User's Manual

Model 707747
Control Toolkit for MATLAB

YOKOGAWA
Yokogawa Meters & Instruments Corporation

IM 707747-61E
1st Edition

Foreword

Thank you for purchasing the Control Toolkit for MATLAB (Model 707747).
This user's manual describes the installation procedure, the program model, and the functions of the Control Toolkit for MATLAB. Read this manual along with the Model 707741 WE Control API User's Manual (IM 707741-61E).
After reading the manual, keep it in a convenient location for quick reference whenever a question arises during operation.

Notes

- The contents of this manual describe the Control Toolkit for MATLAB Ver. 1.03. If you are using another version of the Control Toolkit for MATLAB, the information given in this manual may differ from the version that you are using.
- The contents of this manual are subject to change without prior notice as a result of continuing improvements to the instrument's performance and functions.
- 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.

Trademarks

- MATLAB is a registered trademark of The MathWorks, Inc. in the United States.
- Microsoft, Windows, and Windows NT are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries.
- Adobe and Acrobat are trademarks of Adobe Systems incorporated.
- All other company and product names used in this manual are trademarks or registered trademarks of their respective companies.

Revisions

1st Edition: April 2003

1st Edition : April 2003 (YK)

Terms and Conditions of the Software License

NOTICE - PLEASE READ CAREFULLY BEFORE USE

Thank you very much for purchasing this medium containing a software program and related documentation provided by Yokogawa Electric Corporation (hereinafter called "Yokogawa"), and the program contained, embedded, inserted or used in the medium (hereinafter called the "Yokogawa Software Program").

By opening this package or plastic wrapping (hereinafter called "Package") enclosing the Yokogawa Software Program, you acknowledge that you understand and agree to the "Terms and Conditions of the Software License" (hereinafter called "Terms and Conditions") which is written in the documentation and separately attached. Accordingly, the Terms and Conditions bind you.

The Yokogawa Software Program and its related documentation including ownership of copyright shall remain the exclusive property of Yokogawa or those third parties from whom sublicensed software in the Yokogawa Software Program is licensed.

Yokogawa hereby grants you permission to use the Yokogawa Software Program on the conditions that you agree to the Terms and Conditions before you open the Package and/or install it in or onto a computer.

IF YOU DO NOT AGREE TO THE TERMS AND CONDITIONS, YOU CANNOT OPEN THE PACKAGE, AND MUST IMMEDIATELY RETURN IT TO YOKOGAWA OR ITS DESIGNATED PARTY.

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: Model 707747 Control Toolkit for MATLAB
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.

How to Read This Manual

This manual covers only the settings that are specific to the software and functions for MATLAB. The manual has been prepared with the premise that it be read along with the Model 707741 WE Control API User's Manual (IM 707741-61E).
For details on the settings of functions and settings of ASCII commands, see the WE Control API User's Manual (IM 707761-61E).

Contents

1. Overview

The Control Toolkit for MATLAB is a toolkit based on the interface functions (WE Control API) for controlling the WE7000. By using this toolkit, software applications on MATLAB by MathWorks, Inc. can be created.

mex Function Groups that are Installed

The WE Control Toolkit for MATLAB consists of the following mex function groups. Each mex function corresponds to a single WE Control API function (one-to-one correspondence).

- **Initialization**
  Enables you to select the communication method or set communication parameters according to the communication module that you are using. Initializes the measuring station.
- **Handle/Link**
  Retrieves the station handle, retrieves the module handle, retrieves the link handle, and releases the handle.
- **Measuring Station Control**
  Retrieves a list of measuring station names or information, controls the standby power of the measuring station, restarts the measuring station, and so on.
- **Module Control**
  Retrieves the module information.
- **Settings**
  Initializes the current settings, initializes preset values, saves/updates/copies the setup data, sets or retrieves the scale conversion information, sets various settings, and retrieves the setup data.
- **Synchronization between Modules**
  Sets the trigger bus of the measuring station, sets the input/output of the EXT I/O connector, sets the input and polarity of the TRIG IN terminal, sets the time base, sets the arming signal, and retrieves settings.
- **GUI Control**
  Shows/Hides the module operation panel, shows/hides the trigger setting dialog box, and shows/hides the scale conversion setting dialog box.
- **Measurement Control**
  Starts/Stops/Single-start of measurement module operation, issues latch commands, retrieves acquisition data, retrieves instantaneous data, retrieves data after scale conversion, retrieves automated measurement values of waveform parameters, saves/loads pattern data, and sets overrun detection.
- **Waveform Parameter Computation**
  Executes waveform parameter computation and waveform parameter computation using raw data.
- **Filter API for the WE7281 4-CH, 100kS/s D/A Module**
  Converts the waveform data of a specified file into s16, w32, w7281 format.
- **Others**
  Queries whether the measurement of the block has been completed.

In addition, the file operation functions are included.

Supported OSs

Microsoft Windows 95/98/Me, Windows NT 4.0, Windows 2000 Pro or Windows XP Professional/Home Edition

Development Environments Supported

MATLAB 6.1 (R12.1) or 6.5 (R13)

Note

The WE Control API (707741) sold separately is required for using this software.

Installed Files

When you install the Control Toolkit for MATLAB, the directory \YOKOGAWA\WE7000 is created under the directory where MATLAB is saved (the default directory for MATLAB 6.5 is C:\MATLAB6p5\toolbox), and the following files are copied to the directory.

IM707747-61E.pdf	User's Manual
*.dll	Mex function for WE control
*.m, *.fig	Sample programs
readme *.txt	Information about the sample programs

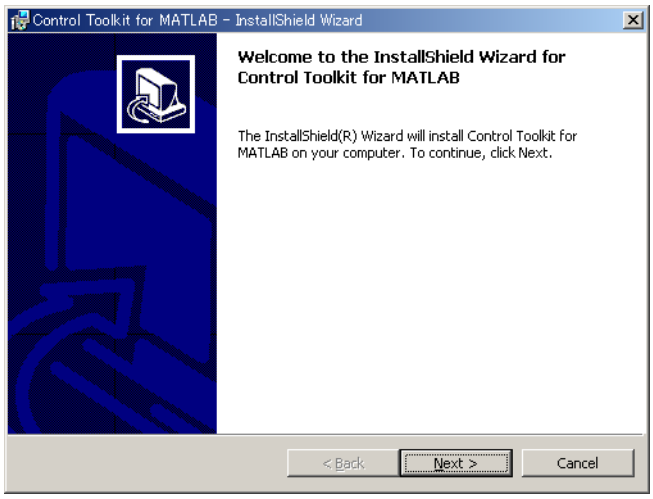
Note

If MATLAB is installed to a directory other than the default directory, change the directory in which the toolkit is installed accordingly.

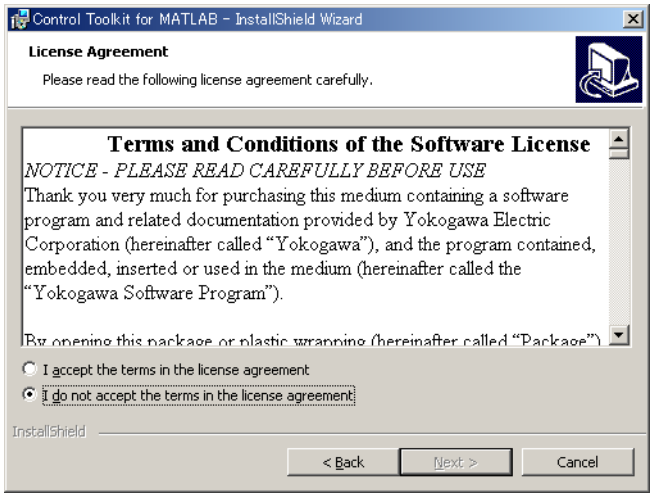
2. Installation

This chapter explains the procedure for installing the Control Toolkit for MATLAB.
WE Control API must be installed before you install the Control Toolkit for MATLAB.

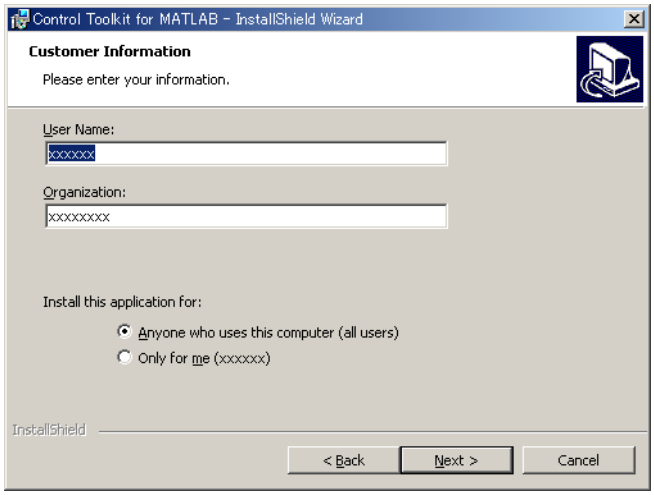
1. Start Windows.
2. Insert the “Control toolkit for MATLAB” setup disk into the CD-ROM drive. An installer automatically starts and the following dialog box opens. If the program does not start automatically, choose Start > Run, then type setup.exe in the Name box and click **OK**. The following dialog box appears. Click **Next**.



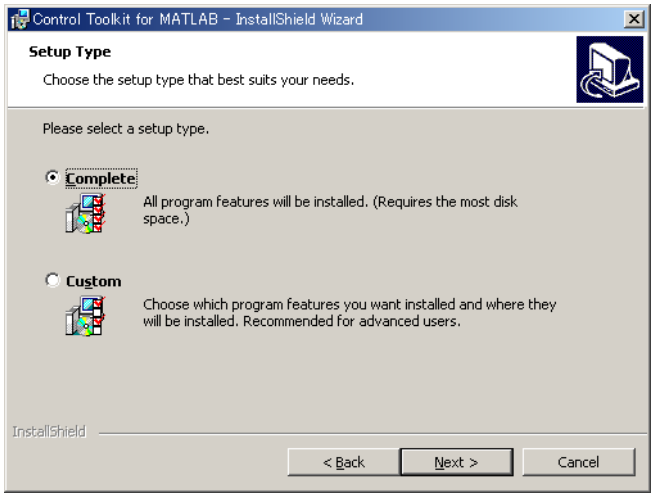
3. The following dialog box appears containing license agreement information. Confirm the license agreement, click the I accept the terms in the license agreement option button, and click **Next**.



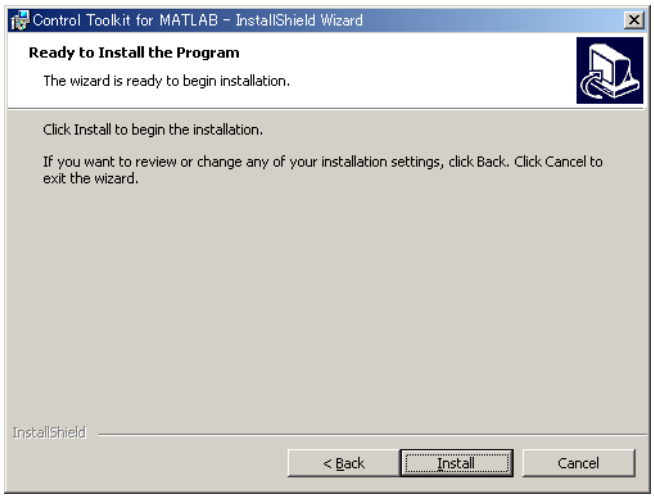
4. The following dialog box appears for registering the name and the organization of the user. After entering the appropriate information into each box, click **Next**.



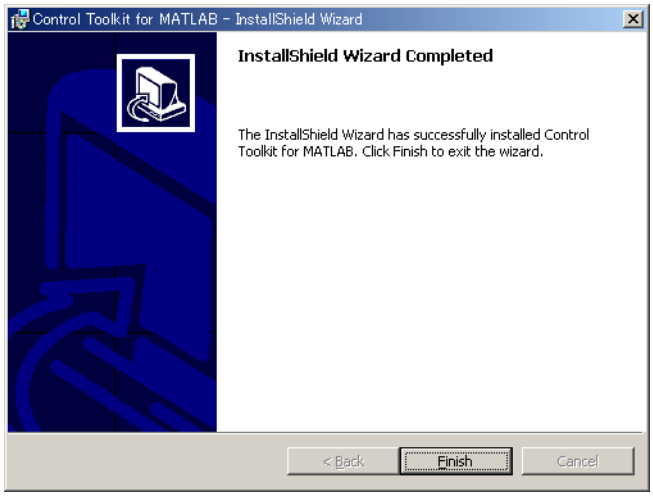
5. A dialog box appears for you to confirm the start of the installation. To proceed with the default installation (Complete), click **Next**. To select which components to install, or to change the installation destination, choose the Custom option then click **Next**.



6. A dialog box appears confirming that you wish to begin installation. Click **Install**.



7. The installation starts and a dialog box appears indicating the progress of the installation. A dialog box appears notifying you that the installation has been completed. Click **Finish**.



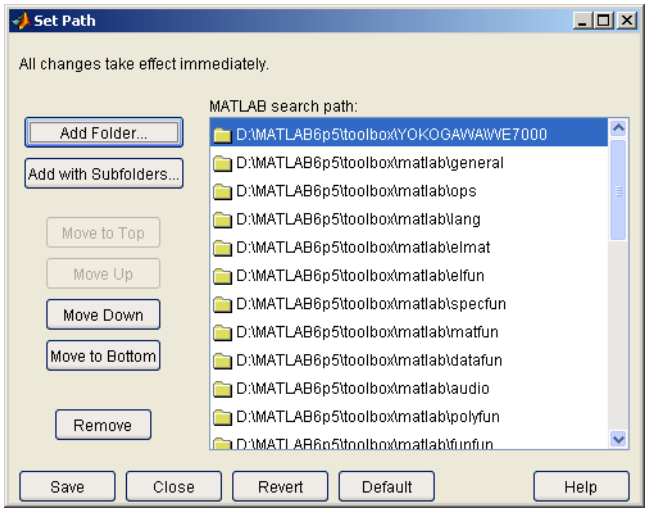
Note

If you installed MATLAB to a directory other than the default directory, change the installation directory of the Control Toolkit for MATLAB.

3. Setting the Path

The path defines the location where MATLAB automatically searches for files. The path is specified by starting MATLAB after installing the Control Toolkit. Normally, MATLAB searches for files in the current directory when a program or command is executed. If the file does not exist in the current directory, the path is searched.

1. From the File menu, choose Set Path or type “pathtool” in the command prompt window. The Set Path dialog box opens.

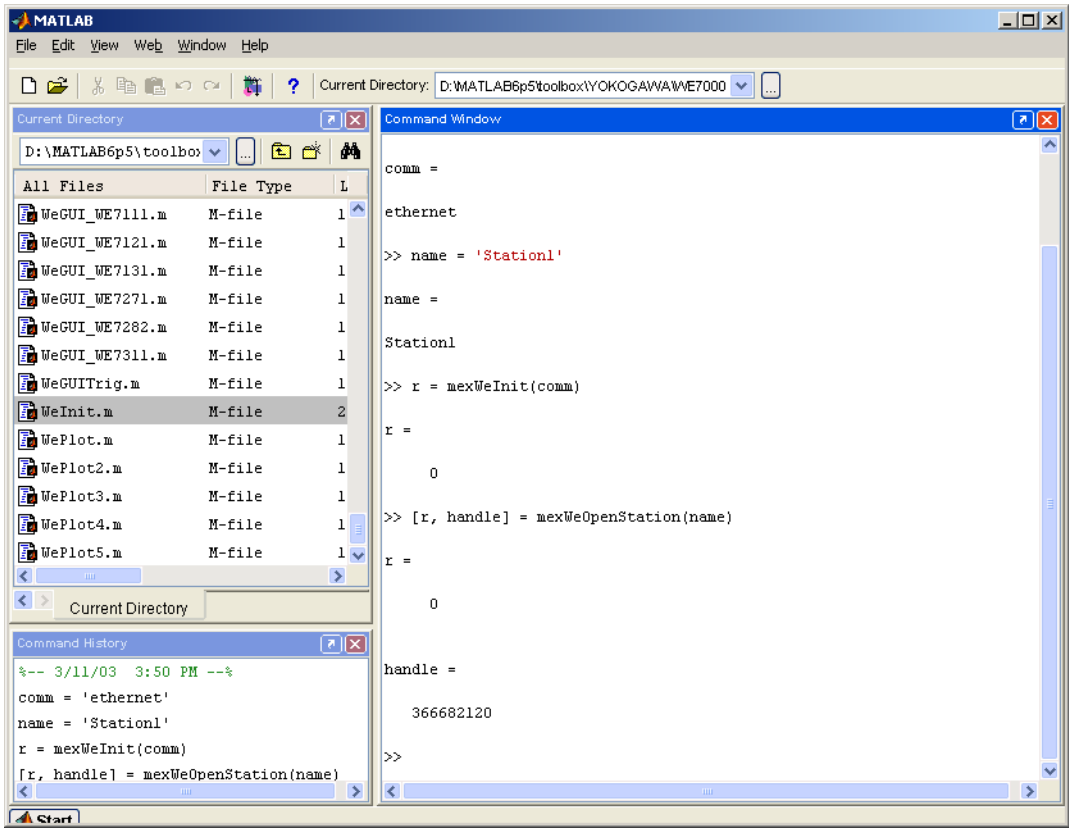


2. Click **Add Folder** to specify the installation destination folder that was specified during installation (C:\MATLAB6p1\toolbox\YOKOGAWA\WE7000 or C:\MATLAB6p5\toolbox\YOKOGAWA\WE7000 by default).
3. Return to the Set Path dialog box and click **Save** followed by **Close** to complete the setting.

4. Executing the Control Toolkit for MATLAB

On the Control Toolkit for MATLAB, programming is done interactively, which is a feature of MATLAB. Below is an example of function usage on the MATLAB command window and M-files.

MATLAB Screen



Execution Example on the Command Window

```
>> comm = 'ethernet'
comm =
ethernet
>> name = 'Station1'
name =
Station0
>> r = mexWeInit(comm)
r =
     0
>> [r, handle] = mexWeOpenStation(name)
r =
     0
handle =
   513138704
>> sw = 1
sw =
     1
>> r = mexWePower(handle, sw)
r =
     0
>> module = 'WE7271'
module =
WE7271
>> [r, mh] = mexWeOpenModule(handle, module, 1)
r =
     0
mh =
   361272136
>> [r, wh] = mexWeShowModuleWindow(mh)
r =
     0
wh =
      459594
>>
```

M-file Example

```
comm = 'ethernet'
name = 'Station1'
r = mexWeInit(comm)
[r, handle] = mexWeOpenStation(name)
sw = 1
r = mexWePower(handle, sw)
mh7121 = 0
mh7282 = 0
mh = 0
module = 'WE7271'
[r, mh] = mexWeOpenModule(handle, module, 1)
[r, wh] = mexWeShowModuleWindow(mh)
module = 'WE7281';
if mh7282==0 % Skip if already open
   [r, mh7282] = mexWeOpenModule(handle, module, 1);
   r = mexWeSetControl(mh7282, 'Operation Mode', 'FG');
   r = mexWeSetControl(mh7282, 'FG:CH1:Func', 'Sine');
   r = mexWeSetControl(mh7282, 'FG:CH1:Freq Start', '10');
   r = mexWeSetControl(mh7282, 'FG:CH1:On', 'On');
   r = mexWeSetControl(mh7282, 'FG:Output', 'On');
end
[r, sw] = mexWeIsModuleWindow(mh7282);
if sw==0 % Open module window if closed
   [r, wh7282] = mexWeShowModuleWindow(mh7282);
end

module = 'WE7271';
if mh==0 % Skip if already open
   [r, mh] = mexWeOpenModule(handle, module, 1);
   r = mexWeSetControl(mh, 'Sampling Interval', '1E-3');
   r = mexWeSetControl(mh, 'Memory Partition', '2');
end
[r, sw] = mexWeIsModuleWindow(mh);
if sw==0 % Open module window if closed
   [r, wh] = mexWeShowModuleWindow(mh);
end
% Set the scaling parameters
para = struct('a', 1.0, 'b', 0.0, 'ch', 0);
para(1).a = 1.0;
para(1).b = 0.0;
para(2).a = 1.0;
para(2).b = 0.0;
para(3).a = 1.0;
para(3).b = 0.0;
para(4).a = 1.0;
para(4).b = 0.0;
% Set the X-axis value
X(1:1000) = 0:0.001:0.999;
N = 30; % Set the repetition count
for n=1:N
   % Single start
   r = mexWeStartSingle(mh, 30);
   % Retrieve raw data
   [r, outPara, recSize, buf] = mexWeGetScaleData(mh, -1, 0, para, 32000, 50);
```

```
    Y1(1:1000) = buf(1:1000);
    Y2(1:1000) = buf(1001:2000);
    Y3(1:1000) = buf(2001:3000);
    Y4(1:1000) = buf(3001:4000);
    % Display on graph
    plot(X,Y1,X,Y2,X,Y3,X,Y4)
    drawnow
end
```

5. Details of Functions

The function names obtained by removing “mex” from the mex function names correspond to the WE Control API functions.
For details on mex functions, see chapter 6, “Details of Functions” in the WE Control API User’s Manual (IM 707741-61E). However, there is no API function that corresponds to mexWeIsBlockEnd.

5.1 The List of Functions

Initialization

mex Function Name	API Function Name	Description	Page
mexWeInit	WeInit	Initialization.	5-5
mexWeExit	WeExit	Termination.	5-5

Handle/Link

mex Function Name	API Function Name	Description	Page
mexWeOpenStation	WeOpenStation	Get the station handle.	5-6
mexWeOpenModule	WeOpenModule	Get the module handle.	5-6
mexWeLinkStation	WeLinkStation	Get the station link handle.	5-6
mexWeLinkModule	WeLinkModule	Get the module link handle.	5-7
mexWeCloseHandle	WeCloseHandle	Release handles.	5-7

Station Control

mex Function Name	API Function Name	Description	Page
mexWeGetStationList	WeGetStationList	Get the list of station names.	5-8
mexWeGetStationInfo	WeGetStationInfo	Get station information.	5-8
mexWePower	WePower	Turn ON/OFF the standby power to the measuring station.	5-8
mexWeRestart	WeRestart	Restart the measuring station.	5-9
mexWeSetStationName	WeSetStationName	Set the station name and the comment.	5-9
mexWeGetStationName	WeGetStationName	Get the station name and the comment.	5-9
mexWeIdentifyStation	WeIdentifyStation	Identify the measuring station.	5-10
mexWeGetPower	WeGetPower	Get the standby power ON/OFF state of the measuring station.	5-10

Module Control

mex Function Name	API Function Name	Description	Page
mexWeGetModuleInfo	WeGetModuleInfo	Get module information.	5-11

Settings

mex Function Name	API Function Name	Description	Page
mexWeInitSetup	WeInitSetup	Initialize the current settings.	5-12
mexWeInitPreset	WeInitPreset	Replace preset values with default values.	5-12
mexWeSaveSetup	WeSaveSetup	Save the current setup data to a file or update the preset values with the current settings.	5-12
mexWeLoadSetup	WeLoadSetup	Load the setup data and update the current settings or update the preset values.	5-13
mexWeCopySetup	WeCopySetup	Copy the setup data between modules.	5-13
mexWeCopyChSetup	WeCopyChSetup	Copy the setup data between channels.	5-13
mexWeCopyChSetupEx	WeCopyChSetupEx	Copy the setup data between channels (extended).	5-14
mexWeSetControl	WeSetControl	Set setup data (character string only).	5-14
mexWeGetControl	WeGetControl	Get setup data (character string only).	5-14
mexWeSetControlEx	WeSetControlEx	Set setup data (extended).	5-15
mexWeGetControlEx	WeGetControlEx	Get setup data (extended).	5-15
mexWeSetScaleInfo	WeSetScaleInfo	Set scale conversion information.	5-16
mexWeGetScaleInfo	WeGetScaleInfo	Get scale conversion information.	5-16

Synchronization between Modules

mex Function Name	API Function Name	Description	Page
mexWeExecManualTrig	WeExecManualTrig	Generate a manual trigger.	5-17
mexWeSetModuleBus	WeSetModuleBus	Set the trigger source/time base source/arming.	5-17
mexWeGetModuleBus	WeGetModuleBus	Get the trigger source/time base source/arming settings.	5-18
mexWeSetTrigBusLogic	WeSetTrigBusLogic	Set the trigger bus logic.	5-18
mexWeGetTrigBusLogic	WeGetTrigBusLogic	Get the trigger bus logic.	5-19
mexWeSetEXTIO	WeSetEXTIO	Set the input/output of the trigger/time base I/O terminal of the EXT. I/O connector.	5-19
mexWeGetEXTIO	WeGetEXTIO	Get the input/output setting of the trigger/time base I/O terminal of the EXT. I/O connector.	5-20
mexWeSetTRIGIN	WeSetTRIGIN	Set the input and polarity of the trigger signal entering the TRIG IN terminal.	5-20
mexWeGetTRIGIN	WeGetTRIGIN	Get the input and polarity of the trigger signal entering the TRIG IN terminal.	5-21
mexWeSetClockBusSource	WeSetClockBusSource	Set the time base.	5-21
mexWeGetClockBusSource	WeGetClockBusSource	Get the time base settings.	5-22
mexWeSetRcvTrigPacket	WeSetRcvTrigPacket	Set the receive station for trigger packets.	5-22
mexWeSetSndTrigPacket	WeSetSndTrigPacket	Set the source station for trigger packets.	5-23
mexWeFireTrigPacket	WeFireTrigPacket	Generate a trigger packet.	5-23
mexWeSetSndClockPacket	WeSetSndClockPacket	Set the source station for time base packets.	5-23
mexWeSetRcvClockPacket	WeSetRcvClockPacket	Set the receive station for time base packets.	5-24
mexWeFireClockPacket	WeFireClockPacket	Issue a time base packet.	5-24
mexWeOutputEXTIOEvent	WeOutputEXTIOEvent	Set the event output of the EXT. I/O connector.	5-24
mexWeExecManualArming	WeExecManualArming	Generate an arming signal.	5-25
mexWeSetArmingSource	WeSetArmingSource	Set the arming source.	5-25
mexWeGetArmingSource	WeGetArmingSource	Get the arming source setting.	5-25

GUI Control

mex Function Name	API Function Name	Description	Page
mexWeShowModuleWindow	WeShowModuleWindow	Show the module operation panel.	5-26
mexWeCloseModuleWindow	WeCloseModuleWindow	Close the module operation panel.	5-26
mexWeIsModuleWindow	WeIsModuleWindow	Get the show/hide status of the module operation panel.	5-26
mexWeShowTrigWindow	WeShowTrigWindow	Show the trigger setting dialog box.	5-27
mexWeCloseTrigWindow	WeCloseTrigWindow	Close the trigger setting dialog box.	5-27
mexWeIsTrigWindow	WeIsTrigWindow	Get the show/hide status of the trigger setting dialog box.	5-27
mexWeShowLinearScaleWindow	WeShowLinearScaleWindow	Show the scale conversion setting dialog box.	5-28
mexWeCloseLinearScaleWindow	WeCloseLinearScaleWindow	Close the scale conversion setting dialog box.	5-28
mexWeIsLinearScaleWindow	WeIsLinearScaleWindow	Get the show/hide status of the scale conversion setting dialog box.	5-28

Measurement Control

mex Function Name	API Function Name	Description	Page
mexWeStart	WeStart	Start the measurement module operation.	5-29
mexWeStop	WeStop	Stop the measurement module operation.	5-29
mexWeStartEx	WeStartEx	Start the measurement module operation (extended).	5-29
mexWeStartSingle	WeStartSingle	Single data acquisition.	5-30
mexWeIsRun	WeIsRun	Get the execution status (run/stop) of the measurement module.	5-30
mexWeLatchData	WeLatchData	Issue a latch command.	5-30
mexWeGetAcqDataInfo	WeGetAcqDataInfo	Get acquisition data information.	5-31
mexWeGetAcqDataSize	WeGetAcqDataSize	Get the acquisition data size.	5-31
mexWeGetAcqData	WeGetAcqData	Get acquisition data (raw data).	5-32
mexWeGetAcqDataEx	WeGetAcqDataEx	Get acquisition data (extended).	5-33
mexWeGetCurrentData	WeGetCurrentData	Get instantaneous data.	5-34
mexWeGetScaleCurrentData	WeGetScaleCurrentData	Get the instantaneous data after scale conversion.	5-34
mexWeGetScaleCurrentDataEx	WeGetScaleCurrentDataEx	Get the instantaneous data after scale conversion (extended).	5-35
mexWeGetScaleData	WeGetScaleData	Get scaled data.	5-35
mexWeGetScaleDataEx	WeGetScaleDataEx	Get scaled data (extended).	5-36
mexWeGetMeasureParam	WeGetMeasureParam	Get automated measurement values of waveform parameters.	5-36
mexWeSaveAcqData	WeSaveAcqData	Save acquisition data (raw data).	5-37
mexWeSaveScaleData	WeSaveScaleData	Save scaled data.	5-37
mexWeSaveScaleDataEx	WeSaveScaleDataEx	Save scaled data (extended).	5-38
mexWeSaveAsciiData	WeSaveAsciiData	Save ASCII data.	5-38
mexWeSaveScaleAsciiData	WeSaveScaleAsciiData	Save scaled data to a file in ASCII format.	5-39
mexWeSaveScaleAsciiDataEx	WeSaveScaleAsciiDataEx	Save scaled data to a file in ASCII format (extended).	5-39
mexWeSaveAcqHeader	WeSaveAcqHeader	Save the header (waveform information) file.	5-40
mexWeSavePatternData	WeSavePatternData	Save the pattern data (arbitrary waveform data).	5-40
mexWeLoadPatternData	WeLoadPatternData	Load the pattern data (arbitrary waveform data).	5-40
mexWeLoadPatternDataEx	WeLoadPatternDataEx	Load the pattern data (arbitrary waveform data) (extended).	5-41
mexWeSetOverRun	WeSetOverRun	Enable/Disable overrun detection.	5-41
mexWeGetOverRun	WeGetOverRun	Get the overrun detection status.	5-41

Waveform Parameter Computation

mex Function Name	API Function Name	Description	Page
mexWeExecMeasureParam	WeExecMeasureParam	Execute waveform parameter computation.	5-42
mexWeExecMeasureParamAcqData	WeExecMeasureParamAcqData	Execute waveform parameter computation using raw data.	5-42

Filter API for the WE7281 4-CH, 100kS/s D/A Module

mex Function Name	API Function Name	Description	Page
mexWeWvf2S16GetSize	WeWvf2S16GetSize	Get the byte size when the waveform data in the specified file is converted to s16 format.	5-43
mexWeWvf2W32GetSize	WeWvf2W32GetSize	Get the byte size when the waveform data in the specified file is converted to w32 format.	5-43
mexWeWvf2W7281GetSize	WeWvf2W7281GetSize	Get the byte size when the waveform data in the specified file is converted to w7281 format.	5-44
mexWeWvf2S16	WeWvf2S16	Convert the specified file to s16 format.	5-44
mexWeWvf2W32	WeWvf2W32	Convert the specified file to w32 format.	5-45
mexWeWvf2W7281	WeWvf2W7281	Convert the specified file to w7281 format.	5-45

Others

mex Function Name	API Function Name	Description	Page
mexWeIsBlockEnd	None	Get the end status of the block data read of the measurement module.	5-46

Note

There are no mex functions that correspond to the following API functions.

WeSetQueryControl	Set the setup data and get the data.
WeStopEx	Stop the measurement module operation (extended).
WeStartWithEvent	Start with end notify event.
WeCreateEvent	Request generation of an event.
WeSetEventPattern	Set the factor that triggers the event.
WeResetEventPattern	Clear the factor that triggers the event.
WeSetEventMode	Set the event mode.
WeReleaseEvent	Release the event handle.
WeGetHandle	Get the handle from the second parameter of the event.
WeIsNan	Check whether the data is non-numeric.

5.2 Initialization

mexWeInit

Description

Initialization function. This function must be executed first when using the Control Toolkit for MATLAB from an application program. Execution of this function initializes the network, automatically identifies the stations that are connected, initializes the Control Toolkit execution environment, etc.

Syntax

ret = mexWeInit(comm)

Output Parameters

ret: Returns 0 if successful. Returns an error code if unsuccessful.

Input Parameters

comm: Communication interface type

mexWeExit

Description

Termination function. Terminates the communication driver and Control Toolkit execution environment. Be sure to execute this function at the end of the application program.

Syntax

ret = mexWeExit

Output Parameters

ret: Returns 0 if successful. Returns an error code if unsuccessful.

Input Parameters

None

5.3 Handle/Link

mexWeOpenStation

Description

Gets the station handle for controlling the measuring station by specifying the station name. You can also get the handle (broadcast handle) for controlling all the measuring stations on the network.

Syntax

[ret, stationHandle] = mexWeOpenStation(name)

Output Parameters

ret: Returns 0 if successful. Returns an error code if unsuccessful.
stationHandle: Station handle retrieved.

Input Parameters

name: Station name

mexWeOpenModule

Description

Gets the handle for controlling the module (opens the module).

Syntax

[ret, moduleHandle] = mexWeOpenModule(stationHandle, name, connection)

Output Parameters

ret: Returns 0 if successful. Returns an error code if unsuccessful.
moduleHandle: Module handle retrieved.

Input Parameters

stationHandle: Station handle
name: Module's product name [:number] or slot number
connection: The number of modules you wish to link.

mexWeLinkStation

Description

Gets the station link handle. This handle can be used to control multiple measuring stations simultaneously.

Syntax

[ret, stationLinkHandle] = mexWeLinkStation(num, stationHandle)

Output Parameters

ret: Returns 0 if successful. Returns an error code if unsuccessful.
stationLinkHandle: Station link handle retrieved.

Input Parameters

num: The number of stations you wish to link.
stationHandle: An array of the station handles of the stations you wish to link.

mexWeLinkModule

Description

Gets the module link handle. This handle can be used to control multiple modules simultaneously.

Syntax

[ret, moduleLinkHandle] = mexWeLinkModule(num, moduleHandle)

Output Parameters

ret: Returns 0 if successful. Returns an error code if unsuccessful.
moduleLinkHandle: Module link handle retrieved.

Input Parameters

num: The number of modules you wish to link.
moduleHandle: An array of the module handles of the modules you wish to link.

mexWeCloseHandle

Description

Releases the station handle, module handle, or link handle. Releasing the station handle also releases the module handles and module link handles within the measuring station.

Syntax

ret = mexWeCloseHandle(hHandle)

Output Parameters

ret: Returns 0 if successful. Returns an error code if unsuccessful.

Input Parameters

hHandle: Station handle, station link handle, module handle, or module link handle

5.4 Station Control

mexWeGetStationList

Description

Gets a list of measuring station names within the network.

Syntax

[ret, num, stationList] = mexWeGetStationList

Output Parameters

ret: Returns 0 if successful. Returns an error code if unsuccessful.
num: The number of measuring stations (including the controller) included in the list
stationList: Structure containing a list of measuring station names

Input Parameters

None

mexWeGetStationInfo

Description

Gets the measuring station information such as the installed module's product names and the number of channels per module.

Syntax

[ret, stationInfo] = mexWeGetStationInfo(stationHandle)

Output Parameters

ret: Returns 0 if successful. Returns an error code if unsuccessful.
stationInfo: Station information structure retrieved.

Input Parameters

stationHandle: Station handle

mexWePower

Description

Turns ON/OFF of the standby power of the measuring station.

Syntax

ret = mexWePower(stationHandle, sw)

Output Parameters

ret: Returns 0 if successful. Returns an error code if unsuccessful.

Input Parameters

stationHandle: Station handle
sw: Power state 0 = OFF, 1 = ON

mexWeRestart

Description

Restarts the measuring station. Executes a procedure similar to power-on reset. However, the communication module is not restarted.

Syntax

ret = mexWeRestart(hHandle)

Output Parameters

ret: Returns 0 if successful. Returns an error code if unsuccessful.

Input Parameters

hHandle: Station handle, station link handle, or broadcast handle.

mexWeSetStationName

Description

Sets the station name and the comment for the measuring station.

Syntax

ret = mexWeSetStationName(stationHandle, name, comment)

Output Parameters

ret: Returns 0 if successful. Returns an error code if unsuccessful.

Input Parameters

stationHandle: Station handle
name: Module's product name [:number] or slot number
comment: Comment

mexWeGetStationName

Description

Gets the station name and the comment for the measuring station.

Syntax

[ret, name, comment] = mexWeGetStationName(stationHandle)

Output Parameters

ret: Returns 0 if successful. Returns an error code if unsuccessful.
name: Station name
comment: Comment

Input Parameters

stationHandle: Station handle

mexWeIdentifyStation

Description

For measuring station identification, the LED of the optical interface module of the specified measuring station blinks. This is valid only when the optical interface module is being used as the communication interface.

Syntax

ret = mexWeIdentifyStation(stationHandle)

Output Parameters

ret: Returns 0 if successful. Returns an error code if unsuccessful.

Input Parameters

stationHandle: Station handle or station link handle

mexWeGetPower

Description

Gets the ON/OFF state of the standby power of the measuring station.

Syntax

[ret, sw] = mexWeGetPower(stationHandle)

Output Parameters

ret: Returns 0 if successful. Returns an error code if unsuccessful.
sw: Power supply state 0 = OFF, 1 = ON

Input Parameters

stationHandle: Station handle

5.5 Module Control

mexWeGetModuleInfo

Description

Gets the module information.

Syntax

[ret, moduleInfo] = mexWeGetModuleInfo(moduleHandle, position)

Output Parameters

ret: Returns 0 if successful. Returns an error code if unsuccessful.
moduleInfo: Extended module information structure retrieved.

Input Parameters

moduleHandle: Module handle
position: Link position of the module you wish to retrieve.

5.6 Settings

mexWeInitSetup

Description

Resets the current settings of the measuring station or module to default values.

Syntax

ret = mexWeInitSetup(hHandle, position)

Output Parameters

ret: Returns 0 if successful. Returns an error code if unsuccessful.

Input Parameters

hHandle: Module handle, module link handle, station handle, or station link handle.
position: Module link position

mexWeInitPreset

Description

Replaces preset values of the measuring station or module with default values.

Syntax

ret = mexWeInitPreset(hHandle, position)

Output Parameters

ret: Returns 0 if successful. Returns an error code if unsuccessful.

Input Parameters

hHandle: Module handle, module link handle, station handle, or station link handle.
position: Module link position

mexWeSaveSetup

Description

Saves the current setup data of the measuring station or module to a file or updates the preset values with the current settings. When saving to a file, the file is saved with .set extension.

Syntax

ret = mexWeSaveSetup(hHandle, position, filename)

Output Parameters

ret: Returns 0 if successful. Returns an error code if unsuccessful.

Input Parameters

hHandle: Module handle or station handle.
position: Module link position
filename: File name

mexWeLoadSetup

Description

Updates the current settings of the measuring station or module with the contents of the setup data file or preset values.

Syntax

ret = mexWeLoadSetup(hHandle, position, filename)

Output Parameters

ret: Returns 0 if successful. Returns an error code if unsuccessful.

Input Parameters

hHandle: Module handle or station handle.
position: Module link position
filename: File name

mexWeCopySetup

Description

Copies the current settings of the specified module to a specified module of the same type.

Syntax

ret = mexWeCopySetup(hSrcModule, hDesModule)

Output Parameters

ret: Returns 0 if successful. Returns an error code if unsuccessful.

Input Parameters

hSrcModule: Copy source module handle
hDesModule: Copy destination module handle.

mexWeCopyChSetup

Description

Copies the current setup data of a channel of a module to the specified channel.
This function is valid only when the modules are linked.

Syntax

ret = mexWeCopyChSetup(moduleHandle, srcCh, desCh)

Output Parameters

ret: Returns 0 if successful. Returns an error code if unsuccessful.

Input Parameters

moduleHandle: Module handle
srcCh: Channel number of the copy source
desCh: Channel number of the copy destination

mexWeCopyChSetupEx

Description

Copies the current setup data of a channel of a module to the specified channel.
This function can be used even when the modules are not linked.

Syntax

ret = mexWeCopyChSetupEx(moduleHandle, srcCh, desStartCh, desEndCh)

Output Parameters

ret: Returns 0 if successful. Returns an error code if unsuccessful.

Input Parameters

moduleHandle:	Module handle
srcCh:	Channel number of the copy source
desStartCh:	Channel number of the copy destination
desEndCh:	Last channel number of the copy destination

mexWeSetControl

Description

Sets the setup parameters of a module (character string only).

Syntax

ret = mexWeSetControl(moduleHandle, command, param)

Output Parameters

ret: Returns 0 if successful. Returns an error code if unsuccessful.

Input Parameters

moduleHandle:	Module handle or module link handle.
command:	ASCII command name (string)
param:	Parameter dependent on the command (string)

mexWeGetControl

Description

Gets the module information (character string only).

Syntax

[ret, param] = mexWeGetControl(moduleHandle, command)

Output Parameters

ret: Returns 0 if successful. Returns an error code if unsuccessful.
param:Parameter dependent on the command (string)

Input Parameters

moduleHandle:	Module handle
command:	ASCII command name (string)

mexWeSetControlEx

Description

Sets the setup parameters of a module. You can set the setup parameters by specifying the data type.

Syntax

ret = mexWeSetControlEx(moduleHandle, command, ptype, paramNum, param)

Output Parameters

ret: Returns 0 if successful. Returns an error code if unsuccessful.

Input Parameters

moduleHandle:	Module handle	
command:	ASCII command name (string)	
ptype:	255=WE_NULL	No parameter
	0=WE_UBYTE	8-bit unsigned integer
	1=WE_SBYTE	8-bit signed integer
	16=WE_UWORD	16-bit unsigned integer
	17=WE_SWORD	16-bit signed integer
	32=WE_ULONG	32-bit unsigned integer
	33=WE_SLONG	32-bit signed integer
	34=WE_FLOAT	32-bit real number
	50=WE_DOUBLE	64-bit real number
paramNum:	Number of data points	
param:	Data	

mexWeGetControlEx

Description

Gets the setup data of the module. You can retrieve the setup parameters by specifying the data type.

Syntax

[ret, param] = mexWeGetControlEx(moduleHandle, command, ptype, paramNum)

Output Parameters

ret: Returns 0 if successful. Returns an error code if unsuccessful.
param: Data

Input Parameters

moduleHandle:	Module handle	
command:	ASCII command name (string)	
ptype:	255=WE_NULL	No parameter
	0=WE_UBYTE	8-bit unsigned integer
	1=WE_SBYTE	8-bit signed integer
	16=WE_UWORD	16-bit unsigned integer
	17=WE_SWORD	16-bit signed integer
	32=WE_ULONG	32-bit unsigned integer
	33=WE_SLONG	32-bit signed integer
	34=WE_FLOAT	32-bit real number
	50=WE_DOUBLE	64-bit real number
paramNum:	Number of data points to be retrieved	

mexWeSetScaleInfo

Description

Sets scale conversion information to the measurement module. This function is equivalent to the settings in the convert scale dialog box. The information is stored to the module through operations such as update preset.

Syntax

ret = mexWeSetScaleInfo(moduleHandle, ch, scaleInfo)

Output Parameters

ret: Returns 0 if successful. Returns an error code if unsuccessful.

Input Parameters

moduleHandle:	Module handle
ch:	Channel number
scaleInfo:	Scale conversion table information structure

mexWeGetScaleInfo

Description

Gets scale conversion information of the measurement module.

Syntax

[ret, scaleInfo] = mexWeGetScaleInfo(moduleHandle, ch)

Output Parameters

ret: Returns 0 if successful. Returns an error code if unsuccessful.

Input Parameters

moduleHandle:	Module handle
ch:	Channel number
scaleInfo:	Scale conversion table information structure

5.7 Synchronization between Modules

mexWeExecManualTrig

Description

Generates a manual trigger signal to the trigger bus common to modules in the measuring station. This causes a trigger to be activated on modules whose trigger source is set to the trigger bus.

Syntax

ret = mexWeExecManualTrig(stationHandle, busNo, pulse)

Output Parameters

ret: Returns 0 if successful. Returns an error code if unsuccessful.

Input Parameters

stationHandle:	Station handle, station link handle, or broadcast handle.	
busNo:	1=WE_TRG1	BUSTRG1
	2=WE_TRG2	BUSTRG2
pulse:	0=WE_MANTRGDOWN	Manual trigger down
	1=WE_MANTRGUP	Manual trigger up
	2=WE_MANTRGONESHOT	Manual trigger one-shot

mexWeSetModuleBus

Description

Sets the trigger source/time base source (sampling clock) and the input/output setting of the arming signal of the module.
If a module that does not have trigger signal, time base, and arming signal input/output functions is specified, the setting is discarded.

Syntax

ret = mexWeSetModuleBus(moduleHandle, InItem, OutItem, InClock, ArmItem)

Output Parameters

ret: Returns 0 if successful. Returns an error code if unsuccessful.

Input Parameters

moduleHandle:	Module handle or module link handle.	
InItem:	0=WE_TRGNONE	No bus trigger
	1=WE_TRG1	Use bus trigger 1
	2=WE_TRG2	Use bus trigger 2
OutItem:	0=WE_TRGNONE	No bus trigger
	1=WE_TRG1	Use bus trigger 1
	2=WE_TRG2	Use bus trigger 2
	3=WE_BOTH	Use bus trigger 1 and 2
InClock:	0=WE_CMNCLKNONE	Not use the common clock
	1=WE_CMNCLK	Use the common clock
ArmItem:	0=WE_ARMNONE	Not use arming
	1=WE_ARM	Use arming

mexWeGetModuleBus

Description

Gets the trigger source/time base source (sampling clock) and the input/output setting of the arming signal of the module.
If a module is specified that does not have a trigger source, time base source, or an arming signal input/output function, a “0” is returned for each setting.

Syntax

[ret, InItem, OutItem, InClock, ArmItem] = mexWeGetModuleBus(moduleHandle)

Output Parameters

ret:	Returns 0 if successful. Returns an error code if unsuccessful.	
InItem:	0=WE_TRGNONE	No bus trigger
	1=WE_TRG1	Use bus trigger 1
	2=WE_TRG2	Use bus trigger 2
OutItem:	0=WE_TRGNONE	No bus trigger
	1=WE_TRG1	Use bus trigger 1
	2=WE_TRG2	Use bus trigger 2
	3=WE_BOTH	Use bus trigger 1 and 2
InClock:	0=WE_CMNCLKNONE	Not use the common clock
	1=WE_CMNCLK	Use the common clock
ArmItem:	0=WE_ARMNONE	Not use arming
	1=WE_ARM	Use arming

Input Parameters

moduleHandle: Module handle or module link handle.

mexWeSetTrigBusLogic

Description

Sets the trigger and/or condition of the trigger bus.
This function is valid when multiple modules are using the trigger bus. You can specify the trigger to occur when all trigger conditions are satisfied (AND operation) or when one of the trigger conditions is satisfied (OR operation).

Syntax

ret = mexWeSetTrigBusLogic(stationHandle, item, logic)

Output Parameters

ret: Returns 0 if successful. Returns an error code if unsuccessful.

Input Parameters

stationHandle:	Station handle or station link handle	
item:	1=WE_TRG1	BUSTRG1
	2=WE_TRG2	BUSTRG2
logic:	0=WE_TRGOR	Bus logic OR
	1=WE_TRGAND	Bus logic AND

mexWeGetTrigBusLogic

Description

Gets the trigger and/or condition of the trigger bus.

Syntax

[ret, logic] = mexWeGetTrigBusLogic(stationHandle, item)

Output Parameters

ret: Returns 0 if successful. Returns an error code if unsuccessful.
logic: 0=WE_TRGOR Bus logic OR
1=WE_TRGAND Bus logic AND

Input Parameters

stationHandle: Station handle or station link handle
item: 1=WE_TRG1 BUSTRG1
2=WE_TRG2 BUSTRG2

mexWeSetEXTIO

Description

Sets the input/output condition of the trigger input/output pin and the time base input/output pin of the external input/output connector (EXT. I/O) on the front panel of the measuring station.
These signal pins can input or output signals. These signal pins can be used to (1) pass the external signals to the trigger and clock buses in the station, (2) output the trigger signal or the clock, and (3) provide input signals for other measuring stations to achieve synchronization.

Syntax

[ret, logic] = mexWeGetTrigBusLogic(stationHandle, item)

Output Parameters

ret: Returns 0 if successful. Returns an error code if unsuccessful.
logic: 0=WE_TRGOR Bus logic OR
1=WE_TRGAND Bus logic AND

Input Parameters

stationHandle: Station handle or station link handle
item: 1=WE_TRG1 BUSTRG1
2=WE_TRG2 BUSTRG2

mexWeGetEXTIO

Description

Gets the input/output condition of the trigger input/output pin and the time base input/output pin of the external input/output connector (EXT. I/O) on the front panel of the measuring station.

Syntax

[ret, trig1, trig2, clock] = mexWeGetEXTIO(stationHandle)

Output Parameters

ret:	Returns 0 if successful. Returns an error code if unsuccessful.	
trig1:	0=WE_EXTIOTRGOUT	EXTIO connector output
	1=WE_EXTIOTRGIN	EXTIO connector input
trig2:	0=WE_EXTIOTRGOUT	EXTIO connector output
	1=WE_EXTIOTRGIN	EXTIO connector input
clock:	0=WE_CMNCLKOUT	Set to output pin.
	1=WE_CMNCLKIN	Set to input pin.

Input Parameters

stationHandle: Station handle or station link handle

mexWeSetTRIGIN

Description

Sets the trigger signal input from the external trigger input terminal (TRIG IN) on the front panel of the measuring station to the trigger bus and the polarity of the input signal.

Syntax

ret = mexWeSetTRIGIN(stationHandle, item, polarity)

Output Parameters

ret: Returns 0 if successful. Returns an error code if unsuccessful.

Input Parameters

stationHandle:	Station handle or station link handle	
item:	0=WE_TRGNONE	No bus trigger
	1=WE_TRG1	Use bus trigger 1
	2=WE_TRG2	Use bus trigger 2
	3=WE_BOTH	Use bus trigger 1 and 2
polarity:	0=WE_TRGPOS	TRGIN pin positive polarity
	1=WE_TRGNEG	TRGIN pin negative polarity

mexWeGetTRIGIN

Description

Gets the trigger signal input setting from the external trigger input terminal (TRIG IN) on the front panel of the station to the trigger bus and the polarity setting of the input signal.

Syntax

[ret, item, polarity] = mexWeGetTRIGIN(stationHandle)

Output Parameters

ret:	Returns 0 if successful. Returns an error code if unsuccessful.	
item:	0=WE_TRGNONE	No bus trigger
	1=WE_TRG1	Use bus trigger 1
	2=WE_TRG2	Use bus trigger 2
	3=WE_BOTH	Use bus trigger 1 and 2
polarity:	0=WE_TRGPOS	TRGIN pin positive polarity
	1=WE_TRGNEG	TRGIN pin negative polarity

Input Parameters

stationHandle: Station handle or station link handle

mexWeSetClockBusSource

Description

Sets the time base source that is output to the clock bus (CMNCLK). Only a single source can be specified for the time base source.

Syntax

ret = mexWeSetClockBusSource(stationHandle, source)

Output Parameters

ret: Returns 0 if successful. Returns an error code if unsuccessful.

Input Parameters

stationHandle:	Station handle or station link handle	
source:	0=WE_CMNCLKSRC_NONE	No bus clock source
	1=WE_CMNCLKSRC_TRIGIN	Input from the TRIG IN terminal
	2=WE_CMNCLKSRC_EXTIO	Input time base from the EXT I/O connector
	3=WE_CMNCLKSRC_SLOT0	Slot 0
	4=WE_CMNCLKSRC_SLOT1	Slot 1
	5=WE_CMNCLKSRC_SLOT2	Slot 2
	6=WE_CMNCLKSRC_SLOT3	Slot 3
	7=WE_CMNCLKSRC_SLOT4	Slot 4
	8=WE_CMNCLKSRC_SLOT5	Slot 5
	9=WE_CMNCLKSRC_SLOT6	Slot 6
	10=WE_CMNCLKSRC_SLOT7	Slot 7
	11=WE_CMNCLKSRC_SLOT8	Slot 8

mexWeGetClockBusSource

Description

Gets the time base source setting.

Syntax

[ret, source] = mexWeGetClockBusSource(stationHandle)

Output Parameters

ret:	Returns 0 if successful. Returns an error code if unsuccessful.	
source:	0=WE_CMNCLKSRC_NONE	No bus clock source
	1=WE_CMNCLKSRC_TRIGIN	Input from the TRIG IN terminal
	2=WE_CMNCLKSRC_EXTIO	Input time base from the EXT I/O connector
	3=WE_CMNCLKSRC_SLOT0	Slot 0
	4=WE_CMNCLKSRC_SLOT1	Slot 1
	5=WE_CMNCLKSRC_SLOT2	Slot 2
	6=WE_CMNCLKSRC_SLOT3	Slot 3
	7=WE_CMNCLKSRC_SLOT4	Slot 4
	8=WE_CMNCLKSRC_SLOT5	Slot 5
	9=WE_CMNCLKSRC_SLOT6	Slot 6
	10=WE_CMNCLKSRC_SLOT7	Slot 7
	11=WE_CMNCLKSRC_SLOT8	Slot 8

Input Parameters

stationHandle: Station handle or station link handle

mexWeSetRcvTrigPacket

Description

Sets the receive station for trigger packets.
Up to 8 receiving stations can be specified.
mexWeFireTrigPacket can be used to generate a trigger packet and send it to the stations specified here. The trigger packet can also be generated from the specified trigger source station using mexWeSetSndTrigPacket.

Syntax

ret = mexWeSetRcvTrigPacket(stationHandle, num, name)

Output Parameters

ret: Returns 0 if successful. Returns an error code if unsuccessful.

Input Parameters

stationHandle:	Station handle, station link handle, or broadcast handle.
num:	Number of stations to be specified (up to 8).
name:	Station name structure

mexWeSetSndTrigPacket

Description

Sets the source station for trigger packets.
Up to 8 trigger packet source stations can be specified.
When the trigger bus (BUSTRG 1 or BUSTRG 2) in the measuring station becomes active, the station generates the trigger packet. The receiving station outputs a trigger signal according to the trigger source to the trigger bus (BUSTRG1 or BUSTRG2).

Syntax

ret = mexWeSetSndTrigPacket(stationHandle, num, name)

Output Parameters

ret: Returns 0 if successful. Returns an error code if unsuccessful.

Input Parameters

stationHandle:	Station handle, station link handle, or broadcast handle.
num:	Number of stations to be specified (up to 8).
name:	Station name structure

mexWeFireTrigPacket

Description

Issues a (software) trigger packet.
The trigger packets are sent to the measuring stations specified by mexWeSetRcvTrigPacket.

Syntax

ret = mexWeFireTrigPacket(stationHandle, item)

Output Parameters

ret: Returns 0 if successful. Returns an error code if unsuccessful.

Input Parameters

stationHandle:	Station handle, station link handle, or broadcast handle.
item:	1=WE_TRG1 Use bus trigger 1
	2=WE_TRG2 Use bus trigger 2

mexWeSetSndClockPacket

Description

Sets the source station for time base packets.
The specified measuring station generates time base packets using its own time base signal. Up to 8 clock packet source stations can be specified.

Syntax

ret = mexWeSetSndClockPacket(stationHandle, num, name)

Output Parameters

ret: Returns 0 if successful. Returns an error code if unsuccessful.

Input Parameters

stationHandle:	Station handle, station link handle, or broadcast handle.
num:	Number of stations to be specified (up to 8).
name:	Station name structure

mexWeSetRcvClockPacket

Description

Sets the receive station for time base packets.
Up to 8 receiving stations can be specified.
The measuring station receiving the time base packet generates one pulse of time base signal on its own clock bus for each packet.

Syntax

ret = mexWeSetRcvClockPacket(stationHandle, num, name)

Output Parameters

ret: Returns 0 if successful. Returns an error code if unsuccessful.

Input Parameters

stationHandle:	Station handle, station link handle, or broadcast handle.
num:	Number of stations to be specified (up to 8).
name:	Station name structure

mexWeFireClockPacket

Description

Issues a (software) time base packet.

Syntax

ret = mexWeFireClockPacket(stationHandle)

Output Parameters

ret: Returns 0 if successful. Returns an error code if unsuccessful.

Input Parameters

stationHandle: Station handle, station link handle, or broadcast handle.

mexWeOutputEXTIOEvent

Description

Outputs a (software) event signal to the event output pin of the EXT. I/O connector on the front panel of the measuring station.

Syntax

ret = mexWeOutputEXTIOEvent(stationHandle, pulse)

Output Parameters

ret: Returns 0 if successful. Returns an error code if unsuccessful.

Input Parameters

stationHandle:	Station handle, station link handle, or broadcast handle.	
pulse:	0=WE_EXTIO_OFF	EXTIO output OFF
	1=WE_EXTIO_ON	EXTIO output ON
	2=WE_EXTIO_PULSE	EXTIO output PULSE

mexWeExecManualArming

Description

Generates a manual arming signal.

Syntax

ret = mexWeExecManualArming(stationHandle)

Output Parameters

ret: Returns 0 if successful. Returns an error code if unsuccessful.

Input Parameters

stationHandle: Station handle, station link handle, or broadcast handle.

mexWeSetArmingSource

Description

Sets the arming signal source.

Syntax

ret = mexWeSetArmingSource(stationHandle, item)

Output Parameters

ret: Returns 0 if successful. Returns an error code if unsuccessful.

Input Parameters

stationHandle:	Station handle, station link handle, or broadcast handle.	
item:	0=WE_TRGNONE	No bus trigger
	1=WE_TRG1	Use bus trigger 1
	2=WE_TRG2	Use bus trigger 2

mexWeGetArmingSource

Description

Retrieves the arming signal source setting.

Syntax

[ret, item] = mexWeGetArmingSource(stationHandle)

Output Parameters

ret:	Returns 0 if successful. Returns an error code if unsuccessful.	
item:	0=WE_TRGNONE	No bus trigger
	1=WE_TRG1	Use bus trigger 1
	2=WE_TRG2	Use bus trigger 2

Input Parameters

stationHandle: Station handle, station link handle, or broadcast handle.

5.8 GUI Control

mexWeShowModuleWindow

Description

Shows the operation panel for controlling the measurement module.

Syntax

[ret, windowHandle] = mexWeShowModuleWindow(moduleHandle)

Output Parameters

ret: Returns 0 if successful. Returns an error code if unsuccessful.
windowHandle: Handle of the operation panel shown

Input Parameters

moduleHandle: Module handle

mexWeCloseModuleWindow

Description

Closes the operation panel for controlling the measurement module.

Syntax

ret = mexWeCloseModuleWindow(moduleHandle)

Output Parameters

ret: Returns 0 if successful. Returns an error code if unsuccessful.

Input Parameters

moduleHandle: Module handle

mexWeIsModuleWindow

Description

Gets whether the operation panel for controlling the module is shown.

Syntax

[ret, sw] = mexWeIsModuleWindow(moduleHandle)

Output Parameters

ret: Returns 0 if successful. Returns an error code if unsuccessful.
sw: Operation panel status 1 = Shown, 0 = Hidden

Input Parameters

moduleHandle: Module handle

mexWeShowTrigWindow

Description

Shows the trigger setting dialog box used to set module linking (trigger source, time base source, and arming) within the measuring station.

Syntax

[ret, windowHandle] = mexWeShowTrigWindow(stationHandle)

Output Parameters

ret: Returns 0 if successful. Returns an error code if unsuccessful.
windowHandle: Handle of the trigger setting dialog box shown

Input Parameters

stationHandle: Station handle

mexWeCloseTrigWindow

Description

Closes the trigger setting dialog box used to set module linking (trigger source, time base source, and arming) within the measuring station.

Syntax

ret = mexWeCloseTrigWindow(stationHandle)

Output Parameters

ret: Returns 0 if successful. Returns an error code if unsuccessful.

Input Parameters

stationHandle: Station handle

mexWeIsTrigWindow

Description

Gets whether the trigger setting dialog box used to set module linking (trigger source, time base source, and arming) within the measuring station is open.

Syntax

[ret, sw] = mexWeIsTrigWindow(stationHandle)

Output Parameters

ret: Returns 0 if successful. Returns an error code if unsuccessful.
sw: Operation panel status 1 = Shown, 0 = Hidden

Input Parameters

stationHandle: Station handle

mexWeShowLinearScaleWindow

Description

Shows the convert scale dialog box.

Syntax

[ret, windowHandle] = mexWeShowLinearScaleWindow(moduleHandle)

Output Parameters

ret: Returns 0 if successful. Returns an error code if unsuccessful.
windowHandle: Handle of the convert scale dialog box shown

Input Parameters

moduleHandle: Module handle

mexWeCloseLinearScaleWindow

Description

Closes the convert scale dialog box.

Syntax

ret = mexWeCloseLinearScaleWindow(moduleHandle)

Output Parameters

ret: Returns 0 if successful. Returns an error code if unsuccessful.

Input Parameters

moduleHandle: Module handle

mexWeIsLinearScaleWindow

Description

Gets whether the convert scale dialog box is open.

Syntax

[ret, sw] = mexWeIsLinearScaleWindow(moduleHandle)

Output Parameters

ret: Returns 0 if successful. Returns an error code if unsuccessful.
sw: Convert scale dialog box status 1 = Shown, 0 = Hidden

Input Parameters

moduleHandle: Module handle

5.9 Measurement Control

mexWeStart

Description

Starts the measurement module operation.

Syntax

ret = mexWeStart(hHandle)

Output Parameters

ret: Returns 0 if successful. Returns an error code if unsuccessful.

Input Parameters

hHandle: Module handle, module link handle, station handle, or station link handle.

mexWeStop

Description

Stops the measurement module operation.

Syntax

ret = mexWeStop(hHandle)

Output Parameters

ret: Returns 0 if successful. Returns an error code if unsuccessful.

Input Parameters

hHandle: Module handle, module link handle, station handle, or station link handle.

mexWeStartEx

Description

Starts the measurement module operation (extended).

Syntax

ret = mexWeStartEx(hHandle, blockLen, blockCount, acqCount)

Output Parameters

ret: Returns 0 if successful. Returns an error code if unsuccessful.

Input Parameters

hHandle: Module handle, module link handle, station handle, or station link handle.
blockLen: Number of data points per block (record length)
blockCount: Number of memory partitions (blocks)
Specify the exponent of the 2's power.
acqCount: Number of acquisitions

Note

To stop the measurement module operation, use mexWeStop.

mexWeStartSingle

Description

Acquires the data once on the data acquisition measurement module. When the data acquisition is completed after starting the measurement, the program exits from this function.

Syntax

ret = mexWeStartSingle(hHandle, timeout)

Output Parameters

ret: Returns 0 if successful. Returns an error code if unsuccessful.

Input Parameters

hHandle: Module handle, module link handle, station handle, or station link handle.
timeout: Timeout value (s)

mexWeIsRun

Description

Gets the execution status (Run status/Stop status) of the measurement module.

Syntax

[ret, state] = mexWeIsRun(moduleHandle)

Output Parameters

ret: Returns 0 if successful. Returns an error code if unsuccessful.
state: 1 = Run, 0 = Stop

Input Parameters

moduleHandle: Module handle

mexWeLatchData

Description

Issues the latch command that specifies the range of acquisition data to be retrieved when the acquisition mode is set to free run.

Syntax

ret = mexWeLatchData(moduleHandle)

Output Parameters

ret: Returns 0 if successful. Returns an error code if unsuccessful.

Input Parameters

moduleHandle: Module handle

mexWeGetAcqDataInfo

Description

Reads the acquisition data information of the measurement module.

Syntax

[ret, info, infoNum] = mexWeGetAcqDataInfo(moduleHandle, ch, blockNo)

Output Parameters

ret: Returns 0 if successful. Returns an error code if unsuccessful.
info: Data information structure
infoNum: The number of data information structure retrieved

Input Parameters

moduleHandle: Module handle
ch: Channel number
blockNo: Block number

mexWeGetAcqDataSize

Description

Reads the acquisition data size (number of bytes) of the measurement module.
The number of data bytes is adjusted according to the data type.

Syntax

[ret, pointNum, dataSize, ptype, chNum] = mexWeGetAcqDataSize(moduleHandle, ch, blockNo)

Output Parameters

ret: Returns 0 if successful. Returns an error code if unsuccessful.
pointNum: Number of data points to be retrieved
dataSize: Total number of bytes to be retrieved.
ptype: Type of data to be retrieved

255=	WE_NULL	No parameter
0=	WE_UBYTE	8-bit unsigned integer
1=	WE_SBYTE	8-bit signed integer
4=	WE_BIT8	8-bit logic type
16=	WE_UWORD	16-bit unsigned integer
17=	WE_SWORD	16-bit signed integer
20=	WE_BIT16	16-bit logic type
32=	WE_ULONG	32-bit unsigned integer
33=	WE_SLONG	32-bit signed integer
36=	WE_BIT32	32-bit logic type
34=	WE_FLOAT	32-bit real number
50=	WE_DOUBLE	64-bit real number
52=	WE_BIT64	64-bit logic type

chNum: Total number of channels that can currently acquire data

Input Parameters

moduleHandle: Module handle
ch: Channel number
blockNo: Block number

mexWeGetAcqData

Description

Reads the acquisition data from the measurement module.
The data is raw data that is obtained after A/D conversion. The data format depends on the measurement module. The relevant information can be obtained with the mexWeGetAcqDataInfo function.

Syntax

[ret, recSize, buf, ptype] = mexWeGetAcqData(moduleHandle, ch, blockNo, bufSize)

Output Parameters

ret: Returns 0 if successful. Returns an error code if unsuccessful.
recSize: Data size (number of bytes) received.
buf: Data buffer
ptype: Type of data to be retrieved

255=WE_NULL	No parameter
0=WE_UBYTE	8-bit unsigned integer
1=WE_SBYTE	8-bit signed integer
4=WE_BIT8	8-bit logic type
16=WE_UWORD	16-bit unsigned integer
17=WE_SWORD	16-bit signed integer
20=WE_BIT16	16-bit logic type
32=WE_ULONG	32-bit unsigned integer
33=WE_SLONG	32-bit signed integer
36=WE_BIT32	32-bit logic type
34=WE_FLOAT	32-bit real number
50=WE_DOUBLE	64-bit real number
52=WE_BIT64	64-bit logic type

Input Parameters

moduleHandle: Module handle
ch: Channel number
blockNo: Block number
bufSize: Data buffer size

mexWeGetAcqDataEx

Description

Reads the acquisition data from the measurement module. Unlike mexWeGetAcqData, this function can read the interpolated data (Peak-to-peak (MIN-MAX) data).

Syntax

[ret, recSize, buf, ptype] = mexWeGetAcqDataEx(moduleHandle, ch, blockNo, startPoint, endPoint, ppNum, interpolation, bufSize)

Output Parameters

ret: Returns 0 if successful. Returns an error code if unsuccessful.
recSize: Data size (number of bytes) received.
buf: Data buffer
ptype: Type of data to be retrieved

255=WE_NULL	No parameter
0=WE_UBYTE	8-bit unsigned integer
1=WE_SBYTE	8-bit signed integer
4=WE_BIT8	8-bit logic type
16=WE_UWORD	16-bit unsigned integer
17=WE_SWORD	16-bit signed integer
20=WE_BIT16	16-bit logic type
32=WE_ULONG	32-bit unsigned integer
33=WE_SLONG	32-bit signed integer
36=WE_BIT32	32-bit logic type
34=WE_FLOAT	32-bit real number
50=WE_DOUBLE	64-bit real number
52=WE_BIT64	64-bit logic type

Input Parameters

moduleHandle: Module handle
ch: Channel number
blockNo: Block number
startPoint: Start position of the data to be retrieved
endPoint: End position of the data to be retrieved
ppNum: Number of display data sets (Pair of MIN and MAX values)
interpolation: Data interpolation type selection

0=WE_INTER_LINE	Linear interpolation
1=WE_INTER_SIN	Sine interpolation
2=WE_INTER_PULSE	Pulse interpolation

bufSize: Data buffer size

mexWeGetCurrentData

Description

Reads the instantaneous data from the measurement module.
This function reads the newest data.

Syntax

[ret, recSize, buf, ptype] = mexWeGetCurrentData(moduleHandle, ch, bufSize)

Output Parameters

ret: Returns 0 if successful. Returns an error code if unsuccessful.
recSize: Data size (number of bytes) received.
buf: Data buffer
ptype: Type of data to be retrieved

255=WE_NULL	No parameter
0=WE_UBYTE	8-bit unsigned integer
1=WE_SBYTE	8-bit signed integer
16=WE_UWORD	16-bit unsigned integer
17=WE_SWORD	16-bit signed integer
32=WE_ULONG	32-bit unsigned integer
33=WE_SLONG	32-bit signed integer
34=WE_FLOAT	32-bit real number
50=WE_DOUBLE	64-bit real number

Input Parameters

moduleHandle: Module handle
ch: Channel number
bufSize: Data buffer size

mexWeGetScaleCurrentData

Description

Reads the data obtained after scaling the instantaneous values of the measurement module.

Syntax

[ret, recParam, recSize, buf] = mexWeGetScaleCurrentData(moduleHandle, ch, ptype, param, bufSize)

Output Parameters

ret: Returns 0 if successful. Returns an error code if unsuccessful.
recParam: Scale parameter information structure received
recSize: Data size (number of bytes) received.
buf: Data buffer

Input Parameters

moduleHandle: Module handle
ch: Channel number
ptype: Type of data being read
34=WE_FLOAT
50=WE_DOUBLE
param: Scale parameter information structure
bufSize: Data buffer size

mexWeGetScaleCurrentDataEx

Description

Reads the data obtained after scaling the instantaneous values of the measurement module. Uses the scale conversion values that were specified using mexWeSetScaleInfo or mexWeShowLinearScaleWindow, that are stored in the module.

Syntax

[ret, recSize, buf] = mexWeGetScaleCurrentDataEx(moduleHandle, ch, ptype, bufSize)

Output Parameters

ret: Returns 0 if successful. Returns an error code if unsuccessful.
recSize: Data size (number of bytes) received.
buf: Data buffer

Input Parameters

moduleHandle: Module handle
ch: Channel number
ptype: Type of data being read
34=WE_FLOAT
50=WE_DOUBLE
bufSize: Data buffer size

mexWeGetScaleData

Description

Reads the data obtained after scaling the acquisition data of the measurement module.
The A/D-converted acquisition data is converted to physical values and then converted using the specified scale values.

Syntax

[ret, recParam, recSize, buf] = mexWeGetScaleData(moduleHandle, ch, blockNo, param, bufSize, ptype)

Output Parameters

ret: Returns 0 if successful. Returns an error code if unsuccessful.
recParam: Scale parameter information structure received
recSize: Data size (number of bytes) received.
buf: Data buffer

Input Parameters

moduleHandle: Module handle
ch: Channel number
blockNo: Block number
param: Scale parameter information structure
bufSize: Data buffer size
ptype: Type of data being read
34=WE_FLOAT
50=WE_DOUBLE

mexWeGetScaleDataEx

Description

Reads the data obtained after scaling the acquisition data of the measurement module. Uses the scale conversion values that were specified using mexWeSetScaleInfo or mexWeShowLinearScaleWindow, that are stored in the module.

Syntax

[ret, recSize, buf] = mexWeGetScaleDataEx(moduleHandle, ch, blockNo, bufSize, ptype)

Output Parameters

ret:	Returns 0 if successful. Returns an error code if unsuccessful.
recSize:	Data size (number of bytes) received.
buf:	Data buffer

Input Parameters

moduleHandle:	Module handle
ch:	Channel number
blockNo:	Block number
bufSize:	Data buffer size
ptype:	Type of data being read 34=WE_FLOAT 50=WE_DOUBLE

mexWeGetMeasureParam

Description

Gets the automated measurement values of waveform parameters from the measurement module. The automated measurement values are the results analyzed by the measurement modules. For modules that do not have the automated measurement function, the mexWeExecMeasureParam can be used.

Syntax

[ret, item, itemNum] = mexWeGetMeasureParam(moduleHandle, ch, blockNo, startPoint, endPoint)

Output Parameters

ret:	Returns 0 if successful. Returns an error code if unsuccessful.
item:	Automated measurement information structure of waveform parameters
itemNum:	The number of automated measurement information structures

Input Parameters

moduleHandle:	Module handle
ch:	Channel number
blockNo:	Block number
startPoint:	Start point of the data for the automated measurement of waveform parameters.
endPoint:	End point of the data for the automated measurement of waveform parameters.

mexWeSaveAcqData

Description

Saves the acquisition data (raw data) of the measurement module to a file.

Syntax

ret = mexWeSaveAcqData(moduleHandle, ch, blockNo, filename, htype)

Output Parameters

ret: Returns 0 if successful. Returns an error code if unsuccessful.

Input Parameters

moduleHandle:	Module handle
ch:	Channel number
blockNo:	Block number
filename:	File name
htype:	Header file creation
	0 = Not create the header file
	1 = Create the header file

mexWeSaveScaleData

Description

Saves the scaled acquisition data of the measurement module to a file in binary format.

Syntax

[ret, recParam] = mexWeSaveScaleData(moduleHandle, ch, blockNo, param, filename, htype)

Output Parameters

ret:	Returns 0 if successful. Returns an error code if unsuccessful.
recParam:	Scaled value information structure retrieved

Input Parameters

moduleHandle:	Module handle
ch:	Channel number
blockNo:	Block number
param:	Scale value information structure
filename:	File name
htype:	Header file creation
	0 = Not create the header file
	1 = Create the header file

mexWeSaveScaleDataEx

Description

Saves the scaled acquisition data of the measurement module to a file in binary format. Uses the scale conversion values that were specified using mexWeSetScaleInfo or mexWeShowLinearScaleWindow, that are stored in the module.

Syntax

ret = mexWeSaveScaleDataEx(moduleHandle, ch, blockNo, filename, htype)

Output Parameters

ret: Returns 0 if successful. Returns an error code if unsuccessful.

Input Parameters

moduleHandle:	Module handle
ch:	Channel number
blockNo:	Block number
filename:	File name
htype:	Header file creation
	0 = Not create the header file
	1 = Create the header file

mexWeSaveAsciiData

Description

Saves the acquisition data of the measurement module converted to physical values to a file in ASCII format (CSV format). Physical values are not the scaled values, but values in the measurement unit of the module. For example, the physical value is voltage on the digitizer module.

Syntax

ret = mexWeSaveAsciiData(moduleHandle, ch, blockNo, filename, htype)

Output Parameters

ret: Returns 0 if successful. Returns an error code if unsuccessful.

Input Parameters

moduleHandle:	Module handle
ch:	Channel number
blockNo:	Block number
filename:	File name
htype:	Header file creation
	0 = Not create the header file
	1 = Create the header file

mexWeSaveScaleAsciiData

Description

Saves the scaled acquisition data of the measurement module to a file in ASCII format (CSV format).

Syntax

[ret, recParam] = mexWeSaveScaleAsciiData(moduleHandle, ch, blockNo, param, filename, htype)

Output Parameters

ret: Returns 0 if successful. Returns an error code if unsuccessful.
recParam: Scaled value information structure retrieved

Input Parameters

moduleHandle: Module handle
ch: Channel number
blockNo: Block number
param: Scale value information structure
filename: File name
htype: Header file creation
0 = Not create the header file
1 = Create the header file

mexWeSaveScaleAsciiDataEx

Description

Saves the scaled acquisition data of the measurement module to a file in ASCII format (CSV format). Uses the scale conversion values that were specified using mexWeSetScaleInfo or mexWeShowLinearScaleWindow, that are stored in the module.

Syntax

ret = mexWeSaveScaleAsciiDataEx(moduleHandle, ch, blockNo, filename, htype)

Output Parameters

ret: Returns 0 if successful. Returns an error code if unsuccessful.

Input Parameters

moduleHandle: Module handle
ch: Channel number
blockNo: Block number
filename: File name
htype: Header file creation
0 = Not create the header file
1 = Create the header file

mexWeSaveAcqHeader

Description

Creates the header file containing the acquisition data waveform information of the measurement module. The file that is created is the same file created using mexWeSaveAcqData.

Syntax

ret = mexWeSaveAcqHeader(moduleHandle, ch, filename)

Output Parameters

ret: Returns 0 if successful. Returns an error code if unsuccessful.

Input Parameters

moduleHandle:	Module handle
ch:	Channel number
filename:	File name

mexWeSavePatternData

Description

Saves the pattern data that is dependent on the measurement module to a file.
The pattern data is different for each module. Some modules do not have pattern data defined.
Pattern data is, for example, the arbitrary waveform data of the WE7121 and the digital pattern data of the WE7131.

Syntax

ret = mexWeSavePatternData(moduleHandle, ch, filename)

Output Parameters

ret: Returns 0 if successful. Returns an error code if unsuccessful.

Input Parameters

moduleHandle:	Module handle
ch:	Channel number
filename:	File name

mexWeLoadPatternData

Description

Loads the pattern data that is dependent on the measurement module. The pattern data is different for each measurement module. Some measurement modules do not have pattern data defined. Pattern data is, for example, the arbitrary waveform data of the WE7121 and the digital pattern data of the WE7131.

Syntax

ret = mexWeLoadPatternData(moduleHandle, ch, filename)

Output Parameters

ret: Returns 0 if successful. Returns an error code if unsuccessful.

Input Parameters

moduleHandle:	Module handle
ch:	Channel number
filename:	File name

mexWeLoadPatternDataEx

Description

Loads the waveform data that is retrieved using the specified parameter from the specified file (wvf or csv format) to the measurement module.

Syntax

ret = mexWeLoadPatternDataEx(moduleHandle, command, filename, ch, blockNo)

Output Parameters

ret: Returns 0 if successful. Returns an error code if unsuccessful.

Input Parameters

moduleHandle: Module handle
command: ASCII Command
filename: File name
ch: Channel number
blockNo: Block number

mexWeSetOverRun

Description

Sets whether to detect overruns.

Syntax

ret = mexWeSetOverRun(moduleHandle, sw)

Output Parameters

ret: Returns 0 if successful. Returns an error code if unsuccessful.

Input Parameters

moduleHandle: Module handle
sw : Overrun detection setting
0 = Not detected (do not stop even when overrun occurs)
1 = Detected (stop when overrun occurs)

mexWeGetOverRun

Description

Gets whether overruns are detected.

Syntax

[ret, sw] = mexWeGetOverRun(moduleHandle)

Output Parameters

ret: Returns 0 if successful. Returns an error code if unsuccessful.
sw: Overrun detection status
0 = Not detected (do not stop even when overrun occurs)
1 = Detected (stop when overrun occurs)

Input Parameters

moduleHandle: Module handle

5.10 Waveform Parameter Computation

mexWeExecMeasureParam

Description

Performs waveform parameter computation on physical data.
mexWeGetMeasureParam performs the automated computation of waveform parameters on the measurement module, but this function computes the parameters on the PC.

Syntax

[ret, item] = mexWeExecMeasureParam(data, ptype, dt, startPoint, endPoint)

Output Parameters

ret: Returns 0 if successful. Returns an error code if unsuccessful.
item: Structure for storing the computed results of waveform parameters

Input Parameters

data: Computed data of waveform parameter
ptype: Parameter type

34=WE_FLOAT	32-bit real number
50=WE_DOUBLE	64-bit real number

dt: Sampling interval (s)
startPoint: Starting point of the data for computing the parameter values.
endPoint: End point of the data for computing the parameter values.

mexWeExecMeasureParamAcqData

Description

Performs waveform parameter computation on acquisition data (raw data).
mexWeGetMeasureParam performs the automated computation of waveform parameters on the measurement module, but this function computes the parameters on the PC.

Syntax

[ret, item] = mexWeExecMeasureParamAcqData(data, ptype, gain, offset, dt, startPoint, endPoint)

Output Parameters

ret: Returns 0 if successful. Returns an error code if unsuccessful.
item: Structure for storing the computed results of waveform parameters

Input Parameters

data: Computed data of waveform parameter
ptype: Parameter type

0=WE_UBYTE	8-bit unsigned integer
1=WE_SBYTE	8-bit signed integer
16=WE_UWORD	16-bit unsigned integer
17=WE_SWORD	16-bit signed integer
32=WE_ULONG	32-bit unsigned integer
33=WE_SLONG	32-bit signed integer

gain: Gain (range)
offset: Offset
dt: Sampling interval (s)
startPoint: Starting point of the data for computing the parameter values.
endPoint: End point of the data for computing the parameter values.

5.11 Filter API for the WE7281 4-CH, 100kS/s D/A Module

mexWeWvf2S16GetSize

Description

Gets the byte size when waveform data retrieved using the specified parameter from the specified file (wvf or csv format) is converted to the arbitrary waveform data format for the FG mode (s16).

Syntax

[ret, size] = mexWeWvf2S16GetSize(filename, ch, block)

Output Parameters

ret: Returns 0 if successful. Returns an error code if unsuccessful.
size: Data size (bytes)

Input Parameters

filename: Waveform data file name
ch: Channel number of the input file
block: Block number of the input file

mexWeWvf2W32GetSize

Description

Gets the byte size when waveform data retrieved using the specified parameter from the specified file (wvf or csv format) is converted to the sweep waveform data format for the FG mode (w32).

Syntax

[ret, size] = mexWeWvf2W32GetSize(filename, ch, block)

Output Parameters

ret: Returns 0 if successful. Returns an error code if unsuccessful.
size: Data size (bytes)

Input Parameters

filename: Waveform data file name
ch: Channel number of the input file
block: Block number of the input file

mexWeWvf2W7281GetSize

Description

Gets the byte size when waveform data retrieved using the specified parameter from the specified file (wvf or csv format) is converted to the arbitrary waveform data format (w7281) for the AG mode.

Syntax

[ret, size] = mexWeWvf2W7281GetSize(filename, ch, block)

Output Parameters

ret: Returns 0 if successful. Returns an error code if unsuccessful.
size: Data size (bytes)

Input Parameters

filename: Waveform data file name
ch: Channel number of the input file
block: Block number of the input file

mexWeWvf2S16

Description

Converts waveform data retrieved using the specified parameter from the specified file (wvf or csv format) to the arbitrary waveform data format for the FG mode (s16).

Syntax

[ret, buf, recSize] = mexWeWvf2S16(filename, ch, block, bufSize)

Output Parameters

ret: Returns 0 if successful. Returns an error code if unsuccessful.
buf: Data retrieved
recSize: Size of the data retrieved (byte size)

Input Parameters

filename: Waveform data file name
ch: Channel number of the input file
block: Block number of the input file
bufsize: Data size (bytes)

mexWeWvf2W32

Description

Converts waveform data retrieved using the specified parameter from the specified file (wvf or csv format) to the sweep waveform data format for the FG mode (w32).

Syntax

[ret, buf, recSize] = mexWeWvf2W32(filename, ch, block, bufSize)

Output Parameters

ret: Returns 0 if successful. Returns an error code if unsuccessful.
buf: Data retrieved
recSize: Size of the data retrieved (byte size)

Input Parameters

filename: Waveform data file name
ch: Channel number of the input file
block: Block number of the input file
bufsize: Data size (bytes)

mexWeWvf2W7281

Description

Converts waveform data retrieved using the specified parameter from the specified file (wvf or csv format) to the arbitrary waveform data format for the AG mode (w7281).

Syntax

[ret, buf, recSize] = mexWeWvf2W7281(filename, ch, block, bufSize)

Output Parameters

ret: Returns 0 if successful. Returns an error code if unsuccessful.
buf: Data retrieved
recSize: Size of the data retrieved (byte size)

Input Parameters

filename: Waveform data file name
ch: Channel number of the input file
block: Block number of the input file
bufsize: Data size (bytes)

5.12 Others

mexWeIsBlockEnd

Description

Gets the end status of the block data read of the measurement module.

Syntax

[ret, sw] = mexWeIsBlockEnd(moduleHandle, blockNo)

Output Parameters

ret: Returns 0 if successful. Returns an error code if unsuccessful.
sw: End status 0 = Stop, 1 = Run

Input Parameters

moduleHandle: Module handle
blockNo: Block number

Note

This function is an original function of this Control Toolkit. It does not correspond to the WE Control API.

6. File Operation Functions

The function names obtained by removing “mex” from the mex function names correspond to the WE Control API functions.
For details on mex functions, see chapter 9, “File Operation Functions” in the WE Control API User’s Manual (IM 707741-61E).

6.1 The List of Functions

Single File Access

mex Function Name	API Function Name	Description	Page
mexWeDPHeaderReadS	WeDPHeaderReadS	Read the header file of the single file.	6-3
mexWeDPDataRead	WeDPDataRead	Read the data file of the single file.	6-3
mexWeDPHeaderWriteS	WeDPHeaderWriteS	Write the header file of the single file.	6-4
mexWeDPDataWrite	WeDPDataWrite	Write the data file of the single file.	6-4

Sequential File Access

mex Function Name	API Function Name	Description	Page
mexWeDPHeaderCsReadS	WeDPHeaderCsReadS	Read the header file of the sequential file.	6-5
mexWeDPCsRead	WeDPCsRead	Read the data file of the sequential file.	6-5
mexWeDPHeaderCsWriteS	WeDPHeaderCsWriteS	Write the header file of the sequential file.	6-6
mexWeDPCsWrite	WeDPCsWrite	Write the data file of the sequential file.	6-6

Access the Specified Item of the Header File

mex Function Name	API Function Name	Description	Page
mexWeDPHeaderItemRead	WeDPHeaderItemRead	Read the data of the specified item.	6-7
mexWeDPHeaderItemWrite	WeDPHeaderItemWrite	Write the data of the specified item.	6-7

Data Operation

mex Function Name	API Function Name	Description	Page
mexWeDPGetSampleChNum	WeDPGetSampleChNum	Get the number of samples and number of channels.	6-8
mexWeDPGetBlockNum	WeDPGetBlockNum	Get the number of blocks.	6-8
mexWeDPInitializeAcqInfo	WeDPInitializeAcqInfo	Store the required data in the data information structure.	6-8
mexWeDPScaleConvert	WeDPScaleConvert	Scale convert the data.	6-9

Read Acquisition Data Information

mex Function Name	API Function Name	Description	Page
mexWeGetAcqDataInfoEx	WeGetAcqDataInfoEx	Retrieves the acquisition data information.	6-10

6.2 Single File Access

mexWeDPHeaderReadS

Description

Reads the data from the header file by specifying the block number.

Syntax

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

Output Parameters

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)

Input 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)

mexWeDPDataRead

Description

Reads the data from the data file by specifying the block number.

Syntax

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

Output Parameters

ret: Returns 0 if successful. Returns an error code if unsuccessful.
data: Read data

Input 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

Note

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

mexWeDPHeaderWriteS

Description

Writes the header information at once to the header file by specifying the block.

Syntax

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

Output Parameters

ret: Returns 0 if successful. Returns an error code if unsuccessful.

Input 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

mexWeDPDataWrite

Description

Writes the data to the data file in units of blocks.

Syntax

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

Output Parameters

ret: Returns 0 if successful. Returns an error code if unsuccessful.

Input 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

6.3 Sequential File Access

mexWeDPHeaderCsReadS

Description

Collectively reads the header information from a header file.

Syntax

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

Output Parameters

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)

Input 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)

mexWeDPCsRead

Description

Reads the data from the data files (sequential files) by specifying the number of samples.

Syntax

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

Output Parameters

ret: Returns 0 if successful. Returns an error code if unsuccessful.
data: Read data

Input 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

Note

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

mexWeDPHeaderCsWriteS

Description

Collectively writes the header information to the header file.

Syntax

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

Output Parameters

ret: Returns 0 if successful. Returns an error code if unsuccessful.

Input 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

mexWeDPCsWrite

Description

Write data to a sequence file.

Syntax

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

Output Parameters

ret: Returns 0 if successful. Returns an error code if unsuccessful.

Input 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

6.4 Access the Specified Item of the Header File

mexWeDPHeaderItemRead

Description

Reads the information of the specified item name and specified channel from the header information of the header file.

Syntax

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

Output Parameters

ret: Returns 0 if successful. Returns an error code if unsuccessful.
data: Read data

Input 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

mexWeDPHeaderItemWrite

Description

Writes data to the specified item name and specified channel in the header information of the header file.

Syntax

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

Output Parameters

ret: Returns 0 if successful. Returns an error code if unsuccessful.

Input 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

6.5 Data Operation

mexWeDPGetSampleChNum

Description

Gets the number of samples and number of channels of the specified file.

Syntax

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

Output Parameters

ret: Returns 0 if successful. Returns an error code if unsuccessful.
SampleNum: Number of data points read
ChNum: Number of channels read

Input Parameters

filename: Name of the file to be read without the extension
blockNo: Number of the block to be read

mexWeDPGetBlockNum

Description

Gets the number of blocks of the specified file.

Syntax

[ret, blockNum] = mexWeDPGetBlockNum(filename)

Output Parameters

ret: Returns 0 if successful. Returns an error code if unsuccessful.
blockNum: Number of block read

Input Parameters

filename: Name of the file to be read without the extension

mexWeDPInitializeAcqInfo

Description

Stores the required data in the data information structure.

Syntax

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

Output Parameters

ret: Returns 0 if successful. Returns an error code if unsuccessful.
AcqInfo: Data information structure

Input Parameters

VMaxData: Max data
VMinData: Min data
sampleNum: Number of data samples
sampInterval: Sampling frequency of the data
infoNum: Number of data information structures

mexWeDPScaleConvert

Description

Scales the specified data and returns the result.

Syntax

[ret, scaleAcqInfo, scaleData] = mexWeDPScaleConvert(scaleA, scaleB, dataForm, data, mode, ChNum, dataAcqInfo, scaleDataForm)

Output Parameters

ret: Returns 0 if successful. Returns an error code if unsuccessful.
scaleAcqInfo: Data information structure
scaleData: Scaled data

Input Parameters

scaleA: Scaling coefficient A
scaleB: Scaling coefficient B
dataForm: Data type
1=WE_UBYTE
17=WE_SWORD
33=WE_SLONG
34=WE_FLOAT
50=WE_DOUBLE
data: Data to be converted
mode: Conversion mode (For details, see the WE Control API User's Manual (IM707741-61E).
ChNum: Number of data information structures (number of channels)
dataAcqInfo: Data information structure
scaleDataForm: ScaleData type
1=WE_UBYTE
17=WE_SWORD
33=WE_SLONG
34=WE_FLOAT
50=WE_DOUBLE

6.6 Read Acquisition Data Information

mexWeGetAcqDataInfoEx

Description

Reads the acquisition data information of the measurement module.

Syntax

[ret, info, infoNum] = mexWeGetAcqDataInfoEx(moduleHandle, ch, blockNo)

Output Parameters

ret:	Returns 0 if successful. Returns an error code if unsuccessful.
info:	Data information structure retrieved
infoNum:	Number of data information structure retrieved

Input Parameters

moduleHandle:	Module handle
ch:	Number of the channel to be retrieved
blockNo:	Number of the block number to be retrieved

Index

M

T

YOKOGAWA

YOKOGAWA ELECTRIC CORPORATION

Headquarters
2-9-32, Nakacho, Musashino-shi, Tokyo, 180-8750 JAPAN

Sales Headquarters
2-9-32, Nakacho, Musashino-shi, Tokyo, 180-8750 JAPAN
Phone : 81-422-52-6194

Branch Sales Offices
Nagoya, Osaka, Hiroshima, Fukuoka, Sapporo, Sendai, Ichihara, Toyoda, Kanazawa, Takamatsu, Okayama, and Kitakyusyu.

Overseas Representative Offices / Service Centers
Beijing, Shanghai (The People's Republic of China), Jakarta (Indonesia), Kuala Lumpur (Malaysia), Bangkok (Thailand)

YOKOGAWA CORPORATION OF AMERICA

Headquarters
2 Dart Road, Newnan, Ga. 30265-1094, U.S.A.
Phone : 1-770-253-7000
Fax : 1-770-251-0029

Branch Sales Offices / Detroit, Chicago, Los Angeles, New Jersey, Oklahoma, Texas, San Jose, Stafford

YOKOGAWA EUROPE B. V.

Headquarters
Databankweg 20 Amersfoort 3821 AL, THE NETHERLANDS
Phone : 31-334-64-1611 Fax : 31-334-64-1610

Branch Sales Offices / Wien (Austria), Zaventem (Belgium), Ratingen (Germany), Madrid (Spain), Runcorn (United Kingdom), Milano (Italy), Velizy Villacoublay (France), Johannesburg (Republic of South Africa), Budapest (Hungary), Stockholm (Sweden)

YOKOGAWA AMERICA DO SUL S.A.

Praca Acapulco, 31 - Santo Amaro. Sao Paulo/SP - BRAZIL
Phone : 55-11-5681-2400 Fax : 55-11-5681-1274

YOKOGAWA ELECTRIC ASIA PTE. LTD.

Head Office
5 Bedok South Road, 469270 SINGAPORE
Phone : 65-6241-9933 Fax : 65-6241-2606

YOKOGAWA ELECTRIC KOREA CO., LTD.

Head Office
420-5, Chongchun - 2dong, Pupyong - ku Inchon, 403-032 KOREA
Phone : 82-32-510-3107 Fax : 82-32-529-6304

YOKOGAWA AUSTRALIA PTY. LTD.

Head Office (Sydney)
Centrecourt D1, 25-27 Paul Street North, North Ryde,
N.S.W.2113, AUSTRALIA
Phone : 61-2-9805-0699 Fax : 61-2-9888-1844

YOKOGAWA BLUE STAR LTD.

Head Office
40 / 4 Lavelle Road, Bangalore 560 001, INDIA
Phone : 91-80-2271513 Fax : 91-80-2274270

Sep. '02

<Recycled Paper Used>

Printed in Japan