User's Manual

Model 701991
MATLAB Tool Kit for DL Series

YOKOGAWA
Yokogawa Test & Measurement Corporation

IM 701991-61E
11th 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://tmi.yokogawa.com/

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.
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.
  DLM2000 Series
  DL6000/DLM6000 Series
  DL850 Series
  SL1000
  DLM4000 Series
- This manual supports **MATLAB R2011b** 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, and Windows are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries.
- Adobe and Acrobat are registered trademarks or trademarks of Adobe Systems Incorporated.
- For purposes of this manual, the ® and TM symbols do not accompany their respective registered trademark names or trademark names. Other company and product names are registered trademarks or 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
- 6th Edition: October 2010
- 7th Edition: November 2011
- 8th Edition: March 2013
- 9th Edition: January 2014
- 10th Edition: June 2016
- 11th Edition: October 2017

11th Edition : October 2017 (YMI)

Terms and Conditions of the Software License

Yokogawa Electric Corporation and Yokogawa Test & Measurement Corporation, Japanese corporations (hereinafter called “Yokogawa”), grant 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

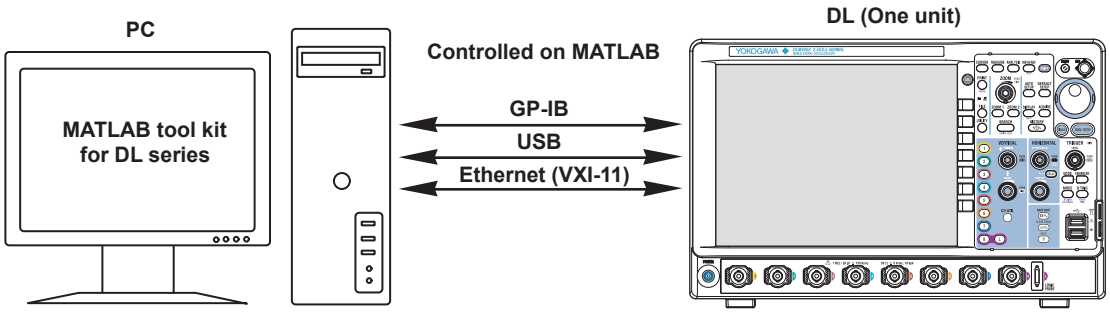
Product Overview

This software program consists of MEX-Functions for DL control, MEX-Functions for 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.
The DL series library is used when the DL is accessed with the MEX-Function for DL control described above.
MEX-Functions for WDF files are a group of functions used to access from the MATLAB environment the waveform data (.wdf extension) that has been saved with the DL.

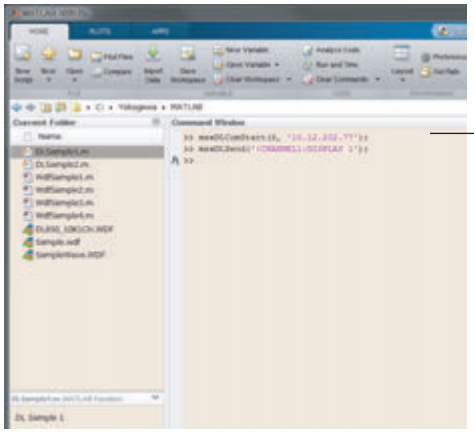
Note

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

This software program connects the PC and the DL and controls the DL.
The method of controlling the DL is to use the MEX-Functions for DL control or the DL communication commands on MATLAB.



Window on MATLAB used to control the DL

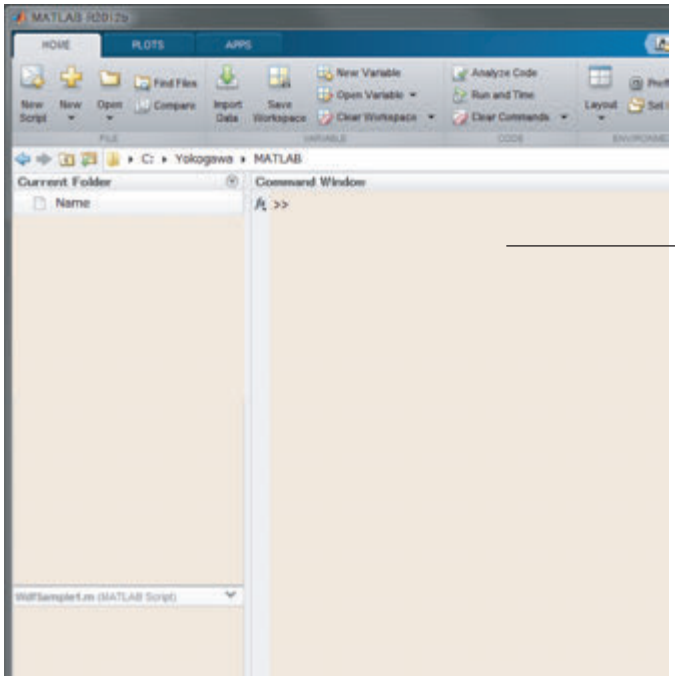


Control the DL using the MEX-Functions for DL control or DL communication commands. Or, retrieve waveform data or history data.

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. By using the MEX-Functions for DL control, you can change the record length and acquire and save waveform data in a format that can be displayed in 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')`

PC System Requirements

- **OS**
  Windows 7(32-bit version and 64bit version)
  Windows 8(32-bit version and 64bit version)
  Windows 8.1(32-bit version and 64bit version)
  Windows 10(32-bit version and 64bit version)

- **CPU**
  Same as that of MATLAB

- **Memory**
  Same as that of MATLAB

- **CRT**
  Same as that of MATLAB

- **Color**
  Same as that of MATLAB

- **Communication Interface**
  GP-IB, Ethernet (VXI II), USB, or RS-232.

- **Controllable DL Series Oscilloscopes**
  DLM2000 Series
  DL6000/DLM6000 Series
  DL850 Series
  SL1000
  DLM4000 Series

- **MATLAB**

R2011b 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.

USB: Yokogawa DLM2000, DL6000/DLM6000,DL850 Series, SL1000 or DLM4000 Digital Oscilloscope with a USB interface.

Ethernet: DLM2000, DL6000/DLM6000,DL850 Series, SL1000 or DLM4000 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 Ethernet 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 Ethernet 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.
- 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.

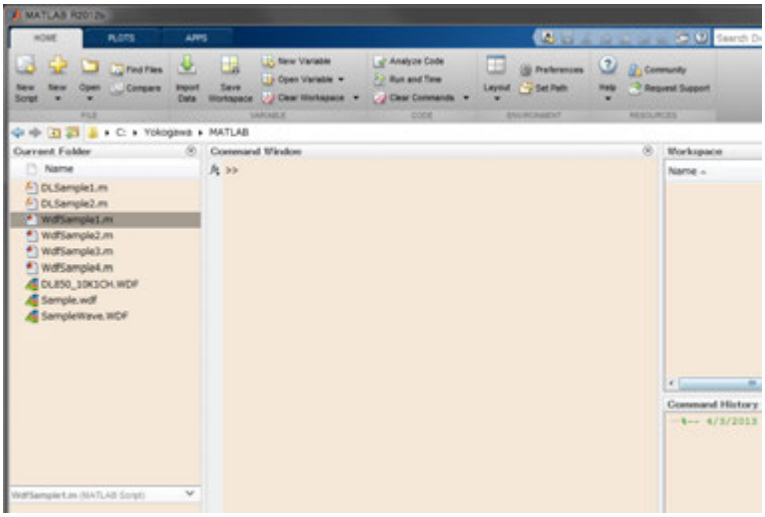
Note

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

1.2 Initializing MATLAB

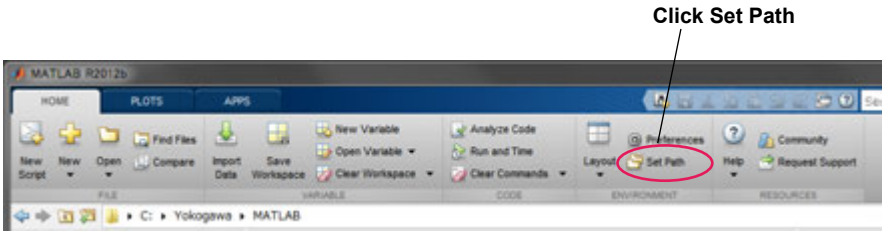
Carry out the procedure below once after installing this software.

1. Start MATLAB.



Setting the MATLAB Path

2. Click Set Path of the toolbar, choose **Set Path**. The Set Path dialog box opens.

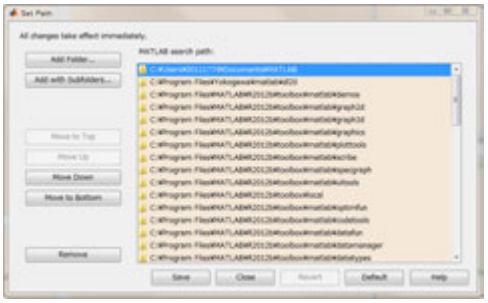


3. Click the **Add Folder** button and select the folder in which the MATLAB tool kit for DL series was installed.

Note

The default installation destination folders are as follows.
C:\Program Files\Yokogawa\matlab\dl20

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



2.1 Setting the DL Series Communication 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.

Set the interface to be used from the front panel of the DL.
On the DLM2000 Series/DL6000/DLM6000 Series/DL850 Series:

UTILITY > Remote Control > Device

When Connecting over USB

To establish a USB connection with the DL Series Oscilloscope, the MATLAB tool kit requires that the USB driver for that DL Series Oscilloscope. The USB driver is on the MATLAB tool kit installation disk.
You can download the latest USB driver from the following web page.
http://www.yokogawa.com/ymi/

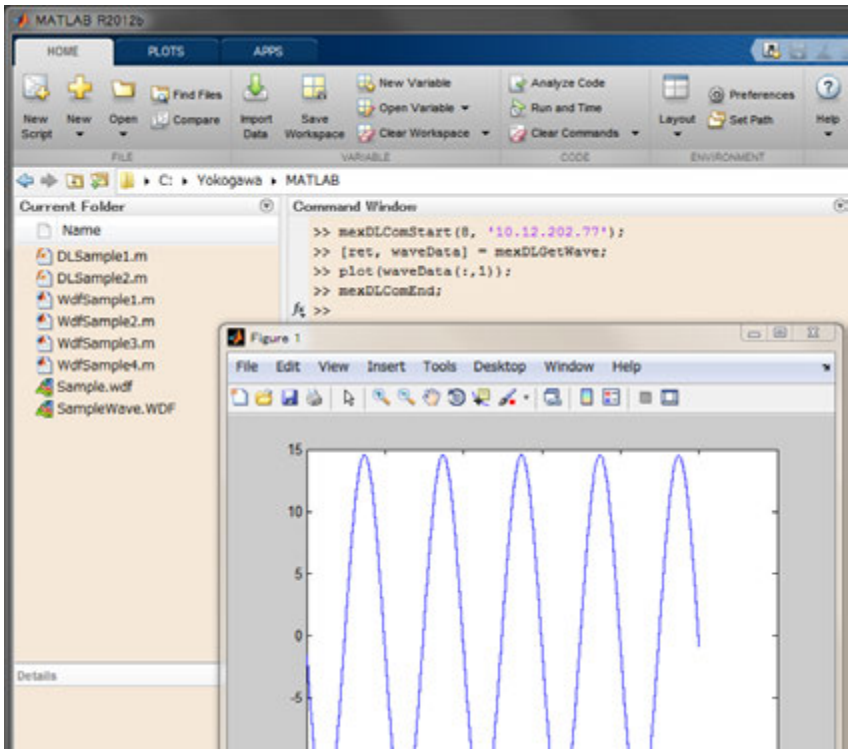
Installing the USB Driver

Run Setup.exe in the YKMUSB folder. The installation wizard starts. For details on the installation procedure, see the manual (IM B9852UT-01E) in the YKMUSB folder.

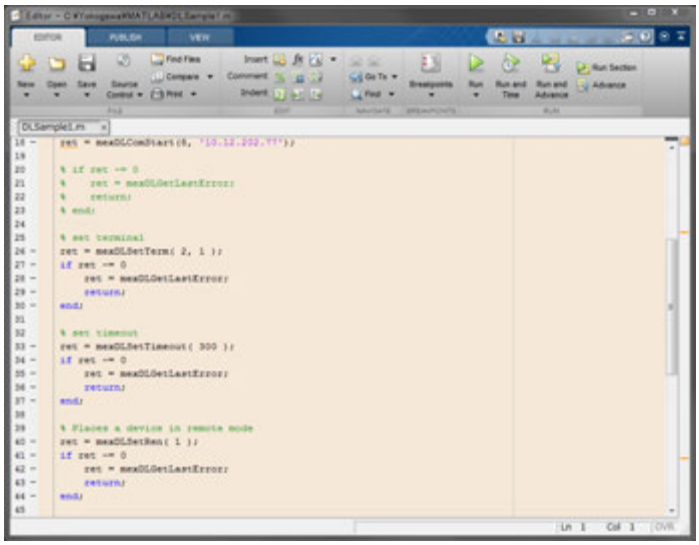
2.2 Executing MEX-Functions for DL Control

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

Setting the Measurement Parameters

```
>> ret = mexDLComStart(8,'10.0.236.100');
>> ret = mexDLSend( 'STOP' );
>> ret = mexDLSend( 'CHANNEL:VDIV 500mV' );
>> ret = mexDLSend( 'ACQUIRE:MODE NORMAL; RLENGTH 1000' );
>> ret = mexDLComEnd;
```

2.3 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 18 )
% You may need to adjust parameters of the interface.


% Example-1:GPIB  ( address = 1 )
% ret = mexDLComStart(1,'1');
% Example-2:USBTMC (7, SirialNo=91K225902 )
% ret = mexDLComStart(7, '91K225902');
% Example-3:Ethernet(VXI-11) (8, address=10.12.202.105 )
ret = mexDLComStart(8, '10.12.202.105');

% 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
%
% Recieve 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 20 )
% You may need to adjust parameters of the interface.

% Example-1:GPIB  ( address = 1 )
% ret = mexDLComStart(1,'1');
% Example-2:USBTMC (7, SirialNo=91K225902 )
% ret = mexDLComStart(7, '91K225902');
% Example-3:Ethernet(VXI-11) (8, address=10.12.202.105 )
ret = mexDLComStart(8, '10.12.202.105');

% 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 );
ret = mexDLSetTimeout( 3000 );
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;
```

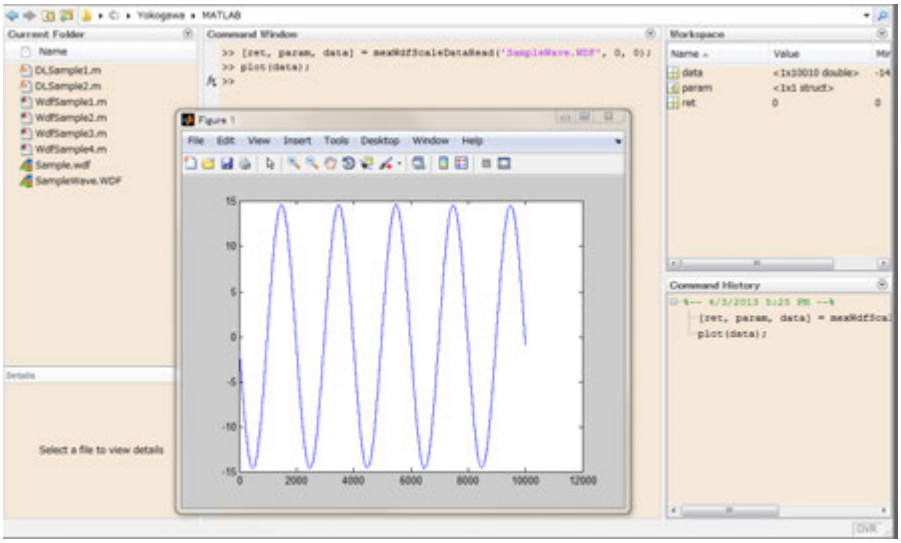
3.1 Using the MEX-Functions for Files

When you install this software program, the MEX-Functions for WDF files are also installed along with the MEX-Functions for DL control in the same folder.

3.2 Execution Example Using the MEX-Functions for Files

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

MATLAB Screen



Execution Example on the Command Window

```
>> [ret, param,data] = mexWdfScaleDataRead('SampleWave.WDF', 0, 0);
>> plot(data);
>>
```

3.3 Sample Programs

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.

There are four sample programs contained in the following files: WdfSample1.m to WdfSample4.m.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%  MATLAB M-File : WdfSample4.m
%  WDF file access sample script- 4
%
%  Copyright (C) Yokogawa Test & Measurement Corporation
%  Software Japan. All rights reserved.
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%
%   Input the Filename
%
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

mex Function Name	Function	Page
`[ret] = mexDLComS tart( wire, Adr );`	Initializes the line and connects the line to the specified device.	4-3
`[ret] = mexDLDeviceClear();`	Executes the clearing (SDC) of the selected device. A dedicated command for the GP-IB.	4-4
`[ret] = mexDLSend( Msg );`	Sends a message to the device.	4-4
`[ret] = mexDLSendByLength( Msg, len );`	Sends the specified number of bytes of the message to the device.	4-5
`[ret] = mexDLSendSetup();`	Prepares to send a message to the device.	4-5
`[ret] = mexDLSendOnly( Msg, len, end );`	Sends the specified number of bytes of the message to the device.	4-5
`[ret,buf,size] = mexDLReceive( blen );`	Receives a message from the device.	4-6
`[ret] = mexDLReceiveSetup();`	Prepares to retrieve a message from the device.	4-6
`[ret,buf,size] = mexDLReceiveOnly( blen );`	Receives a message (after preparation) from the device.	4-7
`[ret,length] = mexDLReceiveBlockHeader();`	Receives the header section of the block data sent from the device, and returns the number of data bytes that follow.	4-7
`[ret,buf,size,endFlag] = mexDLReceiveBlockData( blen );`	Receives the data section of the block data sent from the device.	4-8
`[endFlag] = mexDLCheckEnd();`	Returns whether the message from the device is finished. Can be used on the GP-IB, USB, or Ethernet interface.	4-8
`[ret] = mexDLSetRen( flag );`	Sets the device in remote or local mode. Use of an interface other than GP-IB is limited to Yokogawa products.	4-9
`[errorID] = mexDLGetLastError();`	Returns the number of the last error that occurred due to a MATLAB command.	4-10
`[ret] = mexDLSetTerm( eos, eot );`	Sets the terminator used in the message transmission/reception.	4-11
`[ret] = mexDLSetTimeout( tmo );`	Sets the communication timeout time.	4-11
`[ret,WaveData,Time] = mexDLGetWave();`	Retrieves the measured data.	4-12
`[ret,WaveData,Time] = mexDLGetWave( traceNum, waveStart, waveEnd );`	Specifies the trace number and waveform range and retrieves the waveform data.	4-13
`[ret,WaveData,Time] = mexDLGetWave( traceNum, subTraceNum, waveStart, waveEnd )`	This is a specialized function for retrieving data from DL850 series sub channels. Specify the trace (channel) number, sub trace (sub channel) number, and waveform range to retrieve the waveform data.	4-14
`[ret,HistoryWave,Time] = mexDLGetHistoryWave ( rec );`	Retrieves the history waveform data.	4-15
`[ret,HistoryWave,Time] =mexDLGetHistoryWave( rec, traceNum, waveStart, waveEnd );`	Specifies the record number, trace number and waveform range and retrieves the history waveform data.	4-16
`[ret,HistoryWave,Time] = mexDLGetHistoryWave( rec, traceNum, subTraceNum, waveStart, waveEnd )`	This is a specialized function for retrieving data from DL850 series sub channels. Specify the record number, trace (channel) number, sub trace (sub channel) number, and waveform range to retrieve the history waveform data.	4-17
`[ret] = mexDLComEnd();`	Closes the line connected to the device.	4-18
`mexDLToolkit;`	Displays the version information of the software.	4-18

[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

USB-TMC : wire = 7

VXI-11: Wire = 8

`Adr` Set the address 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)

USB-TMC: Adr = “Instruments serial number”

ETHERNET (VXI-11): Adr = “IP address”

Example:

```
ret = mexDLComStart(1,'1');
ret = mexDLComStart(7,'27E000001');
ret = mexDLComStart(8,'10.0.222.111');
```

[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.
When using USB-TMC (on instruments other than the DL9000), the USB-TMC-standard InitiateClear is executed. If InitiateClear succeeds, ClearFeature is executed, and then the process finishes.
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(':CANNEL 1: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 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, Time] = mexDLGetWave( );

Function: Retrieves the measured data.

Parameters: None

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

`WaveData` Matrix of waveform data

`Time` `Time from the trigger point (double, can be omitted)`

Description: Return value description

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

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

Time The time from the trigger point for each WaveData point is stored in an array. Time can be omitted.

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.
  - 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.
- Real time math waveform data of the DL850 series cannot be retrieved.

[ret, WaveData, Time] = 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)
`Time` Time from the trigger point (double, can be omitted)

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 cannt be retrieved.

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

`Time` The time from the trigger point for each WaveData point is stored in an array. Time can be omitted.

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

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

Function: This is a specialized function for retrieving data from DL850 series sub channels. Specify the trace (channel) number, sub trace (sub channel) number, and waveform range to retrieve the waveform data. This can only be used on DL850 series multichannel modules (modules that have sub channels). If you try to use this command on a module that does not have sub channels, an error will occur.

Parameters:
- `traceNum` Channel number of the waveform for retrieving the data
- `subTraceNum` Sub channel 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)
- `Time` Time from the trigger point (double, can be omitted)

Details: Parameter description

`traceNum` Specify the channel number. The smallest value is 1. The largest value depends on the number of channels on the connected model. Channel numbers of hidden channels cannot be specified.

`subTraceNum` Specify the sub channel number. The largest value depends on the number of sub channels that the installed multichannel module has. The smallest value is 1. Sub channel numbers of hidden sub 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 omit only one of the parameters). When these parameters are omitted, the entire waveform data of the displayed record length is retrieved.

Return value description

`WaveData` The waveform data of the specified sub channel number in the specified range within the display record length is stored in a matrix. The data of hidden channels cannot be etrieved.

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

`Time` The time from the trigger point for each WaveData point is stored in an array. Time can be omitted.

Example: When SubCH1, SubCH2, and SubCH3 of CH1 are shown on the display (with a display record length of 10 kW)

```
[ret,WaveData] = mexDLGetWave(0);
```

%Retrieve all the waveform data of SubCH1, SubCH2, and SubCH3 of CH1.

```
[ret,WaveData] = mexDLGetWave(1,1);
```

%Retrieve all the waveform data of SubCH1 of CH1.

```
[ret,WaveData] = mexDLGetWave(1,1,1000,4000);
```

%Retrieve the waveform data between 1 kW to 4 kW on SubCH1 of CH1.

[ret, HistoryWave, Time] = 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

`Time` Time from the trigger point (double, can be omitted)

Description: Return value description

`HistoryWave` The waveform data of the specified record number are stored to a matrix for each displaye channel.

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

`Time` The time from the trigger point for each HistryWaveData point is stored in an array. Time can be omitted.

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 all the –20th history waveform data.

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
  1. Right-click My Computer and choose Properties.
  2. Select the Advanced tab and select Performance Options.
  3. Select the Advanced tab, and then click Change under Virtual memory.
  4. Change the size of the virtual memory.
- 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, Time] = 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)

`Time` Time from the trigger point (double, can be omitted)

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 notretrieved.

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

`Time` The time from the trigger point for each HistryWaveData point is stored in an array. Time can be omitted.

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

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

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

Function: This is a specialized function for retrieving data from DL850 series sub channels. Specify the record number, trace (channel) number, sub trace (sub channel) number, and waveform range to retrieve the history waveform data. This can only be used on DL850 series multichannel modules (modules that have sub channels). If you try to use this command on a module that does not have sub channels, an error will occur.

Parameters:
`rec` Record number
`traceNum` Channel number of the waveform for retrieving the data
`subTraceNum` Sub channel 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)
`Time` Time from the trigger point (double, can be omitted)

Details:
Parameter description
`rec` This is the record number. The newest (current) waveform is 0, the waveform previous to that is ,àí1, and so on. For the selectable range, see the respective DL User's Manual.
`traceNum` Specify the channel number. The smallest value is 1. The largest value depends on the number of channels on the connected model. Channel numbers of hidden channels cannot be specified.
`subTraceNum` Specify the sub channel number. The largest value depends on the number of sub channels that the installed multichannel module has. The smallest value is 1. Sub channel numbers of hidden sub 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 omit only one of the parameters). When these parameters are omitted, the entire waveform data of the displayed record length is retrieved.
Return value description
`HistoryWave` The waveform data of the specified record number is stored to a matrix for each displayed channel. The data of hidden channels is no retrieved.

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

`Time` The time from the trigger point for each HistoryWave point is stored in an array. Time can be omitted.

Example:
When SubCH1, SubCH2, and SubCH4 of CH1 are shown on the display (with a display record length of 10 kW)

```
[ret,HistoryWave] = mexDLGetHistoryWave(-20,0);
```

%Retrieve all the –20th history waveform data of SubCH1, SubCH2, and SubCH4 of CH1.

```
[ret,HistoryWave] = mexDLGetHistoryWave(-20,1,4);
```

%Retrieve all the –20th history waveform data of SubCH4 of CH1.

```
[ret,HistoryWave] = mexDLGetHistoryWave(-20,1,4,2000,5000);
```

%Retrieve the data between 2 kW to 5 kW of the –20th history waveform data of SubCH4 of CH1.

[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 Meters & Instruments Corporation
```

4.2 MEX-Functions for WDF Files

4.2.1 List of MEX-Functions for WDF Files

Accessing file information

mex Function Name	Function	Page
mexWdfItemRead	Read the file information	4-20
mexWdfGetBlockNum	Read the number of blocks	4-20
mexWdfGetChNum	Read the number of channels	4-20

Data Operation

mex Function Name	Function	Page
mexWdfDataRead	Read the raw waveform data	4-21
mexWdfDataReadEx	Read the raw waveform data (expanded version)	4-21
mexWdfScaleDataRead	Read the physical value waveform data	4-22
mexWdfScaleDataReadEx	Read the physical value waveform data (expanded version)	4-22

4.2.2 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.	

4.2.3 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.	

4.2.4 Parameters and structure

Parameters

Item entered in the itemName area	dataType	Parameters		Meaning
		ch	block	
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

4.2.5 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