DRIVER DA CAD12/32 PARA WINDOWS 95

1. INTRODUÇÃO

VAD32D.VXD é um driver VxD que gerencia o acesso à placa A/D CAD12/32 no sistema operacional Windows 95.

1.1. Escopo

O objetivo deste documento é apresentar a interface com o driver realizada por módulo em Borland Delphi para Windows 95.

1.2. Aplicabilidade

O driver VAD32D deve ser utilizado somente no sistema operacional Windows 95 ou compatível.

2. INSTALANDO O DRIVER VAD32D

Os arquivos fornecidos para utilização do driver VAD32D são:

VAD32D.VXD	driver da CAD12/32 para Windows 95
VAD32D.DLL	biblioteca DLL de acesso ao driver
VAD32api.pas	módulo em pascal para a interface com o driver
TESTE.EXE	programa de teste do driver / CAD12/32
TESTE.*	arquivo de projeto do programa exemplo em Delphi 2.0
Main.*	formulário e módulo principal do programa exemplo
TesteDIO.*	formulário e módulo de teste de E/S digital
TesteAD.*	formulário e módulo de teste de aquisição por polling
TesteIRQ.*	formulário e módulo de teste de aquisição por interrupção
TESTE.VBP	arquivo de projeto do programa exemplo em Visual Basic 5.0
frmMain.*	formulário e módulo principal do programa exemplo em Visual Basic
Module1.bas	módulo do programa exemplo em Visual Basic
VAD32api.bas	arquivo com as declarações da interface da VAD32D.DLL com o Visual Basic

A instalação do driver VAD32D.VXD é bastante simples, basta copiar os arquivos `VAD32D.VXD` e `VAD32D.DLL` para o diretório onde se encontra o programa aplicativo.

O driver VAD32D.VXD é carregado em tempo de execução do programa aplicativo, não sendo necessário nenhum registro no Windows 95 através do REGEDIT.

Para aplicações em Visual Basic, recomenda-se copias os arquivos VAD32D.VXD e VAD32D.DLL para o diretório do Windows 95.

3. INTERFACE DO DRIVER COM O PROGRAMA APLICATIVO

O acesso do programa aplicativo ao driver é disponibilizado através do módulo em Pascal VAD32api. 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 VAD32api.PAS com as referências à biblioteca VAD32D.DLL.

É fornecido também um exemplo de programa aplicativo implementado em Visual Basic 5.0. As declarações necessárias para a utilização das funções da biblioteca VAD32D.DLL são fornecidas no arquivo VAD32api.bas. Esse arquivo deve ser imcluido no projeto do programa aplicativo em VB5.

3.1. Constantes

Para facilitar a programação do ganho de entrada da CAD12/32, foram definidas as seguintes constantes que podem ser usadas nas primitivas *VAD32D_SetGain* e *VAD32D_WriteCM*:

Constante	Valor	Faixa de entrada
cBi5v0	$F0	-5 a 5 volts
cBi2v5	$E0	-2.5 a 2.5 volts
cBi1v0	$D0	-1 a 1 volts
cBi0v5	$B0	-0.5 a 0.5 volts
cUni5v0	$70	0 a 5 volts
cUni2v5	$60	0 a 2.5 volts
cUni1v0	$50	0 a 1 volts
cUni0v5	$30	0 a 0.5 volts

A tabela seguinte apresenta as constantes retornadas pela primitiva *VAD32D_GetAcquiredData*.

Constante	Valor	Descrição
ieNoError	0	Não ocorreu nenhum erro desde a última chamada da primitiva VAD32D_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.
ieBufOverrun	3	Overrun no buffer de aquisição do driver
ieVPICD_Error	4	Esse erro ocorre quando a interrupção utilizada pela placa A/D não pode ser virtualizada pelo VPICD.
ieEndAcquisition	5	Indica final de aquisição quando foram aquisitadas o número de amostras especificado.
ieDriver	100	Erro de acesso ao driver
ieAcqStop	101	Aquisição não está ativa
ieBufSize	102	Dimensão errada do buffer passado na VAD32D_GetSamples

3.2. Estrutura de Dados do Driver

O driver mantém uma estrutura de dados interna com todas as informações necessárias para a utilização da CAD12/32. Algumas dessas informações (endereço padrão, canal de interrupção, frequência de amostragem

e número de canais a serem aquisitados) são passadas pelo programa aplicativo através de primitivas do VAD32D.

O driver possui uma memória de canais onde são armazenados 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 +/-2.5 volts, canal 12 em +/- 1.0 volts e o canal 15 em +/-5 volts).

O driver possui um buffer circular de 256K amostras para a aquisição de sinais. Por exemplo, para a aquisição de 2 canais a 100 Hz por canal, o buffer teria capacidade de armazenar até 20 minutos sem que o programa aplicativo remova os dados do buffer circular.

4. DESCRIÇÃO DAS PRIMITIVAS DO VAD32Dapi

Neste tópico são descritas as primitivas de acesso ao driver para Windows 95. A tabela abaixo lista as primitivas normalmente utilizadas pelo programa aplicativo.

Primitiva	**Descrição**
VAD32D_W32_Open	Inicia o acesso ao driver
VAD32D_W32_Close	Finaliza o acesso ao driver
VAD32D_GetVersion	Informação a versão e revisão do driver
VAD32D_InitCard	Inicia a CAD12/32
VAD32D_SetGain	Define o ganho do conversor A/D
VAD32D_ReadAI	Leitura de canal de entrada analógica
VAD32D_ClearCM	Limpa a memória de canais da aquisição por interrupção
VAD32D_WriteCM	Escreve na memória de canais da aquisição por interrupção
VAD32D_StartAcquisition	Inicia a aquisição por interrupção
VAD32D_StopAcquisition	Finaliza a aquisição por interrupção
VAD32D_GetAcquiredData	Leitura das amostras do buffer de aquisição por interrupção
VAD32D_GetSamples	Leitura das amostras do buffer de aquisição por interrupção (utilizada por programas aplicativos em Visual Basic 5.0)

As funções da bilioteca DLL são descritas com a sintaxe do Object Pascal.

4.1. Primitiva VAD32D_W32_Open

```
Function VAD32D_W32_Open: WordBool;
```

Esta primitiva abre o acesso ao driver e deve ser a primeira primitiva a ser executada pelo programa aplicativo. A primitiva retorna o valor *true* quando a primitiva foi executada com sucesso ou o valor *false* quando houve algum erro na execução da primitiva . Normalmente o erro ocorre quando o driver não foi encontrado. Neste caso, verifique se o arquivo **VAD32D.VXD** se encontra no diretório onde o programa está sendo executado.

Veja também a primitiva *VAD32D_W32_Close*.

4.2. Primitiva VAD32D_W32_Close

```
Procedure VAD32D_W32_Close;
```

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 *VAD32D_W32_Open* deve haver uma correspondente chamada da *VAD32D_W32_Close*.

Veja também a primitiva *VAD32D_W32_Open*.

4.3. Primitiva VAD32D_GetVersion

```
Function VAD32D_GetVersion: word;
```

Esta primitiva devolve o número da versão do driver. O byte mais significativo (MSB) do valor retornado contém o número da versão e o byte menos significativo o número da revisão. Por exemplo para o driver VAD32D.VXD 1.03 o MSB seria 1 e o LSB seria 3.

4.4. Primitiva VAD32D_InitCard

```
Function VAD32D_InitCard (AbaCAD1232: word;
                          AIntNo: byte): WordBool;
```

Esta primitiva inicia a CAD12/32 e permite o acesso às demais funções do driver.

O programa aplicativo deve passar no parâmetro *AbaCAD1232* o endereço base de I/O da placa CAD12/32. No parâmetro *AIntNo* deve-se especificar o canal de interrupção utilizado pela CAD12/32.

A primitiva *VAD32D_InitCard* retorna o valor *true* se a iniciação da CAD12/32 foi bem sucedida ou o valor *false* se ocorreu algum erro durante a sua execução. Em caso de erro, verifique se os endereços base das placas estão corretos

4.5. Primitiva VAD32D_SetGain

```
Procedure VAD32D_SetGain (GainCode: byte);
```

Esta primitiva permite reprogramar o ganho da CAD12/32 (faixa de entrada do conversor A/D). Ao ser iniciada a CAD12/32, o driver programa o ganho para a faixa de entrada de +/-5 volts. Se desejar uma outra faixa de entrada, o aplicativo deve chamar esta primitiva passando no parâmetro *GainCode* o código de ganho desejado (utilize as constantes listadas na tabela do ítem 3.1).

Veja também a primitiva *V*AD32*D_ReadAI*.

4.6. Primitiva VAD32D_ReadAI

```
Function VAD32D_ReadAI (Channel: byte; Var Value: smallint): WordBool;
```

Esta primitiva realiza a leitura de um canal analógico da CAD12/32.

O programa aplicativo deve passar no parâmetro *Channel* o número do canal A/D (0 a 31) a ser lido.

A faixa de entrada a ser utilizada na conversão do canal A/D deve ser previamente programada através da primitiva *VAD32D_SetGain*.

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 valore de -32768 a 32767. A primitiva retorna *true* se a conversão foi

realizada com sucesso ou *false* se houve algum erro na execução da primitiva.

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

Veja também a primitiva *VAD32D_SetGain*.

4.7. Primitiva VAD32D_ClearCM

```
Procedure VAD32D_ClearCM;
```

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 a ser aquisitado, o aplicativo deverá realizar uma chamada da primitiva *VAD32D_WriteCM*. Antes, porém, o aplicativo deverá limpar a memória de canais através da chamada da primitiva *VAD32D_ClearCM*.

Veja também a primitiva *VAD32D_WriteCM*.

4.8. Primitiva VAD32D_WriteCM

```
Procedure VAD32D_WriteCM (Channel, GainCode: byte);
```

O programa aplicativo deve realizar chamadas sucessivas desta primitiva para informar os canais 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 *VAD32D_ClearCM* para limpar a memória de canais.

Os parâmetros de entrada *Channel* e *GainCode correspondem* respectivamente ao número do canal A/D e o código da faixa de entrada.

Veja também a primitiva *VAD32D_ClearCM*.

4.9. Primitiva VAD32D_StartAcquisition

```
Function VAD32D_StartAcquisition (AFreqAq: single;
                                  AnSamples: longint): WordBool;
```

Esta primitiva inicia a aquisição de sinais por interrupção com os parâmetros setados anteriormente. Ela deve ser executada depois da programação da memória de canais através das funções *VAD32D_ClearCM* e *VAD32D_WriteCM*.

A frequência em que os sinais serão amostrados deve ser especificada em hertz no parâmetro *FreqAq* e o número total de amostras por canal a serem aquisitados deve ser passado no parâmetro *AnSamples*. A primitiva retorna o valor true se a execução foi bem sucedida ou valor false se houve alguma erro na sua execução. As causas possíveis são: nenhum canal programado para aquisição, driver não iniciado ou CAD12/32 não iniciada.

Após a chamada da *VAD32D_StartAcquisition*, o programa aplicativo tem acesso ao andamento da aquisição através da primitiva *VAD32D_GetAcquiredData*.

Veja também as primitivas *VAD32D_ClearCM*, *VAD32D_WriteCM*, *VAD32D_GetAcquiredData* e *VAD32D_StopAcquisition*.

4.10. Primitiva VAD32D_StopAcquisition

```
Procedure VAD32D_StopAcquisition;
```

Esta primitiva encerra a aquisição de sinais por interrupção. Para cada chamada da primitiva *VAD32D_StartAcquisition* deve haver uma correspondente chamada da *VAD32D_StopAcquisition*.

Veja também a primitiva *VAD32D_StartAcquisition*.

4.11. Primitiva VAD32D_GetAcquiredData

```
Function VAD32D_GetAcquiredData
           (Var AcquiredData: TpAcquiredData): byte;
```

A primitiva *VAD32D_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 3.1.

```
Type {---------------- Parâmetro de saída da apiW32_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 256K amostras para a aquisição de sinais. Por exemplo, para a aquisição de 2 canais a 100 Hz por canal, o buffer teria capacidade de armazenar até 20 minutos 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 *VAD32D_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 10 minutos 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).

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.

Esta primitiva não deve ser utilizada em programas aplicativos implementados em Visual Basic. No lugar

desta primitiva, deve ser utilizada a primitiva *VAD32D_GetSamples*.

Veja também as primitivas *VAD32D_StartAcquisition* e *VAD32D_StopAcquisition*.

4.12. Primitiva VAD32D_GetSamples

```
Function VAD32D_GetSamples (Var ieStatus: byte;
          Var iSample, nSampRead: longint; BufSize: Longint;
          Var UserBuffer: TpBufArray): WordBool;
```

A primitiva *VAD32D_GetSamples* é análoga à primitiva *VAD32D_GetAcquiredData* e é normalmente utilizada por programas aplicativos escritos em Visual Basic.

A primitiva retorna *true* se há dados novos aquisitados e transferidos para o *UserBuffer*. O *UserBuffer* é um array com até 8192 posições de smallint onde são copiados as amostras aquisitadas pelo driver. O tamanho do *UserBuffer* em amostras deve ser passado no parâmetro *BufSize*.

O número de amostras copiadas no *UserBuffer* por canal é devolvido pelo driver no parâmetro *nSampRead*. O número da última amostra copiada no buffer é devolvida no parâmetro *iSample*. Através desse parâmetro o programa aplicativo pode acompanhar o andamento da aquisição por interrupção.

O status de erro da aquisição por interrupção é retornada no parâmetro *ieStatus* que possui a mesma coidificação da primitiva *VAD32D_GetAcquiredData*.

Veja também as primitivas *VAD32D_GetAcquiredData*, VAD32D_*StartAcquisition* e *VAD32D_StopAcquisition*.

4.13. Outras Primitivas

Além das primitivas descritas anteriormente, o módulo VAD32Dapi possui outras primitivas que podem ser utilizadas para testes. A tabela abaixo relaciona essas primitivas.

Primitiva	**Descrição**
VAD32D_ReadDI	Leitura dos ports de entrada digital P0 e P1 da CAD12/32
VAD32D_ReadDO	Devolve o último valor escrito nos ports de saída digital P0 e P1 da CAD12/32
VAD32D_WriteDO	Escreve nos ports de saída digital P0 e P1 da CAD12/32
VAD32D_ProgTimer	Programa o timer da CAD12/32
VAD32D_ReadTimer	Leitura o timer da CAD12/32