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

Information in this document is subject to change without notice. No part of this document may be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose, without the express written permission of LDS.

LDS makes no warranties on the software, whether express or implied, nor implied warranties of merchantability or fitness for a particular purpose. LDS does not warrant your data, that the software will meet your requirements, or that the operation will be reliable or error free. The user of the software assumes the entire risk of use of the software and the results obtained from use of the software. LDS 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.

Copyright ©1997-2005 LDS Test and Measurement. LDS is a member of SPX Corporation. All rights reserved

All trademarks and registered trademarks are the property of their respective holders.
Table of Contents

1 OBJECT ACTIVEX SIGNAL READER................................................................. 1

ActiveX Signal Reader Methods ............................................................................................................... 1
LoadSignal.................................................................................................................................................... 1
GetData...................................................................................................................................................... 1
GetSliceValueArray .................................................................................................................................... 2
GetEventInform.......................................................................................................................................... 3

ActiveX Signal Reader Properties.............................................................................................................. 4

ActiveX Signal Reader Enumerations .......................................................................................................... 6
Enum DATAFORMAT ........................................................................................................................................ 6
Enum TRANSUNITS ...................................................................................................................................... 6
Enum WNDTYPE .......................................................................................................................................... 7
Enum WEIGHTING....................................................................................................................................... 7

Adding ActiveX Signal Reader to a Project ................................................................................................. 8
Visual C++ .................................................................................................................................................. 8
Visual Basic ................................................................................................................................................ 14

2 SOFTWARE LICENSE STATEMENT.......................................................... 16

3 MANUAL REVISION HISTORY................................................................. 17

INDEX .............................................................................................................. 18
1 Object ActiveX Signal Reader

ActiveX Signal Reader Methods

ActiveX Signal Reader object provides the following methods:

- **LoadSignal**
  
  Initializes the Signal Reader Object

- **GetData**
  
  Retrieves all signal data between the specified start and end indices

- **GetSliceValueArray**
  
  Retrieves the slice values of a 3D signal

- **GetEventInform**
  
  Retrieves all event data between the specified start and end indices

**LoadSignal**

**LoadSignal** initializes the Signal Reader object.

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

**Parameters**

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

- **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);
```

**ActiveX Signal Reader Methods**

**Parameters**

- **[in]** `nSlice`
  
  The 3D slice number of the data to be retrieved. Use 0 for 2D signals.

- **[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] varStartTimes
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:

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>Version</td>
<td>Active X object version. 1000 means 1.00</td>
<td>Long</td>
</tr>
<tr>
<td>SigVersion</td>
<td>Signal version. 1000 means version 1.00</td>
<td>Long</td>
</tr>
<tr>
<td>NumOfSamples (Xsize)</td>
<td>Number of Samples</td>
<td>Long</td>
</tr>
<tr>
<td>DataType (Xstep)</td>
<td>Used to determine if data is real only, or real and imaginary. 1: Y is real</td>
<td>Long</td>
</tr>
<tr>
<td></td>
<td>data 2: Y is (real, imaginary) data</td>
<td></td>
</tr>
<tr>
<td>XaxisBegin</td>
<td>The starting X value.</td>
<td>Double</td>
</tr>
<tr>
<td>XaxisDelta</td>
<td>Delta for X axis</td>
<td>Double</td>
</tr>
<tr>
<td>XaxisSpacingEven (Xincrease)</td>
<td>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)</td>
<td>BOOL</td>
</tr>
<tr>
<td>XaxisUnit (Xlabel)</td>
<td>Text label that has been associated with the X axis.</td>
<td>BSTR</td>
</tr>
<tr>
<td>LowValidIndex</td>
<td>First valid index of the data buffer</td>
<td>Long</td>
</tr>
<tr>
<td>HighValidIndex</td>
<td>Last valid index of the data buffer</td>
<td>Long</td>
</tr>
<tr>
<td>IsRecordingData</td>
<td>If true if the signal file contains more than one frame of data.</td>
<td>BOOL</td>
</tr>
<tr>
<td>DataFormat</td>
<td>Type of signal. (see description of DATAFORMAT for signal type details)</td>
<td>Enum</td>
</tr>
<tr>
<td>Chan1Dimension</td>
<td>Channel dimension (see description of TRANSUNITS for dimension details)</td>
<td>Enum</td>
</tr>
<tr>
<td>Chan2Dimension</td>
<td>Channel dimension (see description of TRANSUNITS for dimension details)</td>
<td>Enum</td>
</tr>
<tr>
<td>SignalName</td>
<td>Display name for the signal</td>
<td>BSTR</td>
</tr>
<tr>
<td>EULabel</td>
<td>Display text for the Engineer Unit of the signal.</td>
<td>BSTR</td>
</tr>
<tr>
<td>MeasDate</td>
<td>Date the signal was measured. Format: HH:MM:SS (24hr clock)</td>
<td>BSTR</td>
</tr>
<tr>
<td>MeasTime</td>
<td>Time the signal was measured. Format: HH:MM:SS (24hr clock)</td>
<td>BSTR</td>
</tr>
<tr>
<td>NumOfSlice (Zsize)</td>
<td>The value is 1 for 2D signals. If the signal is a 3D signal, the value</td>
<td>Long</td>
</tr>
<tr>
<td></td>
<td>represents the number of slices along</td>
<td></td>
</tr>
</tbody>
</table>
### ActiveX Signal Reader Properties

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

Enum DATAFORMAT

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

Enum TRANSUNITS

<table>
<thead>
<tr>
<th>Unit Type</th>
<th>Integer Equivalent</th>
<th>Unit Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRANS_NONE</td>
<td>0</td>
<td>No Units</td>
</tr>
<tr>
<td>TRANS_ACC</td>
<td>1</td>
<td>Acceleration</td>
</tr>
<tr>
<td>TRANS_VEL</td>
<td>2</td>
<td>Velocity</td>
</tr>
<tr>
<td>TRANS_DISP</td>
<td>3</td>
<td>Displacement</td>
</tr>
<tr>
<td>TRANS_FORCE</td>
<td>4</td>
<td>Force</td>
</tr>
</tbody>
</table>
ActiveX Signal Reader Enumerations

<table>
<thead>
<tr>
<th>Enum WNDTYPE</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Window Type</strong></td>
</tr>
<tr>
<td>WT_OFF</td>
</tr>
<tr>
<td>WT_BARTLET</td>
</tr>
<tr>
<td>WT_HANNING</td>
</tr>
<tr>
<td>WT_HAMMING</td>
</tr>
<tr>
<td>WT_WELCH</td>
</tr>
<tr>
<td>WT_TUKEY</td>
</tr>
<tr>
<td>WT_BLACKMN</td>
</tr>
<tr>
<td>WT_MINSIDE</td>
</tr>
<tr>
<td>WT_MAXDECA</td>
</tr>
<tr>
<td>WT_FLATTOP</td>
</tr>
<tr>
<td>WT_KAISER</td>
</tr>
<tr>
<td>WT_FORCE</td>
</tr>
<tr>
<td>WT_TRIANGLE</td>
</tr>
<tr>
<td>WT_HALFSINE</td>
</tr>
<tr>
<td>WT_FLATTOP_301</td>
</tr>
<tr>
<td>WT_FLATTOP_310</td>
</tr>
<tr>
<td>WT_FLATTOP_BK</td>
</tr>
<tr>
<td>WT_FLATTOP_HP</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Enum WEIGHTING</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Window Type</strong></td>
</tr>
<tr>
<td>WEIGHTING_LINEAR</td>
</tr>
<tr>
<td>WEIGHTING_A</td>
</tr>
<tr>
<td>WEIGHTING_B</td>
</tr>
<tr>
<td>WEIGHTING_C</td>
</tr>
<tr>
<td>WEIGHTING_D</td>
</tr>
<tr>
<td>WEIGHTING_NONE</td>
</tr>
</tbody>
</table>
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”
Adding ActiveX Signal Reader to a Project

- Select Dialog based as the type of application.

![Image of MFC AppWizard - Step 1](image)

- 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”

![Image of ReaderDemo](image)
Adding ActiveX Signal Reader to a Project

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

![ReaderDemo dialog box with Load Signal and Exit buttons](image)

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

![MFC Class Wizard with LoadSignal message highlighted](image)

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”
Adding ActiveX Signal Reader to a Project

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

![Confirm Classes](image)

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

![Workspace](image)
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 bSucceeded;
  bSucceeded =
  m_DactronSignal.LoadSignal("C:\ReaderDemo\demo.sig");
  ```
  ```
  if(bSucceeded)
  {
    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.

![Form1](image)

Step 4 – Add functionality to the User Interface

- Add a Dactron Signal to the default form

```vba
Private DactronSignal As New DacSignal
```

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

```vba
Dim bSucceeded As Boolean
bSucceeded = DactronSignal.LoadSignal("demo.sig")
If (bSucceeded) 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.
Limited Warranty Statement

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 OF 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 licensors 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

<table>
<thead>
<tr>
<th>Date</th>
<th>Manual Version</th>
<th>Software Version</th>
<th>Contents added into the Manual in this version</th>
</tr>
</thead>
<tbody>
<tr>
<td>9 MAR 2003</td>
<td>1.0, 22 pages</td>
<td>1.0</td>
<td>Initial Product Release</td>
</tr>
</tbody>
</table>
Index

A
ActiveX Signal Reader Enumerations .......... 6
ActiveX Signal Reader Methods .................... 1
ActiveX Signal Reader Properties ............... 4
Adding ActiveX Signal Reader to a Project .... 8

E
Enum DATAFORMAT ................................... 6
Enum WEIGHTING ...................................... 7
Enum WNDTYPE ....................................... 7
Enum TRANSUNITS .................................. 6

G
GetData .................................................. 1
GetEventInform ..................................... 1, 3

GetSliceValueArray ................................. 1, 2

L
LoadSignal ............................................ 1

R
Revision History .................................... 17

V
Visual Basic ......................................... 14
Visual C++ ........................................... 8

W
Software Warranty Statement .................... 16