WAC2122A.DLL :

DRIVER DA PLACA CONTROLADORA AC2122 DO ADS2000-IP PARA WINDOWS 9X/NT4/2000

1. INTRODUÇÃO

1.1. Escopo

O objetivo deste documento é apresentar as funções da biblioteca WAC2122A.DLL de acesso à placa controladora AC2122 do ADS2000-IP no Windows.

1.2. Audiência

Esta documentação se destina aos programadores de aplicativos em Borland Delphi 5 e VisualBasic que realizam aquisição de sinais com a placa AC2122.

Para um bom entendimento desta nota de aplicação, recomenda-se que o leitor tenha conhecimento prévio dos seguintes assuntos:

- aquisição de dados
- utilização da placa AC2122 e do ADS2000-IP
- Borland Delphi 5 ou Visual Basic 5
- TCP/IP

1.2. Aplicabilidade

O driver WAC2122A pode ser utilizado em Windows 9x e Windows NT4 em aplicações destinadas à aquisição em sistemas com um único controlador AC2122 (não suporta múltiplos controladores em configuração master/slaves). O driver também não possui suporte para contadores, nem controla módulos de expansão do barramento ADS2000.

Para utilização do driver, o microcomputador deverá ter uma interface de rede Ethernet e o protocolo TCP/IP instalado no Windows.

Para maiores informações sobre a AC2122, consulte o manual de referência técnica da placa.

2. INSTALANDO O DRIVER WAC2122

Os arquivos fornecidos para utilização do driver WAC2122:

WAC2122A.DLL	Biblioteca DLL de acesso à placa controladora AC2122
S145U03A.PDF	Este documento em formato Adobe Acrobat
TESTE.EXE	Programa de teste do driver
Delphi5*.*	Exemplo com programa fonte em Borland Delphi 5
VB5*.*	Exemplo com programa fonte em Visual Basic 5

O driver não necessita de nenhuma instalação especial. Basta copiar o arquivo **`WAC2122A.DLL`** para o diretório do seu programa aplicativo (no caso de aplicações Delphi) ou para o diretório do Windows (no caso de aplicações Delphi e VB5).

3. INTERFACE DO DRIVER COM O PROGRAMA APLICATIVO

O acesso do programa aplicativo ao driver é disponibilizado através do módulo em Pascal WACapi. 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 WACapi.pas com as referências à biblioteca WAC2122A.DLL. A biblioteca DLL deverá ser copiada para o mesmo diretório do programa aplicativo.

3.1. Estrutura de Dados do Driver

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 ±10 volts, canal 12 em ±2.0 volts e o canal 15 em 0-10 volts).

O driver possui um buffer circular de 4M amostras para a aquisição de sinais. Por exemplo, para a aquisição de 4 canais a 1000 Hz por canal, o buffer teria capacidade para armazenar até 17 minutos sem que o programa aplicativo remova os dados do buffer circular.

3.2. Constantes

A tabela seguinte apresenta as constantes referentes aos códigos de erros retornados pelas primitivas.

Constante	Valor	Descrição
cNoError	0	Não ocorreu nenhum erro
cErrorNotConnected	101	Não foi estabelecida conexão com a AC2122
cErrorNoSignal	102	Nenhum sinal habilitado para aquisição
cErrorAcqNotActive	103	Aquisição não foi ativada
cErrorAcqBufOverrun	111	Sobreposição no buffer de aquisição

4. DESCRIÇÃO DAS FUNÇÕES DO WAC2122A.DLL

Neste tópico são descritas as funções da biblioteca DLL de acesso ao driver da placa controladora AC2122. A tabela abaixo lista as funções da biblioteca WAC2122A.DLL.

Primitiva	Descrição
OpenDriver	Inicia o acesso ao driver
CloseDriver	Finaliza o acesso ao driver
GetVersion	Informa a versão e revisão do driver
GetAiCaps	Informa os recursos de entrada analógica disponíveis no driver
ConnectAC2122	Estabelece a conexão de rede com a placa AC2122
DisconnectAC2122	Finaliza a conexão de rede com a placa AC2122
Clear_CM	Limpa a memória de canais da aquisição de sinais
Insert_AI_CM	Insere um canal de entrada analógica para aquisição
AcquisitionSetup	Programa aquisição por sinais
StartAcquisition	Inicia a aquisição de sinais
StopAcquisition	Finaliza a aquisição de sinais
GetSamples	Leitura das amostras do buffer de aquisição
ReadAC2122DigitalPort	Leitura das entradas digitais do AC2122
WriteAC2122DigitalPort	Escrita nas saídas digitais do AC2122

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 OpenDriver

Object Pascal:
```
Function OpenDriver: boolean;
```

Visual Basic:
```
Declare Function OpenDriver Lib "WAC2122A.DLL" () As Byte
```

Esta primitiva abre o acesso ao driver e deve ser a primeira primitiva a ser executada pelo programa aplicativo. A primitiva retorna o valor 1 (true) se foi executada com sucesso.

Veja também a primitiva *CloseDriver*.

4.2. Primitiva CloseDriver

Object Pascal:
```
Procedure CloseDriver;
```

Visual Basic:
```
Declare Sub CloseDriver Lib "WAC2122A.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 *OpenDriver* deve haver uma correspondente chamada da primitiva *CloseDriver*.

Veja também a primitiva *OpenDriver* .

4.3. Primitiva GetVersion

Object Pascal:
```
Procedure GetVersion (Var VersionHigh, VersionLow: byte);
```

Visual Basic:
```
Declare Sub GetVersion Lib "WAC2122.DLL" _
        (VersionHigh As Byte, VersionLow As Byte)
```

Esta primitiva devolve nos parâmetros *VersionHigh* e *VersionLow*, respectivamente o byte mais significativo e o byte menos significativo da versão do driver.

4.4. Primitiva GetAiCaps

Object Pascal:
```
Procedure GetAiCaps (Var nAiChannels, nAiRange: smallint;
                     Var AiRange: TpAiRange);
```

Visual Basic:
```
Declare Sub GetAiCaps Lib "WAC2122.DLL" _
        (nAiChannels As Integer, nAiRange As Integer, _
         AiRange As TpAiRange)
```

E primitiva informa nos parâmetros de saída os recursos de entrada analógica disponibilizados pelo driver. O parâmetro *nAiChannels* informa o número de entradas analógicas da placa AC2122. As entradas analógicas podem ser expandidas com o uso de módulos de expansão para condicionamento de sinais, elevando o número de entradas analógicas para um máximo de 64 entradas, 16 entradas por módulo de expansão.

O valor retornado em *nAiRange* indica o número de faixas de entrada do conversor A/D. No parâmetro *AiRange* é retornado um vetor de 16 posições (apenas 4 posições são usadas por este driver) 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 no vetor. Para a placa AC2122 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

4.5. Primitiva ConnectAC2122

Object Pascal:
```
Function ConnectAC2122 (IP: string) : boolean;
```

Visual Basic:
```
Declare Function ConnectAC2122 Lib "WAC2122.DLL" _
        (ByVal IP As String) As Byte
```

Esta primitiva estabelece a conexão de rede com a placa controladora AC2122. Ela deve ser executa antes das primitivas de aquisição de sinais.

O parâmetro de entrada *IP* é uma string onde deve ser especificada o endereço IP da placa controladora AC2122. Por exemplo, o endereço IP 192.168.1.1. Consulte no manual de referência técnica da AC2122 como configurar o endereço IP da placa.

A primitiva retorna o valor 1 (true) se foi estabelecida com sucesso a conexão de rede com a placa AC2122.

Veja também a primitiva *DisconnectAC2122*.

4.6. Primitiva DisconnectAC2122

Object Pascal:
```
Procedure DisconnectAC2122;
```

Visual Basic:
```
Declare Sub DisconnectAC2122 Lib "WAC2122.DLL" ()
```

Esta primitiva primitiva finaliza a conexão de rede com a placa controladora AC2122.

Veja também a primitiva *ConnectAC2122*.

4.7. Primitiva Clear_CM

Object Pascal:
```
Procedure Clear_CM;
```

Visual Basic:
```
Declare Sub Clear_CM Lib "WAC2122.DLL" ()
```

Esta primitiva limpa a memória de canais utilizada na aquisição de sinais. Através da memória de canais, o aplicativo informa ao driver a relação dos canais a serem aquisitados. Para cada canal analógico a ser aquisitado, o aplicativo deverá realizar uma chamada da primitiva *Insert_AI_CM*. Antes, porém, o aplicativo deverá limpar a memória de canais através da chamada da primitiva *Clear_CM*.

Veja também a primitiva *Insert_AI_CM*.

4.8. Primitiva Insert_AI_CM

Object Pascal:
```
Function Insert_AI_CM (ModAddr, Channel, iRange: byte): boolean;
```

Visual Basic:
```
Declare Function Insert_AI_CM Lib "WAC2122.DLL" _
        (ByVal ModAddr As Byte, ByVal Channel As Byte, _
         ByVal iRange As Byte) As Byte
```

O programa aplicativo deve realizar chamadas sucessivas desta primitiva para informar os canais de entrada analógica a serem lidos na aquisição. 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 *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 4.4. Primitiva GetAiCaps). A primitiva retorna 1 (true) 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 iniciado.

No parâmetro *ModAddr* deve ser passado o endereço do módulo de expansão de condicionamento de sinais. Consulte nos manuais de referência técnica dos módulos de expansão como configurar o endereço do módulo. No parâmetro *Channel* deve ser passado o número do canal de entrada analógica (0 a 15) do módulo. Se não estiver utilizando nenhum módulo de expansão, passe o valor 0 no parâmetro *ModAddr*.

Veja também a primitiva *Clear_CM*.

4.9. Primitiva AcquisitionSetup

Object Pascal:
```
Function AcquisitionSetup (fExtClock: boolean; FsUser: single;
                           Var SampleFreq: single;
                           Var ErrorCode: integer): boolean;
```

Visual Basic:
```
Declare Function AcquisitionSetup Lib "WAC2122.DLL" _
        (ByVal fExtClock As Byte, ByVal FsUser As Single, _
         SampleFreq As Single, ErrorCode As Long) As Byte
```

Esta primitiva prepara a aquisição de sinais e deve ser executada depois da programação da memória de canais através das funções *Clear_CM* e *Insert_AI_CM*.

A frequência em que os sinais serão amostrados deve ser especificada em hertz no parâmetro *FsUser*. A frequência de amostragem efetivamente programada, que depende do número de canais da resolução de frequência da AC2122, é retornada no parâmetro de saída *SampleFreq*. Caso deseje utilizar um clock de aquisição de sinais externo à AC2122, passe 1 (true) no parâmetro de entrada *fExtClock*. Consulte o manual de referência técnica da placa AC2122 como conectar um sinal de clock externo, não se esqueça também de conectar o sinal de trigger externo.

No parâmetro de saída *ErrorCode* é retornado o código de erro na programação da aquisição (veja 3.2 Constantes). A primitiva retorna 1 (true) se ela foi executada com sucesso.

Veja também as primitivas *Clear_CM*, *Insert_AI_CM*, *StartAcquisition*, *GetSamples* e *StopAcquisition*.

4.10. Primitiva StartAcquisition

Object Pascal:
```
Function StartAcquisition : boolean;
```

Visual Basic:
```
Declare Function StartAcquisition Lib "WAC2122.DLL" () As Byte
```

Esta primitiva inicia a aquisição de sinais com os parâmetros programados anteriormente. Ela deve ser executada depois da programação da memória de canais e da preparação da aquisição através das primitivas *Clear_CM*, *Insert_AI_CM* e *AcquisitionSetup*.

A primitiva retorna 1 (true) se foi executada com sucesso. Após a chamada da *StartAcquisition*, o programa aplicativo tem acesso ao andamento da aquisição através da primitiva *GetSamples*.

Veja também as primitivas *Clear_CM*, *Insert_AI_CM*, *AcquisitionSetup*, *GetSamples* e *StopAcquisition*.

4.11. Primitiva StopAcquisition

Object Pascal:
```
Procedure StopAcquisition ;
```

Visual Basic:
```
Declare Sub StopAcquisition Lib "WAC2122.DLL" ()
```

Esta primitiva encerra a aquisição de sinais.

Veja também a primitiva *StartAcquisition*.

4.12. Primitiva GetSamples

Object Pascal:
```
Function GetSamples (Var ErrorCode, iSample, nSampRead: integer;
                     Var UserBuf: TpUserBuf): boolean;
```

Visual Basic:
```
Declare Function GetSamples Lib "WAC2122.DLL" _
        (ErrorCode As Long, iSample As Long, _
         nSampReadAs As Long, UserBuf As TpUserBuf) As Byte
```

Através desta primitiva o programa aplicativo tem acesso às amostras aquisitadas pelo driver. A função retorna o valor 1 (true) se foi executada com sucesso.

O driver possui um buffer circular de 4M amostras para a aquisição de sinais. Por exemplo, para a aquisição de 16 canais a 140 Hz por canal, o buffer teria capacidade de armazenar até 29 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 função *GetSamples* 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 30 minutos sem retirar dados do buffer de aquisição. Depois desse tempo ocorre *overflow* no buffer.

A função retorna no parâmetro de saída *ErrorCode* o código de ocorrência de erro durante a aquisição (veja 3.2 Constantes).

O parâmetro de saída *UserBuf* 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 de inserção na memória de canais). Ou seja, *UserBuf* é uma tabela em que as linhas são os "instantes" de amostragem e as colunas os canais. A função 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 e pelo número de amostras disponíveis no buffer de aquisição do driver. O número de amostras por canal transferidos para o array é retornado no parâmetro de saída *nSampRead*. No parâmetro *iSample* é retornado o número de ordem da última amostra transferida para o array. O número de ordem da amostra se refere ao início da aquisição. Em cada chamada da função *GetSamples* é transferido no máximo um total de 16K amostras.

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).

Veja também a função *StartAcquisition* e *StopAcquisition*.

4.13. Primitiva ReadAC2122DigitalPort

Object Pascal:
```
Fucntion ReadAC2122DigitalPort (Var Value: word): boolean;
```

Visual Basic:
```
Declare Function ReadAC2122DigitalPort Lib "WAC2122A.DLL" _
        (Value As Integer) As Byte
```

Através desta primitiva o programa aplicativo tem acesso aos estados das 8 entradas digitais da placa AC2122.

O valor retornado no parâmetro *Value* é a codificação, bit a bit, dos estados de cada entrada digital, sendo que o bit 0 corresponde à entrada 0, o bit 1 corresponde à entrada 1, e assim por diante. Assim, a faixa de valores esperado em *Value* varia de 0 (todas as entradas digitais zeradas, ou desacionadas) até 255 (todas as entradas digitais em 1, ou acionadas).

A função retorna o valor 1 (true) se foi executada com sucesso.

4.14. Primitiva WriteAC2122DigitalPort

Object Pascal:
```
Fucntion WriteAC2122DigitalPort (Value: word): boolean;
```

Visual Basic:
```
Declare Function WriteAC2122DigitalPort Lib "WAC2122A.DLL" _
        (ByVal Value As Integer) As Byte
```

Através desta primitiva o programa aplicativo tem acesso aos estados das 8 entradas digitais da placa AC2122.

No parâmetro *Value* deve ser especificado o estado em que se deseja colocar cada uma das 8 saídas digitais do AC2122. O valor de *Value* deve ser a codificação, bit a bit, dos estados de cada saída digital, sendo que o bit 0 corresponde à saída 0, o bit 1 corresponde à saída 1, e assim por diante. Assim, a faixa de valores esperado em *Value* varia de 0 (todas as saídas digitais zeradas, ou desacionadas) até 255 (todas as saídas digitais em 1, ou acionadas).

A função retorna o valor 1 (true) se foi executada com sucesso.