ActiveX Signal Reader

User Guide

Rev. 1.0

LDS-Dactron

Phone. (408) 934-9160
Fax. (408) 934-9161
E-mail: technical.support@dactron.spx.com
Web Site: www.lds-group.com

Notice

Table of Contents

1 Object ActiveX Signal Reader

ActiveX Signal Reader Methods

ActiveX Signal Reader object provides the following methods:

LoadSignal	Initialize the Signal Reader Object
GetData	Retrieve all signal data between the specified start and end indices
GetSliceValueArray	Retrieve the slice values of a 3D signal
GetEventInform	Retrieve all event data between the specified start and end indices

LoadSignal

LoadSignal initializes the Signal Reader object.

```
HRESULT LoadSignal(BSTR sFileName, BOOL* pRet);
```

Parameters

[in] *sFileName*

A string that contains the full path of the Dactron Signal file to be loaded. The path must be an absolute path.

[out] *pRet*

True if the signal was loaded successfully, otherwise false.

Remarks

This function must be called first. All data in the object will be invalid if LoadSignal has not been called.

Example

Visual C++:

```
See CDacSignalVCDemoDlg::OnLoadsignal() in
DacSignalVCDemoDlg.cpp of the VC++ demo project.
```

Visual Basic:

See Sub LoadSignal_Click() in DacSignalVBDemoDlg.frm of the VB demo project.

GetData

GetData retrieves all signal data between the specified start and end indices.

```
HRESULT GetData(  long nSlice,
                  long nStart,
                  long nEnd,
                  VARIANT* varBuffer);
```

Parameters

[in] *nSlice*

The 3D slice number of the data to be retrieved. Use 0 for 2Dsignals.

[in] *nStart*

The zero based index of the internal data buffer to be used as the first data point to be retrieved. Typically zero.

[in] *nEnd*

The zero based index of the internal data buffer to be used as the last data point to be retrieved. Typically NumOfSamples-1.

[out] *varBuffer*

An array that contains the signal data.

Remarks

The call will fail if the value of *nSlice* is larger than NumOfSlice, the value of *nStart* is greater than *nEnd*, or *nStart* and/or *nEnd* are outside the valid indices of the signal data buffer.
The signal data for complex signals is organized as follows: Real[i] = varBuffer[i*2], Imaginary[i] = varBuffer[(i*2)+1]

Example

Visual C++:

See CDacSignalVCDemoDlg::OnGetData()in DacSignalVCDemoDlg.cpp of the VC++ demo project.

Visual Basic:

See Sub GetDataBuffer_Click() in DacSignalVBDemoDlg.frm of the VB demo project.

GetSliceValueArray

GetSliceValueArray retrieves the slice values of a 3D signal.

```
HRESULT GetSliceValueArray(VARIANT* varBuffer);
```

Parameters

[out] *varBuffer*

An array that contains the slice axis values. E.g. Z[i] = varBuffer[i]

Remarks

This function can be used to get the slice values for 3D signals that have evenly spaced slice values and those that do not.

Example

Visual C++:

See CDacSignalVCDemoDlg::OnGetslicebuffer() in DacSignalVCDemoDlg.cpp of the VC++ demo project.

Visual Basic:

See Sub SliceData_Click() in DacSignalVBDemoDlg.frm of the VB demo project.

GetEventInform

GetEventInform retrieves all event data between the specified start and end indices.

```
HRESULT GetEventInform(VARIANT* varStartPoints,
                       VARIANT* varStartTimes,
                       VARIANT* varEventLengths,
                       BOOL* bRet);
```

Parameters

[out] *varStartPoints*

An array that contains the starting index of each event in the signal file.

[out] *varStartTimess*

An array that contains the starting time in seconds of each event in the signal file.

[out] *varEventLengths*

An array that contains the length in points of each event in the signal file.

[out] *bRet*

True if the data in the arrays is valid. False otherwise.

Remarks

All arrays returned are parallel. The start Point, Time, and Length of an event [i] can be retrieved from the arrays as varStartPoints[i], varStartTimes[i], varEventLengths[i].

Example

Visual C++:

See CDacSignalVCDemoDlg::OnEventInfo() in DacSignalVCDemoDlg.cpp of the VC++ demo project.

Visual Basic:

See Sub EventInfo_Click() in DacSignalVBDemoDlg.frm of the VB demo project.

ActiveX Signal Reader Properties

ActiveX Signal Reader object provides the following properties:

Property		Type
Name	**Description**	
Version	Active X object version. 1000 means 1.00	Long
SigVersion	Signal version. 1000 means version 1.00	Long
NumOfSamples (Xsize)	Number of Samples	Long
DataType (Xstep)	Used to determine if data is real only, or real and imaginary. 1: Y is real data 2: Y is (real, imaginary) data	Long
XaxisBegin	The starting X value.	Double
XaxisDelta	Delta for X axis	Double
XaxisSpacingEven (Xincrease)	Defines how X axis values should be calculated. If True, X[i] = X[0]+Delta* i, If False, X[i] = X[0]*power(delta , i)	BOOL
XaxisUnit (Xlabel)	Text label that has been associated with the X axis.	BSTR
LowValidIndex	First valid index of the data buffer	Long
HighValidIndex	Last valid index of the data buffer	Long
IsRecordingData	If true if the signal file contains more than one frame of data.	BOOL
DataFormat	Type of signal. (see description of DATAFORMAT for signal type details)	Enum DATAFORMAT
Chan1Dimension	Channel dimension (see description of TRANSUNITS for dimension details)	Eunm TRANSUNITS
Chan2Dimension	Channel dimension (see description of TRANSUNITS for dimension details)	Eunm TRANSUNITS
SignalName	Display name for the signal	BSTR
EULabel	Display text for the Engineer Unit of the signal.	BSTR
MeasDate	Date the signal was measured. Format:	BSTR
MeasTime	Time the signal was measured. Format: HH:MM:SS (24hr clock)	BSTR
NumOfSlice (Zsize)	The value is 1 for 2D signals. If the signal is a 3D signal, the value represents the number of slices along	Long

	the slice axis.	
SliceBegin (Zbegin)	Starting value for the slice axis.	Double
SliceDelta (Zdelta)	Delta for slice axis.	Double
SliceUnit	Display units for slice axis.	BSTR
SliceSpacingEven (Zincrease)	If True, Z[i] = Z[0]+Delta* i If False, retrieve data from slice data buffer. Also see GetSliceValueArray()	BOOL
NumOfEvent	Number of events in a time domain signal.	Long
ChanStatus.IsAvailable	If True, Channel Status values are valid If False, Channel Status values are not valid.	BOOL
ChanStatus.Max	Signal Max	Float
ChanStatus.Min	Signal Min	Float
ChanStatus.Mean	Signal Mean	Float
ChanStatus.Peak	Signal Peak	Float
ChanStatus.RMS	Signal RMS	Float
ChanStatus.PSDRMS	Signal PSD RMS	Float
Advance.SampRate	Rate at which the signal was sampled.	Double
Advance.XallocSize	Total size of the X axis buffer. For some signals this number is larger than the value of NumOfSamples.	Long
Advance.AllocSize	Total buffer size allocated for the signal. Includes X and Z data.	Long
Advance.Order	Order value. Valid only for APSD_RPM and CPSD_RPM signals.	Double
Advance.OrderDelta	Internal data is stored as EU^2/Order. This value is used a conversion factor to convert the signal data to EU^2. Valid only for APSD_RPM and CPSD_RPM signals.	Double
Advance.WindowType	The window function applied to the signal. (see description of WNDTYPE for window type details)	Enum WNDTYPE
Advance.Oct_b	Octave number. If Advance.Oct_b = 3, then 1/3 octave.	Long
Advance.Oct_Weight	Octave weighting. One of A,B,C,D, Linear, or none.	Enum WEIGHTING
Advance.Oct_HighFreq	High end of the analysis range	Float
Advance.Oct_LowFreq	Low end of the analysis range	Float

ActiveX Signal Reader Enumerations

Enum DATAFORMAT

Signal Data Format	Integer Equivalent	Data Format Description
FRF_FREQ	0	Frequency Response
COH_FREQ	1	Coherence
SPE_FREQ	2	Linear Spectrum
APS_FREQ	3	Auto-Power Spectrum
CPS_FREQ	4	Cross-Power Spectrum
APSD_FREQ	5	Auto-Power Spectrum Density
CPSD_FREQ	6	Cross-Power Spectrum Density
SRS3_FREQ	8	SRS3 Spectrum
OCT_FREQ	9	Octave Spectrum
REAL_FREQ	10	Real Frequency
REAL_TIME	11	Time domain signal
ACOR_TIME	12	Auto Correlation
CCOR_TIME	13	Cross Correlation
CPLX_TIME	14	Complex Time
TRANS_FREQ	17	Transmissibility. Data consists of a pair of values (amplitude, phase)
SRS_FREQ	21	SRS Spectrum
HIST_AMPL	22	Histogram
ERROR_SRS_FREQ	24	SRS Error Spectrum
APSD_ORDER	25	Auto-Power Spectrum Density Order
IRF_TIME	26	Impulse Response Time
CEPS_TIME	27	Cepstrum
ORDER_TRACK	28	Order Track. RPM Trace.
CPSD_ORDER	29	Cross-Power Spectrum Density Order
APSD_RPM	30	Order Track Slice
CPSD_RPM	31	Cross Order Track Slice

Enum TRANSUNITS

Unit Type	Integer Equivalent	Unit Description
TRANS_NONE	0	No Units
TRANS_ACC	1	Acceleration
TRANS_VEL	2	Velocity
TRANS_DISP	3	Displacement
TRANS_FORCE	4	Force

TRANS_PRESSURE	6	Pressure
TRANS_VOLT	7	Volts
TRANS_ACC_ANGULAR	8	Angular Acceleration
TRANS_RPM	9	RPM
TRANS_USER1	101	User Defined 1
TRANS_USER2	102	User Defined 2
TRANS_USER3	103	User Defined 3

Enum WNDTYPE

Window Type	Integer Equivalent	Window Type Description
WT_OFF	0	No window
WT_BARTLET	1	Bartlett window
WT_HANNING	2	Hanning window
WT_HAMMING	3	Hamming window
WT_WELCH	4	Welch window
WT_TUKEY	5	Tukey window
WT_BLACKMN	6	Blackman (common) window
WT_MINSIDE	7	Blackman's (4th) Minimum Side Lobe window
WT_MAXDECA	8	Blackman's (4th) Maximum Decay window
WT_FLATTOP	9	Flat top window
WT_KAISER	10	Kaiser-Bessel window
WT_FORCE	11	Force-Exponential Window
WT_TRIANGLE	12	Triangle Window
WT_HALFSINE	13	Half Window
WT_FLATTOP_301	14	Flat top window (P301)
WT_FLATTOP_310	15	Flat top window (p310)
WT_FLATTOP_BK	16	Flat top window (B&K)
WT_FLATTOP_HP	17	Flat top window (HP)

Enum WEIGHTING

Window Type	Integer Equivalent	Window Type Description
WEIGHTING_LINEAR	0	Linear Weighting
WEIGHTING_A	1	A Weighting
WEIGHTING_B	2	B Weighting
WEIGHTING_C	3	C Weighting
WEIGHTING_D	4	D Weighting
WEIGHTING_NONE	5	No Weighting

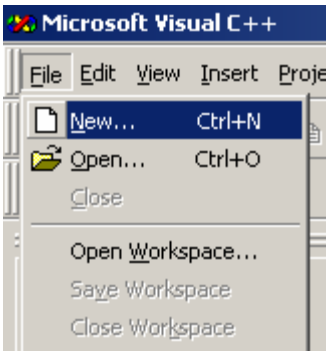
Adding ActiveX Signal Reader to a Project

This section will guide you through the process of adding the Dactron ActiveX Signal Reader to your project. The ActiveX Signal Reader should already be installed on your development PC. If it has not been installed, please install before continuing with this section.

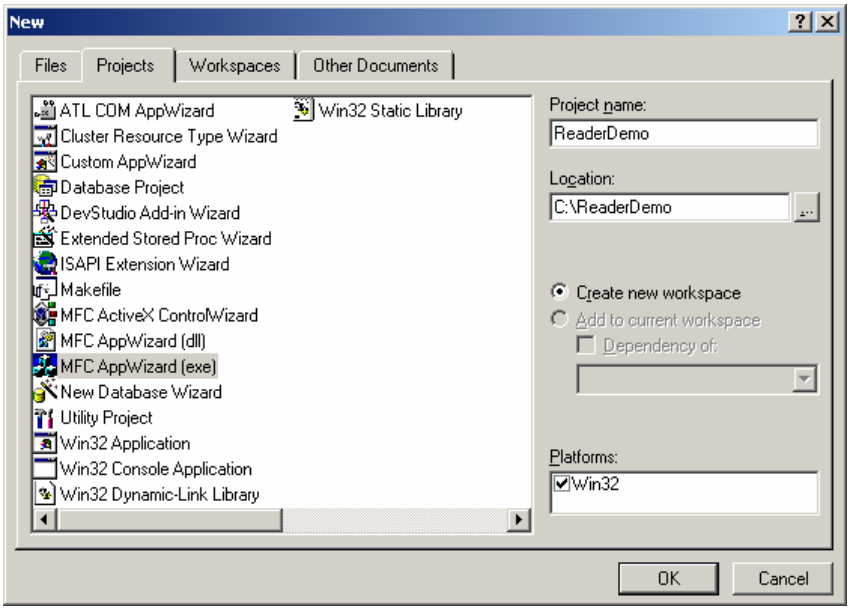
Visual C++

Step 1- Create a dialog based application

- Start Visual C++
- Select "New" from the file menu

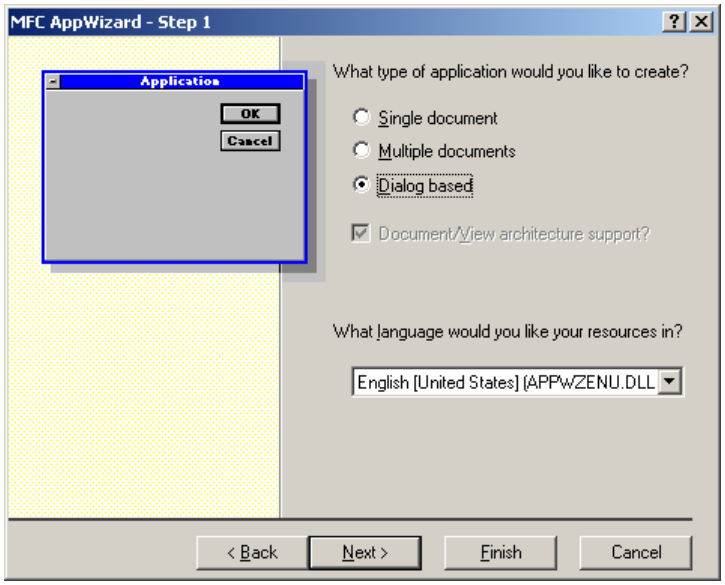


- Click on the Projects tab
- Click "MFCAppWizard(EXE)"
- Enter a name for the project (ReaderDemo)



- Click "Ok"

- Select Dialog based as the type of application.

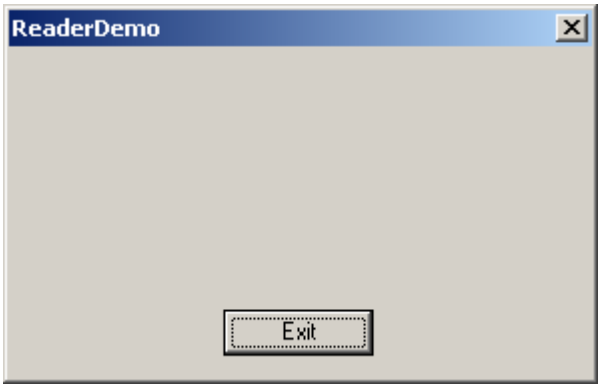


- Click “Finish”
- Click “Ok”

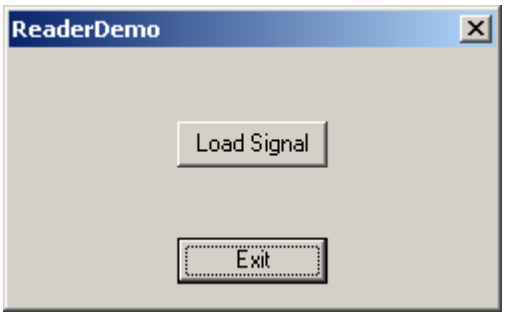
The default dialog based application is now configured.

Step 2 – Modify the User Interface

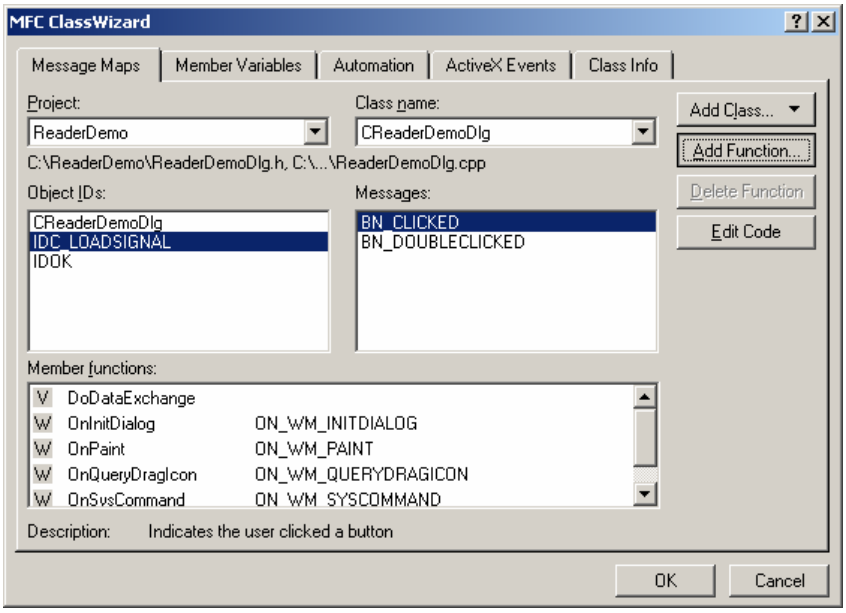
- Remove the “Cancel” button and rename the “Ok” button to “Exit”



- Add a new button and name it “Load Signal”



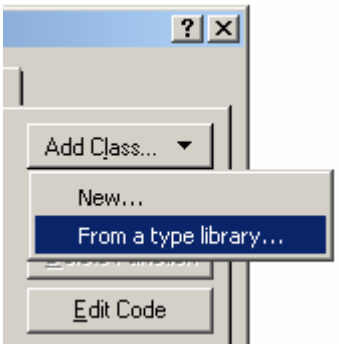
- Use the Class Wizard to add a message handler for the “Load Signal” button called “OnLoadsignal”



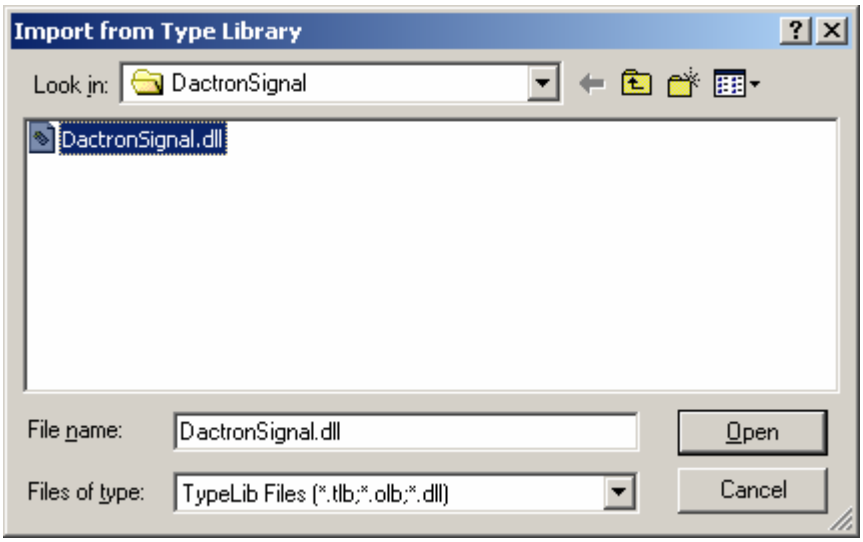
The dialog box has now been configured with two buttons, “Load Signal” and “Ok”.

Step 3 – Add the IDacSignal, IAdvancedSignal, and IChanStatus client interface classes

- Open the Class Wizard
- Click the "Add Class" button and select "From Type Library"

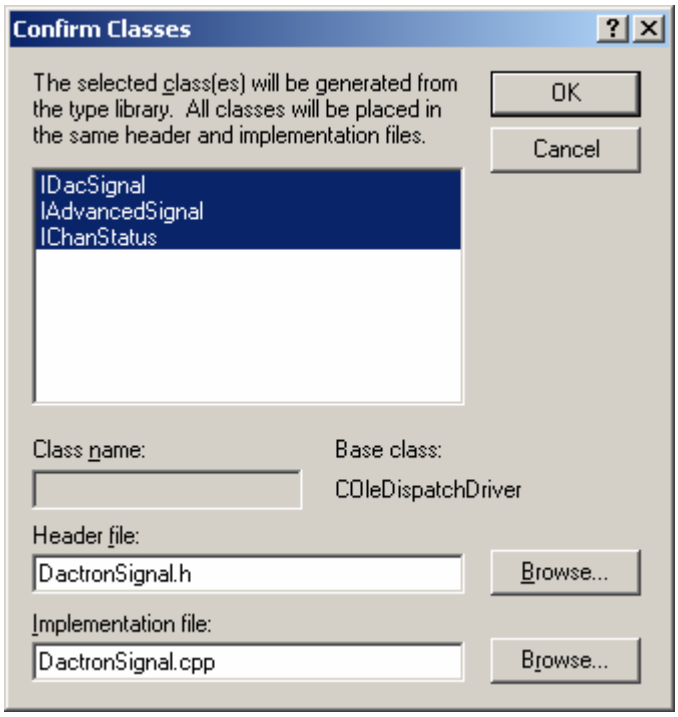


- Locate and select your Dactron Signal Dll.

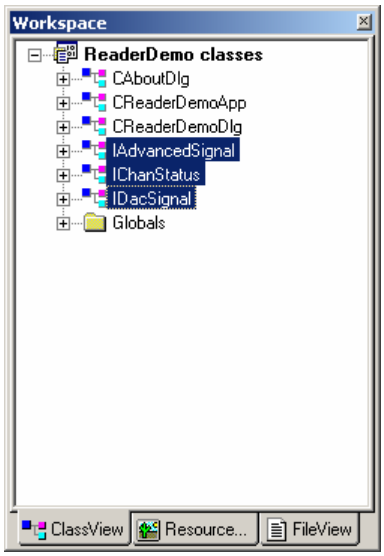


- Click "Open"

- Click “Ok“ to confirm the creation of the new classes.



The wizard will create classes for the IDacSignal, IAdvancedSignal, and IChanStatus interfaces.



Step 4 – Using the ActiveX Signal Reader in the project

- Include the Dactron Signal header in the ReaderDemoDlg.h file and add an IDacSignal member variable to the CReaderDemoDlg class.
```
#include "DactronSignal.h"
…
IDacSignal m_DactronSignal;
...
```

- Add the following lines to CReaderDemoDlg::OnInitDialog() to initialize the COM library and to create an instance of the DacSignal.
```
CoInitialize(NULL);
m_DactronSignal.CreateDispatch("DactronSignal.DacSignal.1");
```

- Add the following code to the OnLoadsignal() message handler to load a Dactron Signal named "demo.sig" when the "Load Signal" button is clicked.
```
BOOL bSucceded;

bSucceded =
m_DactronSignal.LoadSignal("C:\\ReaderDemo\\demo.sig");

if(bSucceded)
{
      MessageBox("Signal loaded",MB_OK);
}
else
{
      MessageBox("Error loading signal",MB_OK);
}
```

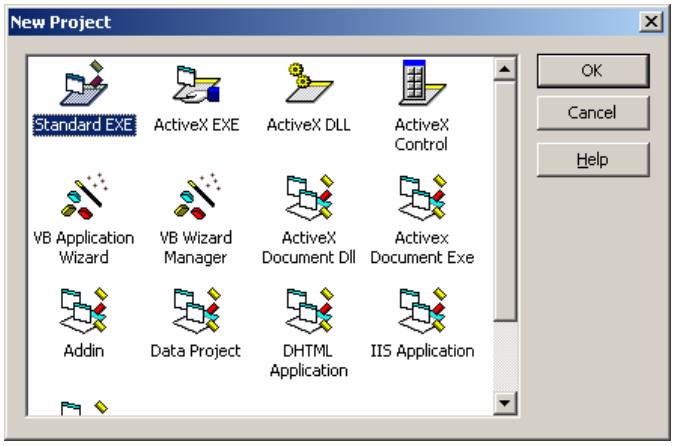
Step 5 – Test

Compile and run the application. When the "Load Signal" button is pressed, the signal in the demo.sig file will be loaded.

Visual Basic

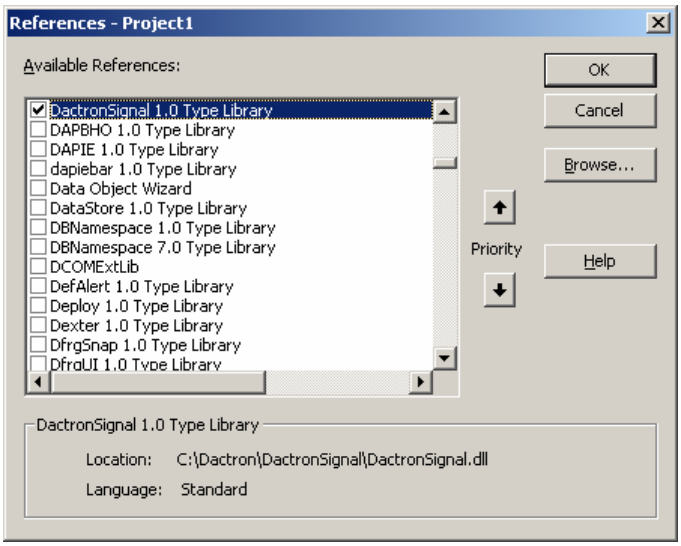
Step 1 – Create a new project

- Start Microsoft Visual Basic
- Select “New Project” from the “File” menu
- Select “Standard EXE”



Step 2 - Add a reference to the Dactron ActiveX Signal Reader

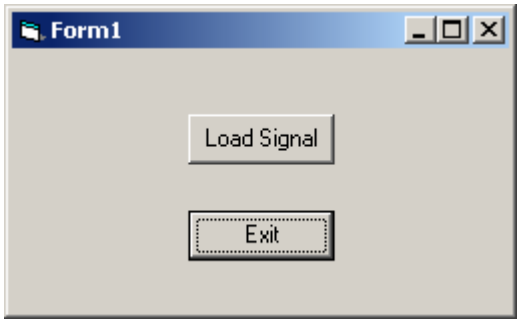
- Select “References…” from the “Project” menu.
- Locate and select the DactronSignal Type Library.



- Click “Ok”

Step 3 – Modify the User Interface

- Add two buttons to the form, one to load the signal and the other to quit the application.



Step 4 – Add functionality to the User Interface

- Add a Dactron Signal to the default form

```
Private DactronSignal As New DacSignal
```

- Add the following code the Load Signal Click event handler.

```
Dim bSucceded As Boolean
bSucceded = DactronSignal.LoadSignal("demo.sig")
If (bSucceded) Then
   MsgBox ("Signal loaded")
Else
   MsgBox ("Error loading Signal")
End If
```

Step 5 – Test

Run application. When the “Load Signal” button is pressed, the signal in the demo.sig file will be loaded.

2 Software License Statement

If you keep the Software, you are agreeing to the terms of this Agreement. Use of the Software is subject to this agreement. If you do not agree to the terms of this Agreement, return this package to LDS for a refund. LDS grants you a non-exclusive right to use this copy of the Software and accompanying materials according to the following:

You may:

a) use and install the Software on only one computer at a time;
b) make one (1) copy of the Software for backup purposes; and
c) transfer the Software from one computer to another so long as the Software is neither used on nor copied onto more than one computer at a time.

You may not:

a) modify, reverse engineer or copy (except for the backup copy) the Software or accompanying materials;
b) rent or transfer the Software.

This license is not a sale. Title and copy rights to the Software and accompanying materials remains with LDS and its licensors. Any breach terminates this Agreement.

LIMITED WARRANTY, DISCLAIMER, AND LIMIT OF LIABILITY

If at any time the software fails to operate as described in the documentation, simply return the original disks to LDS and LDS will replace the software at no charge to you or refund your purchase price, at LDS's option. LDS and its licensors do not warrant your data, that the Software will meet your requirements, or that the operation will be interrupted or error free.

THE WARRANTY ABOVE IS IN LIEU OFF ALL OTHERS, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. SOME STATES HAVE LAWS THAT MAY RESTRICT OR PROHIBIT THE LIMITATIONS SET FORTH ABOVE AND YOUR RIGHTS UNDER LAW MAY VARY ACCORDING TO THE LAWS OF YOUR STATE.

LDS and its licensers shall not be liable for any incidental or consequential damages, including loss of data, lost profits, cost of cover or other special or indirect damages. Your rights under law may vary.

US GOVERNMENT RESTRICTED RIGHTS

The software and documentation are provided with Restricted Rights. Use, duplication, or disclosure by the Government is subject to restrictions as set forth in subparagraph c(1)(ii) of the Rights in Technical Data and Computer Software clause at DFARS 252.227-7013 or subparagraphs c(1) and (2) of the Commercial Computer Software - Restricted Rights at 48 CFR 52.227-19 as applicable. The Manufacturer is LDS Test and Measurement LLC, 8551 Research Way, M/S 140, Middleton, WI 53562.

This license agreement and limited warranty will be governed by the laws of the state of California. If any provision of this agreement is found void or unenforceable, the remainder of the agreement shall remain valid and enforceable.

.

3 Manual Revision History

Date	Manual Version	Software Version	Contents added into the Manual in this version
9 MAR 2003	1.0, 22 pages	1.0	Initial Product Release

Index