

BIBLIOTECA LynxDAS.DLL

PARA ACESSAR DRIVERS DO AqDados 7

1. INTRODUÇÃO

1.1. Escopo

O objetivo deste documento é apresentar as funções da biblioteca LynxDAS.DLL que permite o acesso a placas A/D da Lynx.

1.2. Audiência

Esta documentação se destina aos programadores de aplicativos em LabVIEW 5.0 que realizam aquisição de sinais com as placas CAD12/32, CAD12/36 e CAD12/56.

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

- ☐ aquisição de dados
- ☐ AqDados 7.0
- ☐ Borland Delphi
- ☐ Visual Basic
- ☐ utilização da CAD12/32
- ☐ utilização da CAD12/36
- ☐ utilização da CAD12/56

1.3. Aplicabilidade

A biblioteca LynxDAS.DLL encapsula o acesso a drivers para placas ISA desenvolvidos para o AqDados 7. As placas A/D da Lynx suportadas pela biblioteca DLL e pelos drivers são: CAD12/32, CAD12/36 e CAD12/56.

A versão atual da biblioteca DLL não possui suporte para saídas analógicas e aquisição por DMA.

	Elaboração	Aprovação
Área	SW.P&D	SW.P&D
Nome	Lauro	Lauro
Data	15/mai/2002	15/mai/2002
Visto		

2. INSTALAÇÃO

Os arquivos fornecidos para utilização da biblioteca DLL são:

LynxDAS.DLL	biblioteca DLL de acesso ao driver
LDASapi.pas	módulo em Pascal para a interface com a biblioteca DLL
LDASapi.bas	módulo em Visual Basic para a interface com biblioteca DLL
TESTE.EXE	programa de teste do driver
Delphi*.*	exemplo com programa fonte em Borland Delphi 5.0
VB5*.*	exemplo com programa fonte em Visual Basic 5.0

2.1. Drivers Disponíveis

A biblioteca LynxDAS.DLL utiliza os drivers de aquisição de para placas A/D ISA desenvolvidos para o AqDados 7.

A tabela abaixo lista os drivers disponíveis e as respectivas plataformas Windows.

Nome do Driver	Tipo	Placa A/D	Plataforma Windows
WDM1232a	WDM	CAD12/32	Windows 98/ME/2000/XP
WDM1236a	WDM	CAD12/36	Windows 98/ME/2000/XP
WDM1256a	WDM	CAD12/56	Windows 98/ME/2000/XP
NT1232a	NT4	CAD12/32	Windows NT4/2000/XP
NT1236a	NT4	CAD12/36	Windows NT4/2000/XP
NT1256a	NT4	CAD12/56	Windows NT4/2000/XP

2.2. Instalando um Driver WDM Compatível com o AqDados 7

Os drivers do tipo LynxDAS WDM podem ser instalados nas seguintes plataformas Windows:

- Windows 98
- Windows ME
- Windows 2000
- Windows XP

A seguir é apresentado um roteiro para a instalação de um driver WDM no Windows 98/ME. A instalação no Windows 2000 e no Windows XP é análoga.



Para instalar o driver no Windows 98/ME, siga os seguintes passos:

1. Abra o *Painel de Controle* do Windows selecionando a opção *Painel de Controle* do menu *Iniciar/Configurações* do Windows.
2. Na janela *Painel de Controle* do Windows, ative o ícone *Adicionar novo hardware*. Na caixa de diálogo *Assistente para adicionar novo hardware* apresentada, clique sobre o botão **Avançar**.
3. É apresentada a mensagem *Dispositivo que você deseja instalar está na lista abaixo?*, selecione **Não** e clique sobre o botão **Avançar**.
4. É apresentada a mensagem *Deseja que o Windows procure o seu novo hardware?*, selecione **Não** e clique sobre o botão **Avançar**.
5. O assistente apresenta a lista *Tipo de hardware*. Selecione a opção *Outros dispositivos* ou

LynxDAS Device na lista apresentada e clique sobre o botão **Avançar**.

6. Clique sobre o botão **Com disco**. Informe ao assistente de instalação do Windows onde se encontram os arquivos de instalação do driver. Se for um driver novo, os arquivos estarão normalmente em disquete. Os arquivos podem estar também no subdiretório **Drivers\WDM** localizado no diretório onde foi instalado o *AqDados*. Clique sobre o botão **OK** para continuar.
7. Selecione o modelo da placa A/D e clique sobre o botão **Avançar**.
8. O assistente de instalação do Windows irá continuar a instalação e verificará possíveis conflitos de recursos de hardware (intervalo de endereçamento de E/S e pedido de interrupção). Clique sobre o botão **Avançar**.
9. Clique sobre o botão **Concluir**. O Windows será reiniciado para completar a instalação do driver.

Após a instalação do driver, verifique se os recursos de hardware alocados pelo assistente de instalação do Windows correspondem aos que estão configurados pelos jumpers da sua placa. Consulte o manual de referência técnica da sua placa A/D como configurar o endereço base de E/S da placa e o canal de pedido de interrupção. Você pode também consultar o arquivo de help do driver.

Os recursos de hardware alocados pelo assistente de instalação do Windows ao driver podem ser alterados se eles estiverem em conflito com outras placas ou drivers instalados no seu microcomputador. A seguir é apresentado um roteiro para alterar os recursos de hardware alocados ao driver.

 **Para alterar os recursos de hardware alocados ao driver, siga os seguintes passos:**

1. Abra o *Painel de Controle* do Windows selecionando a opção *Painel de Controle* do menu *Iniciar/Configurações* do Windows.
2. Na janela *Painel de Controle* do Windows, ative o ícone *Sistema*. Na caixa de diálogo *Propriedades do Sistema* apresentada, selecione a pasta *Gerenciador de Dispositivos*.
3. Selecione o modelo da sua placa A/D na lista *LynxDAS* e clique sobre o botão **Propriedades**.
4. Na caixa de diálogo de propriedades do driver da placa A/D, selecione a pasta *Recursos*.
5. Para alterar o canal de pedido de interrupção alocado ao driver, selecione o recurso *Pedido de interrupção* e clique sobre o botão **Alterar configuração**. Na caixa de diálogo apresentada, modifique para o canal de pedido de interrupção desejado. Verifique na parte inferior da caixa de diálogo se há informações de conflito com outros drivers. Para confirmar a seleção, clique sobre o botão **OK**.
6. Para alterar o intervalo de E/S alocado ao driver, selecione o recurso *Intervalo de entrada/saída* e clique sobre o botão **Alterar configuração**. Na caixa de diálogo apresentada, modifique para o intervalo de entrada/saída desejado. Verifique na parte inferior da caixa de diálogo se há informações de conflito com outros drivers. Para confirmar a seleção, clique sobre o botão **OK**.
7. Feche a caixa de diálogo de propriedades do driver.

Se você modificou os parâmetros do driver, não se esqueça de modificar apropriadamente os respectivos jumpers de configuração da placa A/D. Para maiores informações sobre os jumpers de configuração da placa, consulte o manual de referência técnica da placa ou o help do driver.

2.3. Instalando um Driver NT4 Compatível com o AqDados 7

Os kernel mode drivers NT4 compatíveis com o *AqDados 7* podem ser instalados nas seguintes plataformas Windows:

- Windows NT4
- Windows 2000
- Windows XP

Os drivers NT4 podem ser utilizados também no Windows 2000 e no Windows XP, no entanto, caso haja um driver WDM para a sua placa A/D compatível com o *AqDados 7*, recomendamos a instalação do driver WDM no lugar do driver NT4.

Os drivers do tipo kernel mode driver NT4 compatíveis com o *AqDados 7* são instalados com o auxílio do próprio *AqDados*. Para maiores informações, consulte o manual do *AqDados 7*.

3. INTERFACE COM O PROGRAMA APLICATIVO

O acesso do programa aplicativo ao driver é disponibilizado através do módulo em Pascal LDASapi. 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 LDASapi.pas com as referências à biblioteca LynxDAS.DLL. A biblioteca DLL deverá ser copiada para o mesmo diretório do programa aplicativo. Se você estiver utilizando o Visual Basic, copie a biblioteca DLL no diretório do Windows.

É fornecido também o módulo LDASapi.bas em Visual Basic para o interfaceamento com o Visual Basic 5.

3.1. Estrutura de Dados dos Drivers

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 0-5 volts).

O driver possui um buffer circular de 64K amostras para a aquisição de sinais. Por exemplo, para a aquisição de 4 canais a 500 Hz por canal, o buffer teria capacidade de armazenar até 32 segundos sem que o programa aplicativo remova os dados do buffer circular.

3.2. Constantes

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

Constante	Valor	Descrição
ieNoError	0	Não ocorreu nenhum erro desde a última chamada da primitiva <i>LDAS_GetSamples</i>
ieDriver	100	Erro de acesso ao driver
ieAcqStop	101	Aquisição não está ativa
ieLabView	102	Dimensão errada no buffer do LabView passado na função <i>LDAS_GetSamples</i> . Número de colunas é diferente do número de sinais habilitados para aquisição.

4. DESCRIÇÃO DAS FUNÇÕES DA BIBLIOTECA LYNXDAS.DLL

Neste tópico são descritas as funções da biblioteca DLL. A tabela abaixo lista as funções da biblioteca LynxDAS.DLL.

Primitiva	Descrição
LDAS_OpenDriver	Abre o acesso ao driver
LDAS_CloseDriver	Finaliza o acesso ao driver
LDAS_OpenAccess	Inicia o acesso à placa A/D
LDAS_CloseAccessr	Finaliza o acesso à placa A/D
LDAS_GetVersion	Informa a versão e revisão do driver
LDAS_GetAiCaps	Informa os recursos de entrada analógica disponíveis no driver
LDAS_GetDioCaps	Informa os recursos de entrada e saída digital disponíveis no driver
LDAS_ReadAi	Leitura de canal de entrada analógica
LDAS_ReadDi	Leitura dos ports de entrada digital
LDAS_WriteDo	Escrita nos ports de saída digital
LDAS_ClearCM	Limpa a memória de canais da aquisição por interrupção
LDAS_InsertAiCM	Insere um canal de entrada analógica para aquisição
LDAS_InsertDiCM	Insere os ports de saída digital para aquisição
LDAS_AcquisitionSetup	Programa aquisição por interrupção
LDAS_StartAcquisition	Inicia a aquisição por interrupção
LDAS_StopAcquisition	Finaliza a aquisição por interrupção
LDAS_GetSamples	Leitura das amostras do buffer de aquisição por interrupção

4.1. Primitiva LDAS_OpenDriver

Object Pascal:

```
Function LDAS_OpenDriver (szDrvName: pchar): boolean;
```

Visual Basic:

```
Declare Function LDAS_OpenDriver Lib "LynxDAS.DLL" _  
    (ByVal szDrvName As String) As Byte
```

Esta primitiva abre o acesso ao driver e deve ser a primeira primitiva a ser executada pelo programa aplicativo. No parâmetro *szDrvName* deve ser passado o nome do driver a ser utilizado pela biblioteca DLL. Consulte na tabela do tópico 2.1. *Drivers Disponíveis* o nome do driver correspondente à sua placa. Por exemplo para acessar o driver da placa CAD12/32 no Windows 98, especifique no parâmetro *DrvName* a string WDM1232a.

O driver a ser acessado pela biblioteca DLL deve estar previamente instalado.

A primitiva retorna o valor 1 (true) quando foi executada com sucesso. Uma das causas possíveis de falha na execução desta primitiva são:

Causa	Verificação	Solução
Nome do driver está incorreto.	Consulte a tabela do tópico 2.1. <i>Drivers Disponíveis</i> .	Informe o nome correto do driver no parâmetro <i>DrvName</i> .
Driver não instalado	Se o driver a ser utilizado for do tipo NT4, no <i>Painel de Controle</i> do Windows NT, clique sobre <i>Dispositivos</i> . Se o driver estiver instalado, ele deve estar na lista de dispositivos apresentada. Se o driver a ser utilizado for do tipo WDM, no <i>Painel de Controle</i> do Windows, clique sobre o <i>Sistema</i> . Verifique se o driver aparece na lista do gerenciador de dispositivos.	Instale o driver.
Driver não iniciado ou erro na parametrização	No <i>Painel de Controle</i> do Windows NT, clique sobre <i>Dispositivos</i> . Verifique se o driver foi iniciado.	Proceda como descrito no help do driver.
Conflito de I/O ou interrupção com outros driver.	No <i>Painel de Controle</i> do Windows, clique sobre <i>Sistema</i> . Verifique se há conflito de recursos de hardware do driver.	Mude os parâmetros dos driver como descrito no help do driver.
Placa não instalada ou falha na iniciação da placa	Verifique se o endereço base da placa e o canal de interrupção configurado por jumpers na placa estão coerentes com o especificado na parametrização do driver. Para mudar a parametrização do driver consulte o help do driver.	Instale a placa e/ou altere a parametrização do driver.

A primitiva também irá falhar se o driver estiver em uso por outra aplicação.

Veja também a primitiva *LDAS_Close*.

4.2. Primitiva LDAS_CloseDriver

Object Pascal:

```
Procedure LDAS_CloseDriver;
```

Visual Basic:

```
Declare Sub LDAS_CloseDriver Lib "LynxDAS.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 *LDAS_OpenDriver* deve haver uma correspondente chamada da *LDAS_CloseDriver*.

Veja também a primitiva *LDAS_OpenDriver*.

4.3. Primitiva LDAS_OpenAccess

Object Pascal:

```
Function LDAS_OpenAccess: boolean;
```

Visual Basic:

```
Declare Function LDAS_OpenAccess Lib "LynxDAS.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 *zero* quando a primitiva foi executada com sucesso. Uma das causas possíveis são:

Veja também as primitivas *LDAS_CloseAccess* e *LDAS_OpenDriver*.

4.4. Primitiva LDAS_CloseAccess

Object Pascal:

```
Procedure LDAS_CloseAccess;
```

Visual Basic:

```
Declare Sub LDAS_CloseAccess Lib "LynxDAS.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 *LDAS_OpenAccess* deve haver uma correspondente chamada da *LDAS_CloseAccess*.

Veja também a primitiva *LDAS_OpenAccess*.

4.3. Primitiva LDAS_GetVersion

Object Pascal:

```
Function LDAS_GetVersion (Var Version, Release: byte): boolean;
```

Visual Basic:

```
Declare Function LDAS_GetVersion Lib "LynxDAS.DLL" _  
    (Version As Byte, Release As Byte) As Byte
```

Estando o driver instalado, esta primitiva retorna 1 (*true*) e devolve nos parâmetros *Version* e *Release*, respectivamente o byte mais significativo e o byte menos significativo da versão do device driver.

4.4. Primitiva LDAS_GetAiCaps

Object Pascal:

```
Function LDAS_GetAiCaps (Var nAiChannels, nAiRange: smallint;  
                        Var AiRange: TpAiRange): boolean;
```

Visual Basic:

```
Declare Function LDAS_GetAiCaps Lib "LynxDAS.DLL" _  
    (nAiChannels As Integer, nAiRange As Integer, _  
    AiRange As TpAiRange) As Byte
```

Estando o device driver instalado e operando, esta primitiva retorna (1) *true* e informa nos parâmetros de saída os recursos de entrada analógica disponibilizados pelo device driver.

No parâmetro *nAiChannels* é retornado o número de entradas analógicas da placa A/D.

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 6 posições são usadas) 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. Por exemplo, para a placa CAD12/32 são disponíveis as faixas de entrada do A/D apresentadas na tabela.

Índice	Valor do AiRange	Faixa de Entrada
0	-5.0	-5.0 a 5.0 volts
1	-2.5	-2.5 a 2.5 volts
2	-1.0	-1.0 a 1.0 volts
3	-0.5	-0.5 a 0.5 volts
4	5.0	0 a 5.0 volts
5	2.5	0 a 2.5 volts
6	1.0	0 a 1.0 volts
7	0.5	0 a 0.5 volts

Veja também a primitiva *LDAS_GetDioCaps*.

4.5. Primitiva LDAS_GetDioCaps

Object Pascal:

```
Function LDAS_GetDioCaps (Var nDiPorts, nBitsDi: smallint;  
                        Var nDoPorts, nBitsDo: smallint): boolean;
```

Visual Basic:

```
Declare Function LDAS_GetDioCaps Lib "LynxDAS.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 1 (*true*) e informa nos parâmetros de saída os recursos de entrada e saída digital disponibilizados pelo device driver.

O número de ports de entrada digital é retornado no parâmetro *nDiPorts* e o número de pontos por port de entrada digital é retornado em *nBitsDi*.

O número de ports de saída digital é retornado no parâmetro *nDoPorts* e o número de pontos por port de saída digital é retornado em *nBitsDo*.

Veja também a primitiva *LDAS_GetAiCaps*.

4.6. Primitiva LDAS_ReadAi

Object Pascal:

```
Function LDAS_ReadAi (Channel: smallint;  
                     Var Value: smallint): boolean;
```

Visual Basic:

```
Declare Function LDAS_ReadAi Lib "LynxDAS.DLL" _  
    (ByVal Channel As Integer, ByVal iRange As Integer, _  
     Value As Integer) As Byte
```

Esta primitiva realiza a leitura de um canal de entrada analógica da placa A/D. 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 passada no parâmetro *iRange*. Esse valor corresponde ao índice do vetor retornado no parâmetro *AiRange* da primitiva *LDAS_GetAiCaps*. Por exemplo se for passado o valor 0 nesse parâmetro, a faixa de entrada a ser utilizada na conversão A/D corresponderá a primeira faixa de entrada que, no caso da CAD12/32, será ± 5 V.

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 1 (true) 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 a primitiva *LDAS_GetAiCaps*.

4.7. Primitiva LDAS_ReadDi

Object Pascal:

```
Function LDAS_ReadDi (Group: smallint; Var Value: word): boolean;
```

Visual Basic:

```
Declare Function LDAS_ReadDi Lib "LynxDAS.DLL" _  
    (ByVal Group As Byte, Value As Integer) As Byte
```

Esta primitiva realiza a leitura de um port de entrada digital da placa. 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 através da primitiva *LDAS_GetDioCaps*. No caso da CAD12/32 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 *LDAS_WriteDo* e *LDAS_GetDioCaps*.

4.8. Primitiva LDAS_WriteDo

Object Pascal:

```
Function LDAS_WriteDo (Group: smallint; Value: word): boolean;
```

Visual Basic:

```
Declare Function LDAS_WriteDo Lib "LynxDAS.DLL" _
    (ByVal Group As Integer, ByVal Value As Integer) As Byte
```

Esta primitiva realiza a escrita em port de saída digital da placa. 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 através da primitiva *LDAS_GetDioCaps*.

O valor a ser escrito no port de saída digital deve ser passado no parâmetro *Value*. A primitiva retorna 1 (true) 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 *LDAS_ReadDi* e *LDAS_GetDioCaps*.

4.9. Primitiva LDAS_ClearCM

Object Pascal:

```
Procedure LDAS_ClearCM;
```

Visual Basic:

```
Declare Sub LDAS_ClearCM Lib "LynxDAS.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 aqisitados durante a aquisição por interrupção. Para cada canal analógico a ser aqisitado, o aplicativo deverá realizar uma chamada da primitiva *LDAS_InsertAiCM*. Analogamente, deve-se executar a primitiva *LDAS_InsertDiCM* para os ports de entrada digital. Antes, porém, o aplicativo deverá limpar a memória de canais através da chamada da primitiva *LDAs_ClearCM*.

Veja também as primitivas *LDAS_InsertAiCM* e *LDAS_InsertDiCM*.

4.11. Primitiva LDAS_InsertAiCM

Object Pascal:

```
Function LDAS_InsertAiCM (Channel, iRange: smallint): boolean;
```

Visual Basic:

```
Declare Function LDAS_InsertAiCM Lib "LynxDAS.DLL" _
    (ByVal Channel As Integer, ByVal iRange As Integer) As Byte
```

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 aqisitado, deve-se chamar a primitiva *LDAS_ClearCM* 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 a primitiva *LDAS_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 instalado.

Veja também as primitivas *LDAS_InsertDiCM* e *LDAS_ClearCM*.

4.12. Primitiva LDAS_InsertDiCM

Object Pascal:

```
Function LDAS_InsertAiCM (Channel, iRange: smallint): boolean;
```

Visual Basic:

```
Declare Function LDAS_InsertDiCM Lib "LynxDAS.DLL" _  
    (ByVal Group As Integer) As Byte
```

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. Os ports de entrada digital são lidos em grupo de 16 bits e no caso da placa CAD12/32, deve-se passar o valor 0 (zero) neste parâmetro para que o driver leia os ports P0 e P1. 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 *LDAS_InsertAiCM* e *LDAS_ClearCM*.

4.13. Primitiva LDAS_AcquisitionSetup

Object Pascal:

```
Function LDAS_AcquisitionSetup (Var SampleFreq: single): boolean;
```

Visual Basic:

```
Declare Function LDAS_AcquisitionSetup Lib "LynxDAS.DLL" _  
    (SampleFreq As Single) As Byte
```

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

A frequência em que os sinais serão amostrados deve ser especificada em hertz no parâmetro *SampleFreq*. devido à resolução do divisor de frequência da placa A/D , o valor desse parâmetro pode ser alterado na saída da primitiva para o valor efetivo da frequência de amostragem.

A primitiva retorna 1 (true) se foi executada com sucesso.

Veja também as primitivas *LDAS_ClearCM*, *LDAS_InsertAiCM*, *LDAS_InsertDiCM*, *LDAS_StartAcquisition*, *LDAS_GetSamples* e *LDAS_StopAcquisition*.

4.14. Primitiva LDAS _StartAcquisition

Object Pascal:

```
Function LDAS_StartAcquisition: boolean;
```

Visual Basic:

```
Declare Function LDAS_StartAcquisition Lib "LynxDAS.DLL" () As Byte
```

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 *LDAS_ClearCM*, *LDAS_InsertAiCM*, *LDAS_InsertDiCM* e *LDAS_AcquisitionSetup*.

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

Veja também as primitivas *LDAS_ClearCM*, *LDAS_InsertAiCM*, *LDAS_InsertDiCM*, *LDAS_AcquisitionSetup*, *LDAS_GetSamples* e *LDAS_StopAcquisition*.

4.15. Primitiva LDAS _StopAcquisition

Object Pascal:

```
Procedure LDAS_StopAcquisition;
```

Visual Basic:

```
Declare Sub LDAS_StopAcquisition Lib "LynxDAS.DLL" ()
```

Esta primitiva encerra a aquisição de sinais por interrupção.

Veja também a primitiva *LDAS_StartAcquisition*.

4.16. Primitiva LDAS _GetSamples

Object Pascal:

```
Function LDAS_GetSamples (Var Status: TpAcqStatus;  
                          Var Buffer; BufSize: integer): boolean;
```

Visual Basic:

Não disponível para o Visual Basic. Utilize a primitiva *LDAS_GetSamplesVB*.

```
Type {----- Tipo usado na LDAS_GetSamples -----}  
  TpAcqStatus = record  
    ieStatus: word;           // Status da aquisição  
    nSamples: int64;          // Número de amostragens aquisitadas por canal  
    iLast : int64;            // Número de amostragens lidas pela aplicação  
    nSampGot: integer;        // Número de amostragens lidas na primitiva  
  end;
```

Através desta função 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 16 canais a 140 Hz por canal, o buffer teria capacidade de armazenar até 29 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 periodicamente a chamada da função *LDAS_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 29 segundos sem retirar dados do buffer de aquisição. Depois deste tempo ocorre *overflow* no buffer.

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

O parâmetro de saída *Buffer* é um vetor onde serão copiadas as amostras adquiridas pelo driver. O tamanho de *Buffers* em bytes deve ser passado para a primitiva no parâmetro *BufSize*.

A função remove as amostras do buffer de aquisição do driver e as transfere para o *Buffer*. O número de amostras transferidas para o vetor é limitado em função do tamanho do buffer 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 *Status.SampGot*. No parâmetro *Status.iLast* é retornado o número de ordem da última amostra transferida para o *Buffer*. O número de ordem da amostra se refere ao início da aquisição.

A função retorna o valor 1 (true) se foi transferido dados para o array.

Veja também a função e *LDAS_GetSamplesVB*, *LDAS_StartAcquisition* e *LDAS_StopAcquisition*.

4.17. Primitiva LDAS_GetSamplesVB

Object Pascal:

```
Function LDAS_GetSamplesVB (Var ieStatus, nSamples,
                           nSampGot, iLast: integer;
                           Var UserBuf: TpUserBuf): boolean;
```

Visual Basic:

```
Declare Function LDAS_GetSamplesVB Lib "LynxDAS.DLL" _
    (ieStatus As Long, nSamples As Long, nSampGot As Long, _
    iLast As Long, UserBuf As TpUserBuf) As Byte
```

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

```
Type {----- Parâmetro de saída da LDAS_GetSamplesVB -----}
    TpUserBuf = array [0..16383] of smallint;
```

A função retorna o valor 1 (true) se foi transferido dados para o array.

Veja também a função e *LDAS_GetSamples*, *LDAS_StartAcquisition* e *LDAS_StopAcquisition*.