User's Manual

Model 701991
MATLAB Tool Kit for DL Series

YOKOGAWA
Yokogawa Electric Corporation

IM 701991-61E
5th Edition

Product Registration

Thank you for purchasing YOKOGAWA products.

YOKOGAWA provides registered users with a variety of information and services.
Please allow us to serve you best by completing the product registration form accessible from our homepage.

http://www.yokogawa.com/tm/

Thank you for purchasing the MATLAB tool kit for DL series (701991).
This User's Manual contains useful information about the precautions, functions, and operating procedure of the MATLAB tool kit for DL series. To ensure correct use, please read this manual thoroughly before beginning operation. For information about the handling of the WVF File Access Tool Kit that is included in the package, prepare the Model 707712 WVF File Access API or the Model 707741 WE Control API, and see chapter 3 in this manual.
After reading the manual, keep it in a convenient location for quick reference whenever a question arises during operation.
For information about the handling precautions, functions, and operating procedures of Windows, MATLAB, and Yokogawa's DL Series Oscilloscopes, see the respective manuals.

Note

- The contents of this manual are subject to change without prior notice as a result of continuing improvements to the software's performance and functions. The figures given in this manual may differ from those that actually appear on your screen.
- Every effort has been made in the preparation of this manual to ensure the accuracy of its contents. However, should you have any questions or find any errors, please contact your nearest YOKOGAWA dealer as listed on the back cover of this manual.
- Copying or reproducing all or any part of the contents of this manual without the permission of Yokogawa Electric Corporation is strictly prohibited.
- This software program supports the following DL Series Oscilloscopes.
  DL1600 Series
  DL1700E Series
  DL7400 Series
  DL750 and DL750P
  DL9000 Series
  SL1400
- This manual supports MATLAB R2007a (Ver. 7.4.0) or later.
- The TCP/IP software of this product and the document concerning the TCP/IP software have been developed/created by YOKOGAWA based on the BSD Networking Software, Release 1 that has been licensed from the University of California.

Trademarks

- MATLAB is a registered trademark of The MathWorks, Inc. in the United States.
- Microsoft, MS-DOS, Windows, Windows NT, and Windows Vista are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries.
- Adobe and Acrobat are trademarks or registered trademarks of Adobe Systems Incorporated.
- For purposes of this manual, the TM and ® symbols do not accompany their respective trademark names or registered trademark names. Other company and product names are trademarks or registered trademarks of their respective companies.

Revisions

- 1st Edition: July 2004
- 2nd Edition: September 2004
- 3rd Edition: October 2004
- 4th Edition: February 2006
- 5th Edition: April 2007

5th Edition : April 2007 (YK)

Terms and Conditions of the Software License

Yokogawa Electric Corporation, a Japanese corporation (hereinafter called “Yokogawa”), grants permission to use this Yokogawa Software Program (hereinafter called the “Licensed Software”) to the Licensee on the conditions that the Licensee agrees to the terms and conditions stipulated in Article 1 hereof.
You, as the Licensee (hereinafter called “Licensee”), shall agree to the following terms and conditions for the software license (hereinafter called the “Agreement”) based on the use intended for the Licensed Software.
Please note that Yokogawa grants the Licensee permission to use the Licensed Software under the terms and conditions herein and in no event shall Yokogawa intend to sell or transfer the Licensed Software to the Licensee.
Licensed Software Name: 701991 MATLAB Tool Kit for DL Series
Number of License: 1

Article 1 (Scope Covered by these Terms and Conditions)

1.1 The terms and conditions stipulated herein shall be applied to any Licensee who purchases the Licensed Software on the condition that the Licensee consents to agree to the terms and conditions stipulated herein.
1.2 The “Licensed Software” herein shall mean and include all applicable programs and documentation, without limitation, all proprietary technology, algorithms, and know-how such as a factor, invariant or process contained therein.

Article 2 (Grant of License)

2.1 Yokogawa grants the Licensee, for the purpose of single use, non-exclusive and non-transferable license of the Licensed Software with the license fee separately agreed upon by both parties.
2.2 The Licensee is, unless otherwise agreed in writing by Yokogawa, not entitled to copy, change, sell, distribute, transfer, or sublicense the Licensed Software.
2.3 The Licensed Software shall not be copied in whole or in part except for keeping one (1) copy for back-up purposes. The Licensee shall secure or supervise the copy of the Licensed Software by the Licensee itself with great, strict, and due care.
2.4 In no event shall the Licensee dump, reverse assemble, reverse compile, or reverse engineer the Licensed Software so that the Licensee may translate the Licensed Software into other programs or change it into a man-readable form from the source code of the Licensed Software. Unless otherwise separately agreed by Yokogawa, Yokogawa shall not provide the Licensee the source code for the Licensed Software.
2.5 The Licensed Software and its related documentation shall be the proprietary property or trade secret of Yokogawa or a third party which grants Yokogawa the rights. In no event shall the Licensee be transferred, leased, sublicensed, or assigned any rights relating to the Licensed Software.
2.6 Yokogawa may use or add copy protection in or onto the Licensed Software. In no event shall the Licensee remove or attempt to remove such copy protection.
2.7 The Licensed Software may include a software program licensed for re-use by a third party (hereinafter called “Third Party Software”, which may include any software program from affiliates of Yokogawa made or coded by themselves.) In the case that Yokogawa is granted permission to sublicense to third parties by any licensors (sub-licensor) of the Third Party Software pursuant to different terms and conditions than those stipulated in this Agreement, the Licensee shall observe such terms and conditions of which Yokogawa notifies the Licensee in writing separately.
2.8 In no event shall the Licensee modify, remove or delete a copyright notice of Yokogawa and its licenser contained in the Licensed Software, including any copy thereof.

Article 3 (Restriction of Specific Use)

3.1 The Licensed Software shall not be intended specifically to be designed, developed, constructed, manufactured, distributed or maintained for the purpose of the following events:
a) Operation of any aviation, vessel, or support of those operations from the ground;,
b) Operation of nuclear products and/or facilities;,
c) Operation of nuclear weapons and/or chemical weapons and/or biological weapons; or
d) Operation of medical instrumentation directly utilized for humankind or the human body.
3.2 Even if the Licensee uses the Licensed Software for the purposes in the preceding Paragraph 3.1, Yokogawa has no liability to or responsibility for any demand or damage arising out of the use or operations of the Licensed Software, and the Licensee agrees, on its own responsibility, to solve and settle the claims and damages and to defend, indemnify or hold Yokogawa totally harmless, from or against any liabilities, losses, damages and expenses (including fees for recalling the Products and reasonable attorney’s fees and court costs), or claims arising out of and related to the above-said claims and damages.

Article 4 (Warranty)

4.1 The Licensee shall agree that the Licensed Software shall be provided to the Licensee on an “as is” basis when delivered. If defect(s), such as damage to the medium of the Licensed Software, attributable to Yokogawa is found, Yokogawa agrees to replace, free of charge, any Licensed Software on condition that the defective Licensed Software shall be returned to Yokogawa’s specified authorized service facility within seven (7) days after opening the Package at the Licensee’s expense. As the Licensed Software is provided to the Licensee on an “as is” basis when delivered, in no event shall Yokogawa warrant that any information on or in the Licensed Software, including without limitation, data on computer programs and program listings, be completely accurate, correct, reliable, or the most updated.
4.2 Notwithstanding the preceding Paragraph 4.1, when third party software is included in the Licensed Software, the warranty period and terms and conditions that apply shall be those established by the provider of the third party software.
4.3 When Yokogawa decides in its own judgement that it is necessary, Yokogawa may from time to time provide the Licensee with Revision upgrades and Version upgrades separately specified by Yokogawa (hereinafter called “Updates”).
4.4 Notwithstanding the preceding Paragraph 4.3, in no event shall Yokogawa provide Updates where the Licensee or any third party conducted renovation or improvement of the Licensed Software.
4.5 THE FOREGOING WARRANTIES ARE EXCLUSIVE AND IN LIEU OF ALL OTHER WARRANTIES OF QUALITY AND PERFORMANCE, WRITTEN, ORAL, OR IMPLIED, AND ALL OTHER WARRANTIES INCLUDING ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE ARE HEREBY DISCLAIMED BY YOKOGAWA AND ALL THIRD PARTIES LICENSING THIRD PARTY SOFTWARE TO YOKOGAWA.
4.6 Correction of nonconformity in the manner and for the period of time provided above shall be the Licensee’s sole and exclusive remedy for any failure of Yokogawa to comply with its obligations and shall constitute fulfillment of all liabilities of Yokogawa and any third party licensing the Third Party Software to Yokogawa (including any liability for direct, indirect, special, incidental or consequential damages) whether in warranty, contract, tort (including negligence but excluding willful conduct or gross negligence by Yokogawa) or otherwise with respect to or arising out of the use of the Licensed Software.

Article 5 (Infringement)

5.1 If and when any third party should demand injunction, initiate a law suit, or demand compensation for damages against the Licensee under patent right (including utility model right, design patent, and trade mark), copy right, and any other rights relating to any of the Licensed Software, the Licensee shall notify Yokogawa in writing to that effect without delay.

5.2 In the case of the preceding Paragraph 5.1, the Licensee shall assign to Yokogawa all of the rights to defend the Licensee and to negotiate with the claiming party. Furthermore, the Licensee shall provide Yokogawa with necessary information or any other assistance for Yokogawa's defense and negotiation. If and when such a claim should be attributable to Yokogawa, subject to the written notice to Yokogawa stated in the preceding Paragraph 5.1, Yokogawa shall defend the Licensee and negotiate with the claiming party at Yokogawa's cost and expense and be responsible for the final settlement or judgment granted to the claiming party in the preceding Paragraph 5.1.

5.3 When any assertion or allegation of the infringement of the third party's rights defined in Paragraph 5.1 is made, or when at Yokogawa's judgment there is possibility of such assertion or allegation, Yokogawa will, at its own discretion, take any of the following countermeasures at Yokogawa's cost and expense.

a) To acquire the necessary right from a third party which has lawful ownership of the right so that the Licensee will be able to continue to use the Licensed Software;
b) To replace the Licensed Software with an alternative one which avoids the infringement; or
c) To remodel the Licensed Software so that the Licensed Software can avoid the infringement of such third party's right.

5.4 If and when Yokogawa fails to take either of the countermeasures as set forth in the preceding subparagraphs of Paragraph 5.3, Yokogawa shall indemnify the Licensee only by paying back the price amount of the Licensed Software which Yokogawa has received from the Licensee. THE FOREGOING PARAGRAPHS STATE THE ENTIRE LIABILITY OF YOKOGAWA AND ANY THIRD PARTY LICENSING THIRD PARTY SOFTWARE TO YOKOGAWA WITH RESPECT TO INFRINGEMENT OF THE INTELLECTUAL PROPERTY RIGHTS INCLUDING BUT NOT LIMITED TO, PATENT AND COPYRIGHT.

Article 6 (Liabilities)

6.1 If and when the Licensee should incur any damage relating to or arising out of the Licensed Software or service that Yokogawa has provided to the Licensee under the conditions herein due to a reason attributable to Yokogawa, Yokogawa shall take actions in accordance with this Agreement. However, in no event shall Yokogawa be liable or responsible for any special, incidental, consequential and/or indirect damage, whether in contract, warranty, tort, negligence, strict liability, or otherwise, including, without limitation, loss of operational profit or revenue, loss of use of the Licensed Software, or any associated products or equipment, cost of capital, loss or cost of interruption of the Licensee's business, substitute equipment, facilities or services, downtime costs, delays, and loss of business information, or claims of customers of Licensee or other third parties for such or other damages. Even if Yokogawa is liable or responsible for the damages attributable to Yokogawa and to the extent of this Article 6, Yokogawa's liability for the Licensee's damage shall not exceed the price amount of the Licensed Software or service fee which Yokogawa has received. Please note that Yokogawa shall be released or discharged from part or all of the liability under this Agreement if the Licensee modifies, remodels, combines with other software or products, or causes any deviation from the basic specifications or functional specifications, without Yokogawa's prior written consent.

6.2 All causes of action against Yokogawa arising out of or relating to this Agreement or the performance or breach hereof shall expire unless Yokogawa is notified of the claim within one (1) year of its occurrence.

6.3 In no event, regardless of cause, shall Yokogawa assume responsibility for or be liable for penalties or penalty clauses in any contracts between the Licensee and its customers.

Article 7 (Limit of Export)

Unless otherwise agreed by Yokogawa, the Licensee shall not directly or indirectly export or transfer the Licensed Software to any countries other than those where Yokogawa permits export in advance.

Article 8 (Term)

This Agreement shall become effective on the date when the Licensee receives the Licensed Software and continues in effect unless or until terminated as provided herein, or the Licensee ceases using the Licensed Software by itself or with Yokogawa's thirty (30) days prior written notice to the Licensee.

Article 9 (Injunction for Use)

During the term of this Agreement, Yokogawa may, at its own discretion, demand injunction against the Licensee in case that Yokogawa deems that the Licensed Software is used improperly or under severer environments other than those where Yokogawa has first approved, or any other condition which Yokogawa may not permit.

Article 10 (Termination)

Yokogawa, at its sole discretion, may terminate this Agreement without any notice or reminder to the Licensee if the Licensee violates or fails to perform this Agreement. However, Articles 5, 6, and 11 shall survive even after the termination.

Article 11 (Jurisdiction)

Any dispute, controversies, or differences between the parties hereto as to interpretation or execution of this Agreement shall be resolved amicably through negotiation between the parties upon the basis of mutual trust. Should the parties fail to agree within ninety (90) days after notice from one of the parties to the other, both parties hereby irrevocably submit to the exclusive jurisdiction of the Tokyo District Court (main office) in Japan for settlement of the dispute.

Article 12 (Governing Law)

This Agreement shall be governed by and construed in accordance with the laws of Japan. The Licensee expressly agrees to waive absolutely and irrevocably and to the fullest extent permissible under applicable law any rights against the laws of Japan which it may have pursuant to the Licensee's local law.

Article 13 (Severability)

In the event that any provision hereof is declared or found to be illegal by any court or tribunal of competent jurisdiction, such provision shall be null and void with respect to the jurisdiction of that court or tribunal and all the remaining provisions hereof shall remain in full force and effect.

Contents

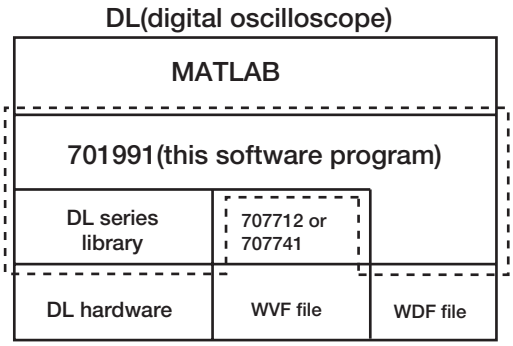
Index

Product Overview

This software program consists of MEX-Functions for DL control, MEX-Functions for WVF/WDF files, and the DL series library.
MEX-Functions for DL control are a group of functions used to control Yokogawa's digital oscilloscopes (DL) and acquire data. The functions can be used on MATLAB to change parameters on the DL such as the measurement range and to load the data from the DL into a MATLAB matrix. In addition, the GUI panel which imitates the DL operation panel can be used to easily control the DL and acquire waveform data.
The DL series library is used when the DL is accessed with the MEX-Function for DL control described above.
MEX-Functions for WVF/WDF files are a group of functions used to access from the MATLAB environment the waveform data (.wvf, and .wdf extension) that has been saved with the DL. Of these two, accessing WVF files saved by the DL using the MEX-Functions for WVF files requires either the WVF File Access API (model 707712) or the WE Control API (model 707741, includes 707712). If you do not have either of these APIs, please purchase the 707712 separately.

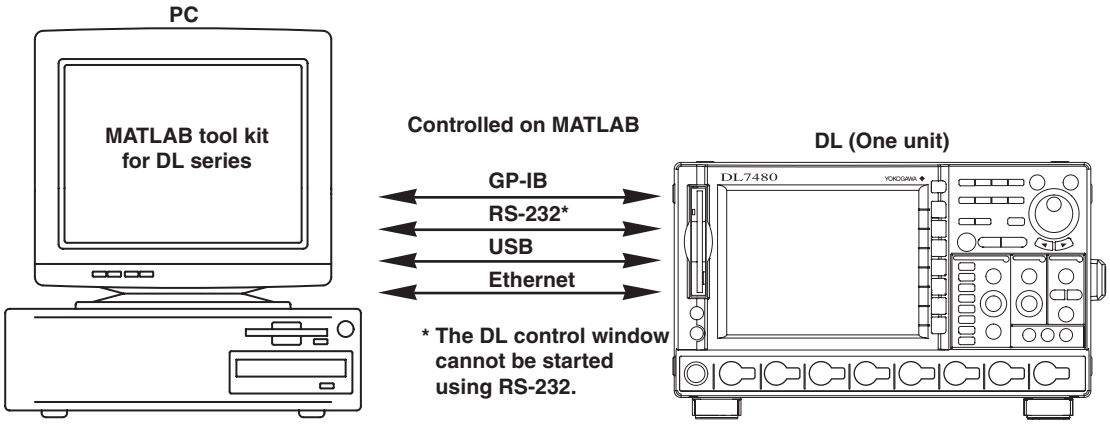
Note

This software program cannot load data that has been acquired with the logic input on the DL.

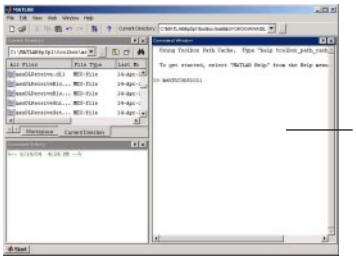


The WVF File Access API (model 707712) is required to access WVF files. Note that the WE Control API (model 707741) includes 707712.

This software program connects the PC and the DL and controls the DL.
There are two method of controlling the DL. One method is to use the MEX-Functions for DL control or the DL communication commands on MATLAB. The other method is to display the DL control window (GUI) on MATLAB and control the DL visually on a software control panel.



Window on MATLAB used to control the DL

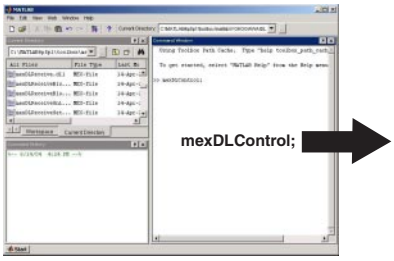


Control the DL using the MEX-Functions for DL control or DL communication commands. Or, retrieve waveform data or history data.

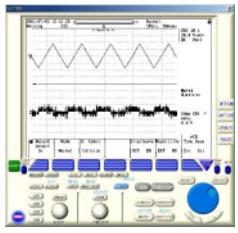
DL control window (GUI) started from MATLAB.

Control the DL visually on a software control panel

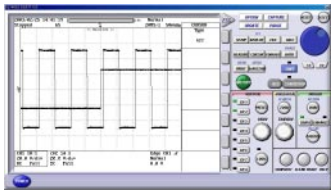
MATLAB window



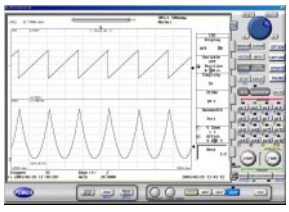
DL1600/DL1700E Series



DL7400 Series



DL750/DL750P



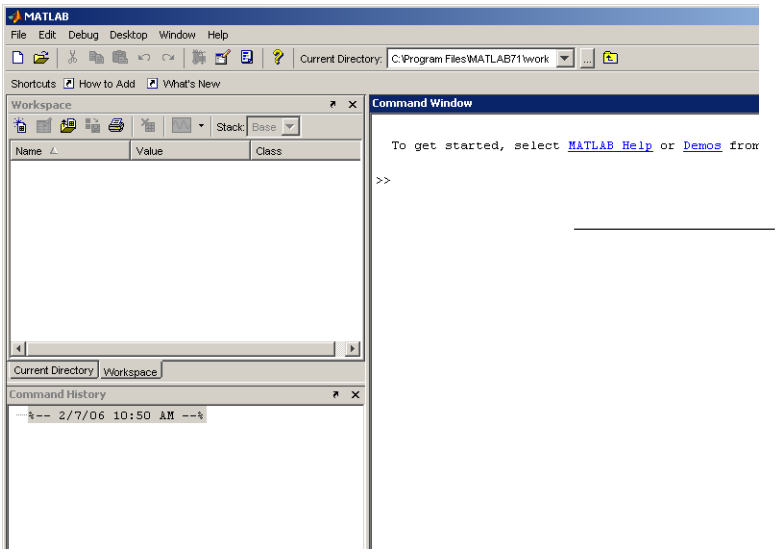
DL9000 Series

SL1400

Controlling the DL on MATLAB

The DL is controlled by using the MEX-Functions for DL control or the DL communication commands from the MATLAB command window. You can use the MEX-Functions for DL control to change the record length setting or save waveform data in a format that can be displayed on MATLAB. For a description of the MEX-Functions for DL control, see section 4.1, “MEX-Functions for DL Control.” For a description of the DL communication commands, see the Communication Interface User’s Manual for the respective DL.

MATLAB Startup Window



Command Window
Control the DL by entering the MEX-Functions for DL control or DL communication commands.

Controlling the DL Using DL Communication Commands

You can enter DL communication commands on the MATLAB Command Window to control the DL. The syntax is shown below. Enter the DL communication command in the Msg parameter of the MEX-Functions for DL control.

Syntax `>mexDLSend(Msg);`

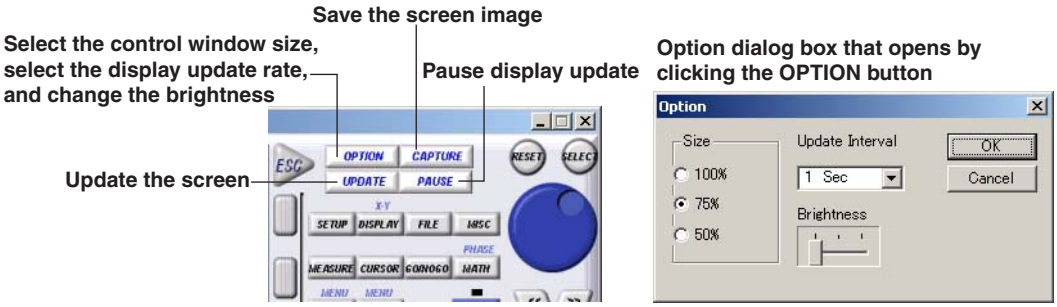
Example `>mexDLSend(':CHANNEL1:DISPLAY 1')`

Starting the Control Window Using MATLAB and Controlling the DL

Use MATLAB to open the front panel image of the DL (control window, see the next page). You can control the DL from your PC as if you are using the DL panel keys.

For the DL1600, DL1700E, and DL7400 series, and the DL750, DL750P, and SL1400

Control Window Example



CAPTURE (Save the Screen Image)

The waveform screen image on the PC can be saved as actual waveform data that is readable from MATLAB. The image can be saved to a file in MAT or CSV format.

OPTION (Select the Control Window Size)

You can select the size of the control window that is displayed on your PC from the list of choices below. When the display resolution of the PC is small, the control window can be displayed reduced in size.

- 100%: Displays the screen image of the DL digital oscilloscope using the same number of pixels as the number of pixels of the entire screen of the connected oscilloscope.
- 75%: Displays the screen image of the DL digital oscilloscope using 75% of the number of pixels of the entire screen of the connected oscilloscope.
- 50%: Displays the screen image of the DL digital oscilloscope using 50% of the number of pixels of the entire screen of the connected oscilloscope.

OPTION (Select the Display Update Rate)

You can select the display update rate of the DL screen image from the following:
1 s to 1 hour, or minimum (fastest update rate on your system. Note that this setting may place heavy load on the network.)
However, the actual display update rate may be slower than the specified update rate depending on the network transmission system or the amount of communication load.

OPTION (Change the Brightness)

You can change the display brightness of the screen image.

UPDATE (Update the Screen)

You can manually update the screen image of the DL. This is used when the display update rate is set to a low value or when the display update is paused.

PAUSE (Pause Display Update)

You can pause the display update operation.
Pausing the display update operation improves the response of the software program such as when turning ON/OFF numerous items at once or when entering values from a keyboard.

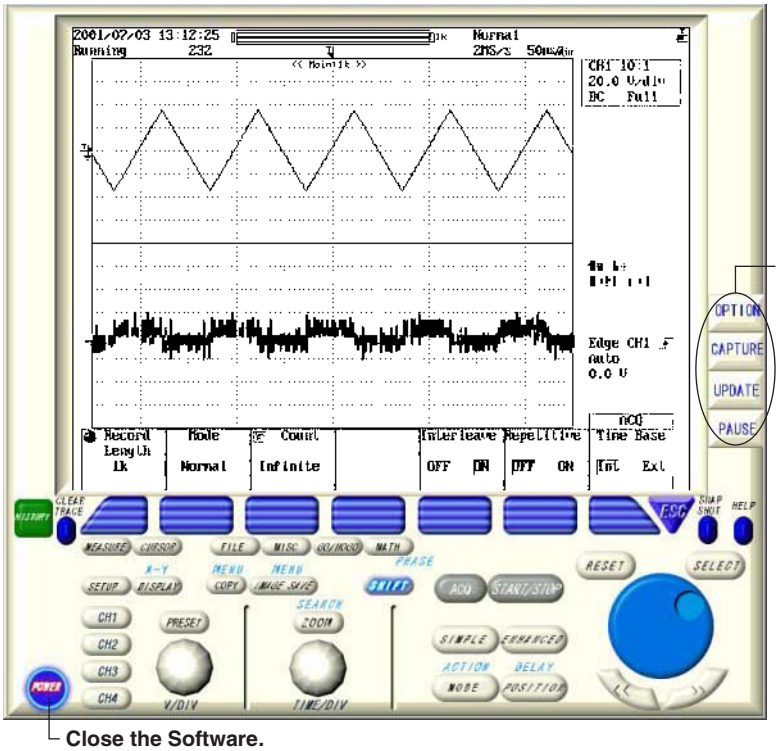
Control Window

Environment Settings Keys

You can enter environmental settings using the keys below. These keys do not exist on the DL.

- OPTION: Set the window size, display update rate, and brightness
- CAPTURE: Save the waveform screen image data
- UPDATE: Execute display update
- PAUSE: Pause display update

- **DL1600/DL1700E Series Digital Oscilloscopes**

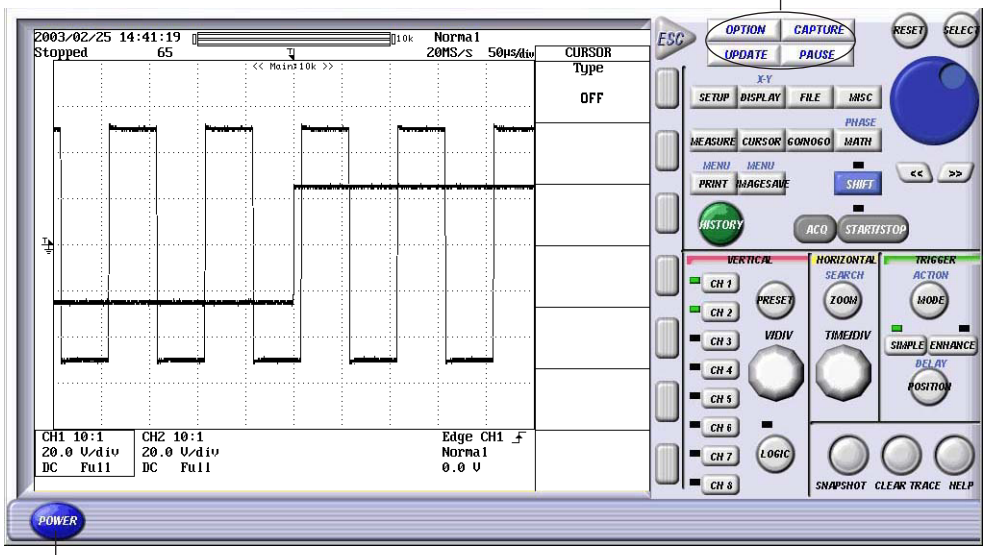


Environment settings keys (Keys not available on the DL1600/DL1700E Series)

Close the Software.

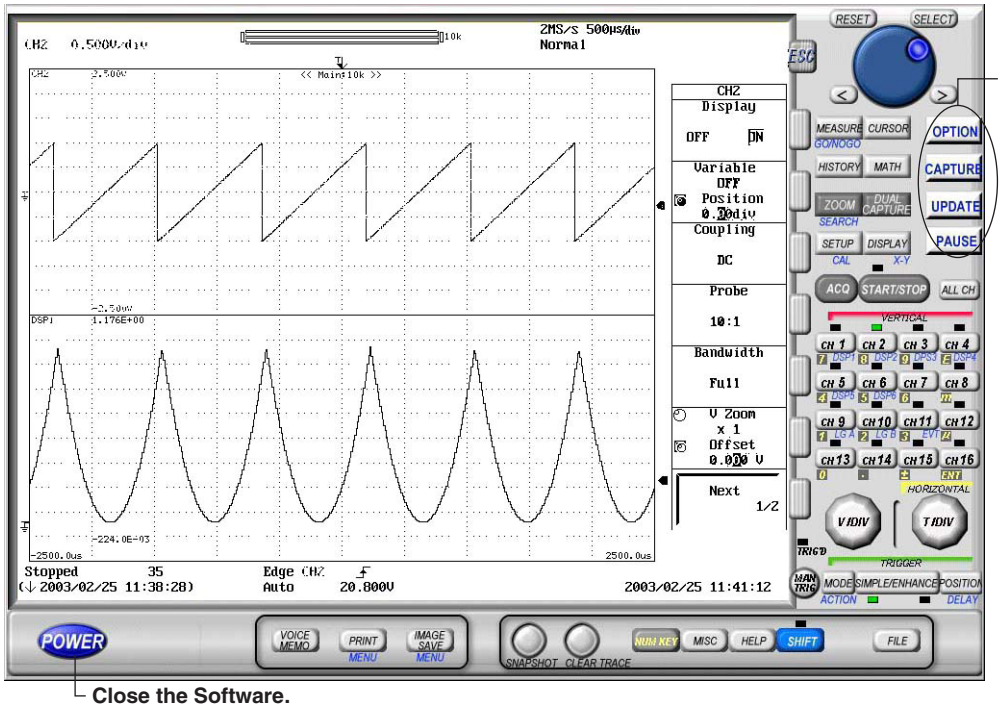
- **DL7400 Series Digital Oscilloscopes**

Environment settings keys (Keys not available on the DL7400 Series)

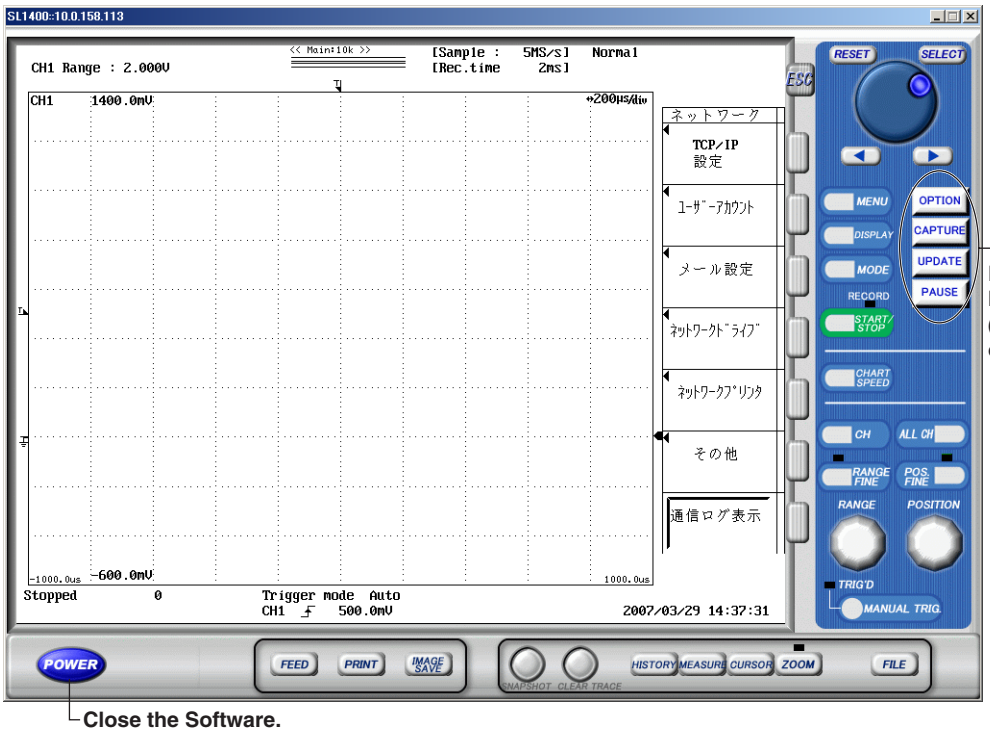


Close the Software.

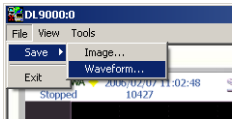
- **DL750/DL750P ScopeCorder**

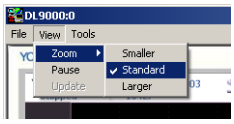


- **SL1400 ScopeCorder LITE**

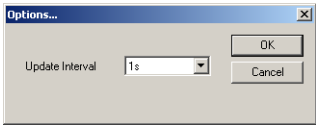


For the DL9000 Series





Option dialog box



File > Save > Image (Save the Screen Image)
An image of the currently displayed screen is saved as a BMP-format file.

File > Save > Waveform (Save the Wave Data)
The waveform screen image on the PC can be saved as actual waveform data that is readable from MATLAB. The image can be saved to a file in MAT or CSV format.

File > Exit (Close the Software)
The software is closed.

View > Zoom > Smaller, Standard, or Larger (Select the Control Window Size)
You can select the size of the control window that is displayed on your PC from the following.
Smaller, Standard, Larger

View > Pause (Pause Display Update)
You can pause the display update operation.
Pausing the display update operation improves the response of the software program such as when turning numerous items ON/OFF at once or when entering values from a keyboard.
To restart display updating, choose **View > Pause** again.

View > Update (Update the Screen)
You can manually update the screen image of the control window. This is used when the display update interval is set to a low value or when the display update is paused.

Tools > Option (Select the Display Update Interval)
You can select a display update interval for the control window from the following.
300 ms, 500 ms, 1 s, 2 s, 5 s, 10 s
However, the actual display update interval may be slower than the specified update interval depending on the network transmission system or the amount of communication load.

PC System Requirements

- **OS**
  Windows 2000 (Service Pack 4), Windows XP (Service Pack 2), or Windows Vista

- **CPU**
  Intel Pentium IV or higher

- **Memory**
  512 MB or more (1024 MB or more recommended)

- **CRT**
  XGA or higher (SXGA or higher recommended)

- **Color**
  256 colors or more (65536 colors (16 bpp) ore more recommended)

- **Mouse**
  A mouse compatible with Windows 2000 (Service Pack 4), Windows XP (Service Pack 2), or Windows Vista.

- **Communication Interface**
  GP-IB, Ethernet, USB, or RS-232.
  However, RS-232 can be used only to carry out communications using the MEX-Functions for DL control on MATLAB.

- **Controllable DL Series Oscilloscopes**
  DL1600 Series
  DL1700E Series
  DL7400 Series
  DL750, DL750P, and SL1400
  DL750, DL750P, and SL1400 module

Model	Note
701250	High-Speed 10 MS/s, 12-Bit Isolation Module (2CH)
701251	High-Speed High-Resolution 1 MS/s, 16-Bit Isolation Module (2CH)
701255	High-Speed 10 MS/s, 12-Bit Non-Isolation Module (2CH)
701260	High-Voltage 100 kS/s, 16-Bit Isolation Module (with RMS, 2CH)
701261	Universal (Voltage/Temp.) Module (2CH)
701262	Universal (Voltage/Temp.) Module (with AAF, 2CH)
701265	Temperature, High Precision Voltage Isolation Module (2CH)
701270	Strain Module (NDIS, 2CH)
701271	Strain Module (DSUB, Shunt-Cal, 2CH)
701275	Acceleration/Voltage Module (with AAF, 2CH)
701280	Frequency Module (2CH)

  DL9000 Series

- **MATLAB**
  R2007a (Ver. 7.4.0) or later

- **Communication Function Necessary on the DL (One of the Following)**

  GP-IB: A Yokogawa product with GP-IB complying with IEEE St'd 488.2

Note

When performing communication using a Yokogawa product, set the terminator to LF and EOI for normal operation and EOI for binary data transmission.

RS-232: RS-232 can be used only to carry out communications using the MEX-Functions for DL control on MATLAB.

Note

Normally, set the parameters as follows:

- 8 bits, no parity, 1 stop bit
- CTS-RTS (hardware handshaking)
- Terminator: LF

USB: Yokogawa DL1600, DL1700E, DL7400, DL750, DL750P, SL1400, or DL9000 Digital Oscilloscope with a USB interface.

Note

Set the terminator to LF and EOI or EOI. Do not turn OFF the power to the PC or the DL when the line is connected.

Ethernet: DL1600, DL1700E, DL7400, DL750, DL750P, SL1400, or DL9000 with an Ethernet interface.

- **Others**

  CD-ROM drive (for installation)

Notes on Using the Software

- Do not perform operations directly on the DL Series Digital Oscilloscope while using this software program. If you do, operation errors can result.
- If the standby mode provided on your PC is activated, the operation of the software may not be able to continue.
  When using the software, turn OFF the standby mode.
- If you run the software using a NIC interface, the line load is 800 KB/s maximum and 400 KB/s or less in normal conditions.
  Consult your network administrator on using the NIC interface.
- Do not set the network or communication parameters of the DL Series Digital Oscilloscope using this software program. The connection may be disconnected.
- Do not execute self-tests using this software program.
- Only a single DL Series Digital Oscilloscope can be controlled by this software program. In addition, simultaneous connections from multiple PCs to a single DL Series Digital Oscilloscope are not allowed.
- You can run multiple instances of the software program on a single PC to control multiple DL Series Digital Oscilloscopes. However, the operation may slow down depending on the specifications of your PC or the line condition. In addition, the program may not operate properly when multiple instances of this program are started depending on the CPU or memory size of your PC.
- The thumbnail preview function of the DL1600/1700E Series Digital Oscilloscopes is not supported. The thumbnail function and preview function of the DL7400 Series Digital Oscilloscopes are not supported.
- If a connection error occurs when connecting to a DL digital oscilloscope, power-cycle the DL.

1.1 Installing the MATLAB Tool Kit for the DL Series

For the installation procedure of the MATLAB tool kit for DL series, read the paper named *Please Read before Installation (MATLAB tool kit for DL series) (IM701991-71E)* that comes with the software.

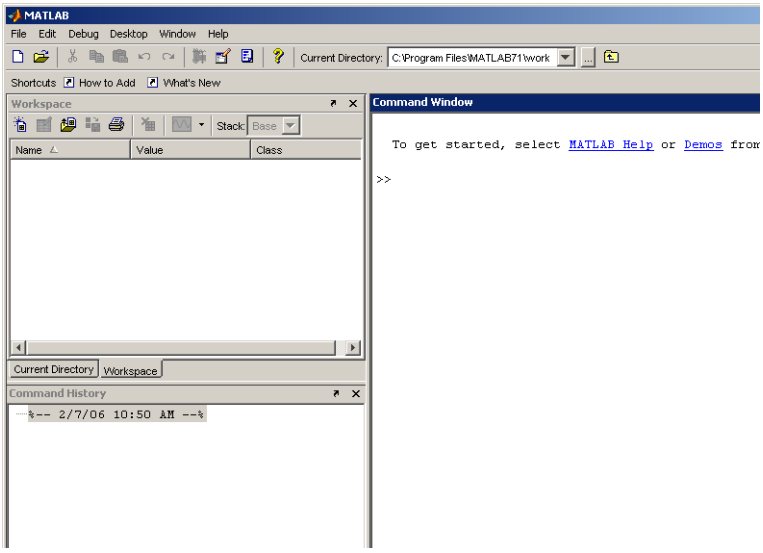
Note

When you install this software program, the MEX-Functions for DL control (see section 4.1) and the MEX-Functions for WVF/WDF (see section 4.2, and 4.3) are also installed.

1.2 Initializing MATLAB

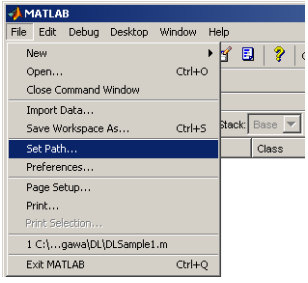
Carry out the procedure below once after installing this software.

1. Start MATLAB.



Setting the MATLAB Path

2. From the **File** menu, choose **Set Path**. The Set Path dialog box opens.

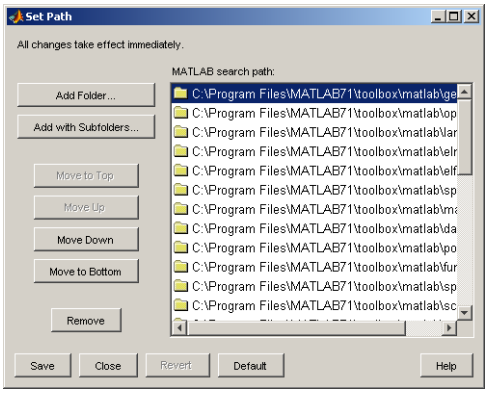


3. Click the **Add Folder** button and select the folder in which the MATLAB tool kit for DL series was installed.

Note

The default installation destination folders are as follows.
C:\Program Files\Yokogawa\matlab\dl

4. Click the **Save** button and then the **Close** button.

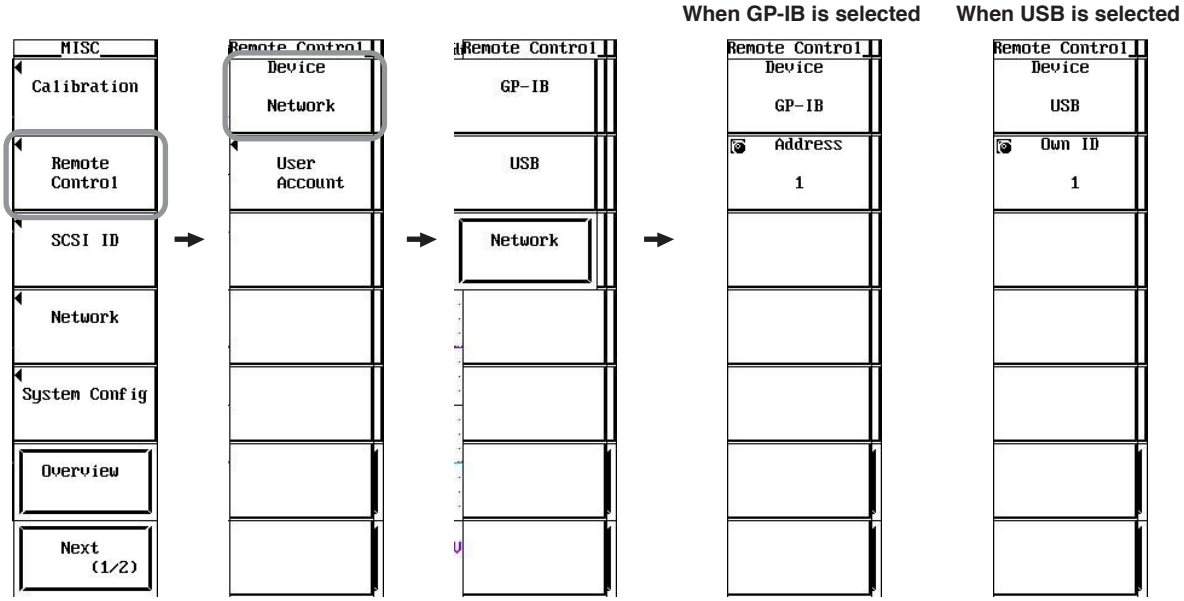


2.1 Setting the DL Series Interface

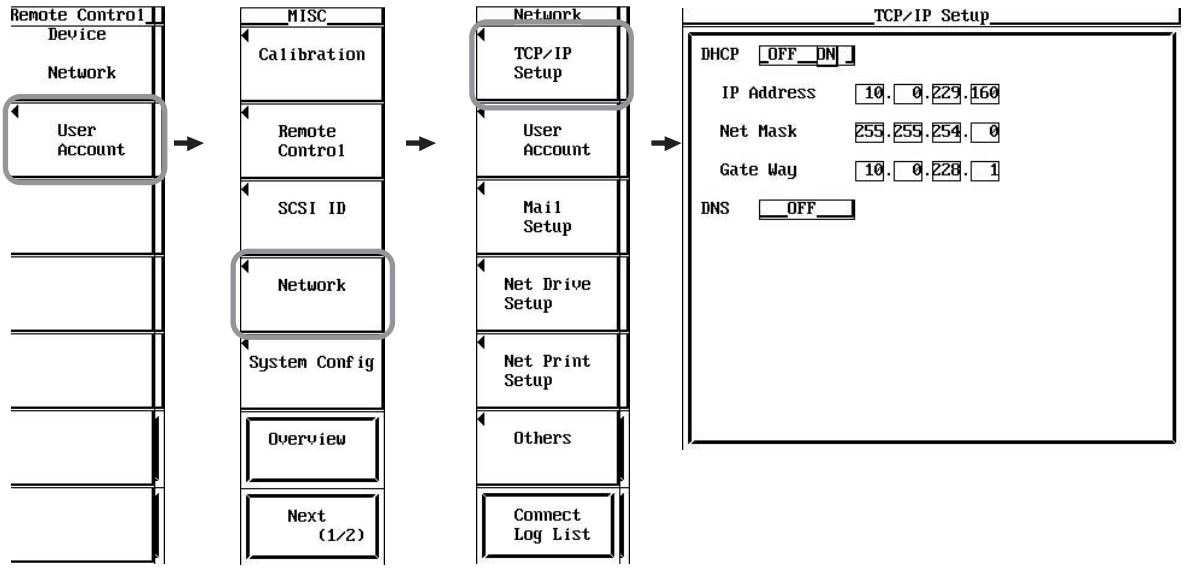
When you install this software program, the MEX-Functions for DL control are also installed. You will be able to control the DL Series oscilloscopes using the MEX-Functions for DL control by carrying out the setup below.

1. Set the interface to be used from the front panel of the DL.
   On the DL1600/DL1700E Series, DL750, and DL750P:
   MISC > Remote Cntl > Device
   On the DL7400 Series: MISC > Remote Control > Device
   On the DL9000 Series: SYSTEM > Remote Control > Device
   On the SL1400: MENU > Next > Remote Cntl > Device

   - **Example (DL7480)**



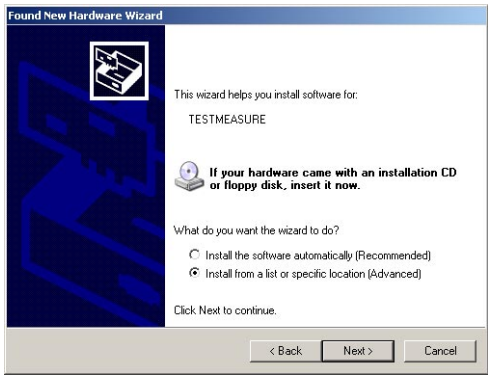
When Network is selected



2.2 Installing the USB Driver

For the DL1600, DL1700E, and DL7400 series, and the DL750, DL750P, and SL1400, when connecting a PC and the DL main unit via USB for the first time, the Found New Hardware Wizard dialog box appears.
For the DL9000 series, see IM701310-91E contained in the CD-ROM.

1. Select the **Install from a list of specific location**.
2. Click **Next**.

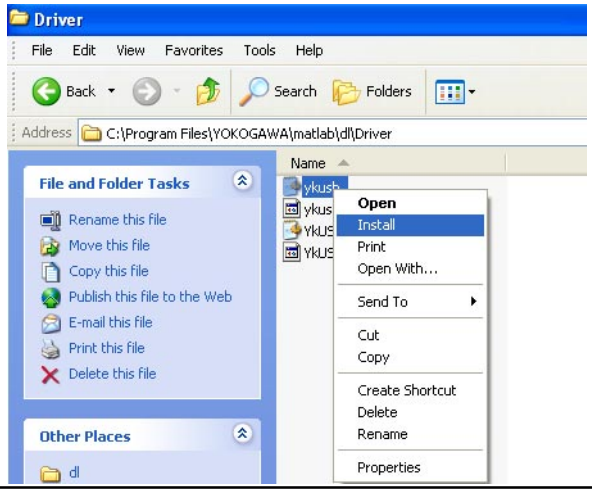


3. Set the following items:
   - Select the **Search for the best driver in these locations**.
   - Clear the **Search removable media** check box.
   - Select the **Include this location in the search** check box.
4. Check that the folder name is as follows. If you installed the software program to a folder other than the default folder, click **Browse** and specify the \Driver folder under the installed location.
   C:\Program Files\Yokogawa\matlab\dl\Driver
5. Click **OK**. The software is installed.

Note

To install the driver manually, right-click the ykusb.inf file in the folder below, and click **Install**. If you installed MATLAB to a folder other than the default folder, the file is located in the \Driver folder in that location.

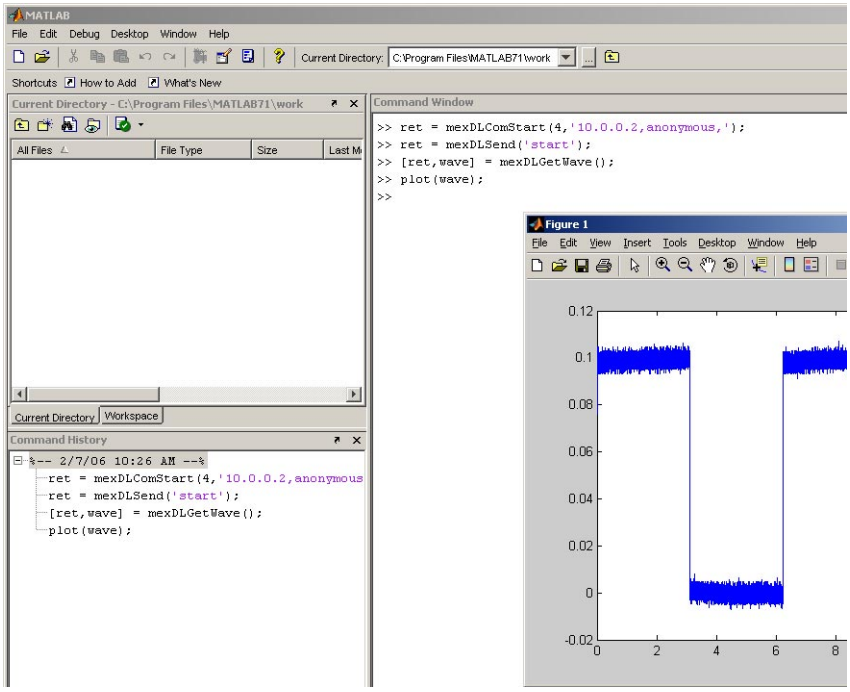
C:\Program Files\Yokogawa\matlab\dl\Driver



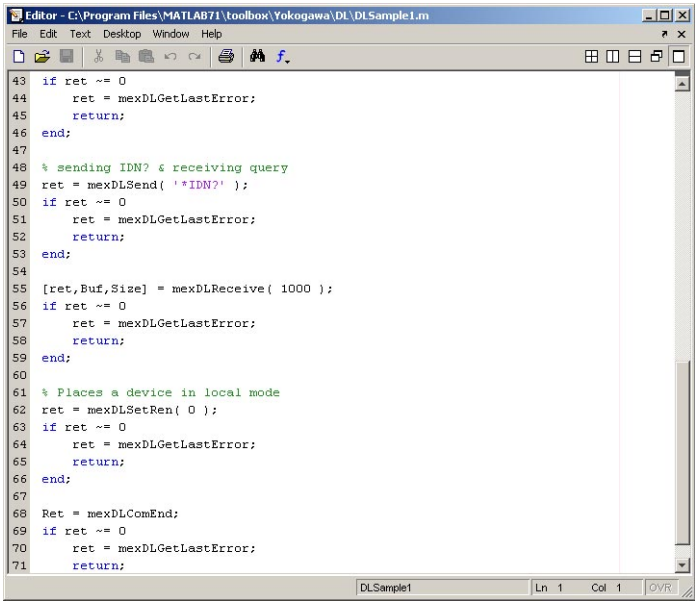
2.3 Executing MEX-Functions for DL Control

The MEX-Functions for DL control are programmed interactively, which is a feature of MATLAB. The DL can be controlled and waveform data can be retrieved into MATLAB by executing a program on the MATLAB command window or through an m file.

Command window



m file editor



Execution Example on the Command Window

```
>> ret = mexDLComStart(4,'10.0.236.100,anonymous,pass');
>> ret = mexDL ( 'STOP' );
>> ret = mexDLSend( 'CHANNEL:VDIV 500mV' );
>> ret = mexDLSend( 'ACQUIRE:MODE NORMAL; RLENGTH 1000' );
>> ret = mexDLComEnd;
```

2.4 Sample Programs Using the MEX-Functions for DL Control

Each sample program is located in the folder in which the MATLAB Tool Kit of DL Series was installed.

Sample Program 1

Below is a program for connecting the communication line and sending commands.
Remove the % character from the program line corresponding to the interface type you are using.
The sample program is DLSample1.m.
Example `[ret, Buf, Size] = DLSample1`

```
function [ret, Buf, Size] = DLSample1()
% DL Sample 1
%
% Basic communication verification.
% Open communication > inquire ID > close communication
%
% Choose an interface type by removing '%' in front of ret
% from the following lines ( between line no. 13 and 20 )
% You may need to adjust parameters of the interface.

% Example-1:GPIB  ( address = 1 )
% ret = mexDLComStart(1,'1');
% Example-2:RS232 ( COM1,19200,8Bit-NoParity-1StopBit,NO-NO )
% ret = mexDLComStart(2,'1,4,0,0');
% Example-3:USB   ( ID = 1 )
% ret = mexDLComStart(3,'1');
% Example-4:Ethernet ( address=10.0.100.100, name=anonymous, password=123 ) % ret =
mexDLComStart(4,'10.0.100.100,anonymous,123');

% if ret ~= 0
%    ret = mexDLGetLastError;
%    return;
% end;

% set terminal
ret = mexDLSetTerm( 2, 1 );
if ret ~= 0
    ret = mexDLGetLastError;
    return;
end;

% set timeout
ret = mexDLSetTimeout( 300 );
if ret ~= 0
    ret = mexDLGetLastError;
    return;
end;

% Places a device in remote mode
ret = mexDLSetRen( 1 );
if ret ~= 0
    ret = mexDLGetLastError;
    return;
end;

% sending IDN? & receiving query
ret = mexDLSend( '*IDN?' );
if ret ~= 0
    ret = mexDLGetLastError;
    return;
end;

[ret,Buf,Size] = mexDLReceive( 1000 );
if ret ~= 0
    ret = mexDLGetLastError;
    return;
end;

% Places a device in local mode
ret = mexDLSetRen( 0 );
if ret ~= 0
    ret = mexDLGetLastError;
    return;
end;

Ret = mexDLComEnd;
if ret ~= 0
    ret = mexDLGetLastError;
    return;
end;
```

Sample Program 2

Below is a program for connecting the communication line, capturing the waveform data when a trigger is activated, and displaying the waveform of the data using the plot function.

Remove the % character from the program line corresponding to the interface type you are using.

The sample program is DLSample2.m.

Example `[ ret, WaveData ] = DLSample2`

```
function [ ret,WaveData ] = DLSample2()
%
% DL Sample 2
%
% Receive data as soon as triggering conditions are met.
% Captured data is stored into 'WaveData' matrix.
% Open communication > start signal acquisition >
%   transfer DL data to MATLAB matrix > close communication
%
% Choose an interface type by removing '%' in front of ret
% from the following lines ( between line no, 15 and 22 )
% You may need to adjust parameters of the interface.

% Example-1:GPIB  ( address = 1 )
% ret = mexDLComStart(1,'1');
% Example-2:RS232 ( COM1,19200,8Bit-NoParity-1StopBit,NO-NO )
% ret = mexDLComStart(2,'1,4,0,0');
% Example-3:USB   ( ID = 1 )
% ret = mexDLComStart(3,'1');
% Example-4:Ethernet ( address=10.0.100.100, name=anonymous, password=123 ) % ret =
mexDLComStart(4,'10.0.100.100,anonymous,123');

% if ret ~= 0
%    ret = mexDLGetLastError;
%    return;
% end;

% set terminal
ret = mexDLSetTerm( 2, 1 );
if ret ~= 0
    ret = mexDLGetLastError;
    return;
end;

% set timeout
ret = mexDLSetTimeout( 300 );
if ret ~= 0
    ret = mexDLGetLastError;
    return;
end;
WaveData = 0;

% Start signal acquisition
Ret = mexDLSend('sstart? 100');
if Ret ~= 0
    Ret = mexDLGetLastError;
    return;
end;

[Ret,Buf,Size] = mexDLReceive( 10 );
if Ret ~= 0
    Ret = mexDLGetLastError;
    return;
end;

% Receive DL data into WaveData matrix if a
% signal is triggered
if strcmp( deblank(Buf(1,:)), ':SST 0' ) == 1
    [Ret,WaveData] = mexDLGetWave;
    plot(WaveData);
    if Ret ~= 0
        Ret = mexDLGetLastError;
        return;
    end;
end;

% Set DL trigger mode to AUTO
mexDLSend(':TRIG:MODE AUTO');

%Close communication port
Ret = mexDLComEnd;
if ret ~= 0
    ret = mexDLGetLastError;
    return;
end;
```

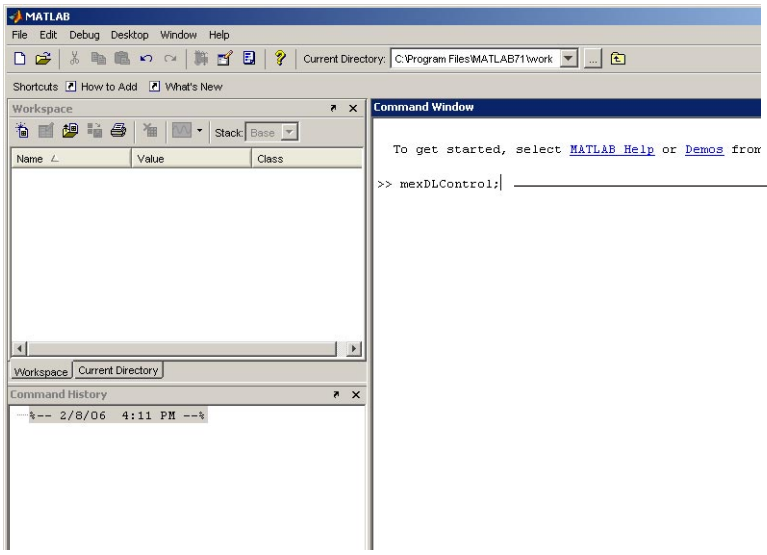
2.5 Starting/Closing the DL Control Window

Starting the DL Control Window

1. Start MATLAB.

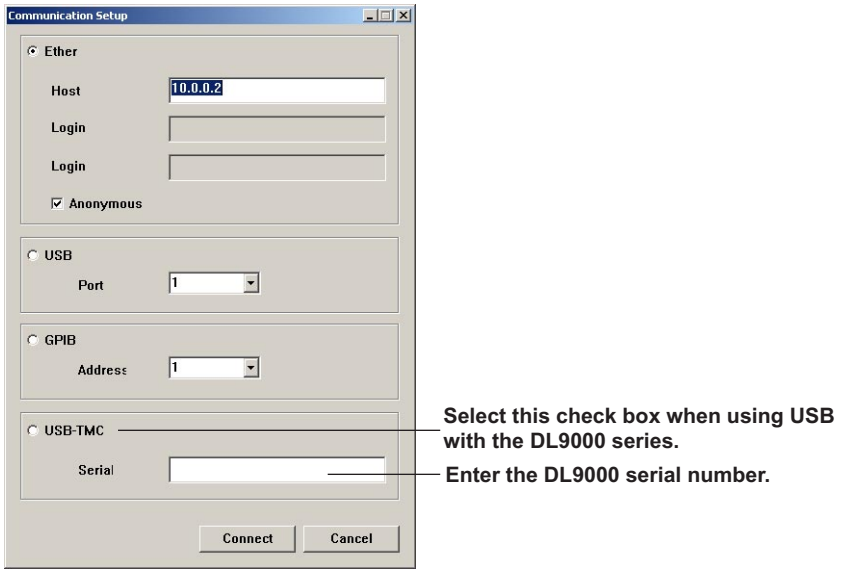
Entering the Command

2. Enter **`mexDLControl;`** in the Command Window. The Communication Setup dialog box opens.



Selecting the Interface

3. Select the interface to be used and set the necessary parameters according to the interface.
4. Click the **Connect** button. The DL control window opens.



Note

- For details on how to operate the DL, see the user's manual for the DL.
- If the line was opened using `mexDLComStart` of the MEX-Functions for DL control, the Communication Setup dialog box does not open. The DL control window (front panel image) appears.

Closing the DL Control Window

For the DL1600, DL1700E, and DL7400 series, and the DL750, DL750P, and SL1400

1. Click the **POWER** button at the lower left corner of the DL control window or the **X** button at the upper right corner of the window.
   The DL control window closes.

For the DL9000 Series

1. Choose **File** > **Exit** from the DL main unit control screen, or click the **X** button in the upper right of the screen.
   The DL control window closes.

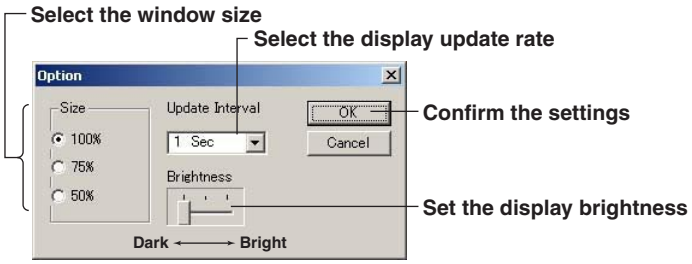
2.6 Operations on the DL Control Window

Gives an example of the operation using a DL1700E series instrument.

Environment Settings

Setting the Window Size, Display Update Rate, and Brightness

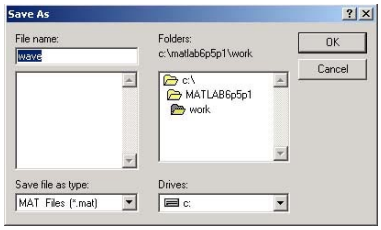
Click **OPTION** on the Control Window to display the following dialog box. Set the items as necessary.



Saving the Waveform Screen Image Data

Click **CAPTURE** on the Control Window. The Save As dialog box opens.
Specify the save destination, file name, and file type. Then, click **OK**. The screen image data is saved.
You can select from the following two file types.

- mat
- csv



Note

If you save the data to a MAT file (*.mat), the file can be displayed on MATLAB.

Updating the Display and Pausing the Display Update Operation

Click **PAUSE** to pause the updating of the screen image of the DL.
Click **UPDATE** to update the screen image of the DL.

Control Window Operation

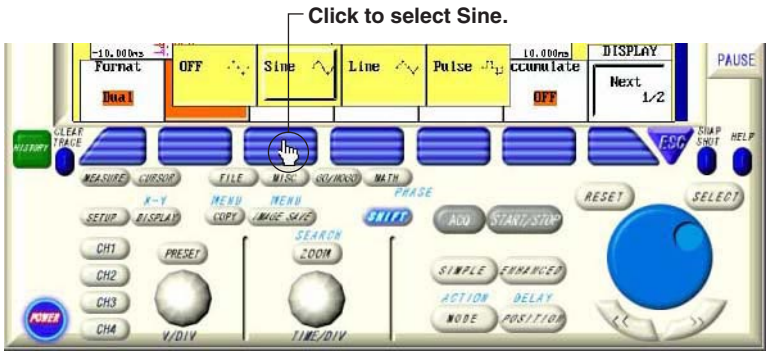
Using the Mouse

The displayed icon and the mouse operation vary depending on where the mouse pointer is located on the control window. The following table shows the mouse operation for each icon and the operation to the DL Series Digital Oscilloscopes.

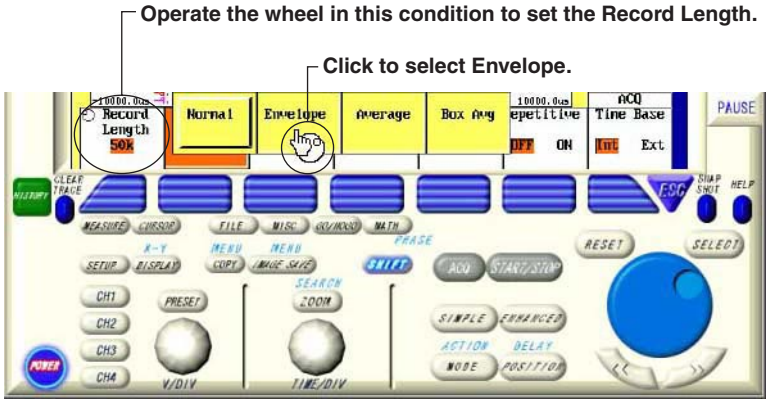
Mouse Pointer Position	Displayed Icon and Mouse Operation	Operation
Panel key		
	Click operation	Same as pressing the panel key
Soft key menu or		
dialog box	Click operation	Same as pressing the soft key or button
	Wheel operation	Same as turning the jog shuttle
Voltage axis information		
display area	Click operation	Same as pressing the CH key
	Wheel operation	Same as turning the V/DIV knob
Time axis information		
display area	Wheel operation	Same as turning the T/DIV knob
Area to the left or right of		
the jog shuttle	Click operation	Same as turning the jog shuttle to the left or right
	Wheel operation	Same as turning the jog shuttle
Area to the left or right of		
the V/DIV knob	Click operation	Same as turning the V/DIV knob to the left or right
	Wheel operation	Same as turning the V/DIV knob
Area to the left or right of		
the T/DIV knob	Click operation	Same as turning the T/DIV knob to the left or right
	Wheel operation	Same as turning the T/DIV knob

Example of Mouse Operation

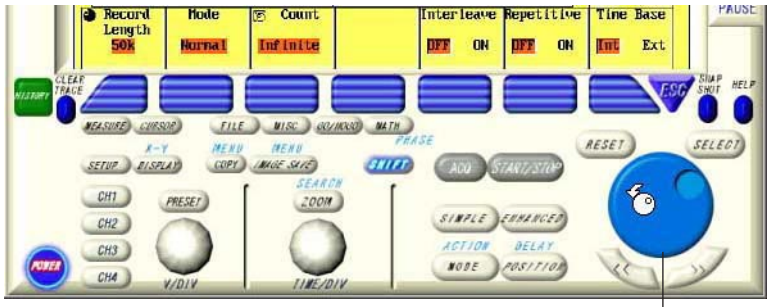
1. Panel Key Operation



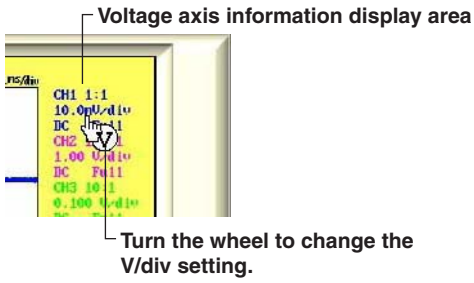
2. Soft Key Menu Operation



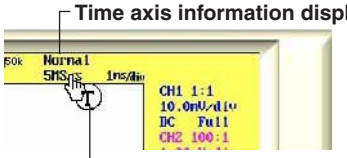
3. Jog Shuttle Operation



4. V/DIV Knob Operation



5. T/DIV Knob Operation



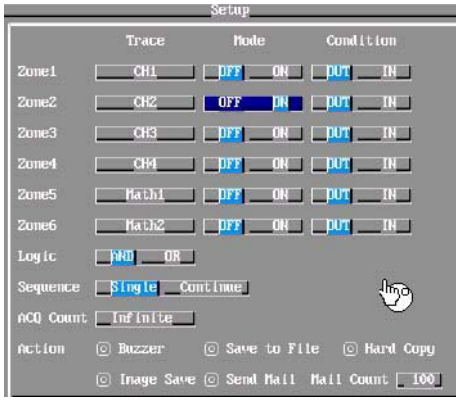
Time axis information display area

Turn the wheel to change the T/div setting.

Click or operate the wheel.

6. Dialog Box Operation

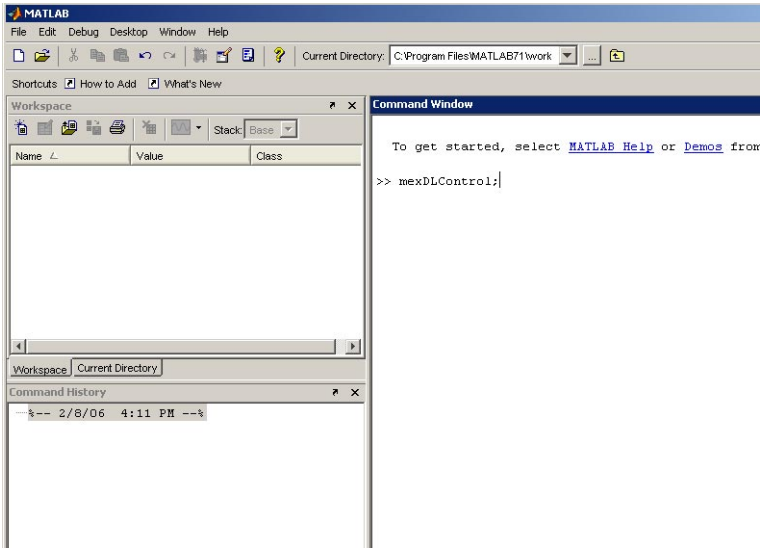
- Move to the item you wish to turn ON: Click the **jog shuttle** or perform wheel operation.
- Set ON/OFF: Click **SELECT** or click directly as shown in the following figure.



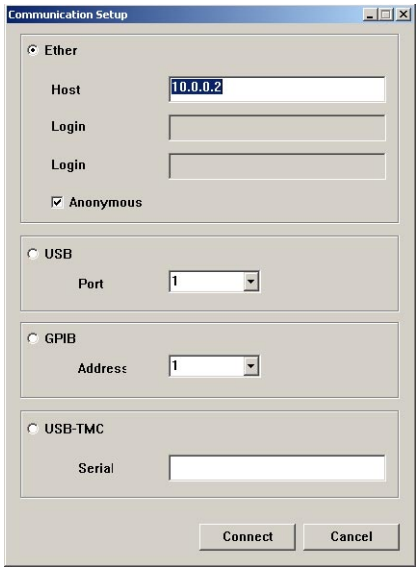
2.7 DL Control Example

Below is an example in which the MEX-Functions for DL control is used on MATLAB to save the waveform image data on the DL7480 and show the waveform on MATLAB.

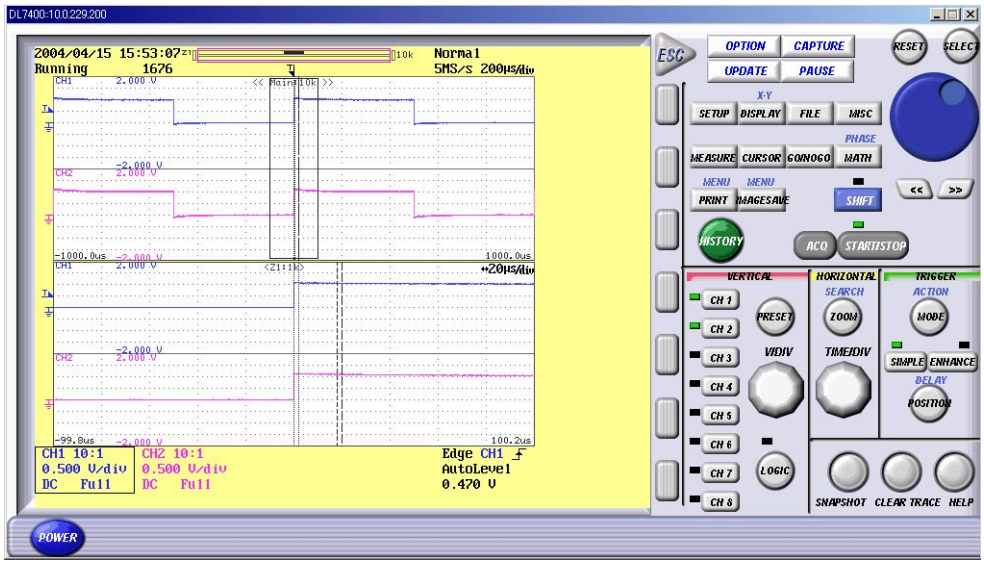
1. Enter **`mexDLControl;`** in the MATLAB Command Window.
   The Communication Setup dialog box opens



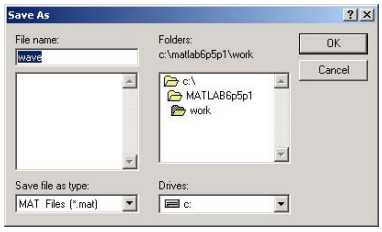
2. Enter the settings as follows:
   - Select the Ether check box.
   - Enter the IP address you checked on the DL screen in the Host Address box.
   - Select the Anonymous check box.
3. Click the **Connect** button.
   The DL7480 front panel image appears.



4. Click **CAPTURE**. The Save As dialog box opens.

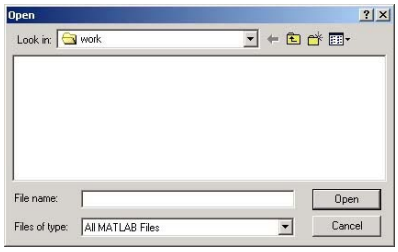


5. Set the file type to MAT (*.mat) and specify the file name and destination.
6. Click **OK**. The DL7480 waveform image data is saved.

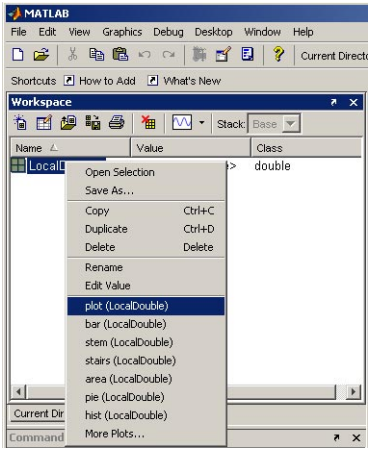


Showing the DL7480 Waveform Image Data on MATLAB

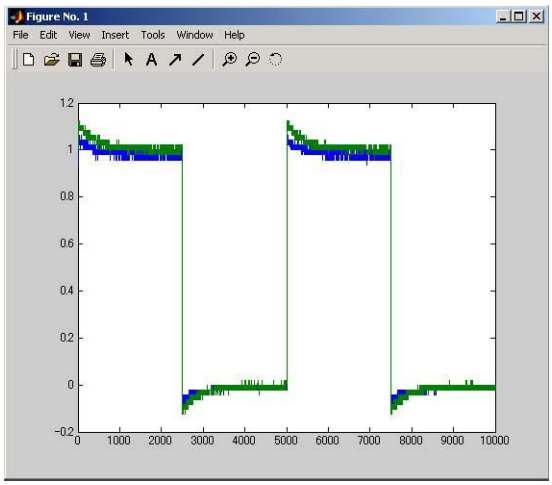
7. Open the MATLAB Workspace window.
8. From the **File** menu, choose **Open**. The Open dialog box opens.
9. Specify the data you wish to display on MATLAB, and click **Open**. The Workspace window shows the file name.



10. Right-click the file on the Workspace window and choose **plot**.



The waveform screen of the DL7480 appears.



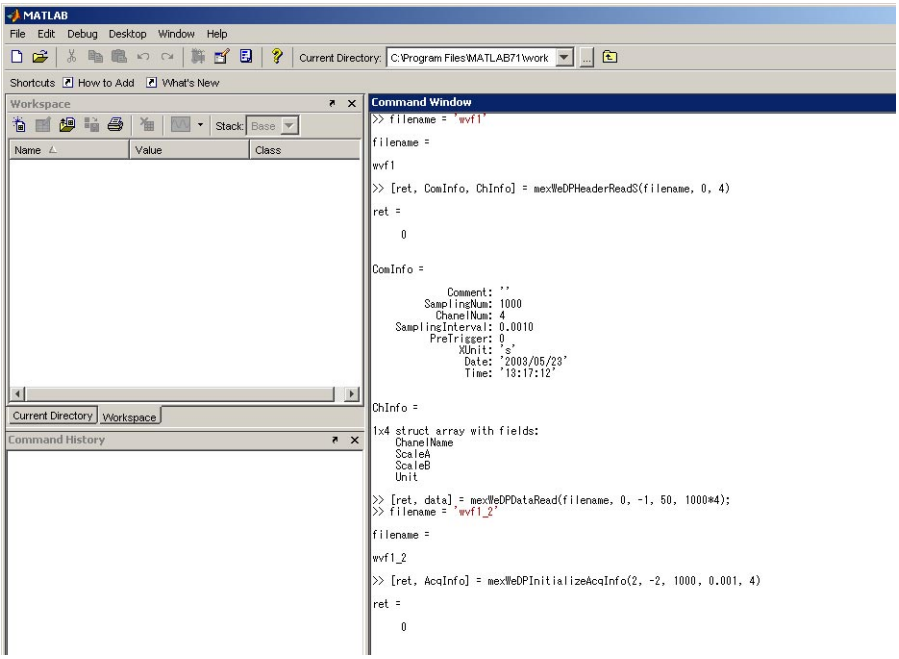
3.1 Using the MEX-Functions for Files

When you install this software program, the MEX-Functions for WVF/WDF files are also installed along with the MEX-Functions for DL control in the same folder. Of these two, using the MEX-Functions for WVF files requires that either the Model 707712 WVF File Access API (sold separately) or the Model 707741 WE Control API be installed.

3.2 Execution Example Using the MEX-Functions for Files

MEX-Functions for WVF/WDF files enables MATLAB-characteristic dialog-based programming. The following are examples of using the MATLAB command window and the MEX-Functions for WVF/WDF files in m files.

MATLAB Screen



Execution Example on the Command Window

```
>> filename = 'wvf1'
filename =
wvf1
>> [ret, ComInfo, ChInfo] = mexWeDPHeaderReadS(filename, 0, 4)
ret =
     0

ComInfo =

            Comment: ''
         SamplingNum: 1000
           ChanelNum: 4
    SamplingInterval: 0.0010
          PreTrigger: 0
               XUnit: 's'
                Date: '2003/05/23'
                Time: '13:17:12'

ChInfo =

1x4 struct array with fields:
    ChanelName
    ScaleA
    ScaleB
    Unit

>> [ret, data] = mexWeDPDataRead(filename, 0, -1, 50, 1000*4);
>> filename = 'wvf1_2'
filename =
wvf1_2
>> [ret, AcqInfo] = mexWeDPInitializeAcqInfo(2, -2, 1000, 0.001, 4)
ret =
     0
```

```
AcqInfo =

1x4 struct array with fields:
    channel
    dataType
    blockNum
    startBit
    effectiveBit
    trigActive
    record
    recordLen
    trigPosition
    time
    interval
    vResolution
    vOffset
    trigLevel
    trigWidth
    plusOverData
    minusOverData
    nonData
    dispMaxData
    dispMinData

>> ret = mexWeDPHeaderWriteS(filename, 0, ComInfo, 4, ChInfo, AcqInfo)
ret =
     0
>> ret = mexWeDPDataWrite(filename, 0, 1000, 4, AcqInfo, 50, data)
ret =
     0
>>
```

m-file Example

```
sourceFilename = 'wvf1';
destinationFilename = 'wvf1_dup';
blockNo = 0;
ch = -1;            % All channel
dataForm = 50;      % WE_DOUBLE
maxData = -Inf;
minData = Inf;

[ret, SampleNum, ChNum] = mexWeDPGetSampleChNum(sourceFilename, blockNo)
SampleNum = double(SampleNum);
ChNum = double(ChNum);
% Read the header file
[ret, ComInfo, ChInfo] = mexWeDPHeaderReadS(sourceFilename, blockNo, ChNum)
samplingInterval = ComInfo.SamplingInterval
% read the data file
[ret, data] = mexWeDPDataRead(sourceFilename, blockNo, ch, dataForm, SampleNum * ChNum);
for ch=1 : ChNum
    [ret, value] = mexWeDPHeaderItemRead(sourceFilename, 'VResolution', ch, blockNo)
    VRes = str2num(value)
    [ret, value] = mexWeDPHeaderItemRead(sourceFilename, 'VOffset', ch, blockNo)
    VOfs = str2num(value)
    for i=(ch-1)*SampleNum+1 : ch*SampleNum
        data(i) = data(i) * VRes + VOfs;    % Convert to the voltage values from the
file values.
    end
    [ret, value] = mexWeDPHeaderItemRead(sourceFilename, 'VMaxData', ch, blockNo)
    work = str2num(value) * VRes + VOfs;
    if maxData < work
        maxData = work
    end
    [ret, value] = mexWeDPHeaderItemRead(sourceFilename, 'VMinData', ch, blockNo)
    work = str2num(value) * VRes + VOfs;
    if minData > work
        minData = work
    end
end
plot(data(1:SampleNum))     % Display on graph
[ret, AcqInfo] = mexWeDPInitializeAcqInfo(maxData, minData, SampleNum, samplingInterval,
ChNum)
AcqInfo(1)
% Write the header file
ret = mexWeDPHeaderWriteS(destinationFilename, blockNo, ComInfo, ChNum, ChInfo, AcqInfo)
for ch=1 : ChNum
    [ret, VUnit] = mexWeDPHeaderItemRead(sourceFilename, 'VUnit', ch, blockNo)
    ret = mexWeDPHeaderItemWrite(destinationFilename, 'VUnit', ch, blockNo, VUnit)
end
% Write the data file
ret = mexWeDPDataWrite(destinationFilename, blockNo, SampleNum, ChNum, AcqInfo,
dataForm, data)
```

3.3 Sample Programs

Sample Program for WVF Files

Below is an example for retrieving the header information of a WVF file and displaying the waveform data using the plot function. Copy Sample.wvf and Sample.hdr to the current directory such MATLAB\work before using them.

The sample program is DLSample3.m.

Example `[ ret ] = DLSample3`

```
function [ ret ] = DLSample3()
%
% DL Sample 3
%
% Open 'Sample.wvf' file located in the current directory.
% Both 'Sample.wvf' and 'Sample.hdr' need to be in the directory.
% DL waveform data is stored into 'data' matrix.
% Either 707741 or 707712 is required to run this program.

file='Sample';
blockNo=0;
ch=-1;
dataForm=50;
dispNum = 0;

[ret,SampleNum,ChNum]=mexWeDPGetSampleChNum(file, blockNo);
SampleNum=double(SampleNum);
ChNum=double(ChNum);

for ch=1:ChNum
      [ret,value]=mexWeDPHeaderItemRead(file,'VDataType', ch, blockNo);
      % Skip data from logic input of the DL series
            if ~strncmp( value, 'B', 1 )
            dispNum = dispNum + 1;
            [ret, data]=mexWEDPDataRead(file,blockNo,ch,dataForm, SampleNum );
            [ret,value]=mexWeDPHeaderItemRead(file,'VResolution', ch, blockNo);
            Vres=str2num(value);
            [ret,value]=mexWeDPHeaderItemRead(file,'VOffset', ch, blockNo);
            Voff=str2num(value);
            wave = data*Vres+ones(1,SampleNum)*Voff;
            if 1 == ch
                  WaveData = wave;
            else
                  WaveData = [WaveData; wave] ;
            end
      end
end

plot( WaveData );
```

Sample Program for WDF Files

Below is an example for retrieving the header information of a WDF file and displaying the waveform data using the plot function.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%      M-File : WdfSample
%      WDF file access sample script- 4
%
%     Copyright (C) 2006 Yokogawa Electric Corporation
%     Software Japan. All rights reserved.
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
filename = input( 'filename = ', 's' );
[ ret, chNum ] = mexWdfGetChNum( filename );
block = 0;
for ch = 0 : double( chNum ) - 1
      %
      %       Get file parameters
      %
      [ret, traceName] = mexWdfItemRead(filename,'TraceName', ch, 0);
      [ret, vScaleUpper] = mexWdfItemRead(filename,'VScaleUpper', ch, block);
      [ret, vScaleLower] = mexWdfItemRead(filename,'VScaleLower', ch, block);
      [ret, hResolution] = mexWdfItemRead(filename,'HResolution', ch, block);
      [ret, hOffset] = mexWdfItemRead(filename,'HOffset', ch, block);
      [ret, vUnit] = mexWdfItemRead(filename,'VUnit', ch, block);
      [ret, hUnit] = mexWdfItemRead(filename,'HUnit', ch, block);
      %
      %      Get waveform data
      %
      clear data x;
      [ret, param, data] = mexWdfScaleDataRead(filename, ch, block);
      x(1 : param.cntOut) = [hOffset : hResolution : hResolution * ( param.cntOut
- 1) + hOffset];
      %
      %      Display waveform data to a graph
      %
      subplot(double(chNum), 1, ch + 1);
      plot(x, data);
      title(traceName);
      axis([x(1) x(param.cntOut) vScaleLower vScaleUpper]);
      ylabel(strcat('Amplitude [',vUnit,']'));
end
xlabel( strcat('Time [',hUnit,']'));
```

4.1 MEX-Functions for DL Control

You can control the DL or retrieve waveform data by entering MEX-Functions for DL control or DL communication commands on the Command Window of MATLAB.

List of MEX-Functions for DL Control

[ret] = mexDLComStart(wire, Adr);

Function: Initializes the line and connects the line to the specified device.

Parameters: `wire` Line type

`Adr` Character sequence of the line-specific address

Return value: `ret ( 0 = OK, 1 = ERROR )`

Description: Parameter description

`wire` Specify the type of line connected to the device to be controlled.
The value for each device is as follows:

GP-IB: wire = 1
RS-232: wire = 2
USB: wire = 3
Ethernet: wire = 4
USB-TMC: wire = 5

`Adr` Set the GP-IB address or the RS-232 value of the device to be controlled as a character string.
Below are the settings for each interface.

GP-IB: Adr = “1” to “30” (GP-IB address of the device)

RS-232: Adr = “port number, baud rate number, bit specification, handshaking number”

Port number	1 = COM1
	2 = COM2
	3 = COM3
Baud rate number	0 = 1200
	1 = 2400
	2 = 4800
	3 = 9600
	4 = 19200
	5 = 38400
	6 = 57600
Bit specification	0 = 8 bit, no parity, 1 stop bit
	1 = 7 bit, even parity, 1 stop bit
	2 = 7 bit, odd parity, 1 stop bit
Handshaking number	0 = NO-NO
	1 = XON-XON
	2 = CTS-RTS

USB: Adr = “1” to “127” (USB ID of the device)

Ethernet: Adr = “server name, user name, password”

Server name	DL server name or IP address
User name	User name
Password	Password

When the user name is anonymous, the password is not necessary.
(The delimiting comma is necessary.)
USB-TMC: Adr = “DL9000 series serial number”

Example:
```
ret = mexDLComStart(1,'1');
ret = mexDLComStart(4'10.0.222.111,user,password');
```


[ret] = mexDLDeviceClear;

Function: Executes the clearing (SDC) of the selected device. A dedicated command for the GP-IB.

Parameters: None

Return value: `ret ( 0 = OK, 1 = ERROR )`

Description: This function applies only to the device connected to the GP-IB; it does nothing to a device connected by a different interface.

Example:
```
ret = mexDLDeviceClear;
```

[ret] = mexDLSend(Msg);

Function: Sends a message to the device.

Parameters: `Msg` Character sequence of the DL communication command

Return value: `ret ( 0 = OK, 1 = ERROR )`

Description: Parameter description

`Msg` Set the DL communication command character string.
To send a single DL communication command in segments, use “mexDLSendSetup” and “mexDLSendOnly”.

Example:
```
ret = mexDLSend(':CANNEL1:DISPLAY 1');
ret = mexDLSend('start');
ret = mexDLSend('stop');
```

[ret] = mexDLSendByLength(Msg, len);

Function: Sends the specified number of bytes of the message to the device.

Parameters:
- `Msg` Character sequence of the DL communication command
- `len` The number of bytes of the DL communication command to be sent

Return value: `ret ( 0 = OK, 1 = ERROR )`

Description: Parameter description
- `Msg` Set the DL communication command.
- `len` Set the number of bytes of the DL communication command to be sent.

Can be sent even when binary data is included in the DL communication command.

To send a single message in segments, use `"mexDLSendSetup"` and `"mexDLSendOnly"`.

Example:
```
ret = mexDLSendByLength(':CHANNEL1:DISPLAY 1', 19);
```

[ret] = mexDLSendSetup;

Function: Prepares to send a message to the device.

Parameters: None

Return value: `ret ( 0 = OK, 1 = ERROR )`

Description: Prepares to send a message to the device.

Execute this function once before transmitting a single message in several transmissions.

Use `mexDLSendOnly` to actually transmit the message.

Example:
```
ret = mexDLSendSetup;
```

[ret] = mexDLSendOnly(Msg, len, end);

Function: Sends the specified number of bytes of the message to the device.

Parameters:
- `Msg` Character sequence of the DL communication command
- `len` The number of bytes of the DL communication command to be sent
- `end` End flag

Return value: `ret ( 0 = OK, 1 = ERROR )`

Description: Parameter description
- `Msg` Set the message.
- `len` Set the number of transmitted bytes of the message.
- `end` Set whether this transmission is the end of the transmission.

Set 1 if this is the end of the transmission. Set 0 to continue the transmission.

Transmits the DL communication command to the specified device.

Can be sent even when binary data is included in the DL communication command.
When the end flag is set to 1, a terminator is transmitted at the end of the DL communication command.
Therefore, while the end flag is 0, the device determines that the transmission is a part of an ongoing message.

Example: `ret = mexDLSendOnly(':CANNEL1:DISPLAY 1', 80,0);`

[ret, buf, size] = mexDLReceive(blen);

Function: Receives a message from the device.

Parameters: `blen` Receive size (in bytes)

Return value: `ret ( 0 = OK, 1 = ERROR )`
`buf` Receive data buffer
`size` The actual number of received bytes.

Description: Parameter description
`blen` Set the maximum number of bytes of the message to be received (this is normally the number of bytes of the buffer).
Return value description
`buf` Set the buffer that will store the received message.
`size` Returns the actual number of received bytes.
Receives a message from the device. If a terminator is detected, the data is received up to the terminator. Otherwise, the data is received up to the number of bytes specified by `blen`.
To receive the data of a message such as `"WAVeform:SEND?"` and `"IMAGe:SEND?"` when communicating with a Yokogawa digital oscilloscope, use `"mexDLReceiveBlockHeader"` and `"mexDLReceiveBlockData"`.

Example: `[ret,buf,size] = mexDLReceive(80);`

[ret] = mexDLReceiveSetup;

Function: Prepares to retrieve a message from the device.

Parameters: None

Return value: `ret ( 0 = OK, 1 = ERROR )`

Description: This function is executed to prepare for the reception of large data from the device in segments.
The actual data is received using `mexDLReceiveOnly`.

Example: `ret = mexDLReceiveSetup;`

[ret, buf, size] = mexDLReceiveOnly(blen);

Function: Receives a message (after preparation) from the device.

Parameters: `blen` Receive size (in bytes)

Return value: `ret ( 0 = OK, 1 = ERROR )`
`buf` Receive data buffer
`size` The actual number of received bytes.

Description: Parameter description
`blen` Set the maximum number of bytes of the message to be received (this is normally the number of bytes of the buffer).
Return value description
`buf` Set the buffer that will store the received message.
`size` Returns the actual number of received bytes.
Use this function when receiving large data in segments.
Receive the message from the specified device after preparing for the reception using `mexDLReceiveSetup`.
If a terminator is detected, the data is received up to the terminator.
Otherwise, the data is received up to the number of bytes specified by `blen`.

Example:
```
[ret,buf,size] = mexDLReceiveOnly(80);
```

[ret, length] = mexDLReceiveBlockHeader;

Function: Receives the header of the block data sent from the device, and returns the number of data bytes that follow.

Parameters: None

Return value: `ret ( 0 = OK, 1 = ERROR )`
`length` The number of bytes of the block data

Description: Return value description
`length` Returns the number of bytes of the block data.
This function is used first when receiving the block data.
The return value “length” contains the number of bytes that follow.
Receive the data by specifying this number of bytes + 1 (for the terminator) in `mexDlReceiveBlockData`.

Example:
```
[ret,length] = mexDLReceiveBlockHeader;
```

[ret, buf, size, endFlag] = mexDLReceiveBlockData(blen);

Function: Receives the data section of the block data sent from the device.

Parameters: `blen` Receive size (in bytes)

Return value: `ret ( 0 = OK, 1 = ERROR )`

`buf` Receive data buffer

`size` The actual number of received bytes.

`endFlag` End flag

Description: Parameter description

`blen` Set the maximum number of bytes of the message to be received (this is normally the number of bytes of the buffer).

Return value description

`buf` Set the buffer that will store the received message.

`size` Returns the actual number of received bytes.

`endFlag` Returns whether the reception of the number of data bytes indicated by X>mexDLReceiveBlockHeader has been completed.

If it is, 1 is returned. If not, 0 is returned.

This command is used to receive block data (message starting with #) Receive the message from the specified device after preparing for the reception using `mexDLReceiveBlockHeader`.

If a terminator is detected, the data is received up to the terminator. Otherwise, the data is received up to the number of bytes specified by `blen`.

Example:
```
[ret,buf,size,endFlag] = mexDLReceiveBlockData(80);
```

[endFlag] = mexDLCheckEnd;

Function: Returns whether the message from the device is finished. Can be used on the GP-IB, USB, or Ethernet interface.

Parameters: None

Return value: end 1 = message remaining, 0 = message is finished

Description: When a sequence of messages is received in segments, this function returns whether all the messages has been received by `mexDLReceiveOnly`.

Example:
```
endFlag = mexDLCheckEnd;
```

[ret] = mexDLSetRen(flag);

Function: Sets the device in remote or local mode. Use of an interface other than GP-IB is limited to Yokogawa products.

Parameters: `flag` Remote (`1`)/Local (`0`)

Return value: `ret ( 0 = OK, 1 = ERROR )`

Description: Parameter description

`flag` To set to remote mode send `1`; to set to local mode send `0`.

The behavior varies slightly depending on the interface type.

For GP-IB, the REN line is set to TRUE/FALSE.

Therefore, remote mode is actually enabled when a message is sent to the device. (Remote/Local operation on the individual device is not carried out.)

In the case of RS-232, USB, and Ethernet, the use of this function is limited to Yokogawa products complying with 488.2 that support the COMMunicate group commands. In this case, remote/local operation on the individual device is possible.

Example:

```
ret = mexDLSetRen(1);
```

[errorID] = mexDLGetLastError;

Function: Returns the number of the last error that occurred due to a MATLAB command (not the error code on the DL).

Parameters: `int id` Device ID

Return value: Error number

Description: Return value description

ErrorID Returns the last error number that occurred on the device.

0x00000000(0)	No error
0x00000001(1)	Timeout
0x00000002(2)	Device Not Found
0x00000004(4)	Open Port Error
0x00000008(8)	Device Not Open
0x00000010(16)	Device Already Open
0x00000020(32)	Controller Not Found
0x00000040(64)	Parameter is illegal
0x00000100(256)	Send Error
0x00000200(512)	Receive Error
0x00000400(1024)	Data is not Block Data
0x00001000(4096)	System Error
0x00002000(8192)	Device ID is Illegal

When the return value of a function including the initialization function is not 0 (= OK), this function is used to retrieve the actual error number.

Example:
```
errorID = mexDLGetLastError;
```

[ret] = mexDLSetTerm(eos, eot);

Function: Sets the terminator used in the message transmission/reception.

Parameters: `eos` Terminator

`eot` EOI

Return value: `ret ( 0 = OK, 1 = ERROR )`

Description: Parameter description

`eos` Set the terminator. Below are the settings.

`eos = 0`: CR+LF

`eos = 1`: CR

`eos = 2`: LF

`eos = 3`: EOI (GPIB) or none (RS-232, USB, or Ethernet)

When the interface is GP-IB and `eos` is 3, use `eot` to specify whether EOI will be used.

`eot` Set whether EOI will be used for the terminator.

This is dedicated to the GP-IB.

In general, when communicating with a Yokogawa product, use the settings below on all interfaces.

```
mexDLSetTerm( 2,1 ); /* eos = LF, eot = TRUE */
```

If `eos = LF` is used when receiving binary data and the binary code contains an LF code, the DL will decide that the data ends there. However, when receiving block data from a Yokogawa product using `"mexDLReceiveBlockHeader"` and `"mexDLReceivceBlockData"`, you do not have to switch the terminator.

Example:

```
ret = mexDLSetTerm(1,0);
```

[ret] = mexDLSetTimeout(tmo);

Function: Sets the communication timeout time.

Parameters: `tmo` Timeout time (100 to 6553600 ms)

Return value: `ret ( 0 = OK, 1 = ERROR )`

Description: Parameter description

`tmo` Sets the timeout value. 100 ms unit.

If `tmo = 0`

GP-IB: Timeout set to infinity.

Others: Timeout not set.

Note

"Infinity" means that the wait time is set to an infinite amount of time. "Not set" means that there is no wait time (responds immediately).

Sets the communication timeout time.

For a Yokogawa product, set the time greater than equal to 30 s. (Even if the timeout time is set long, the overall performance is not affected.)

Example: `ret = mexDLSetTimeout(300);`

[ret, WaveData] = mexDLGetWave;

Function: Retrieves the measured data.

Parameters: None

Return value: `ret ( 0 = OK, 1 = ERROR )`

`WaveData` Matrix of waveform data

Description: Return value description

WaveData All of the waveform data of the specified record length are stored to a matrix for each displayed channel.

$$\text{WaveData} = \begin{pmatrix} W_{1,1} & \cdots & W_{1,Ch} \\ \vdots & & \vdots \\ W_{Len,1} & \cdots & W_{Len,Ch} \end{pmatrix}$$

When retrieving waveform data of long record length, set the timeout time to a relatively large value.

```
>> ret = mexDLSetTimeout(1000);
Data acquired using the logic input on the DL
cannot be stored.
```

Example:

```
[ret,WaveData] = mexDLGetWave;
plot(Wavedata)
plot(Wavedata(:,2));      % Displays the waveform of Ch2
```

Note

When storing large quantities of waveform data into a MATLAB matrix using this command and the memory area needed to store the data cannot be allocated continuously, a message “Out of Memory” appears. The size of continuous memory that can be allocated varies depending on your PC’s environment. If this message appears, take the following measures to retrieve the data.

- Expand the memory swap space
  The procedure for Windows 2000 is described below. For other Windows versions, see the respective Windows manual.
  1. Right-click My Computer and choose Properties.
  2. Select the Advanced tab and select Performance Options.
  3. Click Change and change the virtual memory size.

- Reduce the size of the matrix
  Use the function for retrieving the data by specifying the waveform range
  `[ret,WaveData] = mexDLGetWave( traceNum, waveStart, waveEnd );`
  to divide the large matrix into several smaller matrices. This reduces the amount of data that MATLAB handles at one time.

[ret, WaveData] = mexDLGetWave (traceNum, waveStart, waveEnd);

Function: Specifies the trace number and waveform range and retrieve the waveform data.

Parameters: traceNum Trace number of the waveform for retrieving the data
waveStart Start point of the waveform data to be retrieved (can be omitted)
waveEnd End point of the waveform data to be retrieved (can be omitted)

Return value: `ret ( 0 = OK, 1 = ERROR )`
WaveData Waveform data matrix (each element is double type)

Details: Parameter description

`traceNum` Specify the trace number. The minimum value is 1. The largest value depends on the number of channels on the connected model. If 0 is specified, the waveforms of all displayed channels are retrieved. Trace numbers of hidden channels cannot be specified.

`waveStart,waveEnd` Specify the range of waveform data to be retrieved. The range of waveStart and waveEnd is a value within the display record length. In addition, waveStart and waveEnd can be omitted (you cannot omitted only one of the parameters). When omitted, the entire waveform data of the displayed record length is retrieved.

Return value description

WaveData The waveform data of the specified trace number in the specified range within the display record length is stored in a matrix. The data of hidden channels on the screen cannot be retrieved.

$$\text{WaveData} = \begin{pmatrix} W_{1,1} & \cdot & \cdot & \cdot & W_{1,Ch} \\ \cdot & \cdot & & & \cdot \\ \cdot & & \cdot & & \\ \cdot & & & \cdot & \cdot \\ W_{Len,1} & \cdot & \cdot & \cdot & W_{Len,Ch} \end{pmatrix}$$

Example: When CH1, CH2, and CH4 are shown on the display (with a display record length of 10 kW)

```
[ret,WaveData] = mexDLGetWave(0);
    %Retrieve all the waveform data of CH1, CH2, and CH4.
[ret,WaveData] = mexDLGetWave(1);
    %Retrieve all the waveform data of CH1.
[ret,WaveData] = mexDLGetWave(1,1000,4000);
    %Retrieve the waveform data between 1 kW to 4 kW on CH1
```

Note

- Use the following function to retrieve the specified record length of data from a saved .wvf file using MexFunctions for WVF Files.
  `[ret,data] = mexWeDPCsRead(filename,seriesNo,start,length,ch,dataForm,dataNum)`
  You can retrieve the data by specifying the `start` and `length` parameters.
- If the .wvf files are not consecutive, you can retrieve the data by setting `seriesNo` to `-1`.

[ret, HistoryWave] = mexDLGetHistoryWave (rec);

Function: Retrieves the history waveform data.

Parameters: `Rec` Record No. The newest (current) waveform is 0, the waveform previous to that is –1, and so on. For the selectable range, see the user's manual for the respective DL.

Return value: `ret ( 0 = OK, 1 = ERROR )`

`HistoryWave` Matrix of historical waveform data

Description: Return value description

HistryWaveData The waveform data of the specified record number are stored to a matrix for each displayed channel.

$$\text{HistroyWave} = \begin{pmatrix} W_{1,1} & \cdot & \cdot & \cdot & W_{1,Ch} \\ \cdot & \cdot & & & \cdot \\ \cdot & & \cdot & & \\ \cdot & & & \cdot & \cdot \\ W_{Len,1} & \cdot & \cdot & \cdot & W_{Len,Ch} \end{pmatrix}$$

When retrieving waveform data of long record length, set the timeout time to a relatively large value.

```
>> ret = mexDLSetTimeout(1000);
Data acquired using the logic input on the DL
cannot be stored.
```

Example: `[ret,HistoryWave] = mexDLGetHistoryWave( -20 );`
Retrieve the –20th history waveform.

Note

When storing large quantities of waveform data into a MATLAB matrix using this command and the memory area needed to store the data cannot be allocated continuously, a message "Out of Memory" appears. The size of continuous memory that can be allocated varies depending on your PC's environment. If this message appears, take the following measures to retrieve the data.

- Expand the memory swap space
  The procedure for Windows 2000 is described below. For other Windows versions, see the respective Windows manual.
  1. Right-click My Computer and choose Properties.
  2. Select the Advanced tab and select Performance Options.
  3. Click Change and change the virtual memory size.

- Reduce the size of the matrix
  Use the function for retrieving the data by specifying the waveform range
  `[ret,HistoryWave] = mexDLGetHistoryWave( rec, traceNum, waveStart, waveEnd );`
  to divide the large matrix into several smaller matrices. This reduces the amount of data that MATLAB handles at one time.

[ret, HistoryWave] = mexDLGetHistoryWave (rec, traceNum, waveStart, waveEnd);

Function: Specifies the record number, trace number and waveform range and retrieves the history waveform data.

Parameters:
- `rec` Record number
- `traceNum` Trace number of the waveform for retrieving the data
- `waveStart` Start point of the waveform data to be retrieved (can be omitted)
- `waveEnd` End point of the waveform data to be retrieved (can be omitted)

Return value: `ret ( 0 = OK, 1 = ERROR )`

HistoryWave History waveform data matrix (each element is double type)

Details: Parameter description

`Rec` Record No. The newest (current) waveform is 0, the waveform previous to that is –1, and so on. For the selectable range, see the respective DL User's Manual.

`traceNum` Specify the trace number. The minimum value is 1. The largest value depends on the number of channels on the connected model. If 0 is specified, the waveforms of all displayed channels are retrieved. Trace numbers of hidden channels cannot be specified.

`waveStart,waveEnd` Specify the range of waveform data to be retrieved. The range of waveStart and waveEnd is within the display record length. In addition, waveStart and waveEnd can be omitted (you cannot omitted only one of the parameters). When omitted, the entire waveform data of the displayed record length is retrieved.

Return value description

HistoryWave The waveform data of the specified record number are stored to a matrix for each displayed channel. The data of hidden channels on the screen are not retrieved.

$$\mathbf{HistroyWave} = \begin{pmatrix} W_{1,1} & \cdot & \cdot & \cdot & W_{1,Ch} \\ \cdot & \cdot & & & \cdot \\ \cdot & & \cdot & & \\ \cdot & & & \cdot & \cdot \\ W_{Len,1} & \cdot & \cdot & \cdot & W_{Len,Ch} \end{pmatrix}$$

Example: When CH1, CH2, and CH4 are shown on the display (with a display record length of 10 kW)

```
[ret,HistryWave] = mexDLGetHistoryWave(-20,0);
   %Retrieve the -20th history waveform of CH1, CH2, and CH4.
[ret,HistryWave] = mexDLGetHistoryWave(-20,4);
   %Retrieve the entire data of the -20th history waveform of CH4.
[ret,HistryWave] = mexDLGetHistoryWave(-20,4,2000,
                                       5000);
   %Retrieve the -20th history waveform data of CH4 between 2kW
   and 5 kW.
```

mexDLControl;

Function: Starts the DL control window.
Parameters: None
Return value: None
Example: `mexDLControl;`

[ret] = mexDLComEnd;

Function: Closes the line connected to the device.
Parameters: None
Return value: `ret ( 0 = OK, 1 = ERROR )`
Description: Closes the line that was opened using `mexDLComStart` (initialization function).
Be sure to execute this function when terminating the communication.
Example: `ret = mexDLComEnd;`

mexDLToolkit;

Function: Displays version information about the software program.
Parameters: None
Return value: None
Example:

```
help mexDLToolkit

Model 701991
MATLAB ToolKit for DL Series
Version *.**

All Rights Reserved,
Copyright (c) (year) Yokogawa Electric Corporation
```

4.2 MEX-Functions for WVF Files

The function names obtained by removing “mex” from the mex function names correspond to the WVF File Access API functions.
For details on the mex functions, see chapter 3, “File Operation Functions” in the WVF File Access API User’s Manual (IM707712-61E) or chapter 9, “File Operation Functions” in the WE Control API User’s Manual (IM707741-61E).

4.2.1 List of MEX-Functions for WVF Files

Single File Access

mex Function Name	Function	Page
mexWeDPHeaderReadS	Read the header file of the single file.	4-17
mexWeDPDataRead	Read the data file of the single file.	4-17
mexWeDPHeaderWriteS	Write the header file of the single file.	4-18
mexWeDPDataWrite	Write the data file of the single file.	4-18

Sequential File Access

mex Function Name	Function	Page
mexWeDPHeaderCsReadS	Read the header file of the sequential file.	4-19
mexWeDPCsRead	Read the data file of the sequential file.	4-19
mexWeDPHeaderCsWriteS	Write the header file of the sequential file.	4-20
mexWeDPCsWrite	Write the data file of the sequential file.	4-20

Access to the Specified Item of the Header File

mex Function Name	Function	Page
mexWeDPHeaderItemRead	Read the data of the specified item.	4-21
mexWeDPHeaderItemWrite	Write the data of the specified item.	4-21

Data Operation

mex Function Name	Function	Page
mexWeDPGetSampleChNum	Get the number of samples and number of channels.	4-22
mexWeDPGetBlockNum	Get the number of blocks.	4-22
mexWeDPInitializeAcqInfo	Store the required data in the data information structure.	4-22

4.2.2 Single File Access

[ret, ComInfo, ChInfo] = mexWeDPHeaderReadS(filename, blockNo, ChNum)

Function	Read the header file of the single file.	
Parameters	`filename`:	Name of the file to be read without the extension
	`blockNo`:	Block number to be read (0 origin)
	`ChNum`:	Number of channels to be read (number of ChInfo structures)
Return value	`ret`:	Returns 0 if successful. Returns an error code if unsuccessful.
	`ComInfo`:	Structure of the read information (ComInfo)
	`ChInfo`:	Structure of the read information (ChInfo)
Description	Reads the data from the header file by specifying the block number. Header file of the data acquired using the logic input on the DL cannot be loaded.	

[ret, data] = mexWeDPDataRead(filename, blockNo, ch, dataForm, dataNum)

Function	Read the data file of the single file.	
Parameters	`filename`:	Name of the file to be read without the extension
	`blockNo`:	Number of the block to be read
	`ch`:	Number of the channel to be read
	`dataForm`:	Type of data to be read 1 = WE_UBYTE 17 = WE_SWORD 33 = WE_SLONG 34 = WE_FLOAT 50 = WE_DOUBLE
	`dataNum`:	Number of data points to be read
Return value	`ret`:	Returns 0 if successful. Returns an error code if unsuccessful.
	`data`:	Read data
Description	Reads the data from the data file by specifying the block number.	

Note

The parameter dataNum does not exist in the WE File Access API or the WE Control API function, but is required in the mex function.

ret = mexWeDPHeaderWriteS(filename, blockNo, ComInfo, ChNum, ChInfo, AcqInfo)

Function	Write the header file of the single file.	
Parameters	`filename`:	Name of the file to be written without the extension
	`blockNo`:	Block number to be written (0 origin)
	`ComInfo`:	Structure of the written information (ComInfo)
	`ChNum`:	Number of channels to be written (number of ChInfo structures)
	`ChInfo`:	Structure of the written information (ChInfo)
	`AcqInfo`:	Data information structure to be written
Return value	`ret`:	Returns 0 if successful. Returns an error code if unsuccessful.
Description	Writes the header information at once to the header file by specifying the block.	

ret = mexWeDPDataWrite(filename, blockNo, sampleNum, ChNum, AcqInfo, dataForm, data)

Function	Write the data file of the single file.	
Parameters	`filename`:	Name of the file to be written without the extension
	`blockNo`:	Number of the block to be written
	`sampleNum`:	Number of samples to be written
	`ChNum`:	Number of channels to be written
	`AcqInfo`:	Data information structure to be written
	`dataForm`:	Type of data to be written 1 = WE_UBYTE 17 = WE_SWORD 33 = WE_SLONG 34 = WE_FLOAT 50 = WE_DOUBLE
	`data`:	Data to be written
Return value	`ret`:	Returns 0 if successful. Returns an error code if unsuccessful.
Description	Writes the data to the data file in units of blocks.	

4.2.3 Sequential File Access

[ret, ComInfo, ChInfo] = mexWeDPHeaderCsReadS(filename, seriesNo, ChNum)

Function	Read the header file of the sequential file.	
Parameters	`filename`:	Name of the file to be read without the extension
	`seriesNo`:	First sequence number of the file to be read
	`ChNum`:	Number of channels to be read (number of ChInfo structures)
Return value	`ret`:	Returns 0 if successful. Returns an error code if unsuccessful.
	`ComInfo`:	Structure of the read information (ComInfo)
	`ChInfo`:	Structure of the read information (ChInfo)
Description	Collectively reads the header information from a header file. Header file of the data acquired using the logic input on the DL cannot be loaded.	

[ret, data] = mexWeDPCsRead(filename, seriesNo, start, length, ch, dataForm, dataNum)

Function	Read the data file of the sequential file.	
Parameters	`filename`:	Name of the file to be read without the extension
	`seriesNo`:	First sequence number of the file to be read
	`start`:	Start point of the data to be read
	`length`:	Number of data points to be read
	`ch`:	Number of the channel to be read
	`dataForm`:	Type of data to be read 1 = WE_UBYTE 17 = WE_SWORD 33 = WE_SLONG 34 = WE_FLOAT 50 = WE_DOUBLE
	`dataNum`:	Number of data points to be read
Return value	`ret`:	Returns 0 if successful. Returns an error code if unsuccessful.
	`data`:	Read data
Description	Reads the data from the data files (sequential files) by specifying the number of samples.	

Note

The parameter dataNum does not exist in the WE File Access API or the WE Control API function, but is required in the mex function.

ret = mexWeDPHeaderCsWriteS(filename, seriesNo, ComInfo, ChNum, ChInfo, AcqInfo)

Function	Write the header file of the sequential file.	
Parameters	`filename`:	Name of the file to be written without the extension
	`seriesNo`:	First sequence number of the file to be written
	`ComInfo`:	Structure of the written information (ComInfo)
	`ChNum`:	Number of channels to be written (number of ChInfo structures)
	`ChInfo`:	Structure of the written information (ChInfo)
	`AcqInfo`:	Data information structure to be written
Return value	`ret`:	Returns 0 if successful. Returns an error code if unsuccessful.
Description	Collectively writes the header information to the header file.	

ret = mexWeDPCsWrite(filename, seriesNo, sampleNum, ChNum, AcqInfo, dataForm, data)

Function	Write the data file of the sequential file.	
Parameters	`filename`:	Name of the file to be written without the extension
	`seriesNo`:	First sequence number of the file to be written
	`sampleNum`:	Number of samples to be written
	`ChNum`:	Number of channels to be written
	`AcqInfo`:	Data information structure to be written
	`dataForm`:	Type of data to be written 1 = WE_UBYTE 17 = WE_SWORD 33 = WE_SLONG 34 = WE_FLOAT 50 = WE_DOUBLE
	`data`:	Data to be written
Return value	`ret`:	Returns 0 if successful. Returns an error code if unsuccessful.
Description	Write data to a sequence file.	

4.2.4 Access to the Specified Item of the Header File

[ret, data] = mexWeDPHeaderItemRead(filename, itemName, ch, blockNo)

Function	Read the data of the specified item of the header file.	
Parameters	`filename`:	Name of the file to be read without the extension
	`itemName`:	Name of the item to be read
	`ch`:	Number of the channel to be read
	`blockNo`:	Number of the block to be read
Return value	`ret`:	Returns 0 if successful. Returns an error code if unsuccessful.
	`data`:	Read data
Description	Reads the information of the specified item name and specified channel from the header information of the header file.	

ret = mexWeDPHeaderItemWrite(filename, itemName, ch, blockNo, data)

Function	Write data to the specified item of the header file.	
Parameters	`filename`:	Name of the file to be written without the extension
	`itemName`:	Name of the item to be written
	`ch`:	Number of the channel to be written
	`blockNo`:	Number of the block to be written
	`data`:	Data to be written
Return value	`ret`:	Returns 0 if successful. Returns an error code if unsuccessful.
Description	Writes data to the specified item name and specified channel in the header information of the header file.	

4.2.5 Data Operation

[ret, SampleNum, ChNum] = mexWeDPGetSampleChNum(filename, blockNo)

Function	Get the number of samples and number of channels.	
Parameters	`filename`:	Name of the file to be read without the extension
	`blockNo`:	Number of the block to be read
Return value	`ret`:	Returns 0 if successful. Returns an error code if unsuccessful.
	`SampleNum`:	Number of data points read
	`ChNum`:	Number of channels read
Description	Gets the number of samples and number of channels of the specified file.	

[ret, blockNum] = mexWeDPGetBlockNum(filename)

Function	Get the number of blocks.	
Parameters	`filename`:	Name of the file to be read without the extension
Return value	`ret`:	Returns 0 if successful. Returns an error code if unsuccessful.
	`blockNum`:	Number of block read
Description	Gets the number of blocks of the specified file.	

[ret, AcqInfo] = mexWeDPInitializeAcqInfo(VMaxData, VMinData, sampleNum, sampInterval, infoNum)

Function	Set the data in the data information structure.	
Parameters	`VMaxData`:	Max data
	`VMinData`:	Min data
	`sampleNum`:	Number of data samples
	`sampInterval`:	Sampling frequency of the data
	`infoNum`:	Number of data information structures
Return value	`ret`:	Returns 0 if successful. Returns an error code if unsuccessful.
	`AcqInfo`:	Data information structure
Description	Stores the required data in the data information structure.	

4.3 MEX-Functions for WDF Files

4.3.1 List of MEX-Functions for WDF Files

Accessing file information

mex Function Name	Function	Page
mexWdfItemRead	Read the file information	4-24
mexWdfGetBlockNum	Read the number of blocks	4-24
mexWdfGetChNum	Read the number of channels	4-24

Data Operation

mex Function Name	Function	Page
mexWdfDataRead	Read the raw waveform data	4-25
mexWdfDataReadEx	Read the raw waveform data (expanded version)	4-25
mexWdfScaleDataRead	Read the physical value waveform data	4-26
mexWdfScaleDataReadEx	Read the physical value waveform data (expanded version)	4-26

4.3.2 Accessing file information

[ret, data] = mexWdfItemRead(filename, itemName, ch, block)

Function	Read the file information	
Parameters	`filename`:	Name of file (with extension) to be read
	`itemName`:	Name of item to be read (see Parameters)
	`ch`:	Channel numbers to be read (0-)
	`block`:	Block numbers to be read (0-)
Return value	`ret`:	Returns 0 if successful. Returns an error code if unsuccessful.
	`data`:	Read data
Description	Gets the specified file information (channels, blocks, and items) from the specified WDF file.	

[ret, blockNum] = mexWdfGetBlockNum(filename)

Function	Read the number of blocks	
Parameters	`filename`:	Name of file (with extension) to be read
Return value	`ret`:	Returns 0 if successful. Returns an error code if unsuccessful.
	`blockNum`:	Number of blocks read
Description	Gets the number of blocks in the specified WDF file.	

[ret, chNum] = mexWdfGetChNum(filename)

Function	Read the number of channels	
Parameters	`filename`:	Name of file (with extension) to be read
Return value	`ret`:	Returns 0 if successful. Returns an error code if unsuccessful.
	`chNum`:	Number of channels read
Description	Gets the number of channels in the specified WDF file.	

4.3.3 Data Operation

[ret, param, data] = mexWdfDataRead(filename, ch, block)

Function	Read the raw waveform data	
Parameters	`filename`:	Name of file (with extension) to be read
	`ch`:	Channel numbers to be read (0-)
	`block`:	Block numbers to be read (0-)
Return value	`ret`:	Returns 0 if successful. Returns an error code if unsuccessful.
	`param`:	Structure of the read information (see Structure)
	`data`:	Read data (array)
Description	Gets the specified channel and block data from the specified WDF file.	

[ret, param, data] = mexWdfDataReadEx(filename, ch, block, start,length)

Function	Read the raw waveform data (expanded version)	
Parameters	`filename`:	Name of file (with extension) to be read
	`ch`:	Channel numbers to be read (0-)
	`block`:	Block numbers to be read (0-)
	`start`:	Start point of the data to be read
	`length`:	Number of data points to be read
Return value	`ret`:	Returns 0 if successful. Returns an error code if unsuccessful.
	`param`:	Structure of the read information (see Structure)
	`data`:	Read data (array)
Description	Gets a specified range of specified channel and block data from a specified file.	

[ret, param, data] = mexWdfScaleDataRead(filename, ch, block)

Function	Read the physical value waveform data	
Parameters	`filename`:	Name of file (with extension) to be read
	`ch`:	Channel numbers to be read (0-)
	`block`:	Block numbers to be read (0-)
Return value	`ret`:	Returns 0 if successful. Returns an error code if unsuccessful.
	`param`:	Structure of the read information (see Structure)
	`data`:	Read data (array)
Description	Gets the specified channel and physical value block data from the specified WDF file. Multiply VResolution by the file's raw data (signed 16-bit integer), and add VOffset.	

[ret, param, data] = mexWdfScaleDataReadEx(filename, ch, block,start, length)

Function	Read the physical value waveform data (expanded version)	
Parameters	`filename`:	Name of file (with extension) to be read
	`ch`:	Channel numbers to be read (0-)
	`block`:	Block numbers to be read (0-)
	`start`:	Start point of the data to be read
	`length`:	Number of data points to be read
Return value	`ret`:	Returns 0 if successful. Returns an error code if unsuccessful.
	`param`:	Structure of the read information (see Structure)
	`data`:	Read data (array)
Description	Gets only a specified range of the specified channel and physical value block data from the specified WDF file. Multiply VResolution by the file's raw data (signed 16-bit integer), and add VOffset.	

4.3.4 Parameters and structure

Parameters

Item entered in the itemName area	dataType	Parameters ch	Parameters block	Meaning
Comment	string	N	N	Comment string
Version	string	N	N	Version string
Model	string	N	N	Model name
TraceNumber	UINT	N	N	Number of channels
BlockNumber	UINT	N	N	Number of blocks
TraceName	string	Y	N	Channel name
BlockSize	UINT	Y	Y	Block size
VDataType	UINT	Y	Y	Data type
VUnit	string	Y	Y	Vertical axis unit string
VResolution	double	Y	Y	Vertical axis resolution
VOffset	double	Y	Y	Vertical axis offset
VScaleUpper	double	Y	Y	Vertical axis upper limit scale value
VScaleLower	double	Y	Y	Vertical axis lower limit scale value
HResolution	double	Y	Y	Horizontal axis resolution
HOffset	double	Y	Y	Horizontal axis offset
HUnit	string	Y	Y	Horizontal axis unit string
Date	string	Y	Y	Date
Time	string	Y	Y	Time
DateTime	string	Y	Y	Date and Time
VIllegalData	double	Y	Y	Loss value
VMaxData	double	Y	Y	Maximum value
VMinData	double	Y	Y	Minimum value
SplitNumMain	UINT	N	N	Main display resolution
SplitNumZ1	UINT	N	N	Zoom-1 display resolution
SplitNumZ2	UINT	N	N	Zoom-2 display resolution
TraceColor0	UINT	Y	N	Waveform color (normal)
TraceColor1	UINT	Y	N	Waveform color (neutral)

* Y: Required, N: Ignore

Structure

Parameters	Meaning	Value
ch	Channels to be read	0-
block	Blocks to be read	0-
start	Read start point	0-
count	Number of points to be read	1-
waveType	Output waveform type	0: Measured raw waveforms (AD values) 1: Waveforms converted from physical values
dataType	Output waveform data type	0: Unsigned 8-bit integer 1: Signed 8-bit integer 4: 8-bit logical value 16: Unsigned 16-bit integer 17: Signed 16-bit integer 20: 16-bit logical value 32: Unsigned 32-bit integer 33: Signed 32-bit integer 36: 32-bit logical value 48: Unsigned 64-bit integer 49: Signed 64-bit integer 52: 64-bit logical value 34: Single precision real number 50: Double precision real number 256: None
cntOut	Number of output points	Number of successfully read points

4.3.5 Error Code

Error Code	Meaning
0	Concluded successfully
100	File open failed
101	Allocation failed
102	File access error
103	DLL link failed
200	Unsupported version
201	Unsupported format
202	Unknown function
300	Range specification error
301	File handle not found
900	Data obtained when real time measurement failed
901	Illegal data values
902	Data that cannot be loaded (PPsave, Z1/Z2save)
1000	Other error

Index

M

S

T