Direitos Reservados

µDX

Série 100

BIBLIOTECA DE FUNÇÕES

Manual de Utilização

LINGUAGENS C, PASCAL, CLIPPER, BASIC

Rev. 1.3

Abr/2002

DEXTER Indústria e Comércio de Equipamentos Eletrônicos Ltda.

Av. Pernambuco, 1328 Cj.309 - Porto Alegre - RS - Fones: (0xx51) 3343-2378, 3343-5532

E-mail: dexter@dexter.ind.br Internet: www.dexter.ind.br

Introdução

Este documento apresenta e descreve os sub-programas (funções) disponíveis para utilização dos comandos da rede DXNET em programas feitos na linguagem C, Pascal, Clipper ou ainda Basic.

Através deles é possível elaborar programas que possam comunicar-se com o Controlador Programável µDX para monitorar ou alterar estados de nodos, conteúdos de variáveis, ler ou gravar a memória de programa, acertar o relógio interno, interromper ou iniciar a execução.

Esta biblioteca de funções está presente em alguns arquivos contidos no disquete que, além deste manual, compõe o produto.

Os sub-programas para linguagem C estão contidos no arquivo UDXC.OBJ. O arquivo T.C é um programa de exemplo para uso destes sub-programas. Ele foi feito e compilado através do Turbo C 2.0. Apesar da incompatibilidade existente na prática entre este compilador e o Microsoft C os sub-programas em UDXC.OBJ são úteis em qualquer caso, uma vez que foram mantidas as regras de passagem de parâmetros e de ambiente. Para linkar o arquivo UDXC.OBJ com seu programa em C é fornecido o BAT LINKA.BAT. Note que ele pressupõe que o programas estejam no diretório corrente e as bibliotecas C e CH estejam no subdiretório LIB. Além disso, ao compilar seu programa (gerando o objeto (.OBJ) que será "linkado" com UDXC.OBJ) é necessário especificar modelo de memória grande (large) ou enorme (huge). Isto porque o arquivo UDXC.OBJ utiliza as rotinas com acesso distante (far).

Já para linguagem Pascal é fornecida uma TPU (UDXINT.TPU) para utilizar os comandos do µDX com seus programas em Pascal. O programa TDX.PAS exemplifica o uso desta ferramenta.

Para Clipper o arquivo que contém as rotinas de interface com o µDX chama-se UDXCL.OBJ. Junto no diskette acompanha um programa escrito em Clipper (TESTE.PRG) que exercita todos os comandos do µDX. Existem ainda dois arquivos sufixo .BAT para definir os parâmetros de compilação de TESTE.PRG (CL.BAT) e os parâmetros de "link" de TESTE.OBJ com UDXCL.OBJ, além das bibliotecas fornecidas com o compilador Clipper (LINK.BAT).

Em Basic é fornecido um programa exemplo (T.BAS), que chama um "include" (MDXBAS.BI) e deve ser linkado com os sub-programas para o µDX (MDXBAS.LIB e MDXBAS.QLB). Note que foi utilizado o QuickBasic da Microsoft (versão 4.5 ou 7.1).

Em todas linguagens o retorno é obtido por uma variável tipo long int. Caso esta variável retorne FFFFFFFFh (-1 em decimal com sinal) significa que a comunicação não foi completada corretamente.

Utilização em C

Existem cinco funções disponíveis:

- ini_udx ( );
- s_destino (int destino);
- que_porta (int porta);
- dxnet_free ( );
- txcom (int comando, int p1, int p2, int p3);

Nas instruções abaixo, na declaração EXTERN, pode-se utilizar qualquer nome para os parâmetros de passagem, conforme for adequado ao programa aplicativo.

ini_udx ( );

Esta função inicializa a comunicação via DXNET através da porta paralela no endereço 378h (normalmente LPT1). A calibração, segundo a velocidade do computador, é automática e não exige atenção especial.

No cabeçalho do programa aplicativo deve ser inserida a seguinte linha, identificando a função como externa:

```
extern void ini_udx ( );
```

s_destino (int destino);

Esta função permite alterar o endereço DXNET para o qual será enviado algum comando. DESTINO=0 (zero) endereça todos os µDX conectados à rede DXNET. Utilize este valor apenas para comunicações com um só µDX (ponto-a-ponto) ou com comandos cuja resposta seja idêntica para todos (apenas um ACK, por exemplo), uma vez que todos responderão juntos e sincronizados, uma resposta igual não deverá chegar no computador de forma errada.

No cabeçalho do programa aplicativo deve ser inserida a seguinte linha, identificando a função como externa:

extern void s_destino (int *destino*);

Exemplo de programa:

```
extern void s_destino (int destino_dxnet);
extern void ini_udx ( );
extern long txcom (int comm, int p1, int p2, int p3);
typedef unsigned long LONG;

void main ()
{
long ll,kk;
/* Inicializacao */
ini_udx();
/* Le status do µDX 1 */
s_destino (1);
ll=txcom(11,0,0,0);
kk=ll;
ll>>=16;
printf ("\nStatus µDX 1: %4x (Word MSB)",ll);
printf ("\nStatus µDX 1: %4x (Word LSB)",kk);
/* Le status do µDX 2 */
s_destino (2);
ll=txcom(11,0,0,0);
kk=ll;
ll>>=16;
printf ("\nStatus µDX 2: %4x (Word MSB)",ll);
printf ("\nStatus µDX 2: %4x (Word LSB)",kk);
}
```

s_modo (int modo);

Esta função permite alterar o modo de comunicação na DXNET. MODO=0 (zero) é o modo de operação normal da DXNET, permitindo acessar a rede DXNET local. Já com MODO=1 (um), a rede DXNET opera em modo modem. Neste modo de operação é possível acessar uma rede DXNET remota via modem ligado ao µDX.endereça todos os µDX conectados à rede DXNET. O modo modem é um pouco mais lento que o modo normal, permitindo o acesso remoto.

No cabeçalho do programa aplicativo deve ser inserida a seguinte linha, identificando a função como externa:

extern void s_modo (int *modo*);

que_porta (int porta);

Esta função permite escolher qual porta paralela (de impressora) será empregada para a comunicação DXNET. PORTA=0 identifica a porta com endereço 378h (normalmente LPT1). PORTA=1 identifica a porta com endereço 278h (normalmente LPT2). Se a variável PORTA for maior que 1 a função irá selecionar a porta paralela com endereço 3BCh (quando o computador possui porta paralela neste endereço as atribuições de LPT1,LPT2 e LPT3 se modificam. LPT1 fica sendo o endereço 3BCh, LPT2 o endereço 378h e LPT3 o endereço 278h).

Não é preciso chamar ini_udx() após alterar a porta empregada.

No cabeçalho do programa aplicativo deve ser inserida a seguinte linha, identificando a função como externa:

extern void que_porta (int *porta*);

Exemplo de programa:

```
extern void s_destino (int destino_dxnet);
extern void ini_udx ( );
extern void que_porta (int porta_dxnet);
extern long txcom (int comm, int p1, int p2, int p3);
typedef unsigned long LONG;

void main ()
{
long ll,kk;
/* Inicializacao */
ini_udx();
/* Le status do µDX 1 na rede DXNET da porta 378h */
s_destino (1);
que_porta (0);
ll=txcom(11,0,0,0);
kk=ll;
ll>>=16;
printf ("\nuDX 1, DXNET 1: %4x (Word MSB)",ll);
printf ("\nuDX 1, DXNET 1: %4x (Word LSB)",kk);
/* Le status do µDX 3 na rede DXNET da porta 278h */
s_destino (3);
que_porta (1);
```

```
ll=txcom(11,0,0,0);
kk=ll;
ll>>=16;
printf ("\uDX 3, DXNET 2: %4x (Word MSB)",ll);
printf ("\uDX 3, DXNET 2: %4x (Word LSB)",kk);
}
```

dxnet_free ( );

Esta função serve para acertar os sinais da porta paralela de forma a garantir que sejam mantidos os níveis lógicos para livre comunicação na DXNET.

Em alguns casos, quando por exemplo o programa aplicativo solicitar ao usuário para trocar o cabo DXNET por um cabo de impressora (para que seja impressa alguma mensagem na impressora), é preciso chamar DXNET_FREE.

Caso contrário, se os níveis lógicos da porta estiverem errados, após o uso com a impressora, os µDX conectados na rede DXNET poderão ficar continuamente esperando por um comunicado (nível da linha DXNET em zero) o que fará com que atrasem o processamento, deixando-os muito lentos.

A função ini_udx ( ); já acerta estes níveis, porém deve-se utilizar DXNET_FREE sempre que o programa aplicativo exigir a troca de cabos.

No cabeçalho do programa aplicativo deve ser inserida a seguinte linha, identificando a função como externa:

```
extern void dxnet_free ( );
```

txcom (int *comando*, int *p1*, int *p2*, int *p3*);

Esta função envia ao µDX destino um comando DXNET que poderá ser respondido com um ou mais bytes de resposta ou um Acknowledge. O valor retornado, um LONGINT (32 bits), poderá ser zero, -1 ou maior que zero.

Utiliza-se -1 (FFFFFFFFh em hexadecimal) para indicar quando houve um erro de comunicação. Este erro pode ocorrer por endereçamento errado, cabo não conectado, porta paralela incorreta, programa não inicializado corretamente (ini_udx não calculou corretamente a velocidade do computador), µDX não energizado, excesso de interferência elétrica na linha de comunicação, respostas diferentes de mais de um µDX com mesmo endereço (ou endereço zero) ou falha interna no µDX.

Resposta igual a zero é obtida nos comandos que esperam apenas um ACK.

Respostas igual ou maiores que zero são esperadas nos comandos com retorno de dados.

Respostas com mais de um byte devem ter o valor da variável LONG separado em bytes para que sejam utilizadas cada parte da resposta.

Comandos DXNET:

Número	Descrição	Resposta
0	Parar	ACK
1	Executar	ACK
2	Lê EEPROM, P1=endereço de word (0-256)	1 WORD
3	Escreve na EEPROM, P1=end, P2=byte alto, P3=byte baixo	ACK
4	Lê grupo de 8 nodos, P1=grupo (0-7)	1 Byte
5	Lê variável, P1=variável (0-15)	1 Byte
6	Força nodo_out, P1=nodo (0-63) + estado (bit 7)	ACK
7	Força variável, P1=variável (0-15), P2=valor	ACK
8	Lê/escreve na expansão, P1=valor de saída	1 Byte
9	Força nodo_DXNET, P1=nodo (0-63) + estado (bit 7)	ACK
10	Reset forçado	sem resposta
11	Informações (STATUS)	3 bytes
12	Acerta relógio, P1=dia/hora, P2=minutos, P3=segundos	ACK
13	Lê relógio	3 bytes
14	Força ocorrência de instruções DXNET	ACK
15	Não utilizado	

O comando 11 (leitura do status do µDX) tem como resposta 3 bytes com as seguintes informações:

Valor lido (LONG) : [Byte 3] [Byte 2] [Byte 1] [Byte 0]

Byte 3 - não usado

Byte 2 - tipo de equipamento (1-µDX Série 100, 2-µDX Série 200)

Byte 1 - versão do firmware

Byte 0 - status e endereço DXNET:

7 6 5 4 3 2 1 0

V V R F E E E E

onde, E = Endereço DXNET
R = 0-parado, 1-executando
F = 1-falha de energia
V = velocidade de ciclo

A velocidade do ciclo pode ser:

00 = 1/16 s.

01 = 1/32 s.

10 = 1/64 s.

11 = 1/256 s.

Por exemplo, se o byte 0 retornar o valor A3h (10100011b) 0 µDX estará com ciclo de 1/32s, executando, não houve falha de rede e seu endereço na DXNET é 3.

Os comandos 12 e 13 tratam do relógio interno do µDX. Os três bytes para acerto ou leitura seguem a seguinte configuração:

Byte 2 - Dia/Hora	Bits7,6,5 - Dia(000=Domingo, 110=Sábado)
	Bits 4,3,2,1,0 - Hora (00-23)
Byte 1 - Minutos	Bits 7,6,5,4,3,2 - Minutos (0-59)
	Bits 1,0 - Quartos de minuto (0-3)

Byte 0 - Segundos Bits 7,6,5,4 - segundos de um quarto de minuto (0-14)

Bits 3,2,1,0 - frações (em unidades de 0,0625s)

O comando 14 faz com que todas as intruções DXNET no programa do µDX forçem suas execuções, enviando aos destinos os estados de nodos ou valores de variáveis. Este comando é útil quando algum µDX na rede DXNET sofre um RESET, perdendo o estado e condições operacionais anteriores.

Neste caso o forçamento DXNET faz com que todos os demais µDX e ele próprio atualizem os parêmetros e nodos importantes para o funcionamento em conjunto. Note-se que este comando é enviado automaticamente pelo próprio µDX que executou o RESET (operação programada no firmware do µDX somente a partir da versão 3.8).

Nota explicativa:

No µDX existem três conjuntos de memória para nodos: NODO_IN, NODO_OUT e NODO_DXNET. Os dois primeiros (nodo_in e nodo_out) são empregados no processamento de cada ciclo do programa: em nodo_in ficam os estados iniciais de cada nodo no começo de um ciclo (incluindo os nodos de entrada e de saída); em nodo_out são armazenados os estados de resultados das instruções, sendo que nodo_out começa todo limpo (em zero) no início do ciclo - no fim do ciclo os estados de nodo_out são transferidos para nodo_in e tudo recomeça. A memória de nodo_dxnet é combinada com a memória nodo_out no início de cada ciclo. Assim é possível forçar que um determinado nodo fique ligado mesmo que o resultado do ciclo o tenha deixado desligado. Note-se que se o resultado foi para o nodo ficar ligado não importará o estado do mesmo nodo na memória nodo_dxnet (a operação combinacional empregada é um OU lógico).

Forçamentos feitos sobre nodo_out terão no máximo a duração de um ciclo a menos que o programa faça o nodo em questão ficar ligado.

A leitura dos nodos (comando 4, lê nodo_in) é feita de 8 em 8 nodos (um byte). Os nodos 0 até 3 são as entradas do µDX e os nodos 4 até 7 são as saídas de relé.

No cabeçalho do programa aplicativo deve ser inserida a seguinte linha, identificando a função como externa:

extern long txcom (int *comando*, int *p1*, int *p2*, int *p3*);

Exemplo de programa:

```
extern void ini_udx ( );
extern long txcom (int comm, int p1, int p2, int p3);
typedef unsigned long LONG;

void main ()
{
long ll;
/* Inicializacao */
ini_udx();
/* Le estado das entradas e saidas */
ll=txcom(4,0,0,0);
printf ("\nEntradas e Saidas: %2x (em hexa)",ll);
/* Forca nodo DXNET 4 ligado (saida de rele 1) */
ll=txcom(9,132,0,0);
}
```

Gráficos em C e interface com uDX

O programa TG.C (subdiretório \c\graficos) foi feito para demonstrar a utilização de rotinas gráficas junto com as rotinas de interface do uDX. O arquivo LINKAG.BAT especifica as diretivas de montagem ("link"), de forma a unir os objetos TG.OBJ, UDXC.OBJ e a biblioteca GRAPHICS.LIB existente no subdiretório \TC\LIB do Turbo C. Note que UDXC.OBJ é o programa objeto fornecido para a interface com o uDX, sendo utilizado tanto para gerar T.EXE quanto para gerar a versão gráfica TG.EXE.

Os arquivos com sufixo .BGI são os controladores gráficos, necessários para a interface com a placa gráfica do computador. A função INITGRAPH() lê para a memória o controlador gráfico correspondente a placa de vídeo instalada no computador. Como não foi especificado nenhum caminho ("path") na função INITGRAPH() usada em TG.C, os arquivos .BGI devem estar no diretório de trabalho corrente.

Utilização em Pascal

A utilização em Pascal é muito semelhante à descrita em C, utilizando-se neste caso a TPU UDXINT.TPU. Os procedimentos e funções disponíveis são:

set_destino (int destino);	(corresponde à s_destino em C)
set_modo (int modo);	(corresponde à s_modo em C)
set_port (int porta);	(corresponde à que_porta em C)
libera_linha;	(corresponde à dxnet_free em C)
tx_com(int comando, p1, p2, p3);	(corresponde à txcom em C)

Note que não existe correspondência para o comando em C ini_udx();. Em Pascal este comando é desnecessário.

O programa TDX.PAS que acompanha o disquete exercita os diversos comandos citados. A sintaxe de utilização, em especial do comando tx_com é exatamente igual à descrita para o comando txcom em C.

Utilização em Clipper

A utilização em Clipper é muito simples e as funções disponíveis são idênticas as explicadas para linguagem C. São elas:

```
ini_udx()
s_destino(int destino)
s_modo(int modo)
que_porta(int porta)
dxnet_free()
txcom(int comando,int p1,int p2,int p3)
```

A sintaxe de uso, em especial da função txcom, é exatamente igual à descrita na página 7. Assim, txcom(7,12,45,0) escreve na variável 12 do µDX corrente o valor 45.

O arquivo UDXCL.OBJ contém as funções de interface com o µDX. Este arquivo deve ser "linkado" com o objeto em Clipper. O diskette possui um programa exemplo (TESTE.PRG) em Clipper. O arquivo CL.BAT ativa a compilação deste programa em Clipper e o arquivo LINK.BAT faz a montagem do .OBJ resultante com UDXCL.OBJ. A listagem de TESTE.PRG mais as diretivas de compilação e "linkagem" existentes em CL.BAT e LINK.BAT devem ser suficientes para um programador experiente em Clipper gerar seu próprio programa aplicativo com o µDX.

Utilização em Basic

A utilização em Basic também é muito simples e as funções disponíveis são idênticas as explicadas para linguagem C (exceto que o sinal de sublinha(_) foi substituído por ponto(.) no nome das rotinas). São elas:

```
ini.udx()
s.destino(int destino)
s.modo(int modo)
que.porta(int porta)
dxnet.free()
txcom(int comando,int p1,int p2,int p3)
```

A sintaxe de uso, em especial da função txcom, é exatamente igual à descrita na página 7. Assim, txcom(7,12,45,0) escreve na variável 12 do µDX corrente o valor 45.

São fornecidos os seguintes programas, em dois subdiretórios (QB45, para a versão 4.5 do QuickBasic; e QB71, para a versão 7.1 do QuickBasic):

T.BAS	Arquivo fonte com exemplo em Basic
T.EXE	Arquivo executável do exemplo em Basic
MDXBAS.QLB	Arquivo para "linkagem"
MDXBAS.LIB	Arquivo para "linkagem"
MDXBAS.BI	Arquivo "include" chamado por T.BAS
QB.BAT	Arquivo "bat" para chamar o QuickBasic

Conforme a versão de seu QuickBasic, utilize um ou outro subdiretório. Copie o conteúdo do subdiretório correspondente à versão de seu QuickBasic para o diretório no disco rígido que contém o compilador Basic. Para chamar o QuickBasic digite QB <enter>. Será executado o arquivo QB.BAT, que já carrega o programa de exemplo T.BAS (No caso de outro programa, basta substituir o nome de T.BAS dentro de QB.BAS ou digitar a linha de comando inserida neste arquivo).

Ao entrar no ambiente do QuickBasic, basta selecionar para gerar um executável (.EXE) e pronto, teremos um programa em Basic capaz de interfacear com o controlador programável µDX. Saia do ambiente do QuickBasic e execute o programa gerado (.EXE).

A listagem de T.BAS deve ser suficiente para um programador experiente em QuickBasic gerar seu próprio programa aplicativo com o µDX.

Garantia

A DEXTER oferece uma garantia de 3 (três) meses, a contar da data da compra, para reposição do diskette contendo a Biblioteca de Funções, no caso de mau funcionamento ou defeitos originários na fábrica.

Esta garantia deixa de vigorar caso o defeito apresentado seja resultante do uso indevido ou incorreto da Biblioteca de Funções para controlador programável µDX, assim como no caso de serem feitas alterações nos arquivos objeto (.OBJ) fornecidos, sem autorização por escrito da DEXTER.

Não estão incluídos nesta garantia os custos com transporte, tanto para recebimento como para devolução.

Esta garantia se restringe as rotinas fornecidas no pacote denominado Biblioteca de Funções, não se estendendo ao processo controlado, nem a sensores e/ou acionamentos ligados ao controlador programável µDX, e nem tampouco ao programa aplicativo do usuário.

A DEXTER não se responsabiliza pela aplicação da Biblioteca de Funções em processos perigosos ou de risco de vida.

DEXTER Indústria e Comércio de Equipamento Eletrônicos Ltda.
Av. Pernambuco, 1328 - Cj:309 CEP:90240-001 Porto Alegre RS
Fones: (0xx51) 3343-2378, 3343-5532
Internet: www.dexter.ind.br
E-mail: dexter@dexter.ind.br