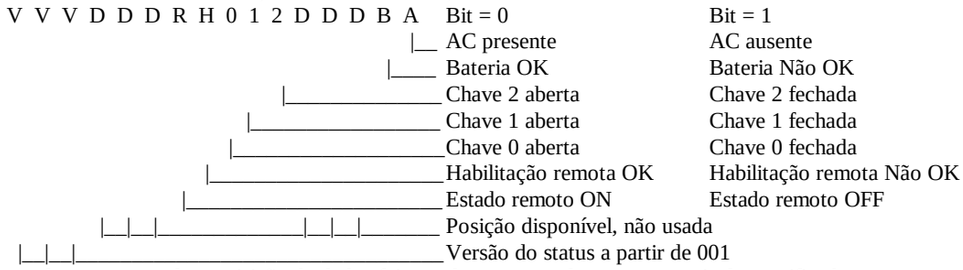
O argumento 'nTerm' informa o número do terminal, que deve ser de 1 a 32.

O argumento 'anStat' é um vetor de elementos numéricos com 11 posições, que deve ser passado por referência para atualizar as informações. As informações estão dispostas no vetor conforme a ordem abaixo:

- último erro de comunicação ocorrido ou zero se recuperado;
- último erro de comunicação, não é limpado após recuperação;
- último erro do *chip* UART ver Máscara de Erros de Comunicação em XPCOMDEF.H;

COM_OVERUN	2	- Erro de *overrun* na recepção
COM_PARITY	4	- Erro de paridade na recepção
COM_FRAME	8	- Erro de *framing* na recepção
COM_BREAK	16	- Detecção de *break*
COM_RXBFUL	128	- Erro de buffer de recepção cheio

- último *status* interno do terminal, 16 bits informando:;



- número de mensagens de requisição de dados feitas pelo concentrador para o terminal especificado;
- número de mensagens de comando transmitidas pelo concentrador para o terminal especificado;
- número de *bytes* de dados transmitidos pelo concentrador para o terminal especificado;
- número de mensagens de dados recebidas pelo concentrador do terminal especificado;
- número de *bytes* de dados recebidos pelo concentrador do terminal especificado;
- número de erros de comunicação ocorridos;
- número de repetições de tentativas de comunicação.

Valor Retornado

A função retorna zero quando a execução for bem sucedida, caso contrário retorna um código de erro sempre menor do que zero. O significado de cada código de erro retornado é fornecido no item “Códigos de Erro”.

Consulte

XPnmStatus.

Exemplo

```
#include “xpcomdef.h”

nError:= 0
anStat:= {0,0,0,0,0,0,0,0,0,0}

nError:= Com_Addr(COM1,568,3)
if nError <> 0
      return 1
endif
nError:= XPnmOpen(COM1,9600,1024,1024,XPN_COMHALF,2)
if nError <> 0
      return 1
endif
nErro:=XPnmTxPoll(COM1,1)
if nError <> 0
      return 1
endif
nErro:= XPnmWaitTx(COM1)
if nError <> 0
      return 1
endif
nError:=XPnmTinfo(COM1,1,@anStat)
if nError <> 0
      return 1
endif
? “Error:          “, anStat[1]
? “LastError:      “, anStat[2]
? “ChipError:      “, anStat[3]
? “Status:         “, anStat[4]
? “TxPolls:        “, anStat[5]
? “TxMsgs:         “, anStat[6]
? “TxData:         “, anStat[7]
? “RxMsgs:         “, anStat[8]
? “RxData:         “, anStat[9]
? “Erros:          “, anStat[10]
? “Retries:        “, anStat[11]
nError:= XPnmClose(COM1)
if nError <> 0
      return 1
endif
quit
```

XPnmStatus

Sumário

XPnmStatus (nCanal)

Descrição

A função 'XPnmStatus' informa ao aplicativo o estado do canal de comunicação especificado. Pode ser utilizada para aguardar que uma mensagem seja transmitida ao coletor antes de prosseguir o programa. Use preferencialmente XPnmWaitTx para aguardar o término de uma transmissão anterior.

O argumento 'nCanal' especifica o canal de comunicação, possibilita o uso de no máximo 16 canais (COM1 a COM16).

Valor Retornado

A função retorna as constantes numéricas abaixo, definidas no arquivo 'XPCOMDEF.H', quando sua execução for bem sucedida:

XPN_STATIDLE - Sem transmissão (Comunicação inativa),
XPN_STATACTIVE - Mensagem em transmissão (Comunicação ativa),
XPN_STATOK - Mensagem transmitida (Comunicação inativa - OK),
XPN_STATERROR - Mensagem não transmitida (Comunicação inativa com erro).

Caso contrário retorna um código de erro sempre menor do que zero. O significado de cada código de erro retornado é fornecido no item “Códigos de Erro”.

Consulte

XPnmWaitTx , XPnmTxMsg, XPnmTxAt, XPnmTxStat, XPnmTxPoll.

Exemplo

```
#include “xpcomdef.h”

nError:= 0
nStatus:= 0
bLoop:= .T.

nError:= Com_Addr(COM1,568,3)
if nError <> 0
	return 1
endif
nError:= XPnmOpen(COM1,9600,1024,1024,XPN_COMHALF,2)
if nError <> 0
	return 1
endif
nError:= XPnmTxAt(COM1,1)
if nError <> 0
	return 1
endif
do while (bLoop)
	nStatus:= XPnmStatus(COM1)
	do case
		case nStatus = XPN_STATIDLE
			? “Comunicacao Inativa.”
			bLoop:= .F.
		case nStatus = XPN_STATACTIVE
			? “Mensagem em transmissao.”
		case nStatus = XPN_STATOK
			? “Requisicao de atencao transmitida.”
			bLoop:= .F.
		case nStatus = XPN_STATERROR
			? “Comunicacao inativa com erro.”
			bLoop:= .F.
		otherwise
			return 1
	endcase
enddo
nError:= XPnmClose(COM1)
```

```
if nError <> 0
    return 1
endif
quit
```

XPnmGetCfg

Sumário

XPnmGetCfg (nCanal, nInfo, nParam)

Descrição

A função 'XPnmGetCfg' é utilizada para se obter informações diversas sobre a configuração da rede XPnet.

O argumento 'nCanal' especifica o canal de comunicação, possibilita o uso de no máximo 16 canais (COM1 a COM16).

O argumento 'nInfo' determina a informação desejada, é definido conforme as constantes abaixo, que já estão declaradas no arquivo *header* da biblioteca:

XPNCFG_OPEN:	Informa se o canal de comunicação está aberto;
XPNCFG_RATE:	Informa o valor da taxa de comunicação, na qual a rede está operando;
XPNCFG_BUFRXMSG:	Informa o tamanho do *buffer* de recepção de mensagens;
XPNCFG_BUFTXMSG:	Informa o tamanho do *buffer* de transmissão de mensagens;
XPNCFG_POLLSTART:	Informa se a varredura automática de terminais está ativa;
XPNCFG_POLLINT:	Informa o intervalo, entre varreduras;
XPNCFG_POLLTERM:	Informa o número de requisições de dados a ser feita em uma varredura a um mesmo terminal, sendo que deve-se especificar o número do terminal (1 a 32) no argumento 'nParam'.

O argumento 'nParam', como mostra acima, é utilizado em apenas um caso, nos demais o valor passado não é levado em consideração.

Valor Retornado

O valor retornado pela função varia de acordo com o especificado no argumento 'nInfo', relacionados abaixo:

XPNCFG_OPEN:	Retorna '1' para o canal aberto, '0' para canal fechado, caso contrário retorna um código de erro;
XPNCFG_RATE:	Retorna o valor da taxa de comunicação, caso contrário retorna um código de erro;
XPNCFG_BUFRXMSG:	Retorna o tamanho do *buffer* de recepção, caso contrário retorna um código de erro;
XPNCFG_BUFTXMSG:	Retorna o tamanho do *buffer* de transmissão, caso contrário retorna um código de erro;
XPNCFG_POLLSTART:	Retorna '1' se a varredura automática de terminais está ativa, '0' quando inativa, caso contrário retorna um código de erro;
XPNCFG_POLLINT:	Retorna o valor do intervalo entre varreduras, caso contrário retorna um código de erro;
XPNCFG_POLLTERM:	Retorna o número de requisições feitas em uma varredura a um mesmo terminal, caso contrário retorna um código de erro;

O valor do código de erro é sempre menor do que zero. Seu significado de é fornecido no item “Códigos de Erro”.

Consulte

XPnmTinfo.

Exemplo

```
#include “xpcomdef.h”

nError:= 0
nStatus:= 0
nInfo:= 0

nError:= Com_Addr(COM1,568,3)
if nError <> 0
    return 1
```

```
endif
nError:= XPnmOpen(COM1,9600,1024,1024,XPN_COMHALF,2)
if nError <> 0
        return 1
endif
nInfo:= XPnmGetCfg(COM1,XPNCFG_OPEN,0)
do case
        case nInfo = 1
                ? "Canal de comunicacao esta aberto."
        case nInfo = 0
                ? "Canal de comunicacao nao esta aberto."
        otherwise
                return 1
endcase
nInfo:= XPnmGetCfg(COM1,XPNCFG_BUFRXMSG,0)
if nInfo < 0
        return 1
else
        ? "Tamanho do buffer de recepcao: ", nInfo
endif
nError:= XPnmClose(COM1)
if nError <> 0
        return 1
endif
quit
```

XPnmError

Sumário

```
XPnmError (nCanal)
```

Descrição

A função 'XPnmError' informa o último erro de comunicação ocorrido durante a execução de uma das funções da biblioteca XPnet.

O argumento 'nCanal' especifica o canal de comunicação, possibilita o uso de no máximo 16 canais (COM1 a COM16).

Valor Retornado

A função retorna zero quando nenhum erro foi encontrado, caso contrário retorna um código de erro sempre menor do que zero. O significado de cada código de erro retornado é fornecido no item "Códigos de Erro".

Exemplo

```
#include "xpcomdef.h"

nError:= 0

nError:= Com_Addr(COM1,568,3)
if nError <> 0
        return 1
endif
nError:= XPnmOpen(COM1,9600,1024,1024,XPN_COMHALF,2)
if nError <> 0
        return 1
endif
nError:= XPnmError(COM1)
? "Erro:      ", nError
nError:= XPnmClose(COM1)
```

```
if nError <> 0
        return 1
endif
quit
```

XPComErMsg

Sumário

```
XPComErMsg (nError)
```

Descrição

A função 'XPComErMsg' recebe um código de erro e retorna um string com a respectiva mensagem de erro.

O argumento 'nError' especifica o código de erro para o qual queremos sua mensagem.

Valor Retornado

A função retorna um string com a mensagem de erro correspondente ao código de erro enviado. O significado de cada código de erro retornado é fornecido no item “Códigos de Erro”.

Exemplo

```
#include “xpcomdef.h”

nError:= 0
cMsg:= ““

nError:= Com_Addr(COM1,568,3)
if nError <> 0
        cMsg:= XPComErMsg(nErro)
        ? cMsg
        return 1
endif
nError:= XPnmOpen(COM1,9600,1024,1024,XPN_COMHALF,2)
if nError <> 0
        cMsg:= XPComErMsg(nErro)
        ? cMsg
        return 1
endif
nError:= XPnmClose(COM1)
if nError <> 0
        cMsg:= XPComErMsg(nErro)
        ? cMsg
        return 1
endif
quit
```

XPnmTxMsg

Sumário

```
XPnmTxMsg(nCanal, nTerm, cMsg)
```

Descrição

A função 'XPnmTxMsg' transmite uma mensagem ao coletor, conforme o destino indicado pelos argumentos 'nCanal' e 'nTerm'. Esta mensagem pode ser retirada, do buffer de recepção de mensagens, pelo programa do coletor. Para que o coletor interprete a mensagem como um comando em XPBASIC, coloque o coletor em modo remoto antes de enviar a mensagem.

O argumento 'nCanal' especifica o canal de comunicação, possibilita o uso de no máximo 16 canais (COM1 a COM16).

O argumento 'nTerm' informa o número do terminal, que deve ser de 1 a 32.

O argumento tipo caractere 'cMsg' contém a mensagem a ser transmitida.

Valor Retornado

A função retorna zero quando a execução for bem sucedida, caso contrário retorna um código de erro sempre menor do que zero. O significado de cada código de erro retornado é fornecido no item “Códigos de Erro”.

Consulte

XPnmWaitTx, XPnmTxPoll, XPnmTxAt, XPnmTxStat, XPnmTxBcnt, XPnmTxBbyt, XPnmTxBFre, XPnmTxBclr.

Exemplo

```
#include “xpcomdef.h”

nStatus:= 0
nError:= 0

nError:= Com_Addr(COM1,568,3)
if nError <> 0
      return 1
endif
nError:= XPnmOpen(COM1,9600,1024,1024,XPN_COMHALF,2)
if nError <> 0
      return 1
endif
nError:= XPnmTxMsg(COM1,1,”Teste”)
if nError <> 0
      return 1
endif
nError:= XPnmWaitTx(COM1)
if nError <> 0
      return 1
endif
nError:= XPnmClose(COM1)
if nError <> 0
      return 1
endif
quit
```

XPnmTxPoll

Sumário

XPnmTxPoll (nCanal, nTerm)

Descrição

A função 'XPnmTxPoll' transmite mensagem de requisição de dados para um terminal. Sendo que os dados recebidos serão armazenados no *buffer* de recepção. Se o terminal não tiver dados enviará o seu status que será removido automaticamente do buffer de recepção do PC e será guardado no campo ‘status interno do terminal’ da estrutura que guarda o ‘status interno do terminal’.

O argumento 'nCanal' especifica o canal de comunicação, possibilita o uso de no máximo 16 canais (COM1 a COM16).

O argumento 'nTerm' informa o número do terminal, que deve ser de 1 a 32.

Como a transmissão é feita por interrupção, a função retorna imediatamente após ter sido chamada, sendo necessário utilizar a função 'XPnmWaitTx' para verificar se a mensagem foi corretamente transmitida. A execução desta função é mal sucedida quando uma outra comunicação já estiver em andamento.

Valor Retornado

A função retorna zero quando a execução for bem sucedida, caso contrário retorna um código de erro sempre menor do que zero. O significado de cada código de erro retornado é fornecido no item “Códigos de Erro”.

Consulte

XPnmWaitTx, XPnmTinfo, XPnmTxMsg, XPnmTxAt, XPnmTxStat, XPnmTxBcnt, XPnmTxBbyt, XPnmTxBfre, XPnmTxBclr, XPnmRxMsg.

Exemplo

```
#include “xpcomdef.h”

nError:= 0
nStatus:= 0

nError:= Com_Addr(COM1,568,3)
if nError <> 0
      return 1
endif
nError:= XPnmOpen(COM1,9600,1024,1024,XPN_COMHALF,2)
if nError <> 0
      return 1
endif
nError:= XPnmTxPoll(COM1,1)
if nError <> 0
      return 1
endif
nError:= XPnmWaitTx(COM1)
if nError <> 0
      return 1
endif
nError:= XPnmClose(COM1)
if nError <> 0
      return 1
endif
quit
```

XPnmTxStat

Sumário

XPnmTxStat (nCanal, nTerm)

Descrição

A função 'XPnmTxStat' requisita o *status* interno de um terminal especificado pelo argumento ‘nCanal”e 'nTerm'. O status fica no buffer de recepção do PC até ser retirado. Não é atualizado o campo “*status* interno do terminal” na estrutura interna de controle da rede lida pela função 'XPnmTinfo'.

O argumento 'nCanal' especifica o canal de comunicação, possibilita o uso de no máximo 16 canais (COM1 a COM16).

O argumento 'nTerm' informa o número do terminal, que deve ser de 1 a 32.

Como a transmissão é feita por interrupção, a função retorna imediatamente após ter sido chamada, sendo necessário utilizar a função 'XPnmWaitTx' para verificar se a mensagem foi corretamente transmitida. A execução desta função é mal sucedida quando uma outra comunicação já estiver em andamento.

Valor Retornado

A função retorna zero quando a execução for bem sucedida, caso contrário retorna um código de erro sempre menor do que zero. O significado de cada código de erro retornado é fornecido no item “Códigos de Erro”.

Consulte

XPnmWaitTx, XPnmTxMsg, XPnmTxPoll, XPnmTxAt, XPnmTxBcnt, XPnmTxBbyt, XPnmTxBfre, XPnmTxBclr.

Exemplo

```
#include “xpcomdef.h”

nStatus:= 0
nError:= 0

nError:= Com_Addr(COM1,568,3)
if nError <> 0
      return 1
endif
nError:= XPnmOpen(COM1,9600,1024,1024,XPN_COMHALF,2)
if nError <> 0
      return 1
endif
nError:= XPnmTxStat(COM1,1)
if nError <> 0
      return 1
endif
nError:= XPnmWaitTx(COM1)
if nError <> 0
      return 1
endif
nError:= XPnmClose(COM1)
if nError <> 0
      return 1
endif
quit
```

XPnmTxAt

Sumário

XPnmTxAt (nCanal, nTerm)

Descrição

A função 'XPnmTxAt' coloca o terminal especificado pelos argumentos 'nCanal' e 'nTerm' em MODO REMOTO e retorna imediatamente. Dai em diante todas mensagens para o terminal serão interpretadas como comandos Basic até que um comando de REMOTE “OFF”seja enviado. A função ‘XPnetmRemote’ chama 'XPnetmTxMsgAttention' ao colocar o coletor em modo remoto e aguarda a transmissão para coletor antes de retornar, garantindo assim que o coletor recebeu o comando de atenção e teve tempo de entrar em modo remoto.

O argumento 'nCanal' especifica o canal de comunicação, possibilita o uso de no máximo 16 canais (COM1 a COM16).

O argumento 'nTerm' informa o número do terminal, que deve ser de 1 a 32.

Como a transmissão é feita por interrupção, a função retorna imediatamente após ter sido chamada, sendo necessário utilizar a função 'XPnmWaitTx' para verificar se a mensagem foi corretamente transmitida. A execução desta função é mal sucedida quando uma outra comunicação já estiver em andamento.

Valor Retornado

A função retorna zero quando sua execução for bem sucedida, caso contrário retorna um código de erro sempre menor do que zero. O significado de cada código de erro sempre menor do que zero retornado é fornecido no item “Códigos de Erro”.

Consulte

XPnmRemote, XPnmWaitTx, XPnmTxMsg, XPnmTxPoll, XPnmTxStat, XPnmTxBcnt, XPnmTxBbyt, XPnmTxBfre, XPnmTxBclr.

Exemplo

```
#include “xpcomdef.h”

nStatus:= 0
nError:= 0

nError:= Com_Addr(COM1,568,3)
if nError <> 0
	return 1
endif
nError:= XPnmOpen(COM1,9600,1024,1024,XPN_COMHALF,2)
if nError <> 0
	return 1
endif
nError:= XPnmTxAt(COM1,1)
if nError <> 0
	return 1
endif
nError:= XPnmWaitTx(COM1)
if nError <> 0
	return 1
endif
nError:= XPnmClose(COM1)
if nError <> 0
	return 1
endif
quit
```

XPnmTxBcnt

Sumário

XPnmTxBcnt (nCanal)

Descrição

A função 'XPnmTxBcnt', retorna o número de mensagens a transmitir na área reservada para o *buffer* de transmissão.

O argumento 'nCanal' especifica o canal de comunicação, possibilita o uso de no máximo 16 canais (COM1 a COM16).

Valor Retornado

A função retorna valor positivo (número de mensagens) ou igual a zero (*buffer* vazio) quando sua execução for bem sucedida, caso contrário retorna um código de erro sempre menor do que zero. O significado de cada código de erro retornado é fornecido no item “Códigos de Erro”.

Consulte

XPnmStatus, XPnmTxMsg, XPnmTxPoll, XPnmTxAt, XPnmTxStat, XPnmTxBbyt, XPnmTxBfre, XPnmTxBclr.

Exemplo

```
#include “xpcomdef.h”

nError:= 0
nCount:= 0

nError:= Com_Addr(COM1,568,3)
if nError <> 0
	return 1
endif
nError:= XPnmOpen(COM1,9600,1024,1024,XPN_COMHALF,2)
if nError <> 0
	return 1
endif
nCount:= XPnmTxBcnt(COM1)
if nCount < 0
	return 1
else
	? “Mensagens a transmitir: ”, nCount
endif
nError:= XPnmClose(COM1)
if nError <> 0
	return 1
endif
quit
```

XPnmTxBbyt

Sumário

XPnmTxBbyt (nCanal)

Descrição

A função 'XPnmTxBbyt', retorna o número de *bytes* armazenados na área reservada para o *buffer* de transmissão.

O argumento 'nCanal' especifica o canal de comunicação, possibilita o uso de no máximo 16 canais (COM1 a COM16).

Valor Retornado

A função retorna valor positivo (*bytes* armazenados) ou igual a zero (*buffer* vazio) quando sua execução for bem sucedida, caso contrário retorna um código de erro sempre menor do que zero. O significado de cada código de erro retornado é fornecido no item “Códigos de Erro”.

Consulte

XPnmStatus, XPnmTxMsg, XPnmTxPoll, XPnmTxAt, XPnmTxStat, XPnmTxBcnt, XPnmTxBclr.

Exemplo

```
#include “xpcomdef.h”

nBytes:= 0
nError:= 0

nError:= Com_Addr(COM1,568,3)
if nError <> 0
	return 1
endif
nError:= XPnmOpen(COM1,9600,1024,1024,XPN_COMHALF,2)
if nError <> 0
	return 1
```

```
endif
nBytes:= XPnmTxBbyt(COM1)
if nBytes < 0
        return 1
else
        ? “ Area ocupada do Buffer de Transmissao: ”, nBytes
endif
nError:= XPnmClose(COM1)
if nError <> 0
        return 1
endif
quit
```

XPnmTxBfre

Sumário

```
XPnmTxBfre (nCanal)
```

Descrição

A função 'XPnmTxBfre', retorna o espaço (em *bytes*) disponível na área reservada para o *buffer* de transmissão.

O argumento 'nCanal' especifica o canal de comunicação, possibilita o uso de no máximo 16 canais (COM1 a COM16).

Valor Retornado

A função retorna valor positivo (*bytes* disponíveis) ou igual a zero (*buffer* cheio) quando sua execução for bem sucedida, caso contrário retorna um código de erro sempre menor do que zero. O significado de cada código de erro retornado é fornecido no item “Códigos de Erro”.

Consulte

XPnmStatus, XPnmTxMsg, XPnmTxPoll, XPnmTxAt, XPnmTxStat, XPnmTxBcnt, XPnmTxBbyt, XPnmTxBclr.

Exemplo

```
#include “xpcomdef.h”

nBytes:= 0
nError:= 0

nError:= Com_Addr(COM1,568,3)
if nError <> 0
        return 1
endif
nError:= XPnmOpen(COM1,9600,1024,1024,XPN_COMHALF,2)
if nError <> 0
        return 1
endif
nBytes:= XPnmTxBfre(COM1)
if nBytes < 0
        return 1
else
        ? “ Area disponivel: ”, nBytes
endif
nError:= XPnmClose(COM1)
if nError <> 0
        return 1
endif
quit
```

XPnmTxBclr

Sumário

XPnmTxBclr (nCanal)

Descrição

A função 'XPnmTxBclr' limpa toda a área reservada para o *buffer* de transmissão, descartando todas as mensagens anteriormente armazenadas.

O argumento 'nCanal' especifica o canal de comunicação, possibilita o uso de no máximo 16 canais (COM1 a COM16).

Valor Retornado

A função retorna zero quando sua execução for bem sucedida, caso contrário retorna um código de erro sempre menor do que zero. O significado de cada código de erro retornado é fornecido no item “Códigos de Erro”.

Consulte

XPnmStatus, XPnmTxMsg, XPnmTxPoll, XPnmTxAt, XPnmTxStat, XPnmTxBcnt, XPnmTxBbyt, XPnmTxBfre.

Exemplo

```
#include “xpcomdef.h”

nError:= 0

nError:= Com_Addr(COM1,568,3)
if nError <> 0
	return 1
endif
nError:= XPnmOpen(COM1,9600,1024,1024,XPN_COMHALF,2)
if nError <> 0
	return 1
endif
nError:= XPnmTxBclr(COM1)
if nError <> 0
	return 1
endif
nError:= XPnmClose(COM1)
if nError <> 0
	return 1
endif
quit
```

XPnmPstart

Sumário

XPnmPstart (nCanal, nIntervalo)

Descrição

A função 'XPnmPstart' inicia a varredura automática, que faz a requisição de dados e recebe o status interno do terminal caso não haja dados, de todos os terminais inseridos na mesma. Após ser executada esta função, a inserção de um terminal na varredura automática deve ser feita pela função 'XPnmPterm'.

O argumento 'nCanal' especifica o canal de comunicação, possibilita o uso de no máximo 16 canais (COM1 a COM16).

O argumento 'nIntervalo' especifica o tempo entre o término de uma requisição de dados de um terminal e o início da requisição de dados do próximo terminal na varredura, determinando assim a freqüência de varredura da rede. Este intervalo de tempo é fornecido em *ticks* (unidades de 0,055 segundos), e deve ser maior que 0.

Valor Retornado

A função retorna zero quando sua execução for bem sucedida, caso contrário retorna um código de erro sempre menor do que zero. O significado de cada código de erro retornado é fornecido no item “Códigos de Erro”.

Consulte

XPnmPstop, XPnmPterm, XPnmTinfo.

Exemplo

```
#include “xpcomdef.h”

nCount:= 0
nError:= 0

nError:= Com_Addr(COM1,568,3)
if nError <> 0
        return 1
endif
nError:= XPnmOpen(COM1,9600,1024,1024,XPN_COMHALF,2)
if nError <> 0
        return 1
endif
nError:= XPnmPstart(COM1, 2)
if nError <> 0
        return 1
endif
nError:= XPnmPterm(COM1,1,2)
if nError <> 0
        return 1
endif
nError:= XPnmPterm(COM1,2,1)
if nError <> 0
        return 1
endif
nCount:= XPnmRxBcnt(COM1)
if nCount < 0
        return 1
elseif nCount = 0
        ? “Nao existem dados disponiveis.”
else
        ? “Existem dados disponiveis.”
endif
nError:= XPnmPStop(COM1)
if nError <> 0
        return 1
endif
nError:=XPnmClose(COM1)
if nError <> 0
        return 1
endif
quit
```

XPnmPstop

Sumário

XPnmPstop (nCanal)

Descrição

A função 'XPnmPstop' termina a varredura automática de requisição de dados previamente inicializada. Após a execução desta função, os dados obtidos através da varredura automática e que ainda não foram retirados do *buffer* de recepção não são descartados.

O argumento 'nCanal' especifica o canal de comunicação, possibilita o uso de no máximo 16 canais (COM1 a COM16).

Valor Retornado

A função retorna zero quando sua execução for bem sucedida, caso contrário retorna um código de erro sempre menor do que zero. O significado de cada código de erro retornado é fornecido no item “Códigos de Erro”.

Consulte

XPnmPstart, XPnmPterm.

Exemplo

```
#include “xpcomdef.h”

nCount:= 0
nError:= 0

nError:= Com_Addr(COM1,568,3)
if nError <> 0
        return 1
endif
nError:= XPnmOpen(COM1,9600,1024,1024,XPN_COMHALF,2)
if nError <> 0
        return 1
endif
nError:= XPnmPstart(COM1, 2)
if nError <> 0
        return 1
endif
nError:= XPnmPterm(COM1,1,2)
if nError <> 0
        return 1
endif
nError:= XPnmPterm(COM1,2,1)
if nError <> 0
        return 1
endif
nCount:= XPnmRxBcnt(COM1)
if nCount < 0
        return 1
elseif nCount = 0
        ? “Nao existem dados disponiveis.”
else
        ? “Existem dados disponiveis.”
endif
nError:= XPnmPStop(COM1)
if nError <> 0
        return 1
endif
nError:=XPnmClose(COM1)
if nError <> 0
```

```
        return 1
    endif
    quit
```

XPnmPterm

Sumário

XPnmPterm (nCanal, nTerm, nRepet)

Descrição

A função 'XPnmPterm' insere um terminal na varredura automática de requisição de dados da rede XPnet, conforme os argumentos 'nCanal' e 'nTerm'.

O argumento 'nCanal' especifica o canal de comunicação, possibilita o uso de no máximo 16 canais (COM1 a COM16).

O argumento 'nTerm' informa o número do terminal, que deve ser de 1 a 32.

O argumento 'nRepet' informa o número de requisições de dados que a rede XPnet deve fazer a um mesmo terminal em uma mesma varredura, este número varia de 0 a 10. Sendo que para se retirar um terminal da varredura, deve-se executar a função 'XPnmPterm', tendo o argumento para número de requisições igual a zero.

Valor Retornado

A função retorna zero quando sua execução for bem sucedida, caso contrário retorna um código de erro sempre menor do que zero. O significado de cada código de erro retornado é fornecido no item “Códigos de Erro”.

Consulte

XPnmPstart, XPnmPstop.

Exemplo

```
#include “xpcomdef.h”

nCount:= 0
nError:= 0

nError:= Com_Addr(COM1,568,3)
if nError <> 0
        return 1
endif
nError:= XPnmOpen(COM1,9600,1024,1024,XPN_COMHALF,2)
if nError <> 0
        return 1
endif
nError:= XPnmPstart(COM1, 2)
if nError <> 0
        return 1
endif
nError:= XPnmPterm(COM1,1,2)
if nError <> 0
        return 1
endif
nError:= XPnmPterm(COM1,2,1)
if nError <> 0
        return 1
endif
nCount:= XPnmRxBcnt(COM1)
if nCount < 0
```

```
        return 1
elseif nCount = 0
        ? “Nao existem dados disponiveis.”
else
        ? “Existem dados disponiveis.”
endif
nError:= XPnmPStop(COM1)
if nError <> 0
        return 1
endif
nError:= XPnmClose(COM1)
if nError <> 0
        return 1
endif
quit
```

XPnmRxMbyt

Sumário

XPnmRxMbyt(nCanal)

Descrição

A função 'XPnmRxMbyt' informa o número de *bytes* da primeira mensagem que está armazenada e ainda não foi retirada da área reservada para o *buffer* de recepção.

O argumento 'nCanal' especifica o canal de comunicação, possibilita o uso de no máximo 16 canais (COM1 a COM16).

Valor Retornado

A função retorna valor positivo (tamanho da mensagem) ou igual a zero (*buffer* vazio) quando sua execução for bem sucedida, caso contrário retorna um código de erro sempre menor do que zero. O significado de cada código de erro retornado é fornecido no item “Códigos de Erro”.

Consulte

XPnmRxMsg, XPnmRxBcnt, XPnmRxBbyt, XPnmRxBfre, XPnmRxBclr.

Exemplo

```
#include “xpcomdef.h”

nError:= 0
nStatus:= 0
nBytes:= 0

nError:=Com_Addr(COM1,568,3)
if nError <> 0
        return 1
endif
nError:= XPnmOpen(COM1,9600,1024,1024,XPN_COMHALF,2)
if nError <> 0
        return 1
endif
nError:=XPnmTxPoll(COM1,1)
if nError <> 0
        return 1
endif
nError:= XPnmWaitTx(COM1)
if nError <> 0
```

```
        return 1
    endif
    nBytes:= XPnmRxMbyt(COM1)
    if nBytes <= 0
        if XPN_RXBEMPTY == nBytes.or. nBytes = 0
            ? “Nao existem dados disponiveis.”
            quit
        endif
        return 1
    endif
    ? “Existem dados disponiveis.”
    nError:=XPnmClose(COM1)
    if nError <> 0
        return 1
    endif
    quit
```

XPnmRxMsg

Sumário

XPnmRxMsg (nCanal, @cMsg)

Descrição

A função 'XPnmRxMsg' retira do *buffer* de recepção do PC uma mensagem, e a coloca a partir da área de retorno apontada pelo argumento 'cMsg'. As mensagens provenientes de um terminal são agrupadas no *buffer* de recepção, sendo cada uma destas uma seqüência de caracteres, onde os primeiros dois caracteres são o número do terminal em ASCII e os demais são a mensagem recebida.

Para que exista mensagem no *buffer* de recepção do PC é necessário executar a função XPnmTxPoll que retira dados do *buffer* de transmissão do coletor e transfere para o PC.

O argumento 'nCanal' especifica o canal de comunicação, possibilita o uso de no máximo 16 canais (COM1 a COM16).

O argumento tipo caractere 'cMsg' é passado por referência, para retirar a mensagem que está armazenada no *buffer* de recepção. As mensagens provenientes de um terminal são agrupadas no *buffer* de recepção, sendo cada uma destas uma seqüência de caracteres, onde os primeiros dois caracteres são o número do terminal em ASCII e os demais a mensagem recebida.

Valor Retornado

A função retorna zero quando sua execução for bem sucedida, caso contrário retorna um código de erro sempre menor do que zero. O significado de cada código de erro retornado é fornecido no item “Códigos de Erro”.

Consulte

XPnmRxMbyt, XPnmRxBcnt, XPnmRxBbyt, XPnmRxBfre, XPnmRxBclr, XPnmTxPoll.

Exemplo

```
#include “xpcomdef.h”

nError:= 0
nBytes:= 0
nStatus:= 0
cMsg:= ““
cTerm:= ““
cDados:= ““

nError:= Com_Addr (COM1,568,3)
if nError <> 0
    return 1
```

```
endif
nError:= XPnmOpen(COM1,9600,1024,1024,XPN_COMHALF,2)
if nError <> 0
        return 1
endif
nError:= XPnmTxPoll(COM1, 1)
nError:= XPnmWaitTx(COM1)
if nError <> 0
        return 1
endif
nBytes:= XPnmRxMbyt(COM1)
if nBytes <= 0
        if XPN_RXBEMPTY == nBytes.or. nBytes = 0
                ? "Nao existem dados disponiveis."
                quit
        endif
        return 1
endif
nError:= XPnmRxMsg(COM1, @cMsg)
if nError <> 0
        return 1
endif
cTerm:= substr(cMsg,1,2)
? "Terminal:    ", cTerm
cDados:= substr(cMsg,3)
? "Mensagem: ", cDados
nError:=XPnmClose(COM1)
if nError <> 0
        return 1
endif
quit
```

XPnmRxBcnt

Sumário

XPnmRxBcnt (nCanal)

Descrição

A função 'XPnmRxBcnt' informa o número de mensagens armazenadas na área reservada para o *buffer* de recepção.

O argumento 'nCanal' especifica o canal de comunicação, possibilita o uso de no máximo 16 canais (COM1 a COM16).

Valor Retornado

A função retorna valor positivo (número de mensagens armazenadas) ou igual a zero (*buffer* vazio) quando sua execução for bem sucedida, caso contrário retorna um código de erro sempre menor do que zero. O significado de cada código de erro retornado é fornecido no item "Códigos de Erro".

Consulte

XPnmRxMsg, XPnmRxMbyt, XPnmRxBbyt, XPnmRxBfre, XPnmRxBclr.

Exemplo

```
#include "xpcomdef.h"

nError:= 0
nStatus:= 0
nCount:= 0
```

```
nError:=Com_Addr(COM1,568,3)
if nError <> 0
        return 1
endif
nError:= XPnmOpen(COM1,9600,1024,1024,XPN_COMHALF,2)
if nError <> 0
        return 1
endif
nError:=XPnmTxPoll(COM1,1)
if nError <> 0
        return 1
endif
nError:= XPnmWaitTx(COM1)
if nError <> 0
        return 1
endif
nCount:= XPnmRxBcnt(COM1)
if nCount < 0
        return 1
elseif nCount = 0
        ? “Nao existem dados disponiveis.”
else
        ? “Existem dados disponiveis.”
endif
nError:=XPnmClose(COM1)
if nError <> 0
        return 1
endif
quit
```

XPnmRxBbyt

Sumário

XPnmRxBbyt (nCanal)

Descrição

A função 'XPnmRxBbyt’ informa o número de *bytes* armazenados na área reservada do *buffer* de recepção.

O argumento 'nCanal' especifica o canal de comunicação, possibilita o uso de no máximo 16 canais (COM1 a COM16).

Valor Retornado

A função retorna valor positivo (número de *bytes* armazenados) ou igual a zero (*buffer* vazio) quando sua execução for bem sucedida, caso contrário retorna um código de erro sempre menor do que zero. O significado de cada código de erro retornado é fornecido no item “Códigos de Erro”.

Consulte

XPnmRxMsg, XPnmRxMbyt, XPnmRxBbyt, XPnmRxBfre, XPnmRxBclr.

Exemplo

```
#include “xpcomdef.h”

nError:= 0
nStatus:= 0
nBytes:= 0
```

```
nError:=Com_Addr(COM1,568,3)
if nError <> 0
        return 1
endif
nError:= XPnmOpen(COM1,9600,1024,1024,XPN_COMHALF,2)
if nError <> 0
        return 1
endif
nError:= XPnmTxPoll(COM1,1)
if nError <> 0
        return 1
endif
nError:= XPnmWaitTx(COM1)
if nError <> 0
        return 1
endif
nBytes:= XPnmRxBbyt(COM1)
if nBytes < 0
        return 1
elseif nBytes = 0
        ? “Nao existem dados disponiveis.”
else
        ? “Existem dados disponiveis.”
endif
nError:= XPnmClose(COM1)
if nError <> 0
        return 1
endif
quit
```

XPnmRxBfre

Sumário

XPnmRxBfre (nCanal)

Descrição

A função 'XPnmRxBfre' informa o numero de *bytes* livres na área reservada para o *buffer* de recepção.

O argumento 'nCanal' especifica o canal de comunicação, possibilita o uso de no máximo 16 canais (COM1 a COM16).

Valor Retornado

A função retorna valor positivo (número de *bytes* disponíveis) ou igual a zero (*buffer* cheio) quando sua execução for bem sucedida, caso contrário retorna um código de erro sempre menor do que zero. O significado de cada código de erro retornado é fornecido no item “Códigos de Erro”.

Consulte

XPnmRxMbyt, XPnmRxMsg, XPnmRxBcnt, XPnmRxBbyt, XPnmRxBclr.

Exemplo

```
#include “xpcomdef.h”

nError:= 0
nStatus:= 0
nBytes:= 0

nError:= Com_Addr(COM1,568,3)
```

```
if nError <> 0
        return 1
endif
nError:= XPnmOpen(COM1,9600,1024,1024,XPN_COMHALF,2)
if nError <> 0
        return 1
endif
nError:= XPnmTxPoll(COM1,1)
if nError <> 0
        return 1
endif
nError:= XPnmWaitTx(COM1)
if nError <> 0
        return 1
endif
nBytes:= XPnmRxBfre(COM1)
if nBytes < 0
        return 1
else
        ? " Area disponivel: ", nBytes
endif
nError:= XPnmClose(COM1)
if nError <> 0
        return 1
endif
quit
```

XPnmRxBclr

Sumário

XPnmRxBclr (nCanal)

Descrição

A função 'XPnmRxBclr' descarta os dados do *buffer* de recepção da rede XPnet.

O argumento 'nCanal' especifica o canal de comunicação, possibilita o uso de no máximo 16 canais (COM1 a COM16).

Valor Retornado

A função retorna zero quando sua execução for bem sucedida, caso contário retorna um código de erro sempre menor do que zero. O significado de cada código de erro retornado é fornecido no item "Códigos de Erro".

Consulte

XPnmRxMbyt, XPnmRxMsg, XPnmRxBclr, XPnmRxBbyt, XPnmRxBfre.

Exemplo

```
#include "xpcomdef.h"

nError:= 0

nError:= Com_Addr(COM1,568,3)
if nError <> 0
        return 1
endif
nError:= XPnmOpen(COM1,9600,1024,1024,XPN_COMHALF,2)
if nError <> 0
        return 1
```

```
endif
nError:= XPnmRxBclr(COM1)
if nError <> 0
        return 1
endif
nError:= XPnmClose(COM1)
if nError <> 0
        return 1
endif
quit
```

Funções Auxiliares de Comunicação com a Rede XPnet

Estas funções são destinadas à transmissão de arquivos e programas do concentrador para os terminais, assim como à recepção de arquivos e programas dos terminais para o concentrador. Para que estas funções sejam executadas é necessário que a rede esteja inicializada operacionalmente e a varredura automática não esteja ativada.

XPnmRemote

Sumário

XPnmRemote(nCanal, nTerm, cRemote)

Descrição

A função 'XPnmRemote' transmite o comando para ativar ou desativar o modo remoto no terminal da rede XPnet e aguarda sua transmissão. Quando ativado, o terminal passa a ser comandado remotamente através de seu canal de comunicação. Desativando-se o modo remoto o terminal retorna ao *prompt*, podendo assim ser operado normalmente. Assim, sempre que o modo remoto não for mais necessário, aconselha-se desativá-lo.

O argumento 'nCanal' especifica o canal de comunicação, possibilita o uso de no máximo 16 canais (COM1 a COM16).

O argumento 'nTerm' informa o número do terminal, que deve ser de 1 a 32.

O argumento ‘cRemote’ define o código da operação a ser relizada pela função:

Argumento	Descrição
XPN_REMOTE	Estabelece modo remoto no coletor;
XPN_LOCAL	Estabelece modo local no coletor sem executar programa;
XPN_LOCALRUN	Estabelece modo local no coletor e executa programa.

Valor Retornado

A função retorna zero quando sua execução for bem sucedida, caso contrário retorna um código de erro sempre menor do que zero. O significado de cada código de erro retornado é fornecido no item “Códigos de Erro”.

Consulte

XPnmTxAt, XPnmTxClk, XPnmWaitTx, XPnmRxFile, XPnmTxFile, XPnmRxProg, XPnmTxProg.

Exemplo

```
#include “xpcomdef.h”

nError:= 0

nError:= Com_Addr(COM1, 568, 3)
if nError <> 0
        return 1
endif
nError:= XPnmOpen(COM1,9600,256,256,XPN_COMHALF,2)
if nError <> 0
        return 1
```

```
endif
nError:= XPnmPstop(COM1)
if nError <> 0
        return 1
endif
nError:= XPnmRemote(COM1, 31, XPN_REMOTE)
if nError <> 0
        return 1
endif
nError:= XPnmRemote(COM1, 31, XPN_LOCAL)
if nError <> 0
        return 1
endif
nError:= XPnmPstart(COM1, 1)
if nError <> 0
        return 1
endif
nError:= XPnmClose(COM1)
if nError <> 0
        return 1
endif
quit
```

XPnmWaitTx

Sumário

XPnmWaitTx (nCanal)

Descrição

A função ‘XPnmWaitTx’ aguarda o término da última transmissão realizada no canal antes de retornar. Deve-se utilizar só quando é necessário aguardar o fim da transmissão antes de prosseguir.

O argumento 'nCanal' especifica o canal de comunicação, possibilita o uso de no máximo 16 canais (COM1 a COM16).

Valor Retornado

A função retorna zero quando sua execução for bem sucedida, caso contrário retorna um código de erro sempre menor do que zero. O significado de cada código de erro retornado é fornecido no item “Códigos de Erro”.

Consulte

XPnmRemote, XPnmTxClk, XPnmRxFile, XPnmTxFile, XPnmRxProg, XPnmTxProg.

Exemplo

```
#include “xpcomdef.h”

nError:= 0

nError:= Com_Addr(COM1, 568, 3)
if nError <> 0
        return 1
endif
nError:= XPnmOpen(COM1,9600,256,256,XPN_COMHALF,2)
if nError <> 0
        return 1
endif
nError:= XPnmPstop(COM1)
if nError <> 0
```

```
        return 1
    endif
    nError:= XPnmRemote(COM1, 31, XPN_REMOTE)
    if nError <> 0
        return 1
    endif
    nError:= XPnmTxClk(COM1, 31)
    if nError <> 0
        return 1
    endif
    nError:= XPnmWaitTx(COM1)
    if nError <> 0
        return 1
    endif
    nError:= XPnmRemote(COM1, 31, XPN_LOCAL)
    if nError <> 0
        return 1
    endif
    nError:= XPnmPstart(COM1, 1)
    if nError <> 0
        return 1
    endif
    nError:= XPnmClose(COM1)
    if nError <> 0
        return 1
    endif
    quit
```

XPnmTxClk

Sumário

XPnmTxClk (nCanal, nTerm)

Descrição

A função 'XPnmTxClk' atualiza relógio e data do terminal especificado. A atualização é feita conforme o relógio e a data do micro concentrador da rede XPnet. Para a execução desta função exige-se que o terminal esteja em modo remoto, isto pode ser feito executando-se previamente a função 'XPnmRemote'. O formato da data do CLIPPER deve ser dd/mm/yy. Para evitar problemas no ano 2000 corrija a mudança do século para 1980 ou 1990.

O argumento 'nCanal' especifica o canal de comunicação, possibilita o uso de no máximo 16 canais (COM1 a COM16).

O argumento 'nTerm' informa o número do terminal, que deve ser de 1 a 32.

Valor Retornado

A função retorna zero quando sua execução for bem sucedida, caso contrário retorna um código de erro sempre menor do que zero. O significado de cada código de erro retornado é fornecido no item “Códigos de Erro”.

Consulte

XPnmRemote, XPnmWaitTx, XPnmRxFile, XPnmTxFile, XPnmRxProg, XPnmTxProg.

Exemplo

```
#include “xpcomdef.h”

Set(_SET_DATEFORMAT, “dd/mm/yy”)
Set(_SET_EPOCH, 1980)
nError:= 0
```

```
nError:= Com_Addr(COM1, 568, 3)
if nError <> 0
        return 1
endif
nError:= XPnmOpen(COM1,9600,256,256,XPN_COMHALF,2)
if nError <> 0
        return 1
endif
nError:= XPnmRemote(COM1, 31, XPN_REMOTE)
if nError <> 0
        return 1
endif
nError:= XPnmTxClk(COM1, 31)
if nError <> 0
        return 1
endif
nError:= XPnmRemote(COM1, 31, XPN_LOCAL)
if nError <> 0
        return 1
endif
nError:= XPnmClose(COM1)
if nError <> 0
        return 1
endif
quit
```

XPnmRxFile

Sumário

XPnmRxFile (nCanal, nTerm, cArq, nNumArqTerm, cbShowMsg)

Descrição

A função 'XPnmRxFile' permite a recepção de um arquivo de dados em formato XPbasic de um terminal para o concentrador da rede XPnet. Para a execução desta função exige-se que o terminal esteja em modo remoto, isto pode ser feito executando-se previamente a função 'XPnmRemote'. Durante o processo de recepção de arquivo pode-se mostrar na tela o número de blocos recebidos usando uma função local chamada a cada fim de bloco. Esta função pode testar o teclado e retornar um valor diferente de zero para abortar a operação.

O argumento 'nCanal' especifica o canal de comunicação, possibilita o uso de no máximo 16 canais (COM1 a COM16).

O argumento 'nTerm' informa o número do terminal, que deve ser de 1 a 32.

O argumento tipo caractere 'cArq' informa o nome e a localização com o qual o arquivo de dados proveniente do terminal deverá ser gravado no concentrador.

O argumento 'nNumArqTerm' informa o número do arquivo no terminal que se deseja receber.

O argumento ‘cbShowMsg’ é o nome da função local que recebe o número de registros processados e pode ser usada escrever este número na tela. Esta função pode também abortar a recepção se testar o teclado e retornar .F. quando determinada tecla for pressionada, retornando .T. a recepção continua. O código de erro retornado é CF_USERABRT. Caso não se deseje usar esta função fazer cbShowMsg = NIL.

Valor Retornado

A função retorna zero quando sua execução for bem sucedida, caso contrário retorna um código de erro sempre menor do que zero. O significado de cada código de erro retornado é fornecido no item “Códigos de Erro”.

Consulte

XPnmRemote, XPnmWaitTx, XPnmTxClk, XPnmTxFile, XPnmRxProg, XPnmTxProg.

Exemplo

```
#include “xpcomdef.h”

cNomeArq:= ““
nError:= 0

nError:= Com_Addr(COM1, 568, 3)
if nError <> 0
      return 1
endif
nError:= XPnmOpen(COM1,9600,256,256,XPN_COMHALF,2)
if nError <> 0
      return 1
endif
nError:= XPnmRemote(COM1, 31, XPN_REMOTE)
if nError <> 0
      return 1
endif
cNomeArq:= “C:\TRIX\RXARQ.BIN”
nError:= XPnmRxFile(COM1, 31, cNomeArq, 3,{|nBlocos| cbExibeMsg(nBlocos)})
if nError == XPN_USERABORT
      ? “Abortado pelo usuario”
elseif nError <> 0
      XPnmClose(COM1)
      return 1
endif
nError:= XPnmRemote(COM1, 31, XPN_LOCAL)
if nError <> 0
      return 1
endif
nError:= XPnmClose(COM1)
if nError <> 0
      return 1
endif
quit

Function ExibeMsg(nNblocos)

      ? “Numero de blocos recebidos: ”, nNblocos
      return .T.
```

XPnmTxFile

Sumário

XPnmTxFile (nCanal, nTerm, cArq, nNumArqTerm, cbShowMsg)

Descrição

A função 'XPnmTxFile' transmite um arquivo de dados em formato XPbasic do concentrador para um terminal da rede XPnet. Para a execução desta função exige-se que o terminal esteja em modo remoto, isto pode ser feito executando-se previamente a função 'XPnmRemote'. Durante o processo de transmissão de arquivo pode-se mostrar na tela o número de blocos transmitidos e o número de blocos total usando uma função local chamada a cada fim de bloco. Esta função pode testar o teclado e retornar um valor diferente de zero para abortar a operação.

O argumento 'nCanal' especifica o canal de comunicação, possibilita o uso de no máximo 16 canais (COM1 a COM16).

O argumento 'nTerm' informa o número do terminal, que deve ser de 1 a 32.

O argumento tipo caractere 'cArq' informa o nome e a localização do o arquivo de dados no concentrador que deverá ser enviado ao terminal.

O argumento 'nNumArqTerm' informa o número do arquivo no terminal para o qual se deseja transmitir.

O argumento ‘cbShowMsg’ é o nome da função local que recebe o número de blocos já transmitidos e o número total de blocos e pode ser usada para escrever estes números na tela. Esta função pode também abortar a transmissão se testar o teclado e retornar .F. quando determinada tecla for pressionada, retornando .T. a recepção continua. O código de erro retornado é XPN_USERABORT. Caso não se deseje usar esta função fazer cbShowMsg = NIL.

Valor Retornado

A função retorna zero quando sua execução for bem sucedida, caso contrário retorna um código de erro sempre menor do que zero. O significado de cada código de erro retornado é fornecido no item “Códigos de Erro”.

Consulte

XPnmRemote, XPnmWaitTx, XPnmTxClk, XPnmRxFile, XPnmRxProg, XPnmTxProg.

Exemplo

```
#include “xpcomdef.h”

cNomeArq:= ““
nError:= 0

nError:= Com_Addr(COM1, 568, 3)
if nError <> 0
      return 1
endif
nError:= XPnmOpen(COM1,9600,256,256,XPN_COMHALF,2)
if nError <> 0
      return 1
endif
nError:= XPnmRemote(COM1, 31, XPN_REMOTE)
if nError <> 0
      return 1
endif
cNomeArq:= “C:\TRIX\TXARQ.BIN”
nError:= XPnmTxFile(COM1, 31, cNomeArq, 3, {|nBlocos, nTotal| cbExibeMsg(nBlocos, nTotal)})
if nError == XPN_USERABORT
      ? “Abortado pelo usuario”
elseif nError <> 0
      XPnmClose(COM1)
      return 1
endif
nError:= XPnmRemote(COM1, 31, XPN_LOCAL)
if nError <> 0
      return 1
endif
nError:= XPnmClose(COM1)
if nError <> 0
      return 1
endif
quit

Function ExibeMsg(nNblocos, nNtotal)

      ? “Numero de blocos recebidos: ”, nNblocos, “/”, nNtotal
      return .T.
```

XPnmRxProg

Sumário

XPnmRxProg (nCanal, nTerm, cArq, cbShowMsg)

Descrição

A função 'XPnmRxProg' permite a recepção de um arquivo de programas em formato XPbasic de um terminal para o concentrador da rede XPnet. Para a execução desta função exige-se que o terminal esteja em modo remoto, isto pode ser feito executando-se previamente a função 'XPnmRemote'. Durante o processo de recepção de programa pode-se mostrar na tela o número de blocos recebidos usando uma função local chamada a cada fim de bloco. Esta função pode testar o teclado e retornar um valor diferente de zero para abortar a operação.

O argumento 'nCanal' especifica o canal de comunicação, possibilita o uso de no máximo 16 canais (COM1 a COM16).

O argumento 'nTerm' informa o número do terminal, que deve ser de 1 a 32.

O argumento tipo caractere 'cArq' informa o nome e a localização com o qual o arquivo de programa proveniente do terminal deverá ser gravado no concentrador.

O argumento ‘cbShowMsg’ é o nome da função local que recebe o número de blocos já recebidos e pode ser usada escrever este número na tela. Esta função pode também abortar a recepção se testar o teclado e retornar .F. quando determinada tecla for pressionada, retornando .T. a recepção continua. O código de erro retornado é XPN_USERABORT. Caso não se deseje usar esta função fazer cbShowMsg = NIL.

Valor Retornado

A função retorna zero quando sua execução for bem sucedida, caso contrário retorna um código de erro sempre menor do que zero. O significado de cada código de erro retornado é fornecido no item “Códigos de Erro”.

Consulte

XPnmRemote, XPnmWaitTx, XPnmTxClk, XPnmRxFile, XPnmTxFile, XPnmTxProg.

Exemplo

```
#include “xpcomdef.h”

cNomeArq:= ““
nError:= 0

nError:= Com_Addr(COM1, 568, 3)
if nError <> 0
        return 1
endif
nError:= XPnmOpen(COM1,9600,256,256,XPN_COMHALF,2)
if nError <> 0
        return 1
endif
nError:= XPnmRemote(COM1, 31, XPN_REMOTE)
if nError <> 0
        return 1
endif
cNomeArq:= “C:\TRIX\RXPROG.BAS”
nError:= XPnmRxProg(COM1, 31, cNomeArq, {|nBlocos| cbExibeMsg(nBlocos)})
if nError == XPN_USERABORT
        ? “Abortado pelo usuario”
elseif nError <> 0
        XPnmClose(COM1)
        return 1
```

```
endif
nError:= XPnmRemote(COM1, 31, XPN_LOCAL)
if nError <> 0
        return 1
endif
nError:= XPnmClose(COM1)
if nError <> 0
        return 1
endif
quit

Function ExibeMsg(nNblocos)

        ? “ Numero de blocos recebidos: ”, nNblocos
        return .T.
```

XPnmTxProg

Sumário

XPnmTxProg (nCanal, nTerm, cArq, cbShowMsg)

Descrição

A função 'XPnmTxFile' transmite um arquivo de programa em formato XPbasic do concentrador para um terminal da rede XPnet. Para a execução desta função exige-se que o terminal esteja em modo remoto, isto pode ser feito executando-se previamente a função 'XPnmRemote'. Durante o processo de transmissão de programa pode-se mostrar na tela o número de blocos transmitidos e o número de blocos total usando uma função local chamada a cada fim de bloco. Esta função pode testar o teclado e retornar um valor diferente de zero para abortar a operação.

O argumento 'nCanal' especifica o canal de comunicação, possibilita o uso de no máximo 16 canais (COM1 a COM16).

O argumento 'nTerm' informa o número do terminal, que deve ser de 1 a 32.

O argumento tipo caractere 'cArq' informa o nome e a localização do o arquivo de programa no concentrador que deverá ser enviado ao terminal.

O argumento ‘cbShowMsg’ é um ponteiro para uma função local que recebe o número de blocos já transmitidos e o número total de blocos e pode ser usada para escrever estes números na tela. Esta função pode também abortar a transmissão se testar o teclado e retornar .F. quando determinada tecla for pressionada, retornando .T. a recepção continua. O código de erro retornado é XPN_USERABORT. Caso não se deseje usar esta função fazer cbShowMsg = NIL.

Valor Retornado

A função retorna zero quando sua execução for bem suce+dida, caso contrário retorna um código de erro sempre menor do que zero. O significado de cada código de erro retornado é fornecido no item “Códigos de Erro”.

Consulte

XPnmRemote, XPnmWaitTx, XPnmTxClk, XPnmRxFile, XPnmTxFile, XPnmRxProg.

Exemplo

```
#include “xpcomdef.h”

cNomeArq:= ““
nError:= 0

nError:= Com_Addr(COM1, 568, 3)
if nError <> 0
        return 1
endif
```

```
nError:= XPnmOpen(COM1,9600,256,256,XPN_COMHALF,2)
if nError <> 0
        return 1
endif
nError:= XPnmRemote(COM1, 31, XPN_REMOTE)
if nError <> 0
        return 1
endif
cNomeArq:= “C:\TRIX\TXPROG.BAS”
nError:= XPnmTxProg(COM1, 31, cNomeArq, {|nBlocos, nTotal| cbExibeMsg(nBlocos, nTotal)})
if nError == XPN_USERABORT
        ? “Abortado pelo usuario”
elseif nError <> 0
        XPnmClose(COM1)
        return 1
endif
nError:= XPnmRemote(COM1, 31, XPN_LOCAL)
if nError <> 0
        return 1
endif
nError:= XPnmClose(COM1)
if nError <> 0
        return 1
endif
quit

Function ExibeMsg(nNblocos, nNtotal)

        ? “Numero de blocos transmitidos: ”, nNblocos, “/”, nNtotal
        return .T.
```

Funções de Comunicação para Protocolo XMODEM

Estas funções são destinadas a fornecer funções adicionais às funções básicas de comunicação, para que se possa realizar comunicação com os coletores de dados TRIX utilizando o protocolo XMODEM com *check sum*. Para utilizar estas funções é necessário que o canal esteja inicializado operacionalmente.

Xm_RxFile

Sumário

Xm_RxFile (nCanal, cArq, cbShowMsg)

Descrição

A função 'Xm_RxFile' recebe um arquivo, de dados ou programa, do coletor de dados TRIX para um concentrador utilizando protocolo XMODEM com *check sum*. O canal de comunicação deve ter sido previamente inicializado operacionalmente.

O argumento 'nCanal' especifica o canal de comunicação, possibilita o uso de no máximo 16 canais (COM1 a COM16).

O argumento tipo caractere 'cArq' informa o nome e a localização com o qual o arquivo proveniente do terminal deverá ser gravado no concentrador.

O argumento ‘cbShowMsg’ é o nome da função local que recebe o número de blocos já recebidos e pode ser usada escrever este número na tela. Esta função pode também abortar a recepção se testar o teclado e retornar .F. quando determinada tecla for pressionada, retornando .T. a recepção continua. O código de erro retornado é XPN_USERABORT. Caso não se deseje usar esta função fazer cbShowMsg = NIL.

Valor Retornado

A função retorna zero quando a execução for bem sucedida, caso contrário retorna um código de erro sempre menor do que zero. O significado de cada código de erro retornado é fornecido no item “Códigos de Erro”.

Consulte

Xm_TxFile.

Exemplo

```
#include “xpcomdef.h”

cNomeArq:= ““
nError:= 0

nError:= Com_Addr(COM1, 568, 3)
if nError <> 0
      return 1
endif
nError:= XPnmOpen(COM1,9600,256,256,XPN_COMHALF,2)
if nError <> 0
      return 1
endif
cNomeArq:= “C:\TRIX\RXARQ.DAT”
nError:= Xm_RxFile(COM1, cNomeArq,{|nBlocos| cbExibeMsg(nBlocos)})
if nError == XPN_USERABORT
      ? “ Abortado pelo usuario”
elseif nError <> 0
      XPnmClose(COM1)
      return 1
endif
nError:= XPnmClose(COM1)
if nError <> 0
      return 1
endif
quit

Function cbExibeMsg(nNblocos)

      ? “ Numero de blocos recebidos: ”, nNblocos
      return .T.
```

Xm_TxFile

Sumário

Xm_TxFile (nCanal, cArq, cbShowMsg)

Descrição

A função 'Xm_TxFile' transmite um arquivo, de dados ou programa, do concentrador para um coletor de dados TRIX utilizando o protocolo XMODEM com *check sum*. O canal de comunicação deve ter sido previamente inicializado operacionalmente.

O argumento 'nCanal' especifica o canal de comunicação, possibilita o uso de no máximo 16 canais (COM1 a COM16).

O argumento tipo caractere 'cArq' informa o nome e a localização com o qual o arquivo proveniente do terminal deverá ser gravado no concentrador.

O argumento ‘cbShowMsg’ é o nome da função local que recebe o número de blocos já transmitidos e o número total de blocos e pode ser usada para escrever estes números na tela. Esta função pode também abortar a transmissão se testar o teclado e retornar .F. quando determinada tecla for pressionada, retornando .T. a transmissão continua. O código de erro retornado é XPN_USERABORT. Caso não se deseje usar esta função fazer cbShowMsg = NIL.

Valor Retornado

A função retorna zero quando a execução for bem sucedida, caso contrário retorna um código de erro sempre menor do que zero. O significado de cada código de erro retornado é fornecido no item “Códigos de Erro”.

Consulte

Xm_RxFile.

Exemplo

```
#include “xpcomdef.h”

cNomeArq:= ““
nError:= 0

nError:= Com_Addr(COM1, 568, 3)
if nError <> 0
      return 1
endif
nError:= XPnmOpen(COM1,9600,256,256,XPN_COMHALF,2)
if nError <> 0
      return 1
endif
cNomeArq:= “C:\TRIX\TXARQ.BIN”
nError:= Xm_TxFile(COM1, cNomeArq, {|nBlocos, nTotal| cbExibeMsg(nBlocos, nTotal)})
if nError == XPN_USERABORT
      ? “Abortado pelo usuario”
elseif nError <> 0
      XPnmClose(COM1)
      return 1
endif
nError:= XPnmClose(COM1)
if nError <> 0
      return 1
endif
quit

Function ExibeMsg(nNblocos, nNtotal)

      ? “Numero de blocos transmitidos: ”, nNblocos, “/”, nNtotal
      return .T.
```

Funções de Conversão de Formato

Estas funções complementam as funções de comunicação propriamente ditas, fornecendo facilidades de conversão do formato XPbasic para ASCII, TEXTO ou DBASE e vice versa.

Descrição detalhada do formato XPbasic pode ser encontrada no Manual de Referência Técnica dos coletores de dados. Este formato admite três tipos de campo: Campo de valor inteiro com valores de -32768 até 32767, campo de valor em ponto flutuante com valores de -1.0E-154 até 5.2E+151 e campo alfanumérico contendondo de 1 a 80 caracteres. São admitidos um máximo de 10 campos.

Arquivo no formato ASCII admite os três tipos de campo, separando-os por vírgula. Os campos podem ter tamanhos variáveis mas precisam estar dentro do tamanho aceito pelo XPbasic. Campos alfanuméricos devem iniciar com aspas e terminar com aspas seguida de vírgula. São admitidas aspas dentro do campo contanto que não sejam seguidas de vírgula.

Arquivo no formato TEXTO só admite um campo alfanumérico de tamanho fixo.

Arquivo no formato DBASE e LOTUS devem conter campos dentro dos limites aceitos pelo formato XPbasic.

ConvASC2XP

Sumário

ConvASC2XP(cFileOrig, cFileDest, aEstrut, cbShowMsg)

Descrição

A função 'ConvASC2XP' converte o arquivo fornecido no formato ASCII para formato XPbasic.

O argumento 'cFileOrig' é uma cadeia de caracteres com o nome do arquivo origem.

O argumento 'cFileDest' é uma cadeia de caracteres com o nome do arquivo destino.

O argumento 'aEstrut' é um vetor com a estrutura do arquivo origem. Cada posição deste vetor contém a descrição de um campo e deve conter I para campo inteiro ou S seguido de um número, que representa o número de caracteres neste campo, para campo string. O formato, F para campo em ponto flutuante não é suportado.

O argumento 'cbShowMsg' é o nome da função local que recebe o número de registros já convertidos e pode ser usada escrever este número na tela. Esta função pode também abortar a recepção se testar o teclado e retornar .F. quando determinada tecla for pressionada, retornando .T. a recepção continua. O código de erro retornado é CF_USERABRT. Caso não se deseje usar esta função fazer cbShowMsg = NIL.

Valor Retornado

A função retorna zero quando a execução for bem sucedida, caso contrário retorna um código de erro sempre menor do que zero. O significado de cada código de erro retornado é fornecido no item "Códigos de Erro".

Consulte

ConvDB2XP, ConvTXT2XP, ConvXP2ASC, ConvXP2DB, ConvXP2TXT.

Exemplo

```
#include "xpcomdef.h"

cInFile:= ""
cOutFile:= ""
aFileFormat:= {}
nError:= 0

aFileFormat[0] = "I"
aFileFormat[1] = "S20"
cInFile:= "C:\TRIX\ARQASC.DAT"
cOutFile:= "C:\TRIX\ARQXPB.DAT"
nErro = ConvASC2XP(cInFile, cOutFile, aFileFormat, {|nRegistros| cbExibeMsg(nRegistros)})
if nError == CF_USERABRT
    ? "Abortado pelo usuario"
elseif nError <> 0
    return 1
endif
quit

Function ExibeMsg(nRegistros)

    ? "Numero de registros recebidos: ", nRegistros
    return .T.
```

ConvDB2XP

Sumário

ConvDB2XP(cFileOrig, cFileDest, aEstrut, cbShowMsg)

Descrição

A função 'ConvDB2XP' converte o arquivo fornecido no formato DBASE para formato XPbasic.

O argumento 'cFileOrig' é uma cadeia de caracteres com o nome do arquivo origem.

O argumento 'cFileDest' é uma cadeia de caracteres com o nome do arquivo destino.

O argumento 'aEstrut' é um vetor com a estrutura do arquivo origem. Cada posição deste vetor contém a descrição de um campo e deve conter I para campo inteiro ou S seguido de um número, que representa o número de caracteres neste campo, para campo string. O formato, F para campo em ponto flutuante não é suportado nem campo tipo DATA no arquivo DBASE.

O argumento 'cbShowMsg' é o nome da função local que recebe o número de registros já convertidos e pode ser usada escrever este número na tela. Esta função pode também abortar a recepção se testar o teclado e retornar .F. quando determinada tecla for pressionada, retornando .T. a recepção continua. O código de erro retornado é CF_USERABRT. Caso não se deseje usar esta função fazer cbShowMsg = NIL.

Valor Retornado

A função retorna zero quando a execução for bem sucedida, caso contrário retorna um código de erro sempre menor do que zero. O significado de cada código de erro retornado é fornecido no item "Códigos de Erro".

Consulte

ConvASC2XP, ConvTXT2XP, ConvXP2ASC, ConvXP2DB, ConvXP2TXT.

Exemplo

```
#include "xpcomdef.h"

cInFile:= ""
cOutFile:= ""
aFileFormat:= {}
nError:= 0

aFileFormat[0] = "I"
aFileFormat[1] = "S20"
cInFile:= "C:\TRIX\ARQDB.DBF"
cOutFile:= "C:\TRIX\ARQXPB.DAT"
nErro = ConvDB2XP(cInFile, cOutFile, aFileFormat, {|nRegistros| cbExibeMsg(nRegistros)})
if nError == CF_USERABRT
    ? "Abortado pelo usuario"
elseif nError <> 0
    return 1
endif
quit

Function ExibeMsg(nRegistros)

    ? "Numero de registros recebidos: ", nRegistros
    return .T.
```

ConvTXT2XP

Sumário

ConvTXT2XP(cFileOrig, cFileDest, aEstrut, cbShowMsg)

Descrição

A função 'ConvTXT2XP' converte o arquivo fornecido no formato TEXTO para formato XPbasic.

O argumento 'cFileOrig' é uma cadeia de caracteres com o nome do arquivo origem.

O argumento 'cFileDest' é uma cadeia de caracteres com o nome do arquivo destino.

O argumento 'aEstrut' é um vetor com a estrutura do arquivo origem. Cada posição deste vetor contém a descrição de um campo e deve conter I para campo inteiro ou S seguido de um número, que representa o número de caracteres neste campo, para campo string. O formato, F para campo em ponto flutuante não é suportado.

O argumento ‘cbShowMsg’ é o nome da função local que recebe o número de registros já convertidos e pode ser usada escrever este número na tela. Esta função pode também abortar a recepção se testar o teclado e retornar .F. quando determinada tecla for pressionada, retornando .T. a recepção continua. O código de erro retornado é CF_USERABRT. Caso não se deseje usar esta função fazer cbShowMsg = NIL.

Valor Retornado

A função retorna zero quando a execução for bem sucedida, caso contrário retorna um código de erro sempre menor do que zero. O significado de cada código de erro retornado é fornecido no item “Códigos de Erro”.

Consulte

ConvDB2XP, ConvASC2XP, ConvXP2ASC, ConvXP2DB, ConvXP2TXT.

Exemplo

```
#include “xpcomdef.h”

cInFile:= ““
cOutFile:= ““
aFileFormat:= {}
nError:= 0

aFileFormat[0] = “I”
aFileFormat[1] = “S20”
cInFile:= “C:\TRIX\ARQTXT.DAT”
cOutFile:= “C:\TRIX\ARQXPB.DAT”
nErro = ConvTXT2XP(cInFile, cOutFile, aFileFormat, {|nRegistros| cbExibeMsg(nRegistros)})
if nError == CF_USERABRT
      ? “ Abortado pelo usuario”
elseif nError <> 0
      return 1
endif
quit

Function ExibeMsg(nRegistros)

      ? “ Numero de registros recebidos: ”, nRegistros
      return .T.
```

ConvXP2ASC

Sumário

ConvXP2ASC(cFileOrig, cFileDest, cbShowMsg)

Descrição

A função 'ConvXP2ASC' converte o arquivo fornecido no formato XPbasic para formato ASCII. A conversão é feita convertendo-se todos os campos tipo inteiro e ponto flutuante para caracteres.

O argumento 'cFileOrig' é uma cadeia de caracteres com o nome do arquivo origem.

O argumento 'cFileDest' é uma cadeia de caracteres com o nome do arquivo destino.

O argumento ‘cbShowMsg’ é o nome da função local que recebe o número de registros já convertidos e pode ser usada escrever este número na tela. Esta função pode também abortar a recepção se testar o teclado e retornar .F. quando determinada tecla for pressionada, retornando .T. a recepção continua. O código de erro retornado é CF_USERABRT. Caso não se deseje usar esta função fazer cbShowMsg = NIL.

Valor Retornado

A função retorna zero quando a execução for bem sucedida, caso contrário retorna um código de erro sempre menor do que zero. O significado de cada código de erro retornado é fornecido no item “Códigos de Erro”.

Consulte

ConvASC2XP, ConvDB2XP, ConvTXT2XP, ConvXP2DB, ConvXP2TXT.

Exemplo

```
#include “xpcomdef.h”

cInFile:= ““
cOutFile:= ““
nError:= 0

cInFile:= “C:\TRIX\ARQXPB.DAT”
cOutFile:= “C:\TRIX\ARQASC.DAT”
nErro = ConvXP2ASC(cInFile, cOutFile, {|nRegistros| cbExibeMsg(nRegistros)})
if nError == CF_USERABRT
      ? “Abortado pelo usuario”
elseif nError <> 0
      return 1
endif
quit

Function ExibeMsg(nRegistros)

      ? “Numero de registros recebidos: ”, nRegistros
      return .T.
```

ConvXP2DB

Sumário

ConvXP2DB(cFileOrig, cFileDest, aEstrut, cbShowMsg)

Descrição

A função 'ConvXP2DB' converte o arquivo fornecido no formato XPbasic para formato DBASE.

O argumento 'cFileOrig' é uma cadeia de caracteres com o nome do arquivo origem.

O argumento 'cFileDest' é uma cadeia de caracteres com o nome do arquivo destino.

O argumento 'aEstrut' é uma matiz com a estrutura do arquivo DBASE destino, é usado o mesmo formato utilizado pelo DBCREATE. Cada linha desta matriz contém quatro valores com a descrição de um campo e pode ter um máximo de 10 linhas. Os valores nas linhas são: Nome do campo, Tipo do campo, Tamanho do campo e Casas decimais do campo. O formato, F para campo em ponto flutuante não é suportado.

A conversão de campos será feita num dos modos abaixo:

Tipo do Campo	
N	Inteiro será convertido para campo numérico no DBASE.
C	Convertido para campo caractere no DBASE.
D	Convertido para Data se string possuir exatamente 8 caracteres

O argumento ‘cbShowMsg’ é o nome da função local que recebe o número de registros já convertidos e pode ser usada escrever este número na tela. Esta função pode também abortar a recepção se testar o teclado e retornar .F. quando determinada tecla for pressionada, retornando .T. a recepção continua. O código de erro retornado é CF_USERABRT. Caso não se deseje usar esta função fazer cbShowMsg = NIL.

Valor Retornado

A função retorna zero quando a execução for bem sucedida, caso contrário retorna um código de erro sempre menor do que zero. O significado de cada código de erro retornado é fornecido no item “Códigos de Erro”.

Consulte

ConvASC2XP, ConvDB2XP, ConvTXT2XP, ConvXP2ASC, ConvXP2TXT.

Exemplo

```
#include “xpcomdef.h”

cInFile:= ““
cOutFile:= ““
aDBFormat:= {}
nError:= 0

aDBFormat := {{“CAMPO1”,”N”, “5”, “0”}, {“CAMPO2”,”C”, “20”, “0”}}
cInFile:= “C:\TRIX\ARQXPB.DAT”
cOutFile:= “C:\TRIX\ARQDB.DBF”
nErro = ConvXP2DB(cInFile, cOutFile, aDBFormat, {|nRegistros| cbExibeMsg(nRegistros)})
if nError == CF_USERABRT
      ? “Abortado pelo usuario”
elseif nError <> 0
      return 1
endif
quit

Function ExibeMsg(nRegistros)

      ? “Numero de registros recebidos: ”, nRegistros
      return .T.
```

ConvXP2TXT

Sumário

ConvXP2TXT(cFileOrig, cFileDest, cbShowMsg)

Descrição

A função 'ConvXP2TXT' converte o arquivo fornecido no formato XPbasic para formato TEXTO. A conversão é feita convertendo-se todos os campos tipo inteiro e ponto flutuante para caracteres.

O argumento 'cFileOrig' é uma cadeia de caracteres com o nome do arquivo origem.

O argumento 'cFileDest' é uma cadeia de caracteres com o nome do arquivo destino.

O argumento ‘cbShowMsg’ é o nome da função local que recebe o número de registros já convertidos e pode ser usada escrever este número na tela. Esta função pode também abortar a recepção se testar o teclado e retornar .F. quando determinada tecla for pressionada, retornando .T. a recepção continua. O código de erro retornado é CF_USERABRT. Caso não se deseje usar esta função fazer cbShowMsg = NIL.

Valor Retornado

A função retorna zero quando a execução for bem sucedida, caso contrário retorna um código de erro sempre menor do que zero. O significado de cada código de erro retornado é fornecido no item “Códigos de Erro”.

Consulte

ConvASC2XP, ConvDB2XP, ConvTXT2XP, ConvXP2DB, ConvXP2ASC.

Exemplo

```
#include “xpcomdef.h”

cInFile:= ““
cOutFile:= ““
nError:= 0

cInFile:= “C:\TRIX\ARQXPB.DAT”
cOutFile:= “C:\TRIX\ARQTXT.DAT”
nErro = ConvXP2TXT(cInFile, cOutFile, {|nRegistros| cbExibeMsg(nRegistros)})
if nError == CF_USERABRT
      ? “Abortado pelo usuario”
elseif nError <> 0
      return 1
endif
quit

Function ExibeMsg(nRegistros)

      ? “Numero de registros recebidos: ”, nRegistros

      return .T.
```

Programas de exemplo para operação ON-Line

Programa a ser executado no PC

Foi incluído no diretório WINDOWS\DEMO\DELPHI32\ONLINE um exemplo de programa que recebe do coletor uma marcacao enviada pelo programa ONLINE.BAS e responde ao coletor com uma mensagem padrão.

Deve ser incluído no diretório do programa o arquivo ONLINE.BAS para poder enviá-lo ao coletor. Este programa está no diretório dos arquivos em XPBASIC.

Para compilar sera necessário incluir no diretório o arquivo XPCOM32.PAS que faz parte do diretório das DLL da biblioteca.

Programa ONLINE.BAS para controle de coletores ON-LINE

O programa de demonstração ONLINE.BAS escrito em XPBASIC incorpora transações para a leitura de cartões no coletor e o envio de comandos para o acionamento de relés, beep e exibição de mensagens. Permite também que o concentrador saiba se houve o acionamento de chaves, indicanto entrada ou saída por porta ou catraca.

Comandos para Controle de acesso ON LINE

No controle de acesso ON LINE o coletor envia o código lido para o PC e este envia ao coletor as ações a serem tomadas. O coletor aguarda esta resposta do PC por 5 segundos antes de escrever a mensagem COLETOR OFF-LINE na tela e voltar a aguardar uma nova marcação.

O coletor executa as ações especificadas pelo PC, acionando os relés, tocando o beep e exibindo a mensagem pelo tempo especificado e examina as chaves para verificar se houve algum acionamento. Em seguida envia ao PC uma resposta com os mesmos dados da última leitura e informação sobre os acionamentos ocorridos. Desta forma o PC não precisa guardar informação sobre quais acionamentos ele solicitou, pois sempre recebe todos os dados da leitura.

Não será enviada uma resposta PC, com os acionamentos de chave, quando a catraca está no repouso e o coletor recebe um comando com os tempos dos relés igual a zero binário.

Toda mensagem enviada do coletor para o PC deverá ser respondida, caso contrário o coletor mostrará a mensagem "COLETOR OFF-LINE".

As leituras de cartão feitas no coletor são enviadas ao PC no formato abaixo:

O<leitura><código><leitor><data><hora>0000000000<4 espaços em branco>

bytes:	26	1	1	8	8	10	4

leitura - Caracteres ASCII com o código lido do cartão ou entrado pelo cartão mestre. Será preenchido de espaços em brancos à direita até completar 26 bytes.

código - Informa o motivo pelo qual esta leitura está sendo enviada:
0 - Quando o cartão foi lido por um dos leitores.

Sempre que a mensagem, com a ação a ser tomada, enviada pelo PC ao coletor, tiver o tempo do rele 1 ou o tempo do rele 2 diferente de zero binário, o coletor envia a última leitura para o PC, substituindo o código 0 acima por:

A - Catraca não girou apos transcorrido tempo do relé.
B - Catraca girou para entrar.
C - Catraca girou para sair.
D - Coletor detectou ambas chaves de giro da catraca (erro de ligação).
E - Catraca estava fora do repouso quando o cartão foi passado. Passagem foi bloqueada.
Este código é retornado ao PC independentemente do tempo dos relés.

leitor - Indica qual foi o leitor onde o cartão foi lido:
n = 2 para cartão lido pelo leitor de barras 1 (padrão).
n = 3 para cartão lido pelo leitor de barras 2.
n = 4 para cartão lido pelo leitor magnético 1.
n = 5 para cartão lido pelo leitor magnético 2.

data - Caracteres ASCII com a data no formato dd/mm/aa

hora - Caracteres ASCII com a hora no formato hh/mm/ss

número - fixo 0000000000.

tempo ms - fixo 4 espaços em branco..

O PC responde com a ação a ser tomada pelo coletor enviando uma string no formato:

	OC<tmp relé 1>	<tmp relé 2>	<tmp beep>	<tmp msg>	<mensagem>	<1 espaço em branco>
bytes:	1(bin)	1(bin)	1(bin)	1(bin)	40	1

OC - Prefixo do comando, indica que o PC está enviando um comando on-line para ser executado pelo coletor. Este comando especifica os tempos de acionamento dos relés do coletor, do beep e da mensagem, assim como a mensagem a ser exibida.

tmp relé 1 - Um byte em binário com o tempo para acionar relé 1 de 0 a 255 em unidades de décimos de segundo. Valor 1 corresponde a 0,1 seg, valor 255 corresponde a 25,5 seg. Valor 0 não aciona o relé.

tmp relé 2 - Um byte em binário com o tempo para acionar relé 2 de 0 a 255 em unidades de décimos de segundo. Valor 1 corresponde a 0,1 seg, valor 255 corresponde a 25,5 seg. Valor 0 não aciona o relé.

tmp beep - Um byte em binário com o tempo para tocar o beep de 0 a 255 em unidades de décimos de segundo. Valor 1 corresponde a 0,1 seg, valor 255 corresponde a 25,5 seg. Valor 0 não toca o beep.

tmp msg - Um byte em binário com o tempo para exibição da mensagem de 0 a 255 em unidades de décimos de segundo. Valor 1 corresponde a 0,1 seg, valor 255 corresponde a 25,5 seg.

mensagem - Caracteres ASCII com duas strings de 20 caracteres, uma para cada linha do display do coletor.

espaço em branco - Qualquer caractere. Foi mantido para compatibilidade com a versão 1.00 que exigia um byte em binário com o valor 0.

Comandos de comunicação OFF LINE com o coletor como receber ou enviar arquivo e mudar data e hora causarão a sua reinicialização.

Operação ON-LINE usando o XTMQUICK

O programa configurador de coletores XTMQUICK da TRIX a partir de sua versão 3.00 incorpora transações para operação ON-LINE. Este programa não é fornecido com a biblioteca de comunicação XPCOMLIB.

Segue abaixo descrição de como configurar o coletor para operação ON-LINE e o formato das transações.

Configuração do XTMQUICK para operação ON-LINE:

Habilitar no menu de **"edição"**, opcao **"comunicações do coletor"** o item
"PC controla acesso" com **'S'**;
Tempo para o coletor reverter para o modo, padrão **1,5 seg**;
O que deve ser feito no modo local:
Fazer consistencia com as listas locais
Bloquear todos
Liberar todos

Comandos para Controle de acesso ON LINE

No controle de acesso ON LINE o coletor envia os dados lidos para o PC e este envia ao coletor as ações a serem tomadas.

Os dados lidos podem ser a marcação lida do cartão completa incluindo os caracteres de identificacao, um código de cartão sem os caracteres de identificação entrado pelo supervisor ou um número entrado pelo usuário.

O arquivo de configuraçao indica se o controle de acesso é feito ON LINE ou local e qual ação o coletor deve tomar caso perca acesso ao PC. Se o coletor for configurado para tratar as marcações caso perca acesso ao PC, deve receber todos os arquivos para tratamento das marcações em modo local.

As leituras de cartão são enviadas ao PC no formato abaixo:

O<leitura><código><data><hora><número><tempo ms>

bytes:	26	2	8	8	10	4(binário, byte menos significativo antes)

leitura - Caracteres ASCII com o código lido do cartão ou entrado pelo cartão mestre. Será preenchido de espaços em brancos à direita até completar 26 bytes.

O campo **leitura** é sempre retornado junto com o valor numérico solicitado pelo PC, decorrente da leitura enviada anteriormente. Isto permite que o PC não precise guardar o estado do coletor e processe como uma nova leitura, retornada com valor numérico.

código - Caracteres ASCII com os códigos abaixo indicando condições especiais:

"0n" - Cartão lido via leitor n (n = 2, 3, 4 ou 5).
"4n" - Número abortado após cartão lido via leitor n (n = 2, 3, 4 ou 5).
"5n" - Número entrado após cartão lido via leitor n (n = 2, 3, 4 ou 5).
"10" a "19" - Tecla 0 a 9 acionada informando tipo de marcação definida pelo usuário, ver arquivo de marcação, campo tipo de marcação. Campo de dados retorna brancos no formato ASCII.
"20" - Chave interna acionada.
"21" - Chave principal acionada, quando configurada para sorteio. Registro 2 do arquivo de configuração deve informar sorteio e chave para sorteio.
"22" - Chave secundária acionada, quando configurada para sorteio. Registro 2 do arquivo de configuração deve informar sorteio e chave para sorteio.
"31" - Supervisor cancelou entrada pelo menu.
"33" - Supervisor alterou data e hora no coletor.
"62" a "65" - Supervisor solicita marcação fazendo consistência.
"72" a "75" - Supervisor solicita marcação sem fazer consistência.
"34" - Supervisor alterou estado do coletor ATIVO / INATIVO.
"36" - Supervisor inicializou modem.
"91" - Coletor liberado pelo supervisor.
"92" - Coletor está sem alimentação AC.
"93" - Coletor informa que está operando no modo off-line e pode ter marcações em sua memória.
"94" a "99" - Código de alarme conforme definido no arquivo de marcação, campo tipo de marcação. Campo de dados retorna brancos no formato ASCII.
"An" - Nenhuma chave acionada após cartão lido via leitor n (n = 2, 3, 4 ou 5).
"Bn" - Chave de entrada acionada após cartão lido via leitor n (n = 2, 3, 4 ou 5).
"Cn" - Chave de saída acionada após cartão lido via leitor n (n = 2, 3, 4 ou 5).
"Dn" - Ambas as chaves acionadas após cartão lido via leitor n (n = 2, 3, 4 ou 5).
"En" - Catraca não está no repouso após cartão lido via leitor n (n = 2, 3, 4 ou 5).

O código "n" especifica o leitor onde foi lido o cartão:

n = 2 para cartão lido via leitor de barra 1.
n = 3 para cartão lido via leitor de barra 2.
n = 4 para cartão lido via leitor magnético 1.
n = 5 para cartão lido via leitor magnético 2.

data - Caracteres ASCII com a data no formato dd/mm/aa

hora - Caracteres ASCII com a hora no formato hh/mm/ss

número - Caracteres ASCII com o número entrado pelo operador. Números permitidos vão de 0.000.000.000 a 3.999.999.999 (retornado sem os pontos). Entrada numérica abortada pelo operador, retorna brancos.

tempo ms - Quatro bytes em binário, byte menos significativo no início e mais significativo no final indicando quanto tempo durou a transação anterior, desde o instante que o coletor colocou a transação no seu buffer de transmissão até o momento em que o PC respondeu a esta transação.

O PC responde com a ação a ser tomada pelo coletor enviando uma string no formato:

O<entrada><tmp relé 1><tmp relé 2><tmp beep><tmp msg><msg><zero final>

bytes:	1	1(bin)	1(bin)	1(bin)	1(bin)	40	1(bin)

entrada - Caractere ASCII com uma das letras:

'C' - indica ao coletor que deve ser executado o comando atuando os dispositivos especificados.
Tempos são números binários de 0 a 255, valor de tempo = 0 não aciona dispositivo,
Mensagem são duas string de 20 caracteres ASCII, uma para cada linha.
'E' - Entra valores mostrando número no display.
'F' - Entra senhas mostrando asteríscos no display.
Solicita ao coletor que deve ser feita entrada de um valor numérico que será posteriormente enviado ao PC no campo número do registro de marcação com o código "52" a "55" ou "42" a "45" indicando a condição especial de

número recebido ou entrada numérica abortada pelo operador. Mensagens são mostradas no display e é solicitada entrada de dados na segunda linha do display. O maior número aceito é 3.999.999.999.
Coloque zero binário no meio da mensagem, onde o cursor deve se posicionar.

‘P’ - Coletor deve apresentar menu do supervisor parcial.

‘S’ - Coletor deve apresentar menu do supervisor completo.

Retornado ao concentrador a leitura digitada com os códigos 62 a 65 ou 72 a 75 ou com a leitura em branco para os códigos 31, 33, 34 e 36.

tmp relé 1 - Um byte em binário com o tempo para acionar relé 1 de 0 a 255 em unidades de décimos de segundo. Valor 1 corresponde a 0,1 seg, valor 255 corresponde a 25,5 seg. Valor 0 não aciona relé.

tmp relé 2 - Um byte em binário com o tempo para acionar relé 2 de 0 a 255 em unidades de décimos de segundo. Valor 1 corresponde a 0,1 seg, valor 255 corresponde a 25,5 seg. Valor 0 não aciona relé.

tmp beep - Um byte em binário com o tempo para tocar o beep de 0 a 255 em unidades de décimos de segundo. Valor 1 corresponde a 0,1 seg, valor 255 corresponde a 25,5 seg. Valor 0 não toca beep.

tmp msg - Um byte em binário com o tempo para exibição da mensagem de 0 a 255 em unidades de décimos de segundo. Valor 1 corresponde a 0,1 seg, valor 255 corresponde a 25,5 seg.

msg - Caracteres ASCII com duas strings de 20 caracteres, uma para cada linha do display do coletor.

zero final - Um byte em binário com o valor 0.

Se o PC não responder no tempo configurado no XTM-Quick (padrão 3 segundos) o coletor faz o tratamento local das marcações, bloqueia todas marcações ou libera todas, conforme arquivo de configuração. Marcação processada localmente ou aceita é armazenada até que seja restabelecida a comunicação com o PC.

O coletor envia mensagem para o concentrador com o campo **leitura** em branco e “93” no campo **código**. Informando que o coletor está operando no modo off-line e pode ter marcações em sua memória.

Para restabelecer comunicação ON LINE o PC deve recolher o arquivo com todas marcações armazenadas no coletor.

Comandos de comunicação OFF LINE com o coletor como Receber ou enviar arquivo, mudar data e hora causarão a sua reinicialização.

Apêndice A - Códigos de Erro

A seguinte lista de erros é referente aos erros retornados pelas funções da "Biblioteca de Funções de Comunicação TRIX". Qualquer valor menor do que 0 retornado por uma função indica que sua execução foi mal sucedida. As constantes de erros, e os respectivos números retornados pelas funções, são definidas no arquivo *header* da biblioteca, **xpcomdef.h xpcom16.h ou xpcom32.h,** e são descritas abaixo:

Os valores atribuídos às constantes não são iguais no ambiente DOS, no ambiente WINDOWS com 16 bits e no ambiente WINDOWS com 32 bits. Use o arquivo de inclusão correto para relacionar o nome da constante com o número retornado pela função.

Códigos de Erro retornados pelas Funções Básicas de Comunicação:

COM_INVPRM:	Parâmetros inválidos do Clipper. A interface Clipper / C detectou número ou tipo de parâmetros diferentes na chamada da função. Note que a função Com_Open tem um parâmetro a mais na XPCOMLIB versão 2.00.
COM_INVCOM:	Canal de comunicação inválido. Foi executada uma função da biblioteca, na qual foi fornecido como argumento um canal de comunicação inexistente.
COM_INVINT:	Número de interrupção de *hardware* inválido. Foi executada uma função da biblioteca, na qual foi fornecido como argumento um número de interrupção inválido.
COM_INVNBT:	Número de bits inválido. Foi executada uma função da biblioteca, na qual foi fornecido como argumento número de *bits* diferente de 7 ou 8.
COM_INVRAT:	Taxa de comunicação inválida. Foi executada uma função da biblioteca, na qual foi fornecido como argumento uma taxa de comunicação inexistente.
COM_INVNSB:	Número de *stop-bits* inválido. Foi executada uma função da biblioteca, na qual foi fornecido como argumento número de *stop-bits* diferente de 1 ou 2.
COM_INVPAR:	Paridade inválida. Foi executada uma função da biblioteca, na qual foi fornecido como argumento paridade diferente de COM_ODDP, COM_EVENP ou COM_NONEP.
COM_INVBUF:	Tamanho de *buffer* inválido. Foi executada uma função da biblioteca, na qual foi fornecido como argumento tamanho do *buffer* menor do que COM_MINRXB ou maior do que COM_MAXRXB.
COM_INVFLW:	Tipo de controle de fluxo inválido. Foi executada uma função da biblioteca, na qual foi fornecido como argumento controle de fluxo diferente de COM_NOFLOW, COM_RTSCTS ou COM_XONXOFF.
COM_ISOPEN:	Canal de comunicação já está aberto. Foi executada a função 'ComAddress' ou 'ComOpen' fornecendo como parâmetro um canal de comunicação que já foi aberto anteriormente.
COM_NTOPEN:	Canal de comunicação não está aberto. Foi executada uma função fornecendo como argumento um canal de comunicação que não foi aberto anteriormente.
COM_NOMEM:	Memória insuficiente para a operação. Não há memória suficiente a ser alocada para a execução da função.
COM_TXBFUL:	*Buffer* de transmissão cheio. *Buffer* de transmissão não possui espaço disponível para transmissão de mensagens.
COM_RXBEMP:	*Buffer* de recepção vazio. *Buffer* de recepção não contém mensagem armazenada.
COM_INVSTE:	Estado de sinal inválido. Foi executada uma função da biblioteca, na qual foi fornecido como argumento estado diferente de COM_ON ou COM_OFF.
COM_INVPORT:	Porta inválida ou erro de *hardware*. Ocorreu um erro no acesso à porta especificada.
COM_INVCFG:	Código de configuração inválido. Foi executada uma função fornecendo como argumento um parâmetro inválido.
COM_RXERROR:	Erro de recepção de caractere. Houve erro na recepção de caractere feita pelo windows.
COM_APIERROR:	Error retornado por uma chamada a uma função da API do Windows.

Códigos de Erro retornados pelas Funções Básicas de Comunicação com a Rede XPnet:

XPN_INVCOM:	Canal de comunicação inválido. Foi executada uma função da biblioteca, na qual foi fornecido como argumento um canal de comunicação inexistente.
XPN_INVTYPE:	Tipo de comunicação inválido. Foi executada a função 'XPnetmOpen' ou 'XPnmOpen' na qual foi fornecido como argumento tipo de comunicação diferente de XPN_COMHALF, XPN_COMAUTO ou XPN_COMFULL. Em DOS não é aceito o argumentos XPN_COMAUTO, Em Windows 95/98/NT não é aceito o argumento XPN_COMHALF.
XPN_INVRETRIES:	Número de tentativas de comunicação inválido. Foi executada a função 'XPnetmOpen' na qual foi fornecido como argumento número máximo de tentativas de retransmissão inválido.
XPN_ISOPEN:	Rede já está aberta para canal especificado. Foi executada a função 'XPnetmOpen' com a rede já inicializada para o canal especificado.
XPN_NOMEM:	Sem memória para operação da rede. Não há memória suficiente a ser alocada para a execução de uma função da rede XPnet.
XPN_NOTOPEN:	Rede não está aberta para canal especificado. Foi executada uma função fornecendo como argumento um canal de comunicação que não foi aberto anteriormente.
XPN_INVTERM:	Número do terminal inválido. Foi executada uma função, fornecendo como argumento 'nTerm' um número de terminal que não está conectado na rede.

XPN_NOTIMERS:	Sem *timers* disponíveis Não há *timers* a serem alocados para que sejam executadas as funções da biblioteca.
XPN_TXBFULL:	*Buffer* de transmissão de mensagens cheio. *Buffer* de transmissão não possui espaço disponível para transmissão de mensagens.
XPN_RXBEMPTY:	*Buffer* de recepção de mensagens vazio. *Buffer* de recepção não contém mensagem armazenada.
XPN_RXBFULL:	*Buffer* de recepção de mensagens cheio. *Buffer* de recepção de mensagens não possui espaço suficiente para receber mais mensagens. Deve ser executada a função 'XPnetmRxMsg' para que sejam retiradas as mensagens armazenadas no *buffer*.
XPN_RXTIMEOUT:	*Time out* durante a recepção. Ocorreu espera muito grande durante a execução de uma função de recepção.
XPN_RXCRC1:	Erro no CRC1. Durante a execução de uma função de recepção de mensagem, houve erro no cálculo do CRC1 (CRC do cabeçalho da mensagem).
XPN_RXCRC2:	Erro no CRC2. Durante a execução de uma função de recepção de mensagem, houve erro no cálculo do CRC2 (CRC dos dados da mensagem).
XPN_POLLON:	Varredura automática já está ativa. Foi executada a função 'XPnetmPollStart' sendo que esta já havia sido executada anteriormente.
XPN_POLLOFF:	Varredura automática não está ativa. Foi executada a função 'XPnetmPollStop' sem a execução prévia da função 'XPnetmPollStart', ou seja está se tentando desativar a varredura automática sem estar ativa.
XPN_POLLINVINT:	Intervalo de varredura automática inválido. Foi executada a função 'XPnetmPollStart' fornecendo-se como argumento 'wIntervalo' um tempo entre requisição de dados não permitido.
XPN_POLLINVN:	Número de requisições de varredura automática inválido. Foi executada a função 'XPnetmPollTerm' fornecendo-se como argumento 'nRepet' um número de repetições não permitido, sendo que este varia de 0 a 10.
XPN_TXTMOUT:	*Time out* durante a transmissão. Uma função de transmissão de dados foi executada, mas houve espera muito grande durante este processo.
XPN_ACKTIMEOUT:	*Time out* de espera de ACK. Ocorreu um problema durante a espera de ACK após ser executada uma função de transmissão de mensagens.
XPN_INVCFG:	Consulta de configuração inválida. Foi executada a função 'XPnetmGetConfig' fornecendo-se como argumento um código de informação 'nInfo' não suportado pela biblioteca XPnet.
XPN_SOHTIMEOUT:	T*ime out* de espera de SOH. Ocorreu um problema durante a espera de SOH após ser executada uma função de atenção ou *polling* a um terminal.
XPN_COMACTIVE:	Comunicação continua ativa após várias tentativas. Função de espera por fim de comunicação XPnetmWaitCom terminou por timeout.

Códigos de Erro retornados pelas Funções Auxiliares de Comunicação com a Rede XPnet:

XPN_NOTTXMSG:	Mensagem não foi transmitida.
XPN_OPENFILE:	Erro na abertura de arquivo - problema na abertura de um arquivo de dados ou programa durante a execução de uma função de transferência.
XPN_READFILE:	Problema na leitura do arquivo - problema na leitura de um arquivo de dados ou programa durante a execução de uma função de transferência para transmissão.
XPN_WRITEFILE:	Problema na escrita do arquivo - problema na escrita de um arquivo de dados ou programa durante a execução de uma função de transferência para recepção.
XPN_ERRBLOCK:	Recepção de bloco de dados inválido. Bloco de dados não está como esperado, com número de terminal errado ou tipo do bloco diferente de 0 (primeiro bloco), 1 (bloco intermediário) ou 2 último bloco.
XPN_USERABORT:	Aborto pelo usuário. Durante a execução de uma função da biblioteca houve a solicitação de aborto pelo usuário. Por exemplo se a função do usuário que exibe número de blocos retornar valor diferente de zero.
XPN_FILEEMPTY:	Arquivo do coletor sem dados. Foi executada uma função da biblioteca que pediu a transmissão de um arquivo do coletor que estava sem dados ou não foi recebido.
XPN_INVREMOP:	Modo de operação inválido. Foi executada a função ‘XPnetmRemote’ na qual foi fornecido como argumento tipo de operação diferente de XPN_REMOTE, XPN_LOCAL ou XPN_LOCALRUN.
XPN_ERRRXMSG:	Erro de recepção. Problema na recepção de um arquivo de dados ou programa numa função de transferência de recepção do tipo 'XPnetmRxFile' ou 'XPnetmRxProg'. Arquivo do coletor não existe ou mensagem transmitida pelo coletor não foi recebida, tinha muitos bytes, tinha número do terminal errado ou tipo de bloco inválido. Programa enviado ao coletor não está escrito em XPBASIC, por exemplo o arquivo de programa do XTM-Quick.
XPN_USRABT:	Aborto pelo usuário. Durante a operação de transferência de um arquivo de dados ou programa, no windows, aparece uma janela que exibe o andamento do processo, com o botão para abortá-lo. Se este botão é acionado a função retorna este código de erro.
XPN_TRANSFER:	Transferência já está ativa no canal especificado. Foi feita uma tentativa de execução de uma função para enviar ou receber um programa ou arquivo num canal que já tem uma transferência ativa.
XPN_NOMEMOP	Não há memória suficiente para a operação.

XPN_REMTMOUT	*Timeout* de modo remoto.

Códigos de Erro retornados pelas Funções de Transmissão e Recepção XMODEM:

XPM_ISOPEN	
XM_ISOPEN	Protocolo já está aberto. Foi executada a função XmodemTxOpen com o protocolo já aberto para o canal especificado.
XPM_NOTOPEN	
XM_NOTOPEN	Protocolo não está aberto. Foi executada uma função do protocolo Xmodem fornecendo como argumento um canal de comunicação que não foi aberto anteriormente.
XPM_NOMEM	
XM_NOMEM	Sem memória para operação. Não há memória suficiente a ser alocada para a execução de uma função do protocolo Xmodem.
XPM_NAKTIMEOUT	
XM_NAKTIMEOUT	*Timeout* de espera de NAK inicial. Ocorreu espera muito grande após envio do NAK inicial no protocolo Xmodem.
XPM_ACKTIMEOUT	
XM_ACKTIMEOUT	*Timeout* de espera de ACK. Ocorreu espera muito grande na confirmação de uma transmissão de bloco ou EOT no protocolo Xmodem.
XPM_RXNAK	
XM_RXNAK	Recepção de NAK após transmissão de bloco. Foi recebido NAK após varias tentativas de transmitir um mesmo bloco no protocolo Xmodem.
XPM_ERRFILE	
XM_ERRFILE	Problema no acesso ao arquivo. Problema no acesso a um arquivo de dados ou programa durante a execução de uma função de transferência no protocolo Xmodem.
XPM_USERABORT	
XM_USERABORT	Aborto pelo usuário. Durante a execução de uma função Xmodem houve a solicitação de aborto pelo usuário. Por exemplo se a função do usuário que exibe número de blocos retornar valor diferente de zero.
XPM_REMABORT	
XM_REMABORT	Aborto remoto. Durante a execução de uma função Xmodem houve a solicitação de aborto pelo terminal remoto.
XPM_RXINVBLK	
XM_RXINVBLK	Recepção de bloco inválido após N tentativas. Ocorreu erro na recepção de um bloco Xmodem, devido a número de bloco errado, complemento errado, comprimento ou *check sum* após várias tentativas.
XPM_SOHTIMEOUT	
XM_SOHTIMEOUT	*Timeout* de recepção de SOH. Ocorreu espera muito grande do caractere de início de bloco SOH no protocolo Xmodem.
XPM_RXTIMEOUT	
XM_RXTIMEOUT	*Timeout* de recepção de caractere. Ocorreu espera muito grande na recepção com protocolo Xmodem.
XM_INVCONFIG	Configuração inválida.

Codigos de Erros retornados pelas funções de comunicação da rede de Rádio Freqüência:

RFN_INVCOM:	Canal de comunicação inválido. Foi executada uma função da biblioteca, na qual foi fornecido como argumento um canal de comunicação inexistente.
RFN_INVRETRIES:	Número de tentativas de comunicação inválido. Foi executada a função ‘RFnetOpen’ na qual foi fornecido como argumento um número máximo de tentativas de retransmissão inválido.
RFN_ISOPEN:	Rede já está aberta para canal especificado. Foi executada a função 'RFnetOpen' com a rede já inicializada para o canal especificado.
RFN_NOMEM:	Sem memória para operação da rede. Não há memória suficiente a ser alocada para a execução de uma função da rede.
RFN_NOTOPEN:	Rede não está aberta para canal especificado. Foi executada uma função fornecendo como argumento um canal de comunicação que não foi aberto anteriormente.
RFN_INVTERM:	Número do terminal inválido. Foi executada uma função, fornecendo como argumento 'nTerm' um número de terminal que não está conectado na rede.
RFN_INVPARM:	Parametro para funcao invalido. Foi passado um parametro não reconhecido para uma função.
RFN_TXBFULL:	*Buffer* de transmissão de mensagens cheio. *Buffer* de transmissão não possui espaço disponível para transmissão de mensagens.
RFN_RXBEMPTY:	*Buffer* de recepção de mensagens vazio. *Buffer* de recepção não contém mensagem armazenada.

RFN_RXBFULL:	*Buffer* de recepção de mensagens cheio. *Buffer* de recepção de mensagens não possui espaço suficiente para receber mais mensagens. Deve ser executada a função 'RFnetRxMsg' para que sejam retiradas as mensagens armazenadas no *buffer*.
RFN_RXTIMEOUT:	*Time out* durante a recepção. Ocorreu espera muito grande durante a execução de uma função de recepção.
RFN_RXCRC:	Erro no CRC. Durante a execução de uma função de recepção de mensagem, houve erro no cálculo do CRC.
RFN_TXTMOUT:	*Time out* durante a transmissão. Uma função de transmissão de dados foi executada, mas houve espera muito grande durante este processo.
RFN_ACKTIMEOUT:	*Time out* de espera de ACK. Ocorreu um problema durante a espera de ACK após ser executada uma função de transmissão de mensagens.
RFN_INVCFG:	Consulta de configuração inválida. Foi executada a função 'RFnetGetConfig' fornecendo-se como argumento um código de informação 'nInfo' não suportado pela biblioteca XPnet.
RFN_THREADPRI:	Erro no estabelecimendo da prioridade de thread.
RFN_NOBASE:	Base fora de operação. Base não responde a comunicação durante RFnetOpen.
RFN_ACTIVE:	Erro: ainda esperando resposta.
RFN_RESULTERR:	Retornou erro de comando ou parâmetro inválido.

Códigos de Erro retornados pelas Funções de Conversão de Arquivos:

XPN_ERRFILE	
CF_ERRFILE	Problema em acesso a arquivo. Não foi possível encontrar o arquivo origem, criar o arquivo destino especificado ou houve erro de escrita no arquivo destino ou leitura no arquivo origem.
CF_NOMEMORY	Problema em alocação de memória durante conversão. Não há memória suficiente para alocação dinâmica dos *buffers* necessários.
XPN_INVFILE	
CF_INVFILE	Formato do Arquivo a ser convertido não foi reconhecido. Na conversão para formato XPbasic a estrutura tipo FileFormat_t passada por referência não foi reconhecida. Variável cFields, número de campos, deve ser entre 1 e 10. Tamanho de campo *string* deve ter no máximo 80 caracteres e deve ter no mínimo o comprimento especificado em Field[n].cLength. Arquivo origem não tem assinatura esperada. Arquivo tipo TEXTO só pode ter campos tipo *string* e comprimento máximo de 800 caracteres por linha. Na conversão a partir de formato XPbasic o arquivo origem não tem a assinatura e o tipo para arquivos de dados XPbasic.
XPN_ERRMAXF	
CF_ERRMAXF	Número de campos maior que 10. Arquivo origem tem número de campos superior ao máximo suportado pelo formato XPbasic ou arquivo destino tem número de campos diferentes que arquivo XPbasic de origem.
XPN_ERRMAXR	Número de registros maior que o máximo.
XPN_ERRSTRUC	
CF_ERRSTRUC	Estruturas incompatíveis. Número de campos do arquivo origem ou tipo de campo diferente do número de campos ou tipo de campo especificado na estrutura FileFormat_t passada por referência. Ex.: Campo XPbasic inteiro convertido para DBASE numérico com casas decimais.
CF_USERABRT	Aborto pelo usuário. Durante a execução de uma função de conversão de arquivos houve a solicitação de aborto pelo usuário. Por exemplo se a função do usuário que exibe número de blocos retornar valor diferente de zero.
XPN_NOFLOAT	
CF_FLOAT	Ponto flutuante não suportado. A bilbioteca de conversão de arquivos em Clipper não suporta conversão de valores em ponto flutuante.

Outros Erros:

1) NO AMBIENTE "DOS" COMPUTADOR TRAVA QUANDO PROGRAMA TERMINA
Programas para DOS que utilizem a biblioteca de comunicação não podem ser abortados com <CONTROL> + <BRAKE>, ou retornar ao DOS sem que a comunicação seja fechada. Caso isto ocorra o PC irá travar pois as interrupções que foram redirecionadas para o tratamento da comunicação não foram restauradas para suas interrupções originais.

O exemplo em CLIPPER que acompanha a biblioteca para DOS (DEMOXTM) mostra como deve ser feito o tratamento de erros de runtime do CLIPPER com fechamento da comunicação antes do retorno ao DOS.

Em C deve-se redirecionar a interrupção INT23 do DOS para desabillitar o <CONTROL> + <BRAKE> e garantir que em todas as saídas do sistema a comunicação está sendo fechada.

2) ERRO NA LINCAGEM PARA "DOS" COM BORLAND C++ 4.0
Caso as funções free(), malloc(), dossetvect(), dosgetvect() apareçam como símbolo indefinido no módulo COMDRV durante a lincagem obtenha *patches* (correções) do Borland 4.0. Estas correções podem ser conseguidas via internet.

3) NO AMBIENTE "WINDOWS" COMPUTADOR TRAVA QUANDO PROGRAMA USANDO A BIBLIOTECA É EXECUTADO
Não deve ser usada a função XPnetmComStatus para implementar loop de espera no ambiente Windows. Usar a função XPnetWaitCom em seu lugar.

4) COLETOR FICA EM MODO REMOTO QUANDO ENVIO DATA E HORA COM REDE RS-485 SOB WINDOWS E OCORRE *TIMEOUT* DE ESPERA DE ACK, ERRO -120 DA DLL.
Se estiver usando placa MOS-485 revisão E ou superior, abrir o jumper JP3 para configurar modo AUTO.
Com as versões anteriores da placa MOS-485 deve-se utilizar a interface RS-422 com 4 fios e abrir os jumpers JP4 a JP7 da placa.

5) OCORRE ERRO GERAL DE PROTEÇÃO (GPF) EM FUNÇÕES QUE PASSAM PONTEIROS PARA A BIBLIOTECA.
Verifique se o ponteiro aponta para uma área onde existe espaço disponível. Em linguagens que alocam espaço dinamicamente inicializar a variável com espaços suficientes para receber a mensagem.

6) INTERMITENTEMENTE OCORRE ERRO OU COLETOR NÃO ENTENDE MENSAGEM ENVIADA PARA ELE.
Aumente o número de retries na abertura do canal.

7) NÃO CONSEGUE COMUNICAR COM VIA MODEM USANDO A REDE XPNET.
Verifique que está abrindo a rede com o argumento XPN_COMFULL que liga automaticamente o DTR e o RTS.

8) OCORRE ERRO -123, COMUNICAÇÃO CONTINUA ATIVA APÓS VÁRIAS TENTATIVAS.
Atualize a versão da XPCOMLIB que aumentou o tempo de espera por resposta do coletor.

RESPONSABILIDADES

A TRIX Tecnologia Ltda. não se responsabiliza por nenhum dano ou problema que a utilização dos programas fornecidos com a biblioteca de comunicação possam causar ou apresentar. Os programas são fornecidos apenas como um exemplo. A TRIX também se exime de fornecer qualquer Suporte Técnico referente aos programas ou características dos programas de demostração. O Suporte Técnico para a Biblioteca XPCOMLIB abrange apenas a utilzação das funções da Biblioteca XPCOMLIB e não dos programas de demostração que a acompanham.

Todas as informações contidas neste manual estão sujeitas a alterações sem prévio aviso.

TRIX Tecnologia Ltda.
Suporte: www.trixtec.com.br / suporte@trixtec.com.br
Rua da Paz, 1957 - São Paulo - SP - CEP 04713-002
fone: (011) 5182-3633
fax: (011) 5182-3070