WDM_1256.SYS

DRIVER DA CAD12/56 PARA WINDOWS 98

1. INTRODUÇÃO

WDM_1256.SYS é um kernel mode driver que gerencia o acesso à placa A/D CAD12/56 no sistema operacional Windows 98.

1.1. Escopo

O objetivo deste documento é apresentar a interface com o driver realizada por módulo em Borland Delphi para Windows 98.

1.2. Aplicabilidade

O driver WDM_1256 deve ser utilizado somente no sistema operacional Windows 98 ou compatível e com a placa A/D CAD12/56. Este driver não possui suporte para o contador 2 da placa e não utiliza o DMA.

2. INSTALANDO O DRIVER WDM_1256

Os arquivos fornecidos para utilização do driver WDM_1256:

WDM_1256.SYS	kernel mode driver para Windows 98
WDM_1256.inf	arquivo com informações de instalação do driver utilizado pelo Windows
WDM_1256.DLL	biblioteca DLL de acesso ao driver
TESTE.EXE	programa de teste do driver
Delphi*.*	exemplo com programa fonte em Borland Delphi
VB5*.*	exemplo com programa fonte em Visual Basic 5.0

2.1. Instalando o Driver

Siga as seguintes etapas para instalar o device driver da CAD12/56 no Windows 98:

1. A partir da barra de tarefa, execute **Iniciar/Programas/Configurações/Painel de controle** para entrar na janela *Painel de controle* do Windows 98.
2. Na janela *Painel de controle*, clique sobre o ícone *Adicionar novo hardware*. O Windows apresentará a janela do *Assistente para instalar novo hardware*. Clique sobre o botão *Avançar*. Novas janelas serão apresentadas, clique em avançar até que seja apresentada a pergunta: "Deseja que o Windows procure seu novo hardware?". Clique em não e em seguida clique em *Avançar*.
3. O Assistente de instalação do Windows apresentará uma janela para você selecionar o tipo de hardware a ser instalado. *Selecione Outros dispositivos* e clique sobre o botão *Avançar*.
4. Na janela seguinte clique sobre o botão *Com disco*.
5. Insira o disquete do driver da CAD12/56 para Windows 98 na unidade de disquete A.
6. Na janela *Instalar a partir do disco*, selecione a unidade de disquete A: e clique sobre o botão *Procurar*.
7. Selecione *WDM_1256.INF* e clique sobre o botão *OK*. Clique sobre o botão *OK* novamente para retornar na janela do Assistente de adicionar novo hardware.
8. Na janela do *Assistente* será apresentada a lista de modelos, selecione CAD12/56 e clique sobre o botão *Avançar* e siga as instruções do *Assistente*. Ao pressionar o botão *Avançar*, o Windows poderá demorar para avançar para o próximo passo.

2.2. Como Alterar a Configuração de Hardware do Driver

A placa CAD12/56 utiliza endereços de I/O e um canal de interrupção do sistema. Eventualmente esses recursos especificados para uso do device driver podem estar em conflito com outras placas instaladas no microcomputador. Para resolver esse conflito, você precisará modificar a alocação do recurso em conflito alterando a configuração do driver da CAD12/56 ou do outro driver em conflito.

Para alterar os recursos alocados ao driver da CAD12/56, clique sobre o ícone *Sistema* do *Painel de controle* do Windows 98.

1. Na janela do *Propriedade de Sistema*, seleciona a página *Gerenciador de dispositivos*.
2. Selecione o dispostivo CAD12/56 na categoria CAD1256.
3. Clique sobre o botão *Propriedade*. Será apresentada a janela *Propriedade de CAD12/56*.
4. Selecione a página *Recursos*. Na página será apresentada os conflitos existentes na parte inferior da janela.
5. Se for um conflito de *Intervalo de entrada/saída*, selecione uma outra configuração básica. O driver possui quatro opções de configuração básica (0 a 3) que correspondem respectivamente aos endereços base 300, 310, 340 e 350 da placa, que devem ser selecionadas por jumper na placa CAD12/56 (consulte o manual de referência técnica da placa).
6. Se o conflito for de *Pedido de interrupção*, clique sobre *Pedido de interrupção* e em seguida clique sobre o botão *Alterar configuração*. Na janela apresentada, selecione uma das opções disponíveis que não esteja em conflito com outro dispositivo. A seleção de canal de interrupção na CAD12/56 é

realizada por software e não depende de nenhum jumper de configuração.

7. Siga as demais instruções do Windows.

Para informações mais detalhada do Painel de controle do Windows, consulte documentação da Microsoft.

3. INTERFACE DO DRIVER COM O PROGRAMA APLICATIVO

O acesso do programa aplicativo ao driver é disponibilizado através do módulo em Pascal WDMapi. Esse módulo deve ser incorporado ao projeto do programa aplicativo em Borland Delphi. Se você estiver utilizando outra linguagem de programação, basta criar um módulo equivalente ao WDMapi.pas com as referências à biblioteca WDM_1256.DLL. A biblioteca DLL deverá ser copiada para o mesmo diretório do programa aplicativo. Se você estiver utilizando o Visual Basic, copie a bibliteca DLL no diretório do Windows.

3.1. Estrutura de Dados do Driver

O driver possui uma memória de canais onde são armazenadas a ordem em que os canais serão aquisitados e os seus respectivos ganhos. Com essa estrutura você pode, por exemplo, informar ao driver para aquisitar 4 canais na seguinte ordem e ganhos: canal 0 em ±5 volts, canal 3 em ±10 volts, canal 12 em ±2.0 volts e o canal 15 em 0-10 volts).
O driver possui um buffer circular de 64K amostras para a aquisição de sinais. Por exemplo, para a aquisição de 4 canais a 1000 Hz por canal, o buffer teria capacidade de armazenar até 16 segundos sem que o programa aplicativo remova os dados do buffer circular.

3.2. Constantes

A tabela seguinte apresenta as constantes retornadas pela primitiva *WDM_GetAcquiredData*.

Constante	Valor	Descrição
ieNoError	0	Não ocorreu nenhum erro desde a última chamada da primitiva WDM_GetAcquiredData
ieIntOverrun	1	Indica que houve interrupt overrun. Ou seja, antes do término do tratamento da interrupção, houve um novo pedido de interrupção da placa A/D.
ieAD_Error	2	Indica que houve erro no conversor A/D da placa.
ieBufOverflow	3	Overflow no buffer de aquisição do driver
ieDriver	100	Erro de acesso ao driver
ieAcqStop	101	Aquisição não está ativa
ieBufSize	102	Buffer passado para a WDM_GetAcquiredData é muito pequeno

A tabela seguinte apresenta as constantes retornadas pela primitiva *WDM_AcquisitionSetup*.

Constante	Valor	Descrição
seNoError	0	Nenhum erro
seNoSignal	1	Não foi programado nenhum sinal para a aquisição.

4. DESCRIÇÃO DAS PRIMITIVAS DO WDM_1256

Neste tópico são descritas as primitivas de acesso ao driver WDM_1256. A tabela abaixo lista as primitivas disponibilizadas para o programa aplicativo.

Primitiva	Descrição
WDM_Open	Inicia o acesso ao driver
WDM _Close	Finaliza o acesso ao driver
WDM _DriverInfo	Informações gerais do driver
WDM _GetVersion	Informa a versão e revisão do driver
WDM _GetAIO_Caps	Informa os recursos de entrada e saída analógica disponíveis no driver
WDM _GetDIO_Caps	Informa os recursos de entrada e saída digital disponíveis no driver
WDM _SetAiRange	Seleciona faixa de entrada do conversor A/D
WDM _ReadAi	Leitura de canal de entrada analógica
WDM _WriteAo	Escreve em canal de saída analógica (opcional)
WDM _ReadDI	Leitura da entrada digital
WDM _WriteDO	Escrita na saída digital
WDM _Clear_CM	Limpa a memória de canais da aquisição por interrupção
WDM _Insert_AI_CM	Insere um canal de entrada analógica para aquisição
WDM _Insert_DI_CM	Insere a entrada digital para aquisição
WDM _AcquisitionSetup	Programa aquisição por interrupção
WDM _StartAcquisition	Inicia a aquisição por interrupção
WDM _StopAcquisition	Finaliza a aquisição por interrupção
WDM _GetAcquiredData	Leitura das amostras do buffer de aquisição por interrupção
WDM _GetAcquiredDataVB	Leitura das amostras do buffer de aquisição por interrupção

As funções da bilioteca DLL são descritas com a sintaxe do Object Pascal e do Visual Basic. A convenção de chamada utilizada pela DLL é **stdcall**.

4.1. Primitiva WDM_Open

Object Pascal:
```
Function WDM_Open: dword;
```

Visual Basic:
```
Declare Function WDM_Open Lib " WDM _1256.DLL" () As Long
```

Esta primitiva abre o acesso ao driver e deve ser a primeira primitiva a ser executada pelo programa aplicativo. A primitiva retorna o valor *zero* quando a primitiva foi executada com sucesso. Umas das causas possíveis são:

Causa	Verificação	Solução
Driver não instalado	No Painel de Controle do Windows 98, clique sobre Sistema. Se o driver da CAD12/56 estiver instalado, ele deve estar na lista do Gerenciador de Dispositivos.	Instale o driver
Driver com conflito de recursos de entrada/saída e/ou pedido de interrupção	Verifique em Sistema no Painel de Controle do Windows 98 se há algum conflito com o device driver da CAD12/56.	Instale a placa CAD12/56 e/ou altere a configuração do driver. Veja 2.2;.

Veja também a primitiva *WDM_Close*.

4.2. Primitiva WDM _Close

Object Pascal:
```
Procedure WDM_Close;
```

Visual Basic:
```
Declare Sub WDM_Close Lib "WDM_1256.DLL" ()
```

Esta primitiva finaliza o acesso ao driver e deve ser a última primitiva a ser executada pelo programa aplicativo. Para cada chamada bem sucedida da primitiva *WDM_Open* deve haver uma correspondente chamada da *WDM_Close*.

Veja também a primitiva *WDM_Open*.

4.3. Primitiva WDM _DriverInfo

Object Pascal:
```
Function WDM_DriverInfo (Var DriverInfo: TpDriverInfo): boolean;
```

Visual Basic:
```
Não disponível para Visual Basic. Utilize as primitivas WDM_GetVersion,
WDM_GetAIO_Caps e WDM_GetDIO_Caps.
```

Esta primitiva devolve no parâmetro *DriverInfo* o número da versão do driver, o endereço base de I/O da placa, canal de interrupção utilizado, a frequência de clock do timer para programação da frequência de amostragem, número de entradas analógicas, número de faixas de entrada do conversor A/D e respectivas faixas de entrada, número e ports de entrada digital e número de ports de saída digital. A primitiva retorna *true* se a execução foi realizada com sucesso ou *false* (zero) caso ocorra algum erro na sua execução. A estrutura *TpDriverInfo* possui a seguinte declaração em Object Pascal.

```
Type {--------------------------- Parâmetro de entrada da WDM_DriverInfo --}
  TpAiRange = array [0..15] of single;
  TpDriverInfo = record
    VersionHigh : byte;       { Versão do driver }
    VersionLow  : byte;       { Versão do driver }
    CAD_IoBase  : longint;    { Endereço base da CAD12/56 }
    CAD_IRQ     : longint;    { Canal de interrupção da CAD12/56 }
    CAD_DMA     : longint;    { Não utilizado pelo driver }
    TmrClock    : single;     { Clock do timer (Hz) }
    nAiChannels : smallint;   { Número de entradas analógicas }
    nAiRange    : smallint;   { Número de faixas de entrada do A/D em volts }
    AiRange     : TpAiRange;
                              { Tabela com as faixas de entrada do A/D }
    nAoChannels : smallint;   { Número de saídas analógicas }
    nDiPorts    : smallint;   { Número de ports de entrada digital }
    nBitsDI     : smallint;   { Número de bits por port de entrada digital }
    nDoPorts    : smallint;   { Número de ports de saída digital }
    nBitsDO     : smallint;   { Número de bits por port de saída digital }
  end;
```

O campo *AiRange* da estrutura *TpDriverInfo* é um vetor com os valores das faixas de entrada do conversor A/D em volts. Valores negativos indicam faixa de entrada bipolar, por exemplo o valor -10.0 indica que a faixa de entrada correspondente é de -10 a 10 volts. Os valores positivos indicam que a faixa de entrada é unipolar, por exemplo, o valor 5.0 indica que a faixa de entrada é de 0 a 5 volts. O conteúdo do vetor é apenas informativo para o programa aplicativo. Para selecionar uma faixa de entrada, o programa deverá informar o índice correspondente ao vetor. Para a placa CAD12/56 são disponíveis as faixas de entrada do A/D apresentadas na tabela.

Índice	Valor do AiRange	Faixa de Entrada
0	-10.0	-10.0 a 10.0 volts
1	-5.0	-5.0 a 5.0 volts
2	-2.0	-2.0 a 2.0 volts
3	-1.0	-1.0 a 1.0 volts
4	10.0	0 a 10.0 volts
5	5.0	0 a 5.0 volts
6	2.0	0 a 2.0 volts
7	1.0	0 a 1.0 volts

4.4. Primitiva WDM _GetVersion

Object Pascal:
```
Function WDM_GetVersion (Var VersionHigh, VersionLow: byte): boolean;
```

Visual Basic:
```
Declare Function WDM_GetVersion Lib "WDM_1256.DLL" _
        (VersionHigh As Byte, VersionLow As Byte) As Byte
```

Estando o driver instalado, esta primitiva retorna *true* e devolve nos parâmetros *VersionHigh* e *VersionLow*, respectivamente o byte mais significativo e e o byte menos significativo da versão do device driver.

Veja também a primitiva *WDM_DriverInfo*.

4.5. Primitiva WDM _GetAIO_Caps

Object Pascal:
```
Function WDM_GetAIO_Caps (Var TmrClock: single;
                          Var nAiChannels, nAoChannels, nAiRange: smallint;
                          Var AiRange: TpAiRange): boolean;
```

Visual Basic:
```
Declare Function WDM_GetAIO_Caps Lib "WDM_1256.DLL" _
        (TmrClock As Single, nAiChannels As Integer, nAoChannels As Integer, _
         nAiRange As Integer, AiRange As TpAiRange) As Byte
```

Estando o device driver instalado e operando, esta primitiva retorna *true* e informa nos parâmetros de saída os recursos de entrada e saída analógica disponibilizados pelo device driver. Os parâmetros retornados pela primitiva correspondem aos campos de mesmo nome da estrutura *TpDriverInfo* descrita na primitiva *WDM_DriverInfo*.

Veja também as primitivas *WDM_GetDIO_Caps* e *WDM_DriverInfo*.

4.6. Primitiva WDM _GetDIO_Caps

Object Pascal:
```
Function WDM_GetDIO_Caps (Var nDiPorts, nBitsDI: smallint;
                          Var nDoPorts, nBitsDO: smallint): boolean; export;
```
Visual Basic:
```
Declare Function WDM_GetDIO_Caps Lib "WDM_1256.DLL" _
        (nDiPorts As Integer, nBitsDI As Integer, _
         nDoPorts As Integer, nBitsDO As Integer) As Byte
```

Estando o device driver instalado e operando, esta primitiva retorna *true* e informa nos parâmetros de saída os recursos de entrada e saída digital disponibilizados pelo device driver. Os parâmetros retornados pela primitiva correspondem aos campos de mesmo nome da estrutura *TpDriverInfo* descrita na primitiva *WDM_DriverInfo*.

Veja também as primitivas *WDM_GetAIO_Caps* e *WDM_DriverInfo*.

4.7. Primitiva WDM _SetAiRange

Object Pascal:
```
Function WDM_SetAiRange (iRange: byte): dword;
```

Visual Basic:
```
Declare Function WDM_SetAiRange Lib "WDM_1256.DLL" _
        (ByVal iRange As Byte) As Long
```

Esta primitiva permite selecionar a faixa de entrada da placa A/D. Ao ser iniciada a CAD12/56 o driver programa o ganho para a faixa de entrada de ±10 volts. Se desejar uma outra faixa de entrada, o aplicativo deve chamar esta primitiva passando no parâmetro *iRange* o número da faixa de entrada desejada (veja 4.3. Primitiva WDM_DriverInfo). A primitiva retorna 0 (zero) se foi executada com sucesso. Os erros mais comuns na execução da primitiva são faixa de entrada inválida e driver não instalado.

Veja também as primitivas *WDM_ReadAi* e *WDM_DriverInfo*.

4.8. Primitiva WDM _ReadAi

Object Pascal:
```
Function WDM_ReadAi (Channel: byte; Var Value: smallint): dword;
```

Visual Basic:
```
Declare Function WDM_ReadAi Lib "WDM_1256.DLL" _
        (ByVal Channel As Byte, Value As Integer) As Long
```

Esta primitiva realiza a leitura de um canal de entrada analógica da CAD12/56. O programa aplicativo deve passar no parâmetro *Channel* o número do canal A/D a ser lido. A faixa de entrada a ser utilizada na conversão do canal A/D deve ser previamente programada através da primitiva *WDM_SetAiRange*, caso seja uam faixa diferente da última faixa programada.

No parâmetro de saída *Value* é retornado o valor lido do conversor A/D. O valor lido é representado em complemento de 2 e pode assumir valores de -32768 a 32767. A primitiva retorna 0 (zero) se foi executada com sucesso. Os erros mais comuns na execução da primitiva são canal de entrada analógica inválido, erro no conversor A/D e driver não instalado.

Esta primitiva não pode ser chamada durante a aquisição por interrupção.

Veja também as primitivas *WDM_SeAiRange* e *WDM_DriverInfo*.

4.9. Primitiva WDM _WriteAo

Object Pascal:
```
Function WDM_WriteAo (Channel: byte; Value: smallint): dword;
```

Visual Basic:
```
Declare Function WDM_WriteAo Lib "WDM_1256.DLL" _
        (ByVal Channel As Byte, ByVal Value As Integer) As Long
```

Esta primitiva realiza a escrita em canal de saída analógica da CAD12/56. O programa aplicativo deve passar no parâmetro *Channel* o número do canal D/A a ser atualizado. O valor a ser escrito no D/A deve ser passado no parâmetro *Value*, que deve estar representado em complemento de 2 com valores de -32768 a 32767. Esses valores correspondem, respectivamente, aos limites inferior e superior da saída analógica. A primitiva retorna 0 (zero) se foi executada com sucesso. Os erros mais comuns na execução da primitiva são canal de saída analógica inválido e driver não instalado.

Os canais D/A 1 e 3 quando utilizados como referência para respectivamente os canais D/A 0 e 2, devem ser programados com o sinal trocado. Por exemplo, para colocar a referência do canal D/A 0 em 5 volts, deve-se programar o valor -16384 no canal D/A 1.

Esta primitiva não pode ser chamada durante a aquisição por interrupção.

Veja também a primitiva *WDM_DriverInfo*.

4.10. Primitiva WDM _ReadDI

Object Pascal:
```
Function WDM_ReadDI (Group: byte; Var Value: word): dword;
```

Visual Basic:
```
Declare Function WDM_ReadDI Lib "WDM_1256.DLL" _
        (ByVal Group As Byte, Value As Integer) As Long
```

Esta primitiva realiza a leitura de um port de entrada digital da CAD12/56. O programa aplicativo deve passar no parâmetro *Group* o número do port de entrada digital a ser lido. O número de ports de entrada digital disponíveis na placa pode ser consultado no campo *nDiPorts* da estrutura *TpDriverInfo*. No caso da CAD12/56 é disponível 1 port de entrada digital de 16 bits.

No parâmetro de saída *Value* é retornado o valor lido do port de entrada digital. A primitiva retorna 0 (zero) se foi executada com sucesso. Os erros mais comuns na execução da primitiva são número do port de entrada digital inválido e driver não instalado.

Veja também as primitivas *WDM_WriteDO* e *WDM_DriverInfo*.

4.11. Primitiva WDM _WriteDO

Object Pascal:
```
Function WDM_WriteDO Group: byte; Value: word): dword;
```

Visual Basic:
```
Declare Function WDM_WriteDO Lib "WDM_1256.DLL" _
        (ByVal Group As Byte, ByVal Value As Integer) As Long
```

Esta primitiva realiza a escrita em port de saída digital da CAD12/56. O programa aplicativo deve passar no parâmetro *Group* o número do port de saída digital a ser atualizado. O número de ports de saída digital disponíveis na placa pode ser consultado no campo *nDoPorts* da estrutura *TpDriverInfo*. No caso da CAD12/56 é disponível 1 port de saída digital de 16 bits.

O valor a ser escrito no port de saída digital deve ser passado no parâmetro *Value*. A primitiva retorna 0 (zero) se foi executada com sucesso. Os erros mais comuns na execução da primitiva são número do port de saída digital inválido e driver não instalado.

Veja também as primitivas *WDM_ReadDI* e *WDM_DriverInfo*.

4.12. Primitiva WDM _Clear_CM

Object Pascal:
```
Procedure WDM_Clear_CM;
```

Visual Basic:
```
Declare Sub WDM_Clear_CM Lib "WDM_1256.DLL" ()
```

Esta primitiva limpa a memória de canais utilizada na aquisição por interrupção. Através da memória de canais, o aplicativo informa ao driver a relação dos canais a serem aquisitados durante a aquisição por interrupção. Para cada canal analógico a ser aquisitado, o aplicativo deverá realizar uma chamada da primitiva *WDM_Insert_AI_CM*. Analogamente, deve-se executar a primitiva *WDM_Insert_DI_CM* para os ports de entrada digital. Antes, porém, o aplicativo deverá limpar a memória de canais através da chamada da primitiva *WDM_Clear_CM*.

Veja também as primitivas *WDM_Insert_AI_CM* e *WDM_Insert_DI_CM*.

4.13. Primitiva WDM _Insert_AI_CM

Object Pascal:
```
Function WDM_Insert_AI_CM (Channel, iRange: byte): dword;
```

Visual Basic:
```
Declare Function WDM_Insert_AI_CM Lib "WDM_1256.DLL" _
        (ByVal Channel As Byte, ByVal iRange As Byte) As Long
```

O programa aplicativo deve realizar chamadas sucessivas desta primitiva para informar os canais de entrada analógica a serem lidos na aquisição por interrupção. A ordem de chamada desta primitiva determina a ordem em que os canais serão lidos. Antes da chamada desta primitiva para especificar o primeiro canal a ser aquisitado, deve-se chamar a primitiva *WDM_Clear_CM* para limpar a memória de canais.

Os parâmetros de entrada *Channel* e *iRange* correspondem respectivamente ao número do canal A/D e o

índice da faixa de entrada (veja 4.3. Primitiva WDM_DriverInfo). A primitiva retorna 0 (zero) se foi executada com sucesso. Os erros mais comuns na execução da primitiva são canal de entrada analógica inválido, faixa de entrada inválida, excedeu a capacidade da memória de canais e driver não instalado.

Veja também as primitivas *WDM_Insert_DI_CM* e *WDM_Clear_CM*.

4.14. Primitiva WDM _Insert_DI_CM

Object Pascal:
```
Function WDM_Insert_DI_CM (Group: byte): dword;
```

Visual Basic:
```
Declare Function WDM_Insert_DI_CM Lib "WDM_1256.DLL" _
        (ByVal Group As Byte) As Long
```

O programa aplicativo deve realizar chamadas sucessivas desta primitiva para informar os ports de entrada digital a serem lidos na aquisição por interrupção. A ordem de chamada desta primitiva determina a ordem em que os ports serão lidos.

O parâmetro *Group* corresponde ao número do port de entrada de entrada digital. O port de entrada digital é lidos em grupo de 16 bits e no caso da placa CAD12/56, deve-se passar o valor 0 (zero) neste parâmetro. A primitiva retorna 0 (zero) se foi executada com sucesso. Os erros mais comuns na execução da primitiva são número do port de entrada digital inválido, excedeu a capacidade da memória de canais e driver não instalado.

Veja também as primitivas *WDM_Insert_AI_CM* e *WDM_Clear_CM*.

4.15. Primitiva WDM _AcquisitionSetup

Object Pascal:
```
Function WDM_AcquisitionSetup (SampleFreq: single;
                               Var ErrorCode: dword): dword;
```

Visual Basic:
```
Declare Function WDM_AcquisitionSetup Lib "WDM_1256.DLL" _
        (ByVal SampleFreq As Single, ErrorCode As Long) As Long
```

Esta primitiva prepara a aquisição de sinais por interrupção e deve ser executada depois da programação da memória de canais através das funções *WDM_Clear_CM*, *WDM_Insert_AI_CM* e *WDM_Insert_DI_CM*.

A frequência em que os sinais serão amostrados deve ser especificada em hertz no parâmetro *SampleFreq*. No parâmetro de saída *ErrorCode* é retornado o código de erro na programação da aquisição (veja 3.2 Constantes). A primitiva retorna 0 (zero) se foi executada com sucesso.

Veja também as primitivas *WDM_Clear_CM*, *WDM_Insert_AI_CM*, *WDM_Insert_DI_CM*, *WDM_StartAcquisition*, *WDM_GetAcquiredData* e *WDM_StopAcquisition*.

4.16. Primitiva WDM _StartAcquisition

Object Pascal:
```
Procedure WDM_StartAcquisition;
```

Visual Basic:
```
Declare Sub WDM_StartAcquisition Lib "WDM_1256.DLL" ()
```

Esta primitiva inicia a aquisição de sinais por interrupção com os parâmetros programados anteriormente. Ela deve ser executada depois da programação da memória de canais e da preparação da aquisição através das primitivas *WDM_Clear_CM*, *WDM_Insert_AI_CM*, *WDM_Insert_DI_CM* e *WDM_AcquisitionSetup*.

Após a chamada da *WDM_StartAcquisition*, o programa aplicativo tem acesso ao andamento da aquisição através da primitiva *WDM_GetAcquiredData*.

Veja também as primitivas *WDM_Clear_CM*, *WDM_Insert_AI_CM*, *WDM_Insert_DI_CM*, *WDM_AcquisitionSetup*, *WDM_GetAcquiredData* e *WDM_StopAcquisition*.

4.17. Primitiva WDM _StopAcquisition

Object Pascal:
```
Procedure WDM_StopAcquisition;
```

Visual Basic:
```
Declare Sub WDM_StopAcquisition Lib "WDM_1256.DLL" ()
```

Esta primitiva encerra a aquisição de sinais por interrupção.

Veja também a primitiva *WDM_StartAcquisition*.

4.18. Primitiva WDM _GetAcquiredData

Object Pascal:
```
Function WDM_GetAcquiredData (Var AcquiredData: TpAcquiredData): dword;
```

Visual Basic:
```
Não disponível para Visual Basic. Utilize a primitiva WDM_GetAcquiredDaraVB.
```

A primitiva *WDM_GetAcquiredData* verifica o status do andamento da aquisição por interrupção e obtém as últimas amostras aquisitadas pelo driver. O status de erro da aquisição é retornado pela primitiva conforme codificação descrita na tabela do item 3.2.

```
Type {--------------------------- Parâmetro de saída da WDM_GetAquiredData --}
  TpAcquiredData = record
    ieStatus: byte;    { Status da aquisição }
    nSamples: dword;   { Número de amostras aquisitadas por canal }
    nSampGot: dword;   { Número de amostras transferidas por canal }
    iLast   : dword;   { Número da última amostra transferida }
    UserBuf : array [0..16383] of smallint;
  end;
```

A primitiva retorna no parâmetro *AcquiredData* o status do andamento da aquisição. O campo *ieStatus* do type *TpAcquiredData* tem o mesmo significado do valor retornado pela primitiva. O campo *nSamples* informa o número de amostras aquisitadas por canal desde o início da aquisição.

Através desta primitiva o programa aplicativo tem acesso às amostras aquisitadas pelo driver durante a aquisição de sinais por interrupção.

O driver possui um buffer circular de 64K amostras para a aquisição de sinais. Por exemplo, para a aquisição de 4 canais a 200 Hz por canal, o buffer teria capacidade de armazenar até 80 segundos sem que o programa aplicativo remova os dados do buffer circular. No entanto, o programa aplicativo normalmente possui um loop de processamento onde é realizada periódicamente a chamada da primitiva *WDM_GetAcquiredData* para a leitura dos dados amostrados. O tamanho do buffer de aquisição do driver é utilizado apenas para que não haja perdas de dados durante processamentos demorados do programa aplicativo. No exemplo, o programa aplicativo pode ficar até 80 segundos sem retirar dados do buffer de aquisição. Depois deste tempo ocorre *overflow* no buffer.

O campo *UserBuf* de *AcquiredData* é um array que deve ser interpretado como uma matriz. As linhas da matriz representam o índice da amostragem e as colunas os canais (na ordem da memória de canais e por tipo, entradas analógica e depois as entrada digitais).

A primitiva remove as amostras do buffer de aquisição do driver e as transfere para o *UserBuf*. O número de amostras transferidas para o array é limitado pelo tamanho do array (16K amostras) e pelo número de amostras disponíveis no buffer de aquisição do driver. O número de amostras por canal transferidos para o *UserBuf* é retornado no campo *nSampGot*. No campo *iLast* é retornado o número de ordem da última amostra transferida para o *UserBuf*. O número de ordem da amostra se refere ao início da aquisição.

Veja também as primitivas *WDM_StartAcquisition* e *WDM_StopAcquisition*.

4.19. Primitiva WDM _GetAcquiredDataVB

Object Pascal:
```
Function  WDM_GetAcquiredDataVB (Var ieStatus, nSamples, nSampGot,
                                  iLast: dword; Var UserBuf: TpUserBuf): dword;
```

Visual Basic:
```
Declare Function WDM_GetAcquiredDataVB Lib "WDM_1256.DLL" _
        (ieStatus As Long, nSamples As Long, nSampGot As Long, _
         iLast As Long, UserBuf As TpUserBuf) As Long
```

Esta primitiva é análoga à primitiva *WDM_GetAcquiredData*. Os parâmetros de saída da primitiva correspondem aos campos de mesmo nome da estrutura *TpAcquiredData* descrita na primitiva *WDM_GetAcquiredData*.

Veja também as primitivas *WDM_StartAcquisition*, *WDM_StopAcquisition* e *WDM_GetAcquiredData*.