
Guia do Usuário

Automation Server LynxFile.FileTS

para Delphi, Matlab, LabVIEW e Python

1. INTRODUÇÃO

O *LynxFile.FileTS* é automation server implementado em Microsoft COM (Common Object Model) para leitura de arquivo de série temporal nos formatos LTX, LTD e TEM do AqDados.

O *LynxFile.FileTS* pode ser utilizado com linguagens de programação compatíveis com o Microsoft COM, tais como:

- Delphi Tokyo 10.2
- Turbo Delphi
- Matlab 2010
- LabVIEW 2013
- Python para Windows

1.1. Requisitos

Para a instalação e utilização do automation server são necessários:

- Computador com Windows 7 ou posterior
- 2 GB de memória, recomendamos 4 GB.
- Conhecimento prévio da plataforma de desenvolvimento a ser utilizada: Delphi, Matlab, LabVIEW ou Python.

1.2. Instalação e Arquivos com Exemplos de Uso

Há dois programas de instalação do automation server *LynxFile.FileTS*. Um para instalação no Windows 32 bits e outro para o Windows 64 bits.

O instalador para Windows 64 bits instala e registra o automation server nas versões de 32 e 64 bits.

Os arquivos com os fontes dos programas de exemplo de uso são instalados no diretório especificado na instalação do automation server *LynxFile*. Esses arquivos devem ser utilizados apenas como referência. Uma cópia dos arquivos também é instalada no seguinte subdiretório dos *Documentos Públicos*:

`LynxFile\Examples`

2. DESCRIÇÃO DE USO EM DELPHI

O automation server *LynxFile.FileTS* é fornecido com o programa exemplo *TestLynxTS* com o código fonte do projeto em *Delphi Tokyo 10.2* e *Turbo Delphi*.

2.1. Acesso ao Automation Server

O módulo *LynxFile_TLB.pas* do programa exemplo em Delphi contém a interface com o *LynxFile*.

Esse módulo deve ser adicionado no projeto em Delphi em que o automation server for ser utilizado.

2.1.1. Interface do LynxFile.FileTS

O programa deve declarar uma variável para a interface com o automation server.

```
oLynxTS: IFileTS;
```

A interface *IFileTS* está definida no módulo *LynxFile_TLB*.

2.1.2. Instância do Automation Server

Para acessar o automation *LynxFile.FileTS*, deve-se criar uma instância do automation server *LynxFile.FileTS*.

A instrução abaixo em Pascal cria a instância do automation server e atribui a interface do automation server na variável *oFileTS*.

```
oFileTS := CoFileTS.Create;
```

A função *CoFileTS.Create* está definida no módulo *LynxFile_TLB*.

2.2. Métodos da Interface IFileTS

2.2.1. Método OpenFile

```
Function OpenFile (Const FileName: WideString): WordBool;
```

Este método inicia o acesso para leitura de um arquivo de série temporal.

Parâmetro	Descrição
FileName	Nome do arquivo de série temporal a ser lido. Os formatos aceitos são: TEM, LTD e LTX.

O *OpenFile* retorna *true* se as operações iniciais para a leitura do arquivo ocorreram com sucesso. Caso contrário o método retornará *false*.

O erro pode ser consultado através das propriedades *ErrorCode* e *ErrorCodeStr*.

2.2.2. Método SetLineariz

```
Function SetLineariz (Const fnLineariz: WideString): WordBool;
```

Neste método é passado o nome do arquivo com os tipos e as tabelas de linearização de canais.

Parâmetro	Descrição
fnLineariz	Nome do arquivo com os tipos e as tabelas de linearização de canais. Usualmente o arquivo de linearização é o arquivo <i>Lineartiz.dat</i> fornecido com o <i>AqDados</i> . A aplicação cliente pode passar outro arquivo de linearização. No entanto, esse arquivo deve seguir a formatação definida para esse tipo de arquivo e apresentada no manual do <i>AqDados</i> .

O método retorna *true* se o arquivo de linearização foi carregado corretamente. O erro pode ser consultado através das propriedades *ErrorCode* e *ErrorCodeStr*.

A chamada deste método só é necessária se os arquivos de série temporal lidos através do automation server *LynxFile.FileTS* tiver canais que não sejam lineares.

Se o arquivo de série temporal possui canais com tipos não lineares e o arquivo de linearização não foi passado através do *SetLineariz* ou ocorreu erro no carregamento do arquivo de linearização, a linearização das amostras dos canais não lineares não é realizada. Por exemplo, se o arquivo de série temporal possui canais de termopar e o arquivo de linearização não foi informado, os valores dos canais de termopar serão lidos em mV,

2.2.3. Método SeekReadPos

```
Function SeekReadPos (iSample: int64): WordBool;
```

Este método é utilizado para iniciar a leitura sequencial das amostras dos sinais do arquivo de série temporal.

Parâmetro	Descrição
iSample	Índice da amostra a partir da qual serão lidos os sinais na leitura sequencial do arquivo de série temporal.

O método *SeekReadPos* retorna *false* se o índice da amostra passado neste método for inválido ou ocorreu algum erro na leitura do arquivo de série temporal. O erro pode ser consultado através das propriedades *ErrorCode* e *ErrorCodeStr*.

A leitura das amostras dos sinais é realizada através do método *ReadSample* que informa o valor da amostra em unidade de engenharia do sinal especificado no parâmetro da chamada desse método.

O posicionamento para o próximo instante de amostragem dos sinais é realizado através da chamada do método *PosNextSample*.

2.2.4. Método ReadSample

```
Function ReadSample (iSignal: smallint): double;
```

Este método é utilizado na leitura sequencial das amostras do arquivo de série temporal.

Parâmetro	Descrição
iSignal	Índice do sinal a ser lido. O valor passado em <i>iSignal</i> deve estar entre 0 e <i>nChannels</i> -1. A propriedade <i>nChannels</i> é o número de canais ativos no arquivo de série temporal.

O valor retornado pelo *ReadSample* é o valor da amostra em unidade de engenharia do sinal especificado correspondente à posição atual da leitura sequencial.

2.2.5. Método PosNextSample

```
Function PosNextSample (Step: integer): WordBool;
```

Este método incrementa o índice da leitura sequencial do arquivo de série temporal.

Parâmetro	Descrição
Step	Valor do incremento do índice da amostra na leitura sequencial. Usualmente o valor passado neste parâmetro é 1.

O método *PosNextSample* incrementa o índice do número da amostragem para a leitura sequencial das amostras.

O método retorna *false* se o índice da amostragem ultrapassou o número de amostras por canal do arquivo ou se ocorreu erro de leitura do arquivo.

O erro pode ser consultado através das propriedades *ErrorCode* e *ErrorCodeStr*.

A posição atual do índice da leitura sequencial pode ser consultada através da propriedade *iSample*.

2.2.6. Método ReadBuffer

```
Function ReadBuffer (iSignal: smallint; Pos: int64; N: integer;  
                    Var pBuffer: PSafeArray; Out NOut): WordBool;
```

Este método é utilizado para a leitura de um bloco de amostras de um sinal do arquivo de série temporal.

Parâmetro	Descrição
iSignal	Índice do sinal a ser lido. O valor passado em <i>iSignal</i> deve estar entre 0 e <i>nChannels</i> -1. A propriedade <i>nChannels</i> é o número de canais ativos no arquivo de série temporal.
Pos	Índice da amostra a partir da qual serão lidas as amostras do sinal no arquivo de série temporal.
N	Número de amostras a serem lidas.
pBuffer	Ponteiro para um safarray onde serão passadas as amostras em unidade de engenharia do sinal especificado. O safarray deve ter as seguintes características: <ul style="list-style-type: none">• Tipo de elemento: double.• Número de dimensões: 1.• Vetor de M elementos ou Matriz M x 1 ou 1 x M.<ul style="list-style-type: none">○ M é o número total de amostras que o vetor ou a matriz comporta.○ M deve ser maior ou igual ao valor do parâmetro <i>N</i>.
NOut	Número de amostras transferidas para o buffer. O valor retornado em <i>NOut</i> pode ser menor que <i>N</i> se atingiu o final do arquivo.

A função retorna *true* se a leitura foi realizada com sucesso.

Se a função retornar *false*, consulte o erro através das propriedades *ErrorCode* e *ErrorCodeStr*.

2.3. Enumerador `ecErrorCode`

O enumerador `ecErrorCode` é utilizado na propriedade `ErrorCode` para informar o código do último erro.

ErrorCode	Valor	Descrição
<code>ecNoError</code>	0	Nenhum erro.
<code>ecInvLinFile</code>	1	Erro no formato do arquivo de linearização
<code>ecInvFileType</code>	2	Tipo de arquivo não é suportado.
<code>ecFileNotFound</code>	3	Arquivo não foi encontrado
<code>ecFileOpenError</code>	4	Erro na abertura do arquivo
<code>ecFileSeekError</code>	5	Erro de posicionamento do arquivo
<code>ecFileReadError</code>	6	Erro na leitura do arquivo
<code>ecInvSignal</code>	7	Sinal inválido.
<code>ecInvSnType</code>	8	Tipo de linearização de um canal é inválido
<code>ecInvSampPos</code>	9	Índice do número da amostragem passado no método <code>SeekReadPos</code> ou <code>ReadBuffer</code> é inválido
<code>ecInvBuffer</code>	10	Safearray passado no método <code>ReadBuffer</code> é inválido
<code>ecEofFile</code>	11	Atingiu o final do arquivo
<code>ecNotOpened</code>	12	Arquivo não foi especificado
<code>ecInvStep</code>	13	PosNextSample: Step inválido
<code>ecInvChannel</code>	14	Canal inválido

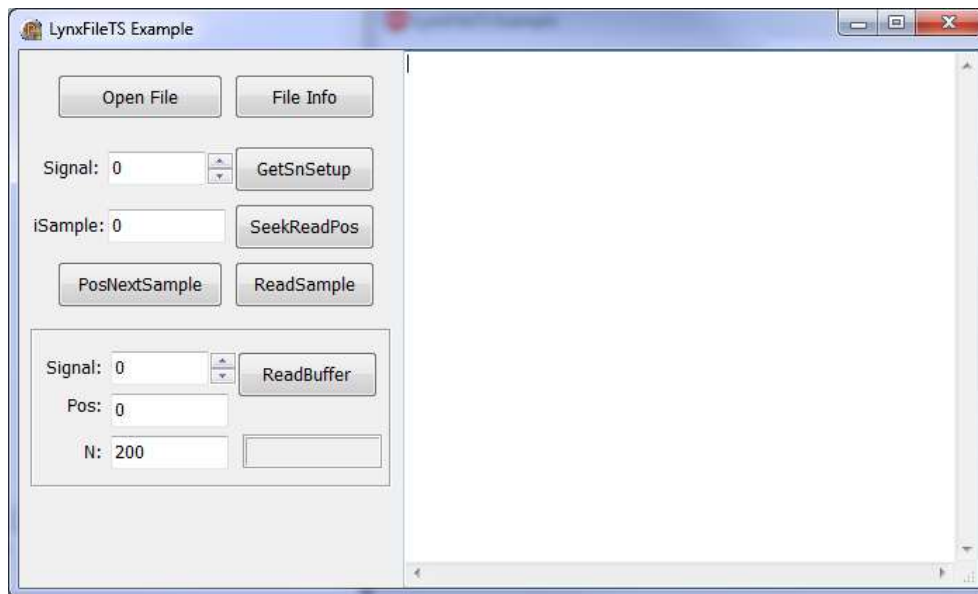
2.4. Propriedades da Interface IFileTS

Propriedade	Type	R/W	Index	Descrição
FileName	BSTR WideString	R		Nome do arquivo de série temporal
Comment	BSTR WideString	R		Comentário do arquivo
AcqDateTime	TDateTime	R		Data e horário da aquisição de sinais
SampleFreq	double	R		Frequência de amostragem
nSamples	__int64 Int64	R		Número de amostras por canal
nEvents	short Smallint	R		Número de eventos registrados no arquivo
EvenTime	double	R	iEvent	Instante em segundos correspondente ao evento <i>iEvent</i>
EventNote	BSTR WideString	R	iEvent	Nota correspondente ao evento <i>iEvent</i>
nChannels	short Smallint	R		Número de canais
SnName	BSTR WideString	R	iSignal	Nome do sinal <i>iSignal</i> . O índice <i>iSignal</i> é o número de ordem do sinal de entrada analógica ou contador. O primeiro canal A/D habilitado para aquisição tem o número de ordem 0, o segundo canal habilitada tem o número de ordem 1 e assim sucessivamente.
SnUnit	BSTR WideString	R	iSignal	Unidade de engenharia do sinal <i>iSignal</i>
SnDesc	BSTR WideString	R	iSignal	Comentário do sinal <i>iSignal</i>
SnHiLim	double	R	iSignal	Limite superior da escala do sinal <i>iSignal</i>
SnLoLim	double	R	iSignal	Limite inferior da escala do sinal <i>iSignal</i>
SnType	short Smallint	R	iSignal	Tipo de linearização do sinal <i>iSignal</i>
MapSnToCh	short Smallint	R	iSignal	Informa o canal A/D correspondente ao sinal <i>iSignal</i>
MapChToSn	short Smallint	R	Channel	Informa o índice do sinal correspondente ao canal <i>Channel</i> . Informa -1 se o canal não está ativo no arquivo.
iSample	__int64 Int64	R		Índice do número da amostragem na leitura sequencial.
ErrorCode	integer	R		Código de erro (veja os valores no tópico 2.3. <i>Enumerador ecErrorCode</i>).
ErrorCodeStr	BSTR WideString	R		String correspondente ao <i>ErrorCode</i>

2.5. Programa Exemplo em Delphi Tokyo 10.2

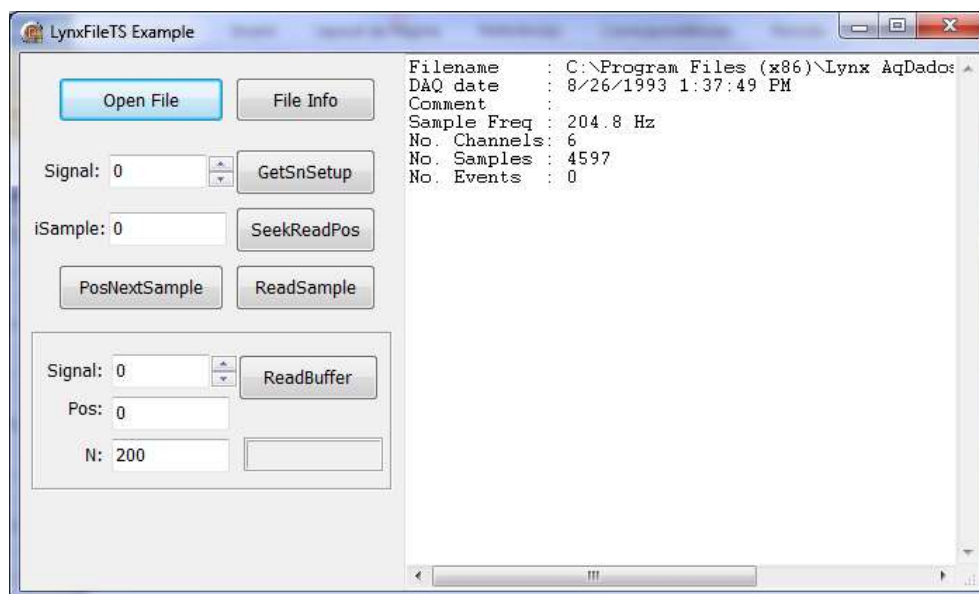
O automation server *LynxFile.FileTS* é fornecido com o programa exemplo *TestLynxFileTS* com código fonte em Delphi Tokyo 10.2 e Turbo Delphi.

A figura seguinte apresenta a janela do programa de teste.



A janela do programa *TestLynxFileTS* apresenta os seguintes botões:

- ❖ **Open File**
Permite selecionar e abrir um arquivo de série temporal.
- ❖ **File Info**
Apresenta as informações do arquivo de série temporal aberto.



- ❖ **GetSnSetup**
Apresenta as propriedades do sinal selecionado no controle *Signal*. São listadas as propriedades: *SnName*, *SnUnit*, *SnHiLim*, *SnLoLim* e o número do canal.
- ❖ **SeekReadPos**
Posiciona o índice do número da amostra na leitura sequencial. O índice da amostra é

especificado no controle *iSample*.

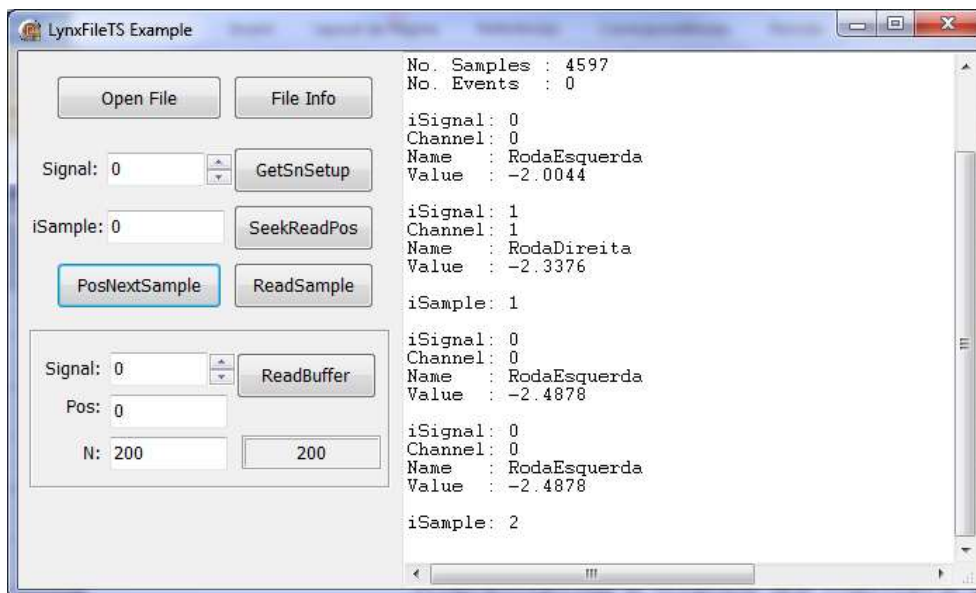
❖ **ReadSample**

Apresenta o valor da amostra do sinal especificado no controle *Signal* correspondente à posição atual do número da amostra na leitura sequencial.

❖ **PosNextSample**

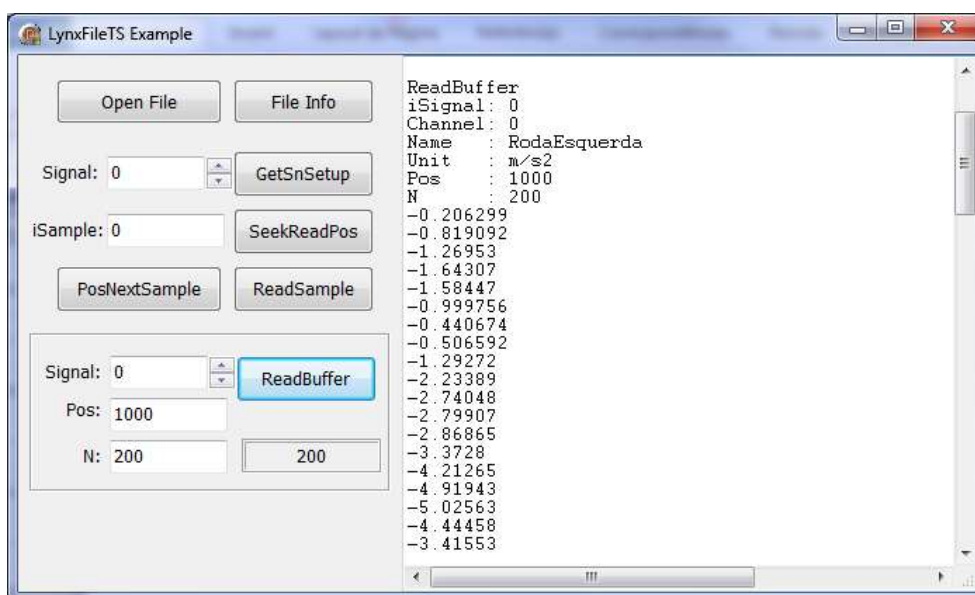
Incrementa o índice do número da amostra na leitura sequencial. A posição atual informada pela propriedade *iSample* é listada no *Memo1*.

A figura abaixo apresenta o *Memo1* após sucessivas execuções do *ReadSample* e *PosNextSample*.



❖ **ReadBuffer**

Efetua a leitura de um bloco de amostras do sinal selecionado no controle *Signal*. O índice da amostra no arquivo e o número de amostras a serem lidas são especificados nos controles *Pos* e *N*. As primeiras 100 amostras retornadas pela função são listadas no *Memo1*.



3. DESCRIÇÃO DE USO EM MATLAB

3.1. Create COM Server

A instância do automation server do *FileLynx.FileTS* é criada através da função *actxserver* do Matlab.

```
h = actxserver('LynxFile.FileTS');
```

Após criar a instância do *LynxFile.FileTS*, tem-se acesso às propriedades e aos métodos da interface *IFileTS* implementada pelo automation server.

A variável ***h*** é o handle da instância do automation server criado pela chamada da função *actxserver*.

No decorrer desta documentação será utilizada a variável ***h*** como a instância do automation server *LynxFile.FileTS*.

3.2. Close Reference

Para finalizar a instância do servidor COM *LynxFile* deve ser executada a função delete.

```
h.delete;
```

3.3. Acesso às Propriedades da Interface IFileTS

Para acessar uma propriedade da interface *IFileTS*, basta referenciar o nome da propriedade através do *handle* da instância do automation server *LynxFile.FileTS*.

A tabela seguinte lista as propriedades da interface *IFileTS*.

Propriedade	Type	R/W	Index	Descrição
FileName	String	R		Nome do arquivo de série temporal
Comment	String	R		Comentário do arquivo
AcqDateTime	DateTime	R		Data e horário da aquisição de sinais
SampleFreq	double	R		Frequência de amostragem
nSamples	int64	R		Número de amostras por canal
nEvents	int16	R		Número de eventos registrados no arquivo
EvenTime	double	R	iEvent	Instante em segundos correspondente ao evento <i>iEvent</i>
EventNote	string	R	iEvent	Nota correspondente ao evento <i>iEvent</i>
nChannels	int16	R		Número de canais
SnName	string	R	iSignal	Nome do sinal <i>iSignal</i> . O índice <i>iSignal</i> é o número de ordem do sinal de entrada analógica ou contador. O primeiro canal A/D habilitado para aquisição tem o número de ordem 0, o segundo canal habilitada tem o número de ordem 1 e assim sucessivamente.
SnUnit	string	R	iSignal	Unidade de engenharia do sinal <i>iSignal</i>
SnDesc	string	R	iSignal	Comentário do sinal <i>iSignal</i>
SnHiLim	double	R	iSignal	Limite superior da escala do sinal <i>iSignal</i>
SnLoLim	double	R	iSignal	Limite inferior da escala do sinal <i>iSignal</i>
SnType	int16	R	iSignal	Tipo de linearização do sinal <i>iSignal</i>
MapSnToCh	int16	R	iSignal	Informa o canal A/D correspondente ao sinal <i>iSignal</i>
MapChToSn	int16	R	Channel	Informa o índice do sinal correspondente ao canal <i>Channel</i> . Informa -1 se o canal não está ativo no arquivo.
iSample	int64	R		Índice do número da amostragem na leitura sequencial.
ErrorCode	int32	R		Código de erro (veja os valores no tópico 2.3. <i>Enumerador ecErrorCode</i>).
ErrorCodeStr	string	R		String correspondente ao <i>ErrorCode</i>

Por exemplo, para verificar a frequência de amostragem do arquivo de série temporal pode-se consultar a propriedades *SampleFreq*.

```
r = h.SampleFreq;
```

Para obter o nome do primeiro canal ativo do arquivo de série temporal pode-se utilizar o seguinte comando em Matlab:

```
r = h.SnName(0);
```

Para consultar o valor de todas as propriedades do servidor COM *LynxFile.FileTS* na linha de comando do Matlab, execute o seguinte comando:

```
h.get
```

3.4. Acesso aos Métodos da Interface IFileTS

Os métodos da interface com automation server *LynxFile.FileTS* são acessados através do handle da sua instância.

Por exemplo, na linha abaixo é chamado o método *ReadSample*.

```
r = h.ReadSample(0);
```

A tabela seguinte lista os métodos da interface *IFileTS*.

Função	Descrição
OpenFile	Informa o nome do arquivo de série temporal e inicia o acesso para leitura do arquivo
SetLineariz	Define o arquivo com os tipos e as tabelas de linearização de sinais.
SeekReadPos	Posiciona a leitura sequencial das amostras dos canais a partir de uma posição do arquivo de série temporal
ReadSample	Retorna o valor da amostra em unidade de engenharia de um sinal correspondente à posição atual da leitura sequencial do arquivo
PosNextSample	Incrementa o índice da posição de leitura sequencial das amostras
ReadBuffer	Leitura de um bloco de amostras de um sinal do arquivo

Para consultar a lista de métodos servidor COM *LynxFile.FileTS* na linha de comando do Matlab, execute o seguinte comando:

```
h.methods('-full')
```

3.4.1. Método OpenFile

Este método inicia o acesso para leitura de um arquivo de série temporal.

```
r = h.OpenFile(FileName);
```

Parâmetro	Descrição
FileName	Parâmetro de entrada do tipo string . Nome do arquivo de série temporal a ser lido. Os formatos aceitos são: TEM, LTD e LTX.
r	O <i>OpenFile</i> retorna <i>true</i> se as operações iniciais para a leitura do arquivo ocorreram com sucesso. Caso contrário o método retornará <i>false</i> . O erro pode ser consultado através das propriedades <i>ErrorCode</i> e <i>ErrorCodeStr</i> .

3.4.2. Método SetLineariz

Neste método é passado o nome arquivo com os tipos e as tabelas de linearização de canais.

```
r = h.SetLineariz(fnLineariz);
```

Parâmetro	Descrição
fnLineariz	Parâmetro de entrada do tipo string . Nome do arquivo com os tipos e as tabelas de linearização de canais. Usualmente o arquivo de linearização é o arquivo <code>Lineartiz.dat</code> fornecido com o <code>AqDados</code> . A aplicação cliente pode passar outro arquivo de linearização. No entanto, esse arquivo deve seguir a formatação definida para esse tipo de arquivo e apresentada no manual do <code>AqDados</code> .
r	O método retorna <i>true</i> se o arquivo de linearização foi carregado corretamente. O erro pode ser consultado através das propriedades <code>ErrorCode</code> e <code>ErrorCodeStr</code> .

A chamada deste método só é necessária se os arquivos de série temporal lidos através do automation server `LynxFile.FileTS` tiver canais que não sejam lineares.

Se o arquivo de série temporal possui canais com tipos não lineares e o arquivo de linearização não foi passado através do `SetLineariz` ou ocorreu erro no carregamento do arquivo de linearização, a linearização das amostras dos canais não lineares não é realizada. Por exemplo, se o arquivo de série temporal possui canais de termopar e o arquivo de linearização não foi informado, os valores dos canais de termopar serão lidos em mV,

3.4.3. Método SeekReadPos

Este método é utilizado para iniciar a leitura sequencial das amostras dos sinais do arquivo de série temporal.

```
r = h.SeekReadPos (iSample);
```

Parâmetro	Descrição
iSample	Parâmetro de entrada do tipo int64 . Índice da amostra a partir da qual serão lidos os sinais na leitura sequencial do arquivo de série temporal.
r	O método retorna <i>false</i> se o índice da amostra passado neste método for inválido ou ocorreu algum erro na leitura do arquivo de série temporal. O erro pode ser consultado através das propriedades <code>ErrorCode</code> e <code>ErrorCodeStr</code> .

O `SeekReadPos` posiciona o índice da leitura sequencial no número da amostra especificado no parâmetro `iSample`.

A leitura das amostras dos sinais é realizada através do método `ReadSample` que informa o valor da amostra em unidade de engenharia do sinal especificado no parâmetro da chamada desse método.

O posicionamento para o próximo instante de amostragem dos sinais é realizado através da chamada do método `PosNextSample`.

3.4.4. Método ReadSample

Este método é utilizado na leitura sequencial das amostras do arquivo de série temporal.

```
r = h.ReadSample(iSignal);
```

Parâmetro	Descrição
iSignal	Parâmetro de entrada do tipo <i>int16</i> . O valor passado em <i>iSignal</i> deve estar entre 0 e <i>nChannels</i> -1. A propriedade <i>nChannels</i> é o número de canais ativos no arquivo de série temporal.
r	O valor retornado pelo <i>ReadSample</i> é o valor da amostra em unidade de engenharia do sinal especificado correspondente à posição atual da leitura sequencial.

3.4.5. Método PosNextSample

Este método incrementa o índice da leitura sequencial do arquivo de série temporal.

```
r = h.PosNextSample (Step);
```

Parâmetro	Descrição
Step	Parâmetro de entrada do tipo <i>int32</i> . Valor do incremento do índice da amostra na leitura sequencial. Usualmente o valor passado neste parâmetro é 1.
r	O método retorna <i>false</i> se o índice da amostragem ultrapassou o número de amostras por canal do arquivo ou se ocorreu erro de leitura do arquivo. O erro pode ser consultado através das propriedades <i>ErrorCode</i> e <i>ErrorCodeStr</i> .

A posição atual do índice da leitura sequencial pode ser consultada através da propriedade *iSample*.

3.4.6. Método ReadBuffer

Este método é utilizado para a leitura de um bloco de amostras de um sinal do arquivo de série temporal.

```
[r, b, NOut = h.ReadBuffer(iSignal, Pos, N, b);
```

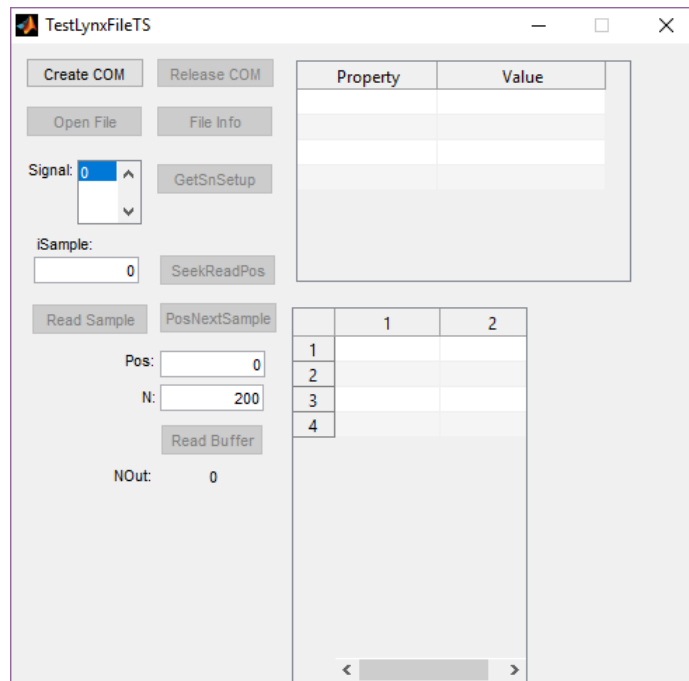
Parâmetro	Descrição
iSignal	Parâmetro de entrada do tipo int16 . Índice do sinal a ser lido. O valor passado em <i>iSignal</i> deve estar entre 0 e <i>nChannels</i> -1. A propriedade <i>nChannels</i> é o número de canais ativos no arquivo de série temporal.
Pos	Parâmetro de entrada do tipo int64 . Índice da amostra a partir da qual serão lidas as amostras do sinal no arquivo de série temporal.
N	Parâmetro de entrada do tipo int32 . Número de amostras a serem lidas.
b	Parâmetro de entrada e saída do tipo Safearray . No parâmetro <i>b</i> deve ser conectado um array do tipo double com tamanho maior ou igual ao número de amostras a serem lidas e informado no parâmetro <i>N</i> .
NOut	Parâmetro de saída do tipo int32 . Número de amostras transferidas para o buffer. O valor retornado em NOut pode ser menor que N se atingiu o final do arquivo.
r	Se a função retorna <i>true</i> se a leitura foi realizada com sucesso. O erro pode ser consultado através das propriedades <i>ErrorCode</i> e <i>ErrorCodeStr</i> .

3.5. Aplicação Exemplo em Matlab

O arquivo *TestLynxFileTS.m* ilustra um exemplo em Matlab para a leitura de arquivo de série temporal nos formatos LTX, LTD e TEM do *AqDados*. A interface gráfica desse exemplo está contida no arquivo *TestLynxFileTS.fig*.

3.5.1. Tela do Aplicativo Exemplo

A interface gráfica desse exemplo está contida no arquivo *TestLynxFileTS.fig* ilustrado na figura abaixo.



O programa exemplo possui os seguintes controles.

- Botão **Create COM**
No tratamento desse botão é criada uma instância do automation server *LynxFile.FileTS* através da função *actxserver* do Matlab.

```
handles.h = actxserver('LynxFile.FileTS');
```

- Botão **Open File**
Este botão abre uma caixa de diálogo para o usuário selecionar o arquivo de série temporal a ser lido. Segue abaixo trecho do código do tratamento deste botão.

```
[fn, fp] = uigetfile('*.LTX; *.LTD; *.TEM', 'Open Time Series File');  
handles.filename = strcat(fp, fn);  
% open time series file  
handles.fOpened = handles.h.OpenFile(handles.filename);  
if handles.fOpened  
    ShowFileInfo(handles);  
...  
...
```

- **Tabela de Propriedades**
Esta tabela apresenta algumas propriedades e a leitura de amostra do arquivo de série temporal. O conteúdo da tabela é atualizado pelo tratamento dos botões da aplicação.

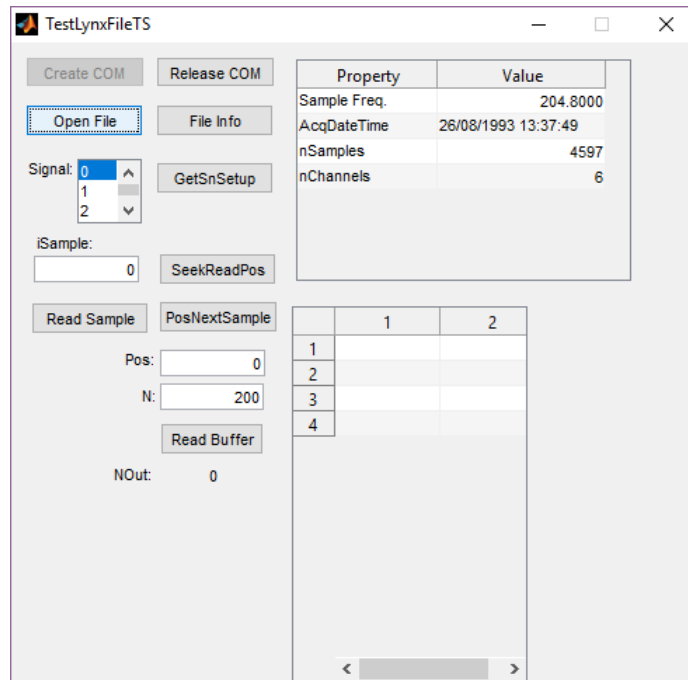
- Botão **File Info**

Este botão consulta algumas propriedades do arquivo e as apresenta na tabela de propriedades.

```

if handles.fOpened
    dat = {'Sample Freq.', handles.h.SampleFreq;
          'AcqDateTime', handles.h.AcqDateTime;
          'nSamples', handles.h.nSamples;
          'nChannels', handles.h.nChannels};
    set(handles.uiTable1, 'data', dat);
end

```



- Botão **GetSnSetup**

Este botão consulta as propriedades de um sinal do arquivo de série temporal. O índice do sinal a ser consultado é selecionado no listbox *iSignal*.

```

if handles.fOpened
    iSignal = get(handles.lbSignal, 'Value') - 1;
    dat = {'Signal', iSignal;
          'Channel', handles.h.MapSnToCh(iSignal);
          'Name', handles.h.SnName(iSignal);
          'Unit', handles.h.SnUnit(iSignal);
          'HiLim', handles.h.SnHiLim(iSignal);
          'LoLim', handles.h.SnHiLim(iSignal)};
    set(handles.uiTable1, 'data', dat);
end

```

- Botão **SeekReadPos**

Quando o método *IFileTS.OpenFile* é executado, o índice do número de amostra para leitura sequencial é posicionado na primeira amostra do arquivo de série temporal. Este botão executa o método *IFileTS.SeekReadPos* que permite posicionar o índice do número da amostra em qualquer posição do arquivo de série temporal. Neste exemplo o índice do número da amostra pode ser especificado pelo usuário no campo *iSample*.

```

if handles.fOpened
    iSample = str2num (get(handles.edISample, 'String'));
    handles.h.SeekReadPos(iSample);
end

```

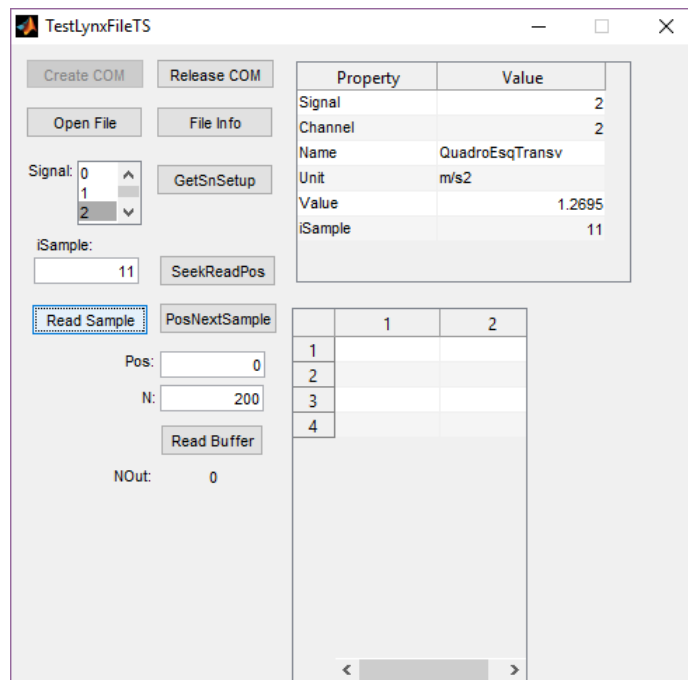
- Botão **ReadSample**

Este botão efetua a leitura de uma amostra de um sinal do arquivo de série temporal na posição atual da leitura sequencial. O índice do sinal a ser consultado é selecionado no listbox *iSignal*. O valor da amostra em unidade de engenharia é apresentado da tabela de propriedades juntamente com as propriedades do sinal e o índice da amostra.

```

if handles.fOpened
    iSignal = get(handles.lbSignal, 'Value') - 1;
    dat = {'Signal', iSignal;
          'Channel', handles.h.MapSnToCh(iSignal);
          'Name', handles.h.SnName(iSignal);
          'Unit', handles.h.SnUnit(iSignal);
          'Value', handles.h.ReadSample(iSignal);
          'iSample', handles.h.iSample};
    set(handles.uiTable1, 'data', dat);
end

```



- Botão **PosNextSample**

O incremento do índice do número da amostra é realizado através da chamada do método *IFileTS.PosNextSample*. No tratamento deste botão esse método é executado com o incremento de um intervalo de amostragem.

```

if handles.fOpened
    handles.h.PosNextSample(1);
    set(handles.edISample, 'String', handles.h.iSample);
end

```

- **Tabela de valores das amostras**

O controle *uiTable1* do tipo *Table* foi acrescentado na janela do exemplo para a apresentação das amostras lidas pelo método *ReadBuffer*.

- Botão **Read Buffer**

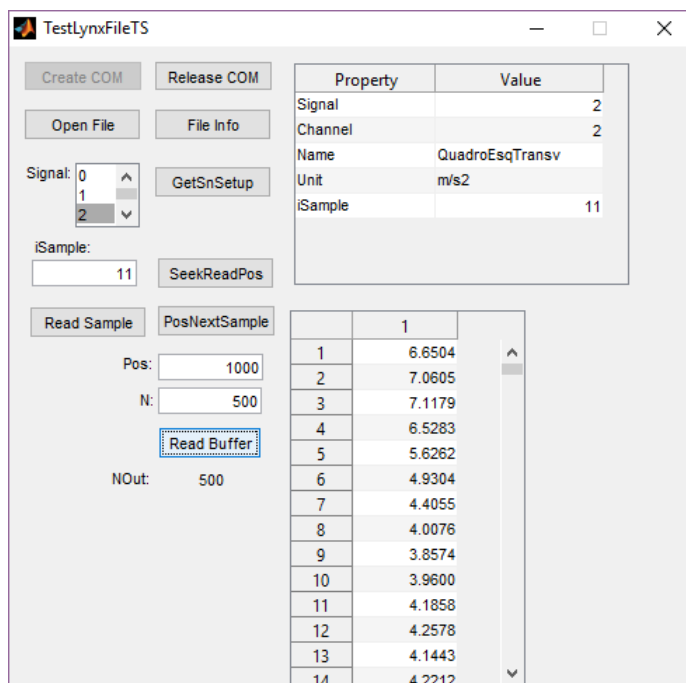
O método *IFileTS.ReadBuffer* permite a leitura de um bloco de amostras de um sinal do arquivo de série temporal. O número do sinal *iSignal*, o índice *Pos* da amostra no arquivo e o número de amostras *N* a serem lidos são passados como parâmetro nesse método. O buffer onde as amostras em unidade de engenharia serão retornadas deve ser uma matriz $1 \times M$ ou $M \times 1$ de tipo *double* com *M* maior que *N*. Neste exemplo foi criada uma matriz com 16384 elementos. O sinal a ser lido é selecionado no listbox *iSignal* e a posição no arquivo e o número de amostras são especificados nos campos *Pos* e *N*.

```

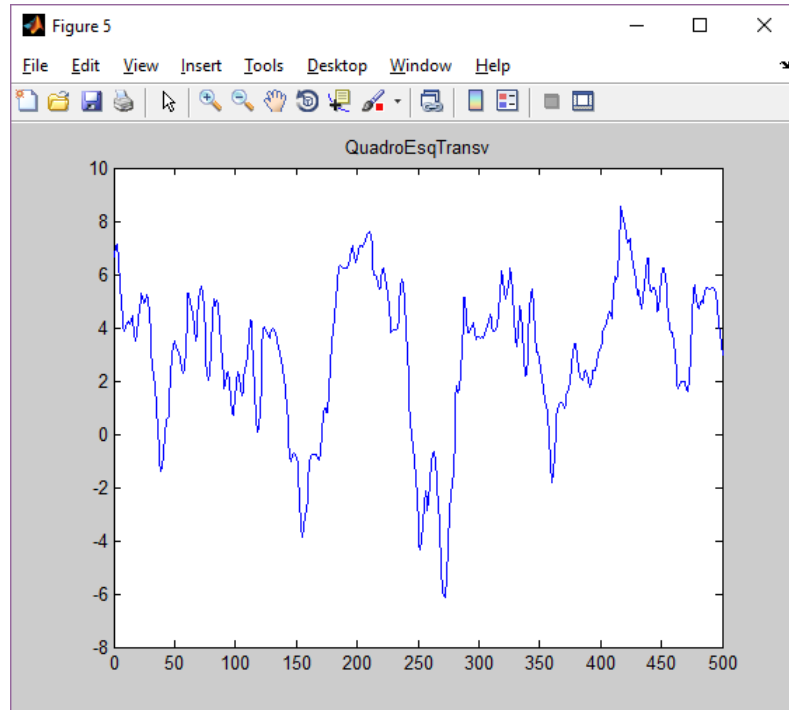
if handles.fOpened
    iSignal = get(handles.lbSignal, 'Value') - 1;
    Pos = str2num (get(handles.edPos, 'String'));
    N = str2num (get(handles.edN, 'String'));
    if N > 16384
        N = 16384;
    end
    [r, handles.Buf, handles.NOut] = handles.h.ReadBuffer (iSignal,
Pos, N, handles.Buf);
    if r
        figure (handles.hPlot);
        set(handles.uiSamples, 'Data', handles.Buf(1,
1:handles.NOut)');
        plot (handles.Buf(1, 1:handles.NOut)');
        title(handles.h.SnName(iSignal));
        set(handles.stNOut, 'String', handles.NOut);
        dat = {'Signal', iSignal;
'Channel', handles.h.MapSnToCh(iSignal);
'Name', handles.h.SnName(iSignal);
'Unit', handles.h.SnUnit(iSignal);
'iSample', handles.h.iSample);
        set (handles.uiTable1, 'data', dat);
    else
        msgbox(handles.h.ErrorCodeStr);
        set (handles.stNOut, 'String', 0);
    end
end
end

```

A figura abaixo ilustra um resultado da chamada do método *ReadBuffer*.



A figura abaixo ilustra um gráfico gerado com as amostras lidas do arquivo com o método *ReadBuffer*.



3.5.2. Arquivo TestLynxFileTS.m

```
function varargout = TestLynxFileTS(varargin)
% TESTLYNXFILETS MATLAB code for TestLynxFileTS.fig
%   TESTLYNXFILETS, by itself, creates a new TESTLYNXFILETS or raises the existing
%   singleton*.
%
%   H = TESTLYNXFILETS returns the handle to a new TESTLYNXFILETS or the handle to
%   the existing singleton*.
%
%   TESTLYNXFILETS('CALLBACK',hObject,eventData,handles,...) calls the local
%   function named CALLBACK in TESTLYNXFILETS.M with the given input arguments.
%
%   TESTLYNXFILETS('Property','Value',...) creates a new TESTLYNXFILETS or raises the
%   existing singleton*. Starting from the left, property value pairs are
%   applied to the GUI before TestLynxFileTS_OpeningFcn gets called. An
%   unrecognized property name or invalid value makes property application
%   stop. All inputs are passed to TestLynxFileTS_OpeningFcn via varargin.
%
%   *See GUI Options on GUIDE's Tools menu. Choose "GUI allows only one
%   instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help TestLynxFileTS

% Last Modified by GUIDE v2.5 14-Aug-2018 13:10:38

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                  'gui_Singleton',   gui_Singleton, ...
                  'gui_OpeningFcn', @TestLynxFileTS_OpeningFcn, ...
                  'gui_OutputFcn',  @TestLynxFileTS_OutputFcn, ...
                  'gui_LayoutFcn',  [], ...
                  'gui_Callback',    []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT

% --- Executes just before TestLynxFileTS is made visible.
function TestLynxFileTS_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to TestLynxFileTS (see VARARGIN)

% Choose default command line output for TestLynxFileTS
handles.output = hObject;

handles.fCreated = false;
handles.fOpened = false;
handles.NOut = 0;
handles.hPlot = figure;
set(handles.hPlot, 'Visible', 'off');

% create a buffer for ReadBuffer
handles.Buf = zeros(1, 16384, 'double');

% Update handles structure
guidata(hObject, handles);
```

```

% UIWAIT makes TestLynxFileTS wait for user response (see UIRESUME)
% uiwait(handles.figure1);

% --- Outputs from this function are returned to the command line.
function varargout = TestLynxFileTS_OutputFcn(hObject, eventdata, handles)
% varargout cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;

% --- Executes on button press in btCreateCOM.
function btCreateCOM_Callback(hObject, eventdata, handles)
% hObject    handle to btCreateCOM (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
handles.h = actxserver('LynxFile.FileTS');
handles.fCreated = true;
guidata(hObject, handles);
set(handles.btCreateCOM, 'Enable', 'Off');
set(handles.btReleaseCOM, 'Enable', 'On');
set(handles.btOpenFile, 'Enable', 'On');
guidata(hObject, handles);

% --- Executes on button press in btReleaseCOM.
function btReleaseCOM_Callback(hObject, eventdata, handles)
% hObject    handle to btReleaseCOM (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
if handles.fOpened
    handles.h.delete;
    handles.fCreated = false;
end
set(handles.btCreateCOM, 'Enable', 'On');
set(handles.btReleaseCOM, 'Enable', 'Off');
set(handles.btOpenFile, 'Enable', 'Off');
guidata(hObject, handles);

% --- Executes when user attempts to close figure1.
function figure1_CloseRequestFcn(hObject, eventdata, handles)
% hObject    handle to figure1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
if handles.fCreated
    handles.h.delete;
    handles.fCreated = false;
    handles.fOpened = false;
end
% Hint: delete(hObject) closes the figure
delete(hObject);

% --- Executes during object creation, after setting all properties.
function figure1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to figure1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% --- Show time series file informations
function ShowFileInfo(handles)
if handles.fOpened
    dat = {'Sample Freq.', handles.h.SampleFreq;
          'AcqDateTime', handles.h.AcqDateTime;
          'nSamples', handles.h.nSamples;

```

```

        'nChannels', handles.h.nChannels};
    set(handles.uiTable1, 'data', dat);
end

% --- Executes on button press in btOpenFile.
function btOpenFile_Callback(hObject, eventdata, handles)
% hObject    handle to btOpenFile (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
if handles.fCreated
    % select time series filename
    [fn, fp] = uigetfile('*.LTX; *.LTD; *.TEM', 'Open Time Series File');
    handles.filename = strcat(fp, fn);
    % open time series file
    handles.fOpened = handles.h.OpenFile(handles.filename);
    if handles.fOpened
        ShowFileInfo(handles);
        nc = handles.h.nChannels;
        if nc > 0
            cs = cell(nc, 1);
            for i=1:nc
                cs(i) = {sprintf('%d', i-1)};
            end
            set(handles.lbSignal, 'String', cs, 'Value', 1);
        end
        set(handles.btFileInfo, 'Enable', 'On');
        set(handles.btGetSnSetup, 'Enable', 'On');
        set(handles.btSeekReadPos, 'Enable', 'On');
        set(handles.btReadSample, 'Enable', 'On');
        set(handles.btPosNextSample, 'Enable', 'On');
        set(handles.btReadBuffer, 'Enable', 'On');
    else
        msgbox(handles.h.ErrorCodeStr);
        set(handles.uiTable1, 'data', []);
    end
end
guidata(hObject, handles);
end

% --- Executes on button press in btFileInfo.
function btFileInfo_Callback(hObject, eventdata, handles)
% hObject    handle to btFileInfo (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
ShowFileInfo(handles)

% --- Executes on selection change in lbSignal.
function lbSignal_Callback(hObject, eventdata, handles)
% hObject    handle to lbSignal (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = cellstr(get(hObject,'String')) returns lbSignal contents as cell
array
%         contents{get(hObject,'Value')} returns selected item from lbSignal

% --- Executes during object creation, after setting all properties.
function lbSignal_CreateFcn(hObject, eventdata, handles)
% hObject    handle to lbSignal (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: listbox controls usually have a white background on Windows.
%         See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
end

```

```

% --- Executes on button press in btGetSnSetup.
function btGetSnSetup_Callback(hObject, eventdata, handles)
% hObject    handle to btGetSnSetup (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
if handles.fOpened
    iSignal = get(handles.lbSignal,'Value') - 1;
    dat = {'Signal', iSignal;
          'Channel', handles.h.MapSnToCh(iSignal);
          'Name', handles.h.SnName(iSignal);
          'Unit', handles.h.SnUnit(iSignal);
          'HiLim', handles.h.SnHiLim(iSignal);
          'LoLim', handles.h.SnHiLim(iSignal)};
    set(handles.uiTable1, 'data', dat);
end

% --- Executes on button press in btPosNextSample.
function btPosNextSample_Callback(hObject, eventdata, handles)
% hObject    handle to btPosNextSample (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
if handles.fOpened
    handles.h.PosNextSample(1);
    set(handles.edISample, 'String', handles.h.iSample);
end

function edISample_Callback(hObject, eventdata, handles)
% hObject    handle to edISample (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edISample as text
%        str2double(get(hObject,'String')) returns contents of edISample as a double

% --- Executes during object creation, after setting all properties.
function edISample_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edISample (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on button press in btSeekReadPos.
function btSeekReadPos_Callback(hObject, eventdata, handles)
% hObject    handle to btSeekReadPos (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
if handles.fOpened
    iSample = str2num (get(handles.edISample,'String'));
    handles.h.SeekReadPos(iSample);
end

% --- Executes on button press in btReadSample.
function btReadSample_Callback(hObject, eventdata, handles)
% hObject    handle to btReadSample (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
if handles.fOpened
    iSignal = get(handles.lbSignal,'Value') - 1;
    dat = {'Signal', iSignal;

```



```

        'Channel', handles.h.MapSnToCh(iSignal);
        'Name', handles.h.SnName(iSignal);
        'Unit', handles.h.SnUnit(iSignal);
        'Value', handles.h.ReadSample(iSignal);
        'iSample', handles.h.iSample};
    set(handles.uiTable1, 'data', dat);
end

function edPos_Callback(hObject, eventdata, handles)
% hObject    handle to edPos (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edPos as text
%        str2double(get(hObject,'String')) returns contents of edPos as a double

% --- Executes during object creation, after setting all properties.
function edPos_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edPos (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%        See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on button press in btReadBuffer.
function btReadBuffer_Callback(hObject, eventdata, handles)
% hObject    handle to btReadBuffer (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
if handles.fOpened
    iSignal = get(handles.lbSignal,'Value') - 1;
    Pos = str2num (get(handles.edPos,'String'));
    N = str2num (get(handles.edN,'String'));
    if N > 16384
        N = 16384;
    end
    [r, handles.Buf, handles.NOut] = handles.h.ReadBuffer (iSignal, Pos, N, handles.Buf);
    if r
        figure (handles.hPlot);
        set(handles.uiSamples, 'Data', handles.Buf(1, 1:handles.NOut)');
        plot (handles.Buf(1, 1:handles.NOut)');
        title(handles.h.SnName(iSignal));
        set(handles.stNOut, 'String', handles.NOut);
        dat = {'Signal', iSignal;
            'Channel', handles.h.MapSnToCh(iSignal);
            'Name', handles.h.SnName(iSignal);
            'Unit', handles.h.SnUnit(iSignal);
            'iSample', handles.h.iSample};
        set (handles.uiTable1, 'data', dat);
    else
        msgbox(handles.h.ErrorCodeStr);
        set (handles.stNOut, 'String', 0);
    end
end
end

function edN_Callback(hObject, eventdata, handles)
% hObject    handle to edN (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edN as text

```

```

%         str2double(get(hObject,'String')) returns contents of edN as a double

% --- Executes during object creation, after setting all properties.
function edN_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edN (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%         See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edNOut_Callback(hObject, eventdata, handles)
% hObject    handle to edNOut (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edNOut as text
%         str2double(get(hObject,'String')) returns contents of edNOut as a double

% --- Executes during object creation, after setting all properties.
function edNOut_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edNOut (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

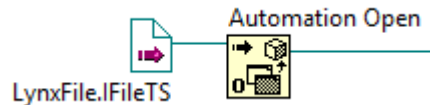
% Hint: edit controls usually have a white background on Windows.
%         See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

```

4. DESCRIÇÃO DE USO EM LABVIEW

4.1. Automation Open

A instância do *LynxFile.IFileTS* é criada através do bloco **Automation Open** do LabVIEW.



Para inserir o bloco *Automation Open*, siga os seguintes passos:

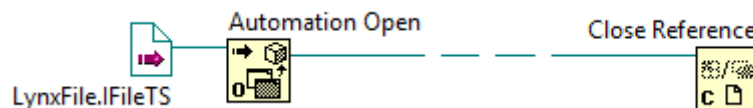
1. Se a janela *Functions* do LabVIEW não estiver visível, selecione *View / Functions*.
2. Na janela *Functions*, abra a paleta *Connectivity* e clique em *ActiveX*.
3. Na paleta *Connectivity / ActiveX*, selecione **Automation Open** e arraste para o seu diagrama de bloco.
4. Crie uma constante para o terminal *automation refnun* do bloco.
5. Clique com o botão direito do mouse sobre o bloco da constante criada no passo anterior e selecione o menu *Select ActiveX Class* no popup menu apresentado. Selecione a classe *LynxFile.IFileTS*.
6. Se a classe *LynxFile.IFileTS* não estiver na lista do passo anterior, selecione *Browse*.
7. Na janela *Select Object From Type Library* aberta, selecione *FileLynx* no combo box *Type Library*, selecione o objeto *FileFile.FileTS* e pressione o botão *OK*.

Após criar a instância do *LynxFile.IFileTS*, tem-se acesso às propriedades e métodos da interface *IFileTS* implementada pelo *automation LynxFile.IFileTS*.

A saída do bloco *Automation Open* é um *handle* que deve ser conectado na entrada *reference* dos blocos de chamadas de métodos e propriedades da interface *IFileTS*.

4.2. Close Reference

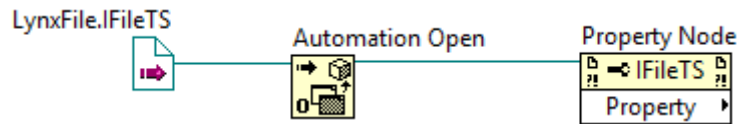
Para cada bloco *Automation Open* incluído no programa em *LabVIEW* deve ser incluída uma chamada do bloco **Close Reference** para finalizar a instância do automation quando ele não for mais necessário.



O bloco *Close Reference* se encontra na paleta de funções *Connectivity / ActiveX*. A entrada *reference* desse bloco deve ser conectada no terminal *reference out* de um bloco *Invoke Node* ou *Property Node* associado ao automation *LynxFile.IFileTS*.

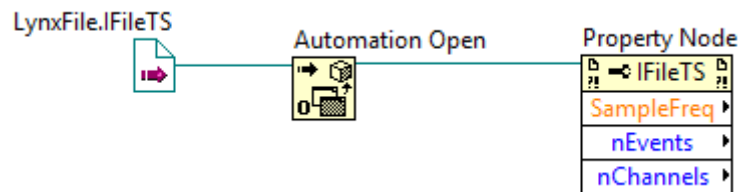
4.3. Acesso às Propriedades da Interface IFileTS

As propriedades da interface *IFileTS* são acessadas através do bloco **Property Node** disponível na paleta de funções *Connectivity / ActiveX* da janela *Functions* do LabVIEW.



Para acessar uma propriedade da interface *IFileTS*:

1. Selecione o bloco de função **Property Node** na paleta *Connectivity / ActiveX*, da janela *Functions* e arraste para o seu diagrama de bloco.
2. Conecte o terminal de entrada **reference** no terminal de saída *Automation Refnum* do bloco *Automation Open* ou no terminal *reference out* de um bloco *Property Node* ou *Invoke Node* que esteja referenciado à interface *IFileTS*. Veja a figura anterior.
3. Clique no campo **Property** do bloco *Property Node* inserido e selecione a propriedade a ser acessada.
4. Para incluir uma nova propriedade no mesmo bloco, clique com o botão direito do mouse sobre o campo de propriedade do bloco e selecione o comando **Add Element** no popup menu apresentado. Outra maneira é dimensionar o tamanho do bloco pela parte inferior com o uso do mouse.



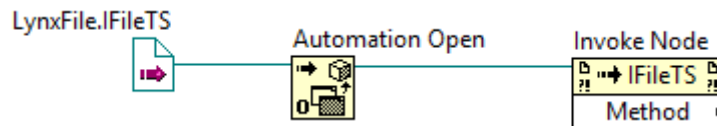
A tabela seguinte lista as propriedades da interface *IFileTS* que podem ser acessadas através do bloco *Property Node*.

Propriedade	Type	R/W	Descrição
FileName	String	R	Nome do arquivo de série temporal
Comment	Boolean	R	Comentário do arquivo
AcqDateTime	DateTime	R	Data e horário da aquisição de sinais
SampleFreq	String	R	Frequência de amostragem
nSamples	I64	R	Número de amostras por canal
nEvents	I16	R	Número de eventos registrados no arquivo
nChannels	I16	R	Número de canais
iSample	I64	R	Índice do número da amostragem na leitura sequencial.
ErrorCode	I32	R	Código de erro (veja os valores no tópico 2.3. <i>Enumerador ecErrorCode</i>)
ErrorCodeStr	String	R	String correspondente ao <i>ErrorCode</i>

As propriedades com índice da interface *IFileTS* são tratadas pelo LabVIEW como métodos e não estão listadas na tabela anterior. Elas estão listadas na tabela de métodos.

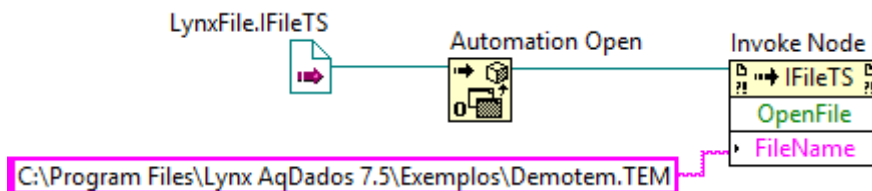
4.4. Acesso aos Métodos da Interface IFileTS

Os métodos da interface *IFileTS* são acessados através do bloco **Invoke Node** disponível na paleta de funções *Connectivity / ActiveX* da janela *Functions* do LabVIEW.



Para acessar um método da interface *IFileTS*:

1. Selecione o bloco de função **Invoke Node** na paleta *Connectivity / ActiveX*, da janela *Functions* e arraste para o seu diagrama de bloco.
2. Conecte o terminal de entrada **reference** no terminal de saída *Automation Refnum* do bloco *Automation Open* ou no terminal *reference out* de um bloco *Property Node* ou *Invoke Node* que esteja referenciado à interface *IFileTS*. Veja a figura anterior.
3. Clique no campo **Method** do bloco **Invoke Node** inserido e selecione o método a ser chamado.
4. Após selecionar um método da interface *IFileTS*, o bloco é atualizado e passa a apresentar terminais de entrada e saída que correspondem aos parâmetros de entrada e saída do método. Se o método for uma função, um terminal de saída é apresentado no campo com o nome do método. A figura abaixo ilustra um exemplo com a chamada do método *OpenFile* da interface *IFileTS*.



A tabela seguinte lista os métodos da interface *IFileTS* que podem ser acessadas através do bloco *Invoke Node*.

Função	Descrição
OpenFile	Informa o nome do arquivo de série temporal e inicia o acesso para leitura do arquivo
SetLineariz	Define o arquivo com os tipos e as tabelas de linearização de sinais.
SeekReadPos	Posiciona a leitura sequencial das amostras dos canais a partir de uma posição do arquivo de série temporal
ReadSample	Retorna o valor da amostra em unidade de engenharia de um sinal correspondente à posição atual da leitura sequencial do arquivo
PosNextSample	Incrementa o índice da posição de leitura sequencial das amostras
ReadBuffer	Leitura de um bloco de amostras de um sinal do arquivo
EvenTime ^{1,2}	Instante em segundos correspondente ao evento <i>iEvent</i> . O parâmetro <i>iEvent</i> , correspondente ao número do evento, deve ser especificado entre 1 e o valor da propriedade <i>nEvents</i> .
EventNote ^{1,2}	Nota correspondente ao evento <i>iEvent</i> O parâmetro <i>iEvent</i> , correspondente ao número do evento, deve ser especificado entre 1 e o valor da propriedade <i>nEvents</i> .
SnName ^{1,3}	Nome do sinal <i>iSignal</i> . O índice <i>iSignal</i> é o número de ordem do sinal de entrada analógica ou contador. O primeiro canal A/D habilitado para aquisição tem o número de ordem 0, o segundo canal habilitada tem o número de ordem 1 e assim sucessivamente.

Função	Descrição
SnUnit ^{1,3}	Unidade de engenharia do sinal <i>iSignal</i>
SnDesc ^{1,3}	Comentário do sinal <i>iSignal</i>
SnHiLim ^{1,3}	Limite superior da escala do sinal <i>iSignal</i>
SnLoLim ^{1,3}	Limite inferior da escala do sinal <i>iSignal</i>
SnType ^{1,3}	Tipo de linearização do sinal <i>iSignal</i>
MapSnToCh ^{1,4}	Informa o canal A/D correspondente ao sinal <i>iSignal</i>
MapChToSn ^{1,5}	Informa o índice do sinal correspondente ao canal <i>Channel</i> . Informa -1 se o canal não está ativo no arquivo.

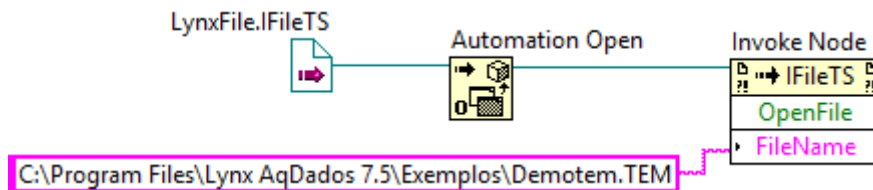
Notas:

1. Esse método é uma propriedade indexada que o LabVIEW trata como método.
2. O parâmetro de entrada *iEvent* (índice da propriedade) desse método corresponde ao número do evento. O valor de *iEvent* deve estar entre 1 e o valor da propriedade *nEvents*.
3. O parâmetro de entrada *iSignal* (índice da propriedade) desse método é o número de ordem do sinal de entrada analógica ou contador. O primeiro canal A/D habilitado para aquisição tem o número de ordem 0, o segundo canal habilitada tem o número de ordem 1 e assim sucessivamente.
4. O parâmetro de entrada *iSignal* (índice da propriedade) desse método é o número de ordem do sinal de entrada analógica ou contador. O primeiro canal A/D habilitado para aquisição tem o número de ordem 0, o segundo canal habilitada tem o número de ordem 1 e assim sucessivamente.
5. O parâmetro de entrada *Channel* (índice da propriedade) desse método é o número do canal.

4.5. Método OpenFile

Este método inicia o acesso para leitura de um arquivo de série temporal.

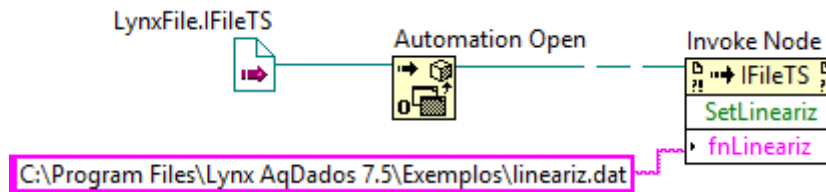
A figura abaixo ilustra um exemplo da chamada do método *IFileTS.OpenFile* disponível através do bloco *Invoke Node* da paleta *Connectivity / ActiveX* da janela *Functions* do LabVIEW.



Terminal	Descrição
FileName	Terminal de entrada do tipo String. Especifique neste parâmetro o nome do arquivo de série temporal a ser lido. Os formatos aceitos são: .TEM, .LTD e .LTX.
Valor retornado	Valor retornado do tipo boolean O <i>OpenFile</i> retorna <i>true</i> se as operações iniciais para a leitura do arquivo ocorreram com sucesso. Caso contrário o método retornará <i>false</i> . O erro pode ser consultado através das propriedades <i>ErrorCode</i> e <i>ErrorCodeStr</i> .

4.6. Método SetLineariz

Este método especifica o nome do arquivo com os tipos e as tabelas de linearização de sinais, usualmente utilizados para linearização de termopares.



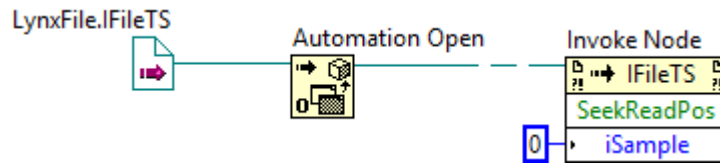
Terminal	Descrição
fnLineariz	Terminal de entrada do tipo String. Nome do arquivo com os tipos e as tabelas de linearização de canais. Usualmente o arquivo de linearização é o arquivo <code>Lineariz.dat</code> fornecido com o <code>AqDados</code> . A aplicação cliente pode passar outro arquivo de linearização. No entanto, esse arquivo deve seguir a formatação definida para esse tipo de arquivo e apresentada no manual do <code>AqDados</code> .
Valor retornado	Valor retornado do tipo boolean O método retorna <code>true</code> se o arquivo de linearização foi carregado corretamente O erro pode ser consultado através das propriedades <code>ErrorCode</code> e <code>ErrorCodeStr</code> .

A chamada deste método só é necessária se os arquivos de série temporal lidos através do automation server `LynxFile.FileTS` tiver canais que não sejam lineares.

Se o arquivo de série temporal possui canais com tipos não lineares e o arquivo de linearização não foi passado através do `SetLineariz` ou ocorreu erro no carregamento do arquivo de linearização, a linearização das amostras dos canais não lineares não é realizada. Por exemplo, se o arquivo de série temporal possui canais de termopar e o arquivo de linearização não foi informado, os valores dos canais de termopar serão lidos em mV,

4.7. Método SeekReadPos

Este método é utilizado para iniciar a leitura sequencial das amostras dos sinais do arquivo de série temporal.



O *SeekReadPos* posiciona o índice da leitura sequencial no número da amostra especificado no parâmetro *iSample*.

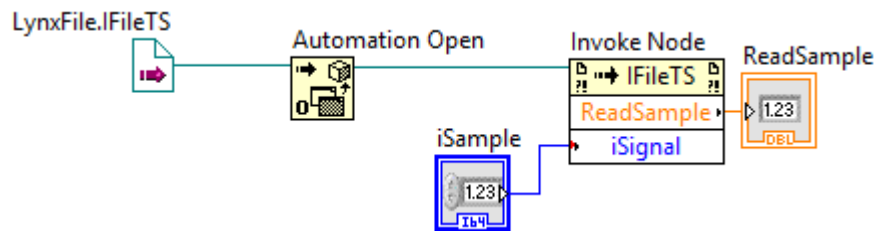
Terminal	Descrição
<i>iSample</i>	Terminal de entrada do tipo l64. Índice da amostra a partir da qual serão lidos os sinais na leitura sequencial do arquivo de série temporal.
Valor retornado	Valor retornado do tipo boolean O método retorna <i>false</i> se o índice da amostra passado neste método for inválido ou ocorreu algum erro na leitura do arquivo de série temporal. O erro pode ser consultado através das propriedades <i>ErrorCode</i> e <i>ErrorCodeStr</i> .

A leitura das amostras dos sinais é realizada através do método *ReadSample* que informa o valor da amostra em unidade de engenharia do sinal especificado no parâmetro da chamada desse método.

O posicionamento para o próximo instante de amostragem dos sinais é realizado através da chamada do método *PosNextSample*.

4.8. Método ReadSample

Este método é utilizado na leitura sequencial das amostras do arquivo de série temporal.

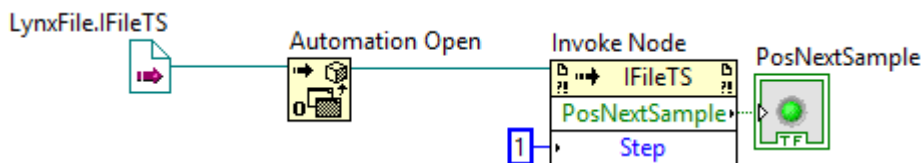


O *SeekReadPos* posiciona o índice da leitura sequencial no número da amostra especificado no parâmetro *iSample*.

Terminal	Descrição
iSignal	Terminal de entrada do tipo I16. Índice do sinal a ser lido. O valor passado em <i>iSignal</i> deve estar entre 0 e <i>nChannels</i> -1. A propriedade <i>nChannels</i> é o número de canais ativos no arquivo de série temporal.
Valor retornado	Valor retornado do tipo DBL O valor retornado pelo <i>ReadSample</i> é o valor da amostra em unidade de engenharia do sinal especificado correspondente à posição atual da leitura sequencial.

4.9. Método PosNextSample

Este método incrementa o índice da leitura sequencial do arquivo de série temporal.

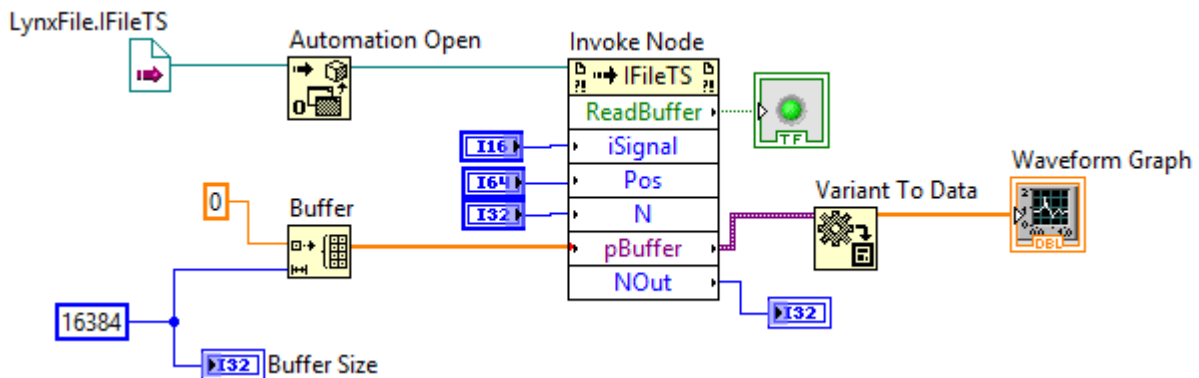


Terminal	Descrição
Step	Terminal de entrada do tipo I16. Valor do incremento do índice da amostra na leitura sequencial. Usualmente o valor passado neste parâmetro é 1.
Valor retornado	Valor retornado do tipo boolean O método retorna <i>false</i> se o índice da amostragem ultrapassou o número de amostras por canal do arquivo ou se ocorreu erro de leitura do arquivo. O erro pode ser consultado através das propriedades <i>ErrorCode</i> e <i>ErrorCodeStr</i> .

A posição atual do índice da leitura sequencial pode ser consultada através da propriedade *iSample*.

4.10. Método ReadBuffer

Este método é utilizado para a leitura de um bloco de amostras de um sinal do arquivo de série temporal.



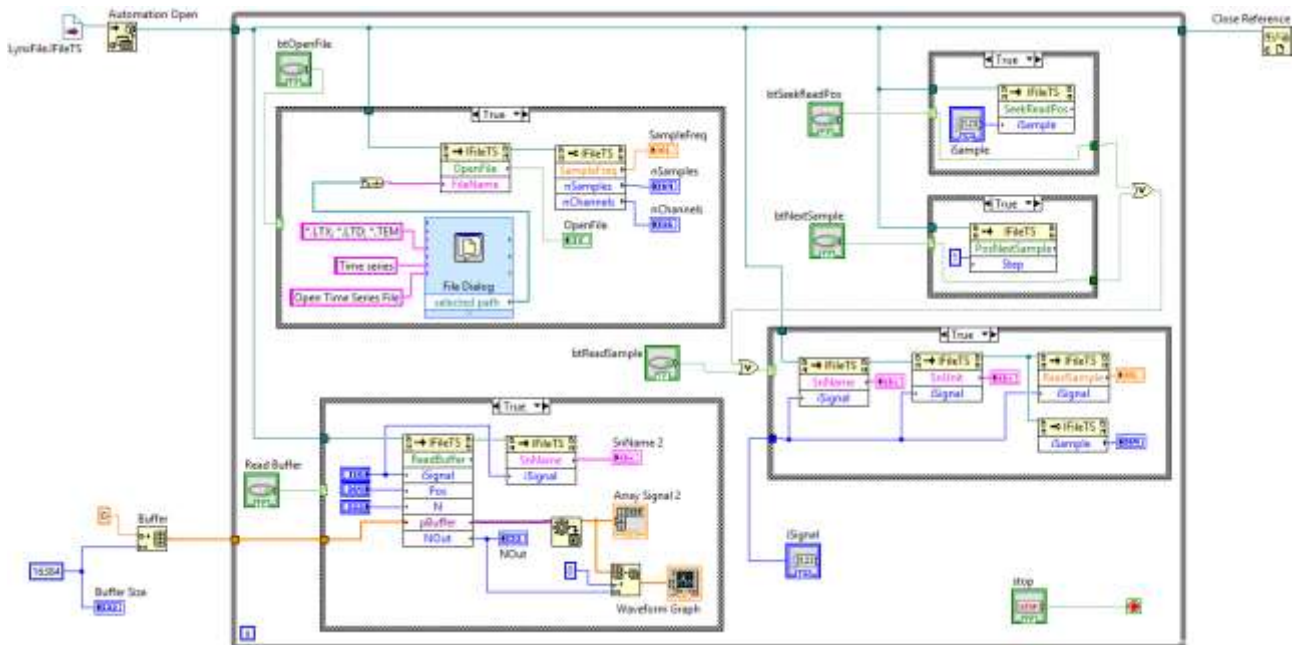
Terminal	Descrição
iSignal	Terminal de entrada do tipo I16. Índice do sinal a ser lido. O valor passado em <i>iSignal</i> deve estar entre 0 e <i>nChannels</i> -1. A propriedade <i>nChannels</i> é o número de canais ativos no arquivo de série temporal.
Pos	Terminal de entrada do tipo I64. Índice da amostra a partir da qual serão lidas as amostras do sinal no arquivo de série temporal.
pBuffer	Terminal de entrada do tipo I32. Número de amostras a serem lidas do sinal especificado.
N	Terminal de entrada do tipo I32. Número de amostras a serem lidas.
pBuffer	Terminal de entrada e saída do tipo Safearray. No terminal de entrada deve ser conectado um array do tipo double com tamanho maior ou igual ao número de amostras a serem lido e informado no parâmetro <i>N</i> . Antes da chamada efetiva do método <i>ReadBuffer</i> , o LabVIEW converte o array para <i>SafeArray</i> . O método <i>ReadBuffer</i> devolve no <i>SafeArray</i> as amostras lidas em unidade de engenharia. Após a execução do método <i>ReadBuffer</i> o LabVIEW converte o <i>SafeArray</i> para um tipo <i>Variant</i> . Sendo assim, deve-se conectar no terminal de saída do bloco Variant to Data para converter o vetor para um array do tipo double do LabVIEW.
NOut	Terminal de saída do tipo I32. Número de amostras que foram efetivamente lidas e retornadas no safearray.
Valor retornado	Valor retornado do tipo boolean O método retorna <i>true</i> se a leitura foi realizada com sucesso. O erro pode ser consultado através das propriedades <i>ErrorCode</i> e <i>ErrorCodeStr</i> .

4.11. Aplicação Exemplo em LabVIEW

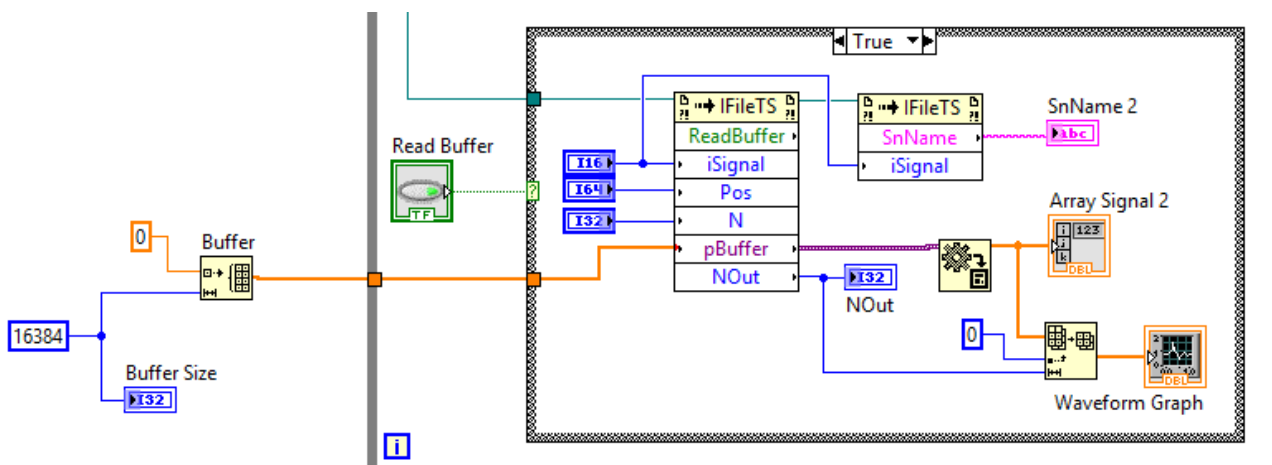
Este programa exemplifica o uso dos principais métodos e propriedades do automation server *LynxFile.FileTS* para a consulta de arquivo de série temporal nos formatos LTX, LTD e TEM do *AqDados*.

4.11.1. Diagrama de Blocos do Exemplo

A figura acima ilustra o diagrama de blocos do programa exemplo *Example_LynxFileTS*.

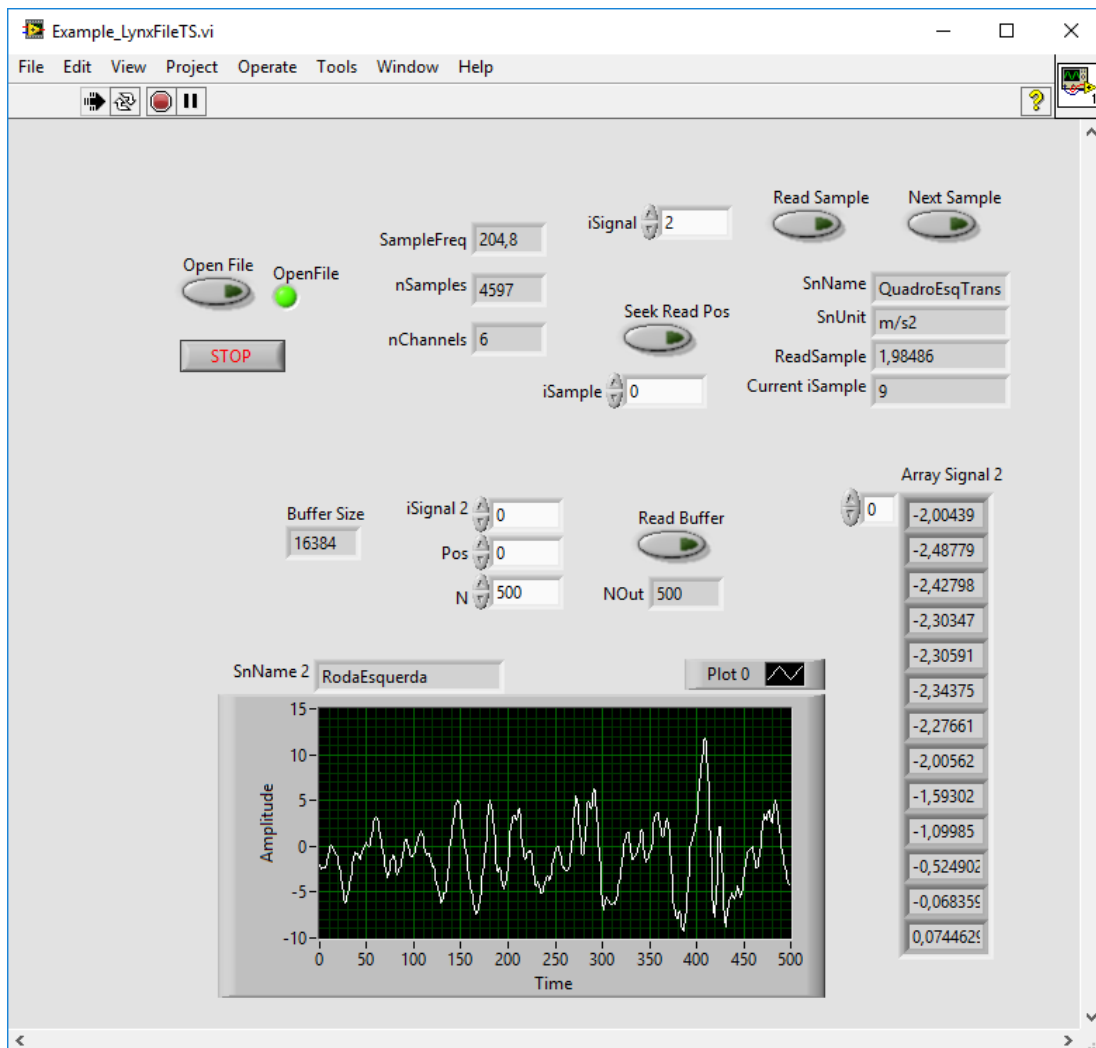


O método *ReadBuffer* do automation server *LynxFile.FileTS* efetua a leitura de um bloco de amostras do arquivo de série temporal. O parâmetro de entrada e saída *pBuffer* é do tipo *safearray* utilizado em objeto *Activex*. No caso do método *ReadBuffer*, o *safearray* deve ser um vetor do tipo *double*. No lado esquerdo do diagrama apresentado abaixo é criado um array *double* de dimensão 16384. Esse array é passado para o *ReadBuffer* permitindo a leitura de um bloco de amostras com tamanho até a dimensão especificada. O bloco *Activex Invoke Node* do LabVIEW não efetua a conversão automática do *safearray* para o formato *double*. O LabVIEW converte o *safearray* para um array do tipo *variant*. Conforme citado na descrição do método *ReadBuffer*, a aplicação em LabVIEW deve utilizar o bloco *Variant to Data* para efetuar a conversão para um array do tipo *double*, conforme exemplificado no lado direito da figura.



4.11.2. Painel do Exemplo

A figura abaixo apresenta a janela do painel do exemplo durante a execução.



Nessa tela estão disponíveis os seguintes botões:

- **Open File**
Utilize o botão *Open File* para selecionar o arquivo de série temporal a ser consultado. Após a abertura do arquivo são consultadas as propriedades *SampleFreq*, *nSamples* e *nChannels* e apresentadas nos respectivos indicadores.
- **Seek Read Pos**
O método *OpenFile* do automation server *LynxFile.FileTS* inicia a leitura sequencial do arquivo de série temporal a partir da primeira amostra do arquivo. O botão *Seek Read Pos* executa a chamada do método *SeekReadPos* do automation server para posicionar a leitura sequencial no índice do número de amostra entrado no controle *iSample*.
- **Read Sample**
O botão *Read Sample* efetua a leitura da amostra do sinal especificado no controle *iSignal*. Além da apresentação do valor da amostra em unidade de engenharia, o tratamento desse botão também efetua a consulta do nome e da unidade do sinal.
- **Next Sample**
Utilize esse botão para incrementar o índice da amostra na leitura sequencial.

- **Read Buffer**
No tratamento deste botão é realizada a leitura de um bloco de amostras do sinal selecionado no controle *iSignal 2*. O índice da primeira amostra e o número de amostra são especificados nos controles *Pos* e *N*. As amostras lidas são apresentadas numa tabela e num gráfico.
- **Stop**
Finaliza a execução do programa exemplo.

5. DESCRIÇÃO DE USO EM PYTHON

5.1. Requisitos

O automation server *LynxFile.FileTS* para leitura de arquivos de séries temporais nos formatos LTX, LTD e TEM do *Lynx AqDados* em Python requer a versão do Python para Windows na plataforma Intel.

São necessários também os seguintes packages do Python:

- **pywin32**
O package *pywin32* é um pacote com extensões Python para Microsoft Windows que disponibiliza acesso ao Win32 API e o suporte a objetos COM. A versão do *pywin32* utilizada no desenvolvimento dos exemplos dessa documentação é o `pywin32-223-cp37-cp37m-win32.whl`.
- **numpy**
O package *numpy* é um pacote desenvolvido para processamento eficiente de arrays multi dimensional. A versão do *numpy* utilizada no desenvolvimento dos exemplos dessa documentação é o `numpy-1.15.0-cp37-none-win32.whl`.

Esses pacotes podem ser obtidos no site do PyPi.org (Python Package Index):

<https://pypi.org/>

O PyPi é um repositório de software para a linguagem de programação Python.

5.2. Instalação dos Pacotes win32py e numpy

Esse tópico pode ser pulado se você já tiver os *packages win32py* e *numpy* instalados no seu computador.

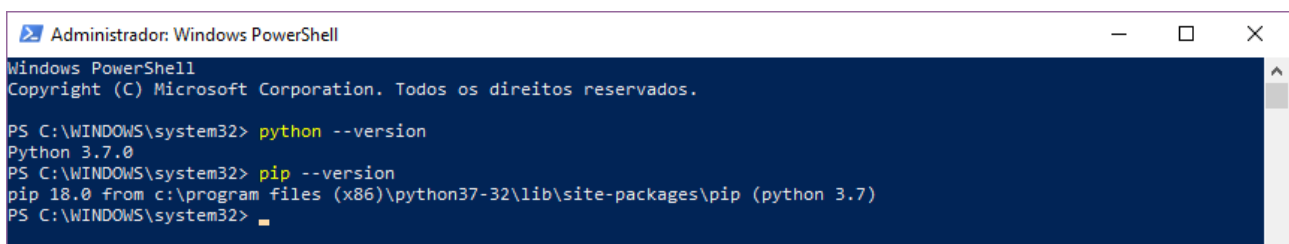
Para a instalação desses pacotes podem ser seguidas as instruções da documentação do Python.

Se preferir, pode-se instalar os pacotes ***win32py*** e ***numpy*** na janela do *PowerShell(Admin)* do Windows 10 ou na janela do *Prompt de Comando* do Windows 7 ou 8. Em ambos os casos deve-se executar esses comandos com privilégio de administrador.

Antes executar os comandos para instalação dos pacotes, deve-se verificar se o *Python* e o *pip* estão instalados no seu computador.

Para verificar se o *Python* e o *pip* se encontram instalados, execute os seguintes comandos no *PowerShell(Admin)* do Windows.

```
python --version  
pip --version
```



```
Administrador: Windows PowerShell  
Windows PowerShell  
Copyright (C) Microsoft Corporation. Todos os direitos reservados.  
PS C:\WINDOWS\system32> python --version  
Python 3.7.0  
PS C:\WINDOWS\system32> pip --version  
pip 18.0 from c:\program files (x86)\python37-32\lib\site-packages\pip (python 3.7)  
PS C:\WINDOWS\system32>
```

Se o *PowerShell* indicar falha na execução dos comandos citados, pode ser que o *Python* não foi instalado ou o diretório do *Python* e/ou *pip* não estão no *path* do Windows.

Na configuração padrão do instalador do Python a opção de inclusão do diretório do Python no *path* do Windows não é habilitada.

Se a execução dos comandos foi bem sucedida, execute os comandos abaixo no *PowerShell(Admin)* do Windows.

```
pip install 'c:\user\Lauro\Downloads\Python\pywin32-223-cp37-cp37m-win32.whl'
```

```
pip install 'c:\user\Lauro\Downloads\Python\numpy-1.15.0-cp37-none-win32.whl'
```

O diretório e o nome dos arquivos citados no comando `pip install` são apenas exemplos. Substitua-os pelo diretório e nomes dos arquivos dos pacotes *win32py* e *numpy* que você baixou do site da *pypi.org*.

5.3. Iniciação do Python

Para acessar o pacote *win32py* e *numpy* no ambiente do Python, é necessário importar as referências a esses pacotes. Para isso execute os seguintes comandos na janela de prompt de comando do Python:

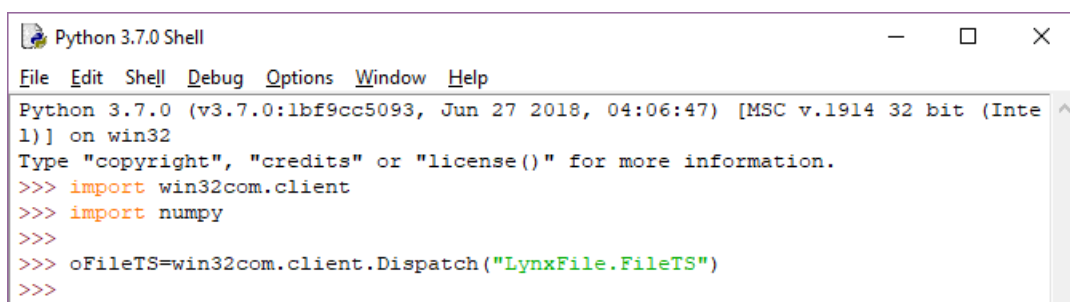
```
import win32com.client
import numpy
```

5.4. Criação da Instância do LynxFile.FileTS

A instância do automation server do *FileLynx.FileTS* é criada através do seguinte comando em Python.

```
oFileTS = win32com.client.Dispatch("LynxFile.FileTS");
```

A figura abaixo ilustra a janela de comandos do Python após importação das referências aos pacotes *win32py* e *numpy* e a criação da instância do *FileLynx.FileTS*.



```
Python 3.7.0 Shell
File Edit Shell Debug Options Window Help
Python 3.7.0 (v3.7.0:1bf9cc5093, Jun 27 2018, 04:06:47) [MSC v.1914 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> import win32com.client
>>> import numpy
>>>
>>> oFileTS=win32com.client.Dispatch("LynxFile.FileTS")
>>>
```

Após criar a instância do *LynxFile.FileTS*, tem-se acesso às propriedades e aos métodos da interface *IFileTS* implementada pelo automation server.

A variável *oFileTS* é o handle da instância do automation server criado pela chamada da *win32com.client.Dispatch*.

No decorrer desta documentação será utilizada a variável *oFileTS* como a instância do automation server *LynxFile.FileTS*.

5.5. Acesso às Propriedades da Interface IFileTS

Para acessar uma propriedade da interface *IFileTS*, basta referenciar o nome da propriedade através do *handle* da instância do automation server *LynxFile.FileTS*.

A tabela seguinte lista as propriedades da interface *IFileTS*.

Propriedade	Type	R/W	Index	Descrição
FileName	String	R		Nome do arquivo de série temporal
Comment	String	R		Comentário do arquivo
SampleFreq	double	R		Frequência de amostragem
nSamples	int64	R		Número de amostras por canal
nEvents	int16	R		Número de eventos registrados no arquivo
EvenTime	double	R	iEvent	Instante em segundos correspondente ao evento <i>iEvent</i>
EventNote	string	R	iEvent	Nota correspondente ao evento <i>iEvent</i>
nChannels	int16	R		Número de canais
SnName	string	R	iSignal	Nome do sinal <i>iSignal</i> . O índice <i>iSignal</i> é o número de ordem do sinal de entrada analógica ou contador. O primeiro canal A/D habilitado para aquisição tem o número de ordem 0, o segundo canal habilitada tem o número de ordem 1 e assim sucessivamente.
SnUnit	string	R	iSignal	Unidade de engenharia do sinal <i>iSignal</i>
SnDesc	string	R	iSignal	Comentário do sinal <i>iSignal</i>
SnHiLim	double	R	iSignal	Limite superior da escala do sinal <i>iSignal</i>
SnLoLim	double	R	iSignal	Limite inferior da escala do sinal <i>iSignal</i>
SnType	int16	R	iSignal	Tipo de linearização do sinal <i>iSignal</i>
MapSnToCh	int16	R	iSignal	Informa o canal A/D correspondente ao sinal <i>iSignal</i>
MapChToSn	int16	R	Channel	Informa o índice do sinal correspondente ao canal <i>Channel</i> . Informa -1 se o canal não está ativo no arquivo.
iSample	int64	R		Índice do número da amostragem na leitura sequencial.
ErrorCode	int32	R		Código de erro (veja os valores no tópico 2.3. <i>Enumerador ecErrorCode</i>).
ErrorCodeStr	string	R		String correspondente ao <i>ErrorCode</i>

Importante: a propriedade *AcqDateTime* não foi listada nessa tabela e não deve ser acessada pela aplicação em Python. O tipo dessa propriedade não é suportado pelo Python. Se essa propriedade for referenciada pela aplicação, ocorrerá um reset do ambiente do Python.

Por exemplo, para verificar a frequência de amostragem do arquivo de série temporal pode-se consultar a propriedades *SampleFreq*.

```
r = oFileTS.SampleFreq;
```

Para obter o nome do primeiro canal ativo do arquivo de série temporal pode-se utilizar o seguinte comando em Python:

```
r = oFileTS.SnName(0);
```

A figura abaixo ilustra um exemplo de consultas a propriedades da interface do automation server.


```
Python 3.7.0 Shell
File Edit Shell Debug Options Window Help
Python 3.7.0 (v3.7.0:1bf9cc5093, Jun 27 2018, 04:06:47) [MSC v.1914 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> import win32com.client
>>> import numpy
>>>
>>> oFileTS=win32com.client.Dispatch("LynxFile.FileTS")
>>> oFileTS.OpenFile("C:\Program Files (x86)\Lynx AqDados 7.5\Exemplos\DEMOTEM.TEM")
True
>>> oFileTS.SampleFreq
204.8000030517578
>>> oFileTS.nSamples
4597
>>> oFileTS.nChannels
6
>>> oFileTS.SnName(0)
'RodaEsquerda'
>>> oFileTS.SnUnit(0)
'm/s2'
>>> oFileTS.SnHiLim(0)
40.0
>>> oFileTS.SnLoLim(0)
-40.0
>>> |
```

5.6. Acesso aos Métodos da Interface IFileTS

Os métodos da interface com automation server *LynxFile.FileTS* são acessados através do handle da sua instância.

Por exemplo, na linha abaixo é chamado o método *ReadSample* para a leitura de amostra do primeiro sinal do arquivo de série temporal.

```
r = oFileTS.ReadSample(0);
```

A tabela seguinte lista os métodos da interface *IFileTS*.

Função	Descrição
OpenFile	Informa o nome do arquivo de série temporal e inicia o acesso para leitura do arquivo
SetLineariz	Define o arquivo com os tipos e as tabelas de linearização de sinais.
SeekReadPos	Posiciona a leitura sequencial das amostras dos canais a partir de uma posição do arquivo de série temporal
ReadSample	Retorna o valor da amostra em unidade de engenharia de um sinal correspondente à posição atual da leitura sequencial do arquivo
PosNextSample	Incrementa o índice da posição de leitura sequencial das amostras
ReadBuffer	Leitura de um bloco de amostras de um sinal do arquivo

5.6.1. Método OpenFile

Este método inicia o acesso para leitura de um arquivo de série temporal.

```
r = oFileTS.OpenFile(FileName);
```

Parâmetro	Descrição
FileName	Parâmetro de entrada do tipo string . Nome do arquivo de série temporal a ser lido. Os formatos aceitos são: TEM, LTD e LTX.
r	O <i>OpenFile</i> retorna <i>true</i> se as operações iniciais para a leitura do arquivo ocorreram com sucesso. Caso contrário o método retornará <i>false</i> . O erro pode ser consultado através das propriedades <i>ErrorCode</i> e <i>ErrorCodeStr</i> .

5.6.2. Método SetLineariz

Neste método é passado o nome arquivo com os tipos e as tabelas de linearização de canais.

```
r = oFileTS.SetLineariz(fnLineariz);
```

Parâmetro	Descrição
fnLineariz	Parâmetro de entrada do tipo string . Nome do arquivo com os tipos e as tabelas de linearização de canais. Usualmente o arquivo de linearização é o arquivo <code>Lineartiz.dat</code> fornecido com o <code>AqDados</code> . A aplicação cliente pode passar outro arquivo de linearização. No entanto, esse arquivo deve seguir a formatação definida para esse tipo de arquivo e apresentada no manual do <code>AqDados</code> .
r	O método retorna <i>true</i> se o arquivo de linearização foi carregado corretamente. O erro pode ser consultado através das propriedades <code>ErrorCode</code> e <code>ErrorCodeStr</code> .

A chamada deste método só é necessária se os arquivos de série temporal lidos através do automation server `LynxFile.FileTS` tiver canais que não sejam lineares.

Se o arquivo de série temporal possui canais com tipos não lineares e o arquivo de linearização não foi passado através do `SetLineariz` ou ocorreu erro no carregamento do arquivo de linearização, a linearização das amostras dos canais não lineares não é realizada. Por exemplo, se o arquivo de série temporal possui canais de termopar e o arquivo de linearização não foi informado, os valores dos canais de termopar serão lidos em mV,

5.6.3. Método SeekReadPos

Este método é utilizado para iniciar a leitura sequencial das amostras dos sinais do arquivo de série temporal.

```
r = oFileTS.SeekReadPos (iSample);
```

Parâmetro	Descrição
iSample	Parâmetro de entrada do tipo int64 . Índice da amostra a partir da qual serão lidos os sinais na leitura sequencial do arquivo de série temporal.
r	O método retorna <i>false</i> se o índice da amostra passado neste método for inválido ou ocorreu algum erro na leitura do arquivo de série temporal. O erro pode ser consultado através das propriedades <code>ErrorCode</code> e <code>ErrorCodeStr</code> .

O `SeekReadPos` posiciona o índice da leitura sequencial no número da amostra especificado no parâmetro `iSample`.

A leitura das amostras dos sinais é realizada através do método `ReadSample` que informa o valor da amostra em unidade de engenharia do sinal especificado no parâmetro da chamada desse método.

O posicionamento para o próximo instante de amostragem dos sinais é realizado através da chamada do método `PosNextSample`.

5.6.4. Método ReadSample

Este método é utilizado na leitura sequencial das amostras do arquivo de série temporal.

```
r = oFileTS.ReadSample(iSignal);
```

Parâmetro	Descrição
iSignal	Parâmetro de entrada do tipo <i>int16</i> . O valor passado em <i>iSignal</i> deve estar entre 0 e <i>nChannels</i> -1. A propriedade <i>nChannels</i> é o número de canais ativos no arquivo de série temporal.
R	O valor retornado pelo <i>ReadSample</i> é o valor da amostra em unidade de engenharia do sinal especificado correspondente à posição atual da leitura sequencial.

5.6.5. Método PosNextSample

Este método incrementa o índice da leitura sequencial do arquivo de série temporal.

```
r = oFileTS.PosNextSample (Step);
```

Parâmetro	Descrição
Step	Parâmetro de entrada do tipo <i>int32</i> . Valor do incremento do índice da amostra na leitura sequencial. Usualmente o valor passado neste parâmetro é 1.
r	O método retorna <i>false</i> se o índice da amostragem ultrapassou o número de amostras por canal do arquivo ou se ocorreu erro de leitura do arquivo. O erro pode ser consultado através das propriedades <i>ErrorCode</i> e <i>ErrorCodeStr</i> .

A posição atual do índice da leitura sequencial pode ser consultada através da propriedade *iSample*.

A figura abaixo ilustra um exemplo de chamadas dos métodos *SeekReadPos*, *ReadSample* e *PosNextSample*.

```
>>> oFileTS.SeekReadPos(1000)
True
>>> oFileTS.ReadSample(0)
-0.206298828125
>>> oFileTS.ReadSample(1)
-0.4345703125
>>> oFileTS.PosNextSample(1)
True
>>> oFileTS.iSample
1001
>>> oFileTS.ReadSample(0)
-0.819091796875
>>> oFileTS.ReadSample(1)
-0.1123046875
>>> |
```

Ln: 37 Col: 4

5.6.6. Método ReadBuffer

Este método é utilizado para a leitura de um bloco de amostras de um sinal do arquivo de série temporal.

```
R = oFileTS.ReadBuffer(iSignal, Pos, N, Buf);  
OU  
r, Buf, NOut = oFileTS.ReadBuffer(iSignal, Pos, N, Buf);
```

Parâmetro	Descrição
iSignal	Parâmetro de entrada do tipo int16 . Índice do sinal a ser lido. O valor passado em <i>iSignal</i> deve estar entre 0 e <i>nChannels</i> -1. A propriedade <i>nChannels</i> é o número de canais ativos no arquivo de série temporal.
Pos	Parâmetro de entrada do tipo int64 . Índice da amostra a partir da qual serão lidas as amostras do sinal no arquivo de série temporal.
N	Parâmetro de entrada do tipo int32 . Número de amostras a serem lidas.
Buf	Parâmetro de entrada e saída do tipo Safearray . No parâmetro Buf deve ser passado um array do tipo double com tamanho maior ou igual ao número de amostras a serem lidas e informadas no parâmetro <i>N</i> .
NOut	Parâmetro de saída do tipo int32 . Número de amostras transferidas para o buffer. O valor retornado em NOut pode ser menor que N se atingiu o final do arquivo.
r	Se a função retorna <i>true</i> se a leitura foi realizada com sucesso. O erro pode ser consultado através das propriedades <i>ErrorCode</i> e <i>ErrorCodeStr</i> .

O vetor passado no parâmetro *Buf* do método *ReadBuffer* deve ser do tipo *double* e ter tamanho suficiente para as leituras realizadas através do *ReadBuffer*. O parâmetro *Buf* é de entrada e saída. No entanto, o suporte ao *Activex* no Python utiliza apenas as informações do tipo e da dimensão do vetor para efetuar a conversão para o tipo *safearray*. Na saída do método *ReadBuffer* o Python não transfere os valores retornados no *safearray* para o parâmetro *Buf*. O conteúdo do *safearray* é transferido pelo Python para um vetor diferente de saída.

Por essa característica do Python, o vetor *Buf* pode ser criado uma única vez e com tamanho adequado para a aplicação. O comando abaixo exemplifica a criação do vetor *Buf* do tipo *double* e com tamanho 200. O vetor é criado através de função do package **numpy**.

```
Buf = numpy.zeros(200);
```

A figura seguinte ilustra um exemplo de chamada do método *ReadBuffer*.

O resultado da função é transferido para a variável *R* no primeiro modo de chamada do método. Nessa estrutura são retornadas as três saídas da função:

- **R[0]**: É o valor retornado da função, *true* ou *false*
- **R[1]**: É a saída *Buf* e corresponde ao vetor com os valores lidos do arquivo de série temporal. Note que a entrada *Buf* é diferente da saída *Buf*.
- **R[2]**: É a saída *NOut* e corresponde ao número de amostras lidas pela função e retornadas

