
**User's
Manual**

**Model MX190
API for the MX100/DARWIN**

Foreword

Thank you for purchasing the API for the MX100/DARWIN. This user's manual explains the operating procedures of the API for the MX100/DARWIN. To ensure correct use, please read this manual thoroughly before beginning operation. After reading the manual, keep it in a convenient location for quick reference whenever a question arises during operation.

Notes

- 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 representative, dealer, or sales office.
- Copying or reproducing all or any part of the contents of this manual without the permission of Yokogawa Electric Corporation is strictly prohibited.
- Use of this software on more than one computer at the same time is prohibited. Use by more than one user is also prohibited.
- Transfer or lending of this software to any third party is prohibited.
- Yokogawa Electric Corporation provides no guarantees other than for physical deficiencies found on the original disk upon opening the product package.
- Yokogawa Electric Corporation shall not be held responsible by any party for any losses or damage, direct or indirect, caused by the use or any unpredictable defect of the product.
- Please do not lose your serial number as it cannot be reissued.
- Serial numbers will not be reissued. Please keep your serial numbers in a safe place.

Trademarks

- Microsoft, Windows, Windows NT, Windows XP, Visual Studio, Visual Basic, Visual C++, and Visual C 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.
- For purposes of this manual, the TM and ® symbols do not accompany their respective trademark names or registered trademark names.
- All other company and product names used in this manual are trademarks or registered trademarks of their respective companies.

Revisions

- 1st Edition: May, 2003
- 2nd Edition: November, 2004

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: API for the MX100/DARWIN

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 licensor 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;
 - 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.

Checking the Contents of the Package

Unpack the box and check the contents before operating the software. If some of the contents are not correct, or if any items are missing or damaged, contact the dealer from whom you purchased them.

Model

Model	Name
MX190	API for the MX100/DARWIN

Package Contents

CD-ROM 1 piece
API for the MX100/DARWIN



CD-ROM 1 piece
User's Manual



Handling Precautions of the CD-ROM

Make sure to observe the precautions below.

WARNING

- **Do not use or store the CD-ROM in a place with excessive dirt or dust.**
 - **Do not touch the non-printed side of the CD-ROM.**
Oil and sweat from the fingertips sticking to the surface can cause malfunctions.
Do not write on the CD-ROM.
 - **Lead from pencils and residue from erasers sticking to the surface can also cause malfunction.**
 - **Do not bend or scratch the CD-ROM.**
Doing so will make the CD-ROM unreadable.
 - **Do not place objects on the CD-ROM.**
Doing so can deform the CD-ROM and make it unusable.
 - **Do not drop from high places.**
Dropping the CD-ROM can make it unusable due to breakage or deformation.
 - **Keep away from direct sunlight and any heating equipment.**
Never use solvents such as alcohol, thinner, and Freon to clean the CD-ROM.
 - **Insert the CD-ROM into the CD-ROM drive with care.**
 - **Do not eject the CD-ROM, or turn OFF or reset the PC while the CD-ROM is being accessed.**
 - **Store the CD-ROM in its storage case.**
Do not leave the CD-ROM in the CD-ROM drive after use. Leaving the CD-ROM out of the storage case can cause it to become dirty or deformed.
-

How to Use This Manual

Structure of this Manual

This user's manual consists of the following twenty-six chapters, appendix, and index.

Ch Title	Description
1 Before Using the Software	Gives an overview of the API for the MX100/DARWIN. Describes the PC requirements needed to run the software, the installation procedures, and other information.
2 API for MX100 using Visual C++ - Visual C++ - program examples, and the classes.	Describes how to apply the API to the MX100 C++. Describes the functions, details of
3 API for MX100 using Visual C. - Visual C - program examples.	Describes how to apply the API to the MX100 Describes the functions and
4 API for MX100 using Visual - Visual Basic - examples.	Describes how to apply the API to the MX100 Basic. Describes the functions and program
5 Functions for the MX100 API - Visual C and Visual Basic - by Visual C and Visual Basic.	Describes the details of the MX100 functions that can be used
6 MX100 API Constants and Types C, and	Describes the details of the constants and types for the MX100 that can be used by Visual C++, Visual Visual Basic.
7 API for DARWIN C++. - Visual C++ - examples, and the details of classes.	Describes how to apply the API to DARWIN using Visual Describes the functions, program the
8 API for DARWIN - Visual C -	Describes how to apply the API to DARWIN using Visual C. Describes the functions and program examples.
9 API for DARWIN - Visual Basic - examples.	Describes how to apply the API to the DARWIN using Visual Basic. Describes the functions and program
10 Functions for the DARWIN API - Visual C and Visual Basic -	Describes the details of the DARWIN functions that can be used by Visual C and Visual Basic.
11 DARWIN Constants and Types C++, Visual C, and Visual Basic.	Describes the details of the constants and types for that can be used by Visual
12 Extension API for MX100 - Visual C++ - program details of the classes.	Describes how to apply the extension API to the MX100 using Visual C++. Describes the functions, examples, and the
13 Extension API for MX100 - Visual C - and program	Describes how to apply the extension API to the MX100 using Visual C. Describes the functions examples.
14 Extension API for MX100 using - Visual Basic - examples.	Describes how to apply the extension API to the MX100 Visual Basic. Describes the functions and program
15 Extension API for MX100 using - Visual Basic.NET - program	Describes how to apply the extension API to the MX100 Visual Basic.NET. Describes the functions and examples.
16 Extension API for MX100 using - Visual C# - program examples.	Describes how to apply the extension API to the MX100 Visual C#. Describes the functions and
17 Functions for the MX100 used Extension API Basic.NET/C#.	Describes the details of the MX100 functions that can be by Visual C, Visual Basic, and Visual - Visual C, Visual Basic, and Visual Basic.NET/C# -

ChTitle	Description
18 MX100 Constants and Types for Extended API - Visual Basic -	Describes the details of the constants and types for the MX100 that can be used by Visual C++, Visual C, Visual Basic, and Visual Basic.NET/C#.
19 DARWIN for Extended API - Visual C++ -	Describes how to apply the extension API to DARWIN using Visual C++. Describes the functions and program examples.
20 DARWIN for Extended API - Visual C -	Describes how to apply the extension API to DARWIN using Visual C. Describes the functions and program examples.
21 DARWIN for Extended API - Visual Basic -	Describes how to apply the extension API to DARWIN using Visual Basic. Describes the functions and program examples.
22 DARWIN for Extended API - Visual Basic.NET -	Describes how to apply the extension API to DARWIN using Visual Basic.NET. Describes the functions and program examples.
23 DARWIN for Extended API - Visual C# -	Describes how to apply the extension API to DARWIN using Visual C#. Describes the functions and program examples.
24 DARWIN for Extended API - Visual C/Visual Basic/ Visual Basic.NET/C# -	Describes the details of the DARWIN functions that can be used by Visual C, Visual Basic, Visual Basic.NET/C#.
25 DARWIN for Extended API Constants and Types	Describes the details of the constants and types for DARWIN that can be used by Visual C++, Visual C, Visual Basic, and Visual Basic.NET/C#.
26 Error messages	Lists and explains error messages.
Appendix	Describes the terminology used by this software and the terminology for the MX100/DARWIN.
Index	An alphabetical index of the contents of the manual.

Scope of the Manual

OS

This manual does not cover the basic operations of Windows XP, Windows 2000, Windows NT 4.0, and Windows 98. For information regarding the basic operations of Windows, see the user's guide that came with Windows. Also, this manual does not cover the Visual Studio environment or languages supported. See the relevant user's manual for information on those topics.

MX100/DARWIN

Describes only the terminology used by the API. For details on the MX100 and DARWIN, see the relevant user's manuals.

Format Used in This Manual

Explanation of Description of Functions

The functions are explained using the following format. This format is also applied to the description of the function members of the classes.

The name of the function.

Syntax for Visual C++ or Visual C.

autoFIFOMX

Syntax
`int autoFIFOMX(DAQMX daqmx, int bAuto);`

Declaration statement for Visual Basic.

Declaration
 Public Declare Function autoFIFOMX Lib "DAQMX" (ByVal daqmx As Long, ByVal bAuto As Long) As Long

Parameters

daqmx Specify the device descriptor. **Description of the parameters.**
 bAuto Specify the valid/invalid value.

Description
 Sets auto control of the FIFO. **Describes the functionality of the function and gives notes.**

Return value
 Returns an error number. **Description of the return value (see "Return value" below).**

Error:
 Not descriptor No device descriptor.

Reference
 CDAQMX::autoFIFO **Error that this function generates.**

Functions that are called when this function is executed (see "Reference" below).

Return Value (Error Numbers)

There are three types of errors: errors generated by the function itself, errors caused by other component functions upon execution of the main function, and communication errors.

The errors generated by the function itself are listed in the `!Error:` section.

The errors generated by one of the component functions that is called when the main function is executed are described in the explanation of those functions which are listed under `!Reference.` **Communication Errors**

Communication execution errors generated when communication is performed using the communication descriptor. A communication error is any error between error numbers 0 and 3.

For a description of error numbers and corrective actions, see section 12.1, `!API Error Messages.`

Reference

A function consists of multiple, simpler functions. The Reference section lists the functions that make up a given function. This information can be used to understand the action of the function and/or troubleshoot errors.

Functions for Visual C and Visual Basic are executed by calling the Visual C++ class instance. Therefore, the Reference section lists the class name and the member function name and/or the function name.

The names are listed in alphabetical order.

Program Description

Sample programs are printed in this manual in monospaced font. Note the following.

- Due to the limitation of the width of a page, a long statement is written across multiple lines.

Terminology Reference

The terminology for the API and the terminology for the MX100 and DARWIN are described in the appendix.

Miscellaneous

Units

K Denotes 1024. Example: 100 KB

M Denotes 1024 K. Example: 10 MB

Symbols

Note Calls attention to information that is important for proper operation of the software.

CONTENTS

Terms and Conditions of the Software License	ii
Checking the Contents of the Package	iv
Handling Precautions of the CD-ROM	v
How to Use This Manual	vi
Format Used in This Manual	viii

Chapter 1 Before Using the API

1.1 Introduction to Functions	1-1
1.2 Software Components and Features	1-4
1.3 Operating Environment	1-6
1.4 Installation	1-7

API (Chapter 2—Chapter 11)

Chapter 2 MX100 - Visual C++ -

2.1 MX100 Class	2-1
2.2 Correspondence between the Functions and Class/Member Functions - MX100 -	2-4
2.3 Programming - MX100/Visual C++ -	2-9
2.4 Details of the MX100/DARWIN Common Class	2-15
2.5 Details of the MX100 Class	2-37

Chapter 3 MX100 - Visual C

3.1 Functions and Their Functionalities - MX100/Visual C -	3-1
3.2 Program - MX100/Visual C -	3-6

Chapter 4 MX100 - Visual Basic

4.1 Functions and Their Functionalities - MX100/Visual Basic -	4-1
4.2 Programming - MX100/Visual Basic -	4-6

Chapter 5 Functions for the MX100 (Visual C/Visual Basic)

5.1 Details of Functions - MX100 (Visual C/Visual Basic) -	5-1
--	-----

Chapter 6 MX100 Constants and Types

6.1 Overview of the MX100 Constants	6-1
6.2 MX100 Constants	6-3
6.3 MX100 Setting Item Numbers	6-16
6.4 Overview of the MX100 Types	6-22
6.5 MX100 Types	6-24

Chapter 7 DARWIN - Visual C++ -

7.1 DARWIN Class	7-1
7.2 Correspondence between the Functions and Class/Member Functions - DARWIN -	7-2
7.3 Programming - DARWIN/Visual C ++ -	7-5
7.4 Details of the DARWIN Class	7-13

Chapter 8 DARWIN - Visual C -

8.1	Functions and Their Functionalities - DARWIN/Visual C -	8-1
8.2	Programming - DARWIN/Visual C -	8-4

Chapter 9 DARWIN - Visual Basic -

9.1	Functions and Their Functionalities - DARWIN/Visual Basic -	9-1
9.2	Programming - DARWIN/Visual Basic -	9-4

Chapter 10 Functions for the DARWIN - Visual C/Visual Basic -

10.1	Details of Functions - DARWIN (Visual C/Visual Basic) -	10-1
------	---	------

Chapter 11 DARWIN Constants and Types

11.1	Overview of the DARWIN Constants	11-1
11.2	DARWIN Constants	11-2
11.3	Overview of the DARWIN Types	11-10
11.4	DARWIN Types	11-11

Extended API (Chapter 12—Chapter 25)

Chapter 12 MX100 for Extended API - Visual C++ -

12.1	MX100 Class	12-1
12.2	Correspondence between the Functions and Class/Member Functions - MX100 -	12-3
12.3	Programming - MX100/Visual C++ -	12-15
12.4	Details of the MX100 Class	12-18

Chapter 13 MX100 for Extended API - Visual C -

13.1	Functions and Their Functionalities - MX100/Visual C -	13-1
13.2	Programming - MX100/Visual C -	13-12

Chapter 14 MX100 for Extended API - Visual Basic -

14.1	Functions and Their Functionalities - MX100/Visual Basic -	14-1
14.2	Programming - MX100/Visual Basic -	14-12

Chapter 15 MX100 for Extended API - Visual Basic.NET -

15.1	Functions and Their Functionalities - MX100/Visual Basic.NET -	15-1
15.2	Programming - MX100/Visual Basic.NET -	15-12

Chapter 16 MX100 for Extended API - C# -

16.1	Functions and Their Functionalities - MX100/C# -	16-1
16.2	Programming - MX100/C# -	16-12

Chapter 17 Functions for the MX100 (Extended API) - Visual C/Visual Basic/ Visual Basic.NET/C# -

17.1	Details of Function - MX00 (Visual C/Visual Basic/Visual Basic.NET/C#) - Status Transition Functions	17-1
17.2	Details of Function - MX00 (Visual C/Visual Basic/Visual Basic.NET/C#) - Retrieval Functions	17-90

Chapter 18 MX100 Constants and Types for Extended API - Visual Basic -

18.1	Overview of the MX100 Constants	18-1
18.2	MX100 Constants	18-3
18.3	MX100 Setting Item Numbers	18-17
18.4	MX100 Types	18-18

Chapter 19 DARWIN for Extended API - Visual C++

19.1	DARWIN Class	19-1
19.2	Correspondence between the Functions and Class/Member Functions - DARWIN - ...	19-2
19.3	Programming - DARWIN/Visual C++ -	19-7
19.4	Functions and Class Members for Loading Instantaneous Value Data	19-9
19.5	Program for Loading Instantaneous Value Data - DARWIN/Visual C++ -	19-12
19.6	Details of the DARWIN Class	19-14

Chapter 20 DARWIN for Extended API - Visual C -

20.1	Functions and Their Functionalities - DARWIN/Visual C -	20-1
20.2	Programming - DARWIN/Visual C -	20-6
20.3	Correspondence between Functions for Instantaneous Value Data Loading and Functions - DARWIN/Visual C -	20-8
20.4	Program for Loading Instantaneous Value Data - DARWIN/Visual C -	20-10

Chapter 21 DARWIN for Extended API - Visual Basic -

21.1	Correspondence between the Functions and Class/Member Functions - DARWIN/Visual Basic -	21-1
21.2	Programming - DARWIN/Visual Basic -	21-6
21.3	Correspondence between Functions for Instantaneous Value Data Loading and Member Functions - DARWIN/Visual Basic -	21-8
21.4	Program for Loading Instantaneous Value Data - DARWIN/Visual Basic -	21-10

Chapter 22 DARWIN for Extended API - Visual Basic.NET -

22.1	Functions and Their Functionalities - DARWIN/Visual Basic.NET -	22-1
22.2	Programming - DARWIN/Visual Basic.NET -	22-6
22.3	Correspondence between Functions for Instantaneous Value Data Loading and Functions - DARWIN/Visual Basic.NET -	22-8
22.4	Program for Loading Instantaneous Value Data -DARWIN/Visual Basic.NET-	22-10

Chapter 23 DARWIN for Extended API - C# -

23.1	Functions and Their Functionalities - DARWIN/C# -	23-1
23.2	Programming - DARWIN/C# -	23-6
23.3	Correspondence between Functions and Instantaneous Value Data Loading and Functions - DARWIN/Visual C# -	23-8
23.4	Program for Loading Instantaneous Value Data -DARWIN/C#-	23-10

Chapter 24 DARWIN for Extended API

- Visual C/Visual Basic/Visual Basic.NET/C# -

24.1	Details of Functions - DARWIN (Visual C/Visual Basic/Visual Basic.NET/C#) - Status Transition Functions	24-1
24.2	Details of Functions - DARWIN (Visual C/Visual Basic/Visual Basic.NET/C#) - Retrieval Functions	24-39
24.3	Details of Functions for Instantaneous Value Loading - DARWIN (Visual C/Visual Basic/Visual Basic.NET/C#) - Status Transition Functions	24-74
24.4	Details of Instantaneous Value Loading Functions - DARWIN (Visual C/Visual Basic/Visual Basic.NET/C#) - Status Transition Functions	24-81

Chapter 25 DARWIN for Extended API Constants and Types

25.1	Overview of the DARWIN Constants	25-1
25.2	DARWIN Constants	25-2
25.3	DARWIN Types	25-14
25.4	Overview of DARWIN Constants for Loading Instantaneous Value Data	25-15
25.5	DARWIN Constants for Loading Instantaneous Value Data	25-16
25.6	DARWIN Types for Loading Instantaneous Value Data	25-20

API and Extended API

Chapter 26 Error Messages

26.1	API Error Messages	26-1
26.2	MX100 - Specific Error Messages	26-3

Appendix

Appendix 1	MX100 Terminology	App-1
Appendix 2	DARWIN Terminology	App-13
Appendix 3	Calculation of the MX100 Timeout Value	App-19
Appendix 4	API Revision History	App-20

Index

1.1 Introduction to Functions

Supported Models

This software provides the API (Application Programming Interface) that supports the MX100 and DARWIN, and the extension API that makes the main API easier to use.

MX100: All models

DARWIN: All models (DA, DC, and DR). However, the Ethernet communication function (Ethernet module) is required.

Supported Languages

Languages supported by the API: Visual C++, Visual C, and Visual Basic

Languages supported by the extension API:

Visual C++, Visual C, Visual Basic, Visual Basic.NET, and Visual C#

Functions for the MX100

Communication Functions

Functions used to communicate with the MX100 via the Ethernet interface.

Setup Functions

For configuring the MX100, modules, and channels.

Data Retrieval Functions

- Retrieval of measured data using FIFO
Retrieves the most recently measured data via the FIFO (first-in first-out) buffer. Also retrieves instantaneous values. The retrieval interval of instantaneous values is 100 ms or more.
- Retrieval of measured data by channel
Retrieves the most recent measured data of the specified channel. Also retrieves instantaneous values. The retrieval interval of instantaneous values is 100 ms or more.
- Retrieval of setup data
Retrieves the system configuration of the MX100, network information such as the IP address, and setup data related to channels.
- Collective retrieval of DO (digital output) data
Collectively retrieves the ON/OFF status of the DO.
- Retrieval of channel information data (channel number, etc.)
- Retrieval of AO/PWM data
Retrieves data that shows the AO/PWM channel output.

- Retrieval of transmission output data
Data that indicates the AO/PWM channel transmission status.
When data is retrieved, it indicates the status of the current transmission output.
When data is sent, it is the specification that controls the start and stop of the transmission output.

Control Functions

- Date/time setting
- Setting of the backup function that saves the measured data to the CF card, and formatting of the CF card
- Initialization and reconfiguration of the system (unit)
- Resetting of alarms (acknowledge alarm)
- Designation of the information displayed on the 7-segment LED

Utilities

Provides functions for converting measured values into character strings, retrieving error message strings, and other utility functions.

Functions for DARWIN

The API provides functions that are equivalent to the DARWIN communication functions. Some of the basic functions of the DARWIN communication functions are provided as API classes (for Visual C++) or functions (for Visual Basic, Visual C, Visual Basic.NET, and C#). Other functions can be implemented using the commands of the DARWIN communication function.

Functions Provided by this Software

- **Communication Functions**
Functions used to communicate with DARWIN via the Ethernet interface.
- **Setup Functions**
Sets the measurement range and alarms.
- **Data Retrieval Functions**
 - Retrieval of the measured data
Retrieves measured data in ASCII or binary format. The values are instantaneous values.

- Retrieval of setup data
Retrieves the setup data of the operation mode, basic setting mode, and A/D calibration mode.
- Retrieval of channel information data (channel number, etc.)
- **Control Functions**
 - Date/time setting
 - Reconfiguring the system
 - RAM clear (initialization of operation mode parameters)
 - Alarm reset
 - Setting mode switching
- **Utilities**
Provides functions for converting measured values into character strings, retrieving error message strings, and other utility functions.

Implementing Functions Corresponding to the Commands of the DARWIN Communication Function

Functions corresponding to commands of the DARWIN communication function can be implemented according to the procedure below.

- Visual C++: Create an inheritance class of CDAQDARWIN and add member functions that execute the commands of the DARWIN communication function using the runCommand member function.
- Visual C and Visual Basic: Create functions that execute the commands of the DARWIN communication function using the runCommandDARWIN function.
- Visual Basic.NET/C#: Create functions that execute the commands of the DARWIN communication function using the runCommandDA100 function of the extension API.

1.2 Software Components and Features

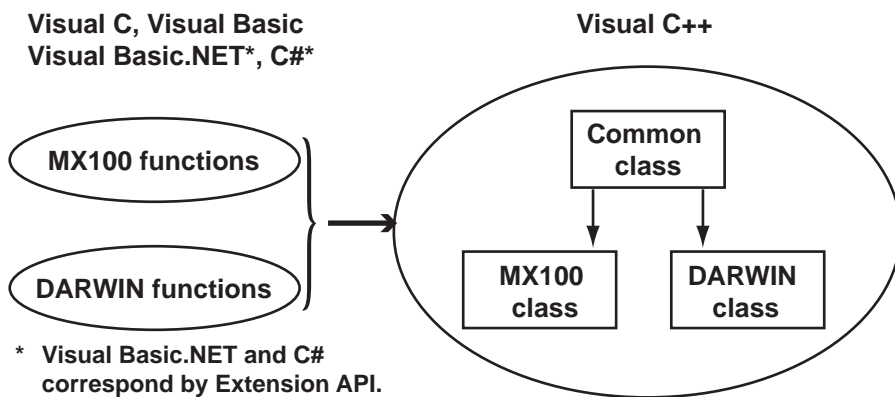
Software Components

The software consists of the API and an extension API.

The software provides Visual C++ classes.

The classes are Common Class, MX100 class, and DARWIN class. The MX100 Class and DARWIN Class are derived from the Common Class.

Functions used by Visual C, Visual Basic, Visual Basic.NET, and C# are provided. Functions for the MX100 and functions for DARWIN are also available. These functions are executed by calling the Visual C++ class instance.



Features

- Control is achieved by retrieving the handle (device descriptor) that represents the connection with the controlled unit. This means that, programs can be created independently of the hardware environment of the MX100 or DARWIN.
- The API provides structures for collection of data corresponding to the function. For example, the MX100 module information structure contains the module type, the number of channels, the scan interval, and other information related to the module. The same structure can be used even when the same data is retrieved using different commands and different formats.
- This API is subordinate to the extension API. Call the API to activate. The extension API holds the current status data internally. The extension API can describe a program without using a structure.

Files Included

The software consists of the files listed below.

Type	Extension	Description
Executable module	.dll	Executable files.
Include file	.h	Include file for Visual C and Visual C++.
Library file	.lib	Files linked in Visual C and Visual C++.
Standard module for Visual Basic	.bas	Files containing definitions for Visual Basic.
API viewer text file	.txt	Files containing functions, types, and constants for Visual Basic in text format. Can be used by the API Viewer of Visual Studio.
Standard module for Visual Basic.NET	.vb	Files containing definitions for Visual Basic.NET.
Standard module for C#	.cs	Files containing definitions for C#.

1.3 Operating Environment

PC (Personal Computer)

The software can be used by installing the API on a PC that meets the conditions below.

Note

- No limitation is placed on the CPU, memory, and hard disk. However, the performance of the software depends on them.
 - At least 10 MB of free space on the hard disk is required for installing the API.
-

Operating System

One of the following operating systems must be running.

- Windows 98
- Windows NT 4.0 SP3 or later
- Windows 2000
- Windows XP

CD-ROM Drive

Used to install the API.

Mouse

Mouse supported by the OS.

Display

Display supported by the OS.

Communication Port

Ethernet port supported by the OS. The TCP/IP protocol must be installed.

Supported Languages

- Visual C++
- Visual Basic.NET (extension API only)
- Visual C
- Visual C# (extension API only)
- Visual Basic

User Development Environment

A supported language must be installed and functional.

You must use Visual Studio 6.0 SP5 or later. The functions of this software are not guaranteed when used with other platforms.

MX100

No limitations in particular.

DARWIN

DA100, DC100, DR130, DR230, or DR240

The Ethernet communication function (Ethernet module) is required.

1.4 Installation

Procedure

1. Turn ON the PC and allow the operating system to start.

Note

Close memory resident programs such as virus protection programs before installation.

2. Insert the CD-ROM containing the software into the CD-ROM drive.
3. After a few moments, the DAQMASTER DARWIN MXAPI window and the Install Shield Wizard window appear. Next, the following screen appears. Follow the instructions on the screen.



If the DAQMASTER DARWIN MXAPI window does not appear, carry out the procedure below.

Click My Computer and double-click the CD-ROM icon. Double-click the Disk 1 icon and double-click Setup.exe in the folder. The DAQMASTER DARWIN MXAPI window and the Install Shield Wizard window appear. Then, the following screen appears. Follow the instructions on the screen.

Note

- **Reinstalling the Software**
If you carry out step 3 on a PC that has the software already installed, a window for confirming the file deletion opens. Click OK to delete the software. Then, carry out step 3 to reinstall the software.
 - You can uninstall the API by double-clicking Add/Remove Programs in the Windows Control Panel.
-

2.1 MX100 Class

The API consists of the MX100/DARWIN Common Class and the class dedicated to the MX100 (see the figure below).

- CDAQChInfo
 - CDAQMXChID
 - CDAQMXChInfo
 - CDAQMXChConfig
 - CDAQDataInfo
 - CDAQMXDataInfo
 - CDAQDateTime
 - CDAQMXDateTime
 - CDAQHandler
 - CDAQMX
 - CDAQMXChConfigData
 - CDAQMXConfig
 - CDAQMXDODData
 - CDAQMXNetInfo
 - CDAQMXSegment
 - CDAQMXStatus
 - CDAQMXSysInfo
 - CDAQMXBalanceData
 - CDAQMXBalanceResult
 - CDAQMXOutputData
 - CDAQMXAOPWMDData
 - CDAQMXTransmit
- : Class common to the MX100 and the DARWIN.
 - : Class dedicated to the MX100.

CDAQChInfo Class

Base class for storing the channel information data.

CDAQDataInfo Class

Base class for storing the measured data.

CDAQDateTime Class

Base class for storing the time information data.

CDAQHandler Class

Handler base class for performing communications with the instrument (MX100/DARWIN).

CDAQMX Class

Class derived from the CDAQHandler class. Provides functions for the MX100.

CDAQMXAOPWMDData

Class for storing the AO/PWM data on the MX100. It is a wrapper class of the MXAOPWMDData structure.

CDAQMXBalanceData

Class for storing the initial balance data on the MX100. It is a wrapper class of the MXBalanceData structure.

CDAQMXBalanceResult

Class for storing the initial balance results on the MX100. It is a wrapper class of the MXBalanceResult structure.

CDAQMXChConfig Class

Class derived from the CDAQMXChID class. Class for storing the channel setup data. It is a wrapper class of the MXChConfig structure.

CDAQMXChConfigData class

Class for storing the channel setup data for all the channels. It is a wrapper class of the MXChConfigData structure.

CDAQMXChID Class

Class derived from the CDAQChInfo class. Class for storing channel ID information. It is a wrapper class of the MXChID structure.

CDAQMXChInfo Class

Class derived from the CDAQMXChID class. Class for storing the channel information data. It is a wrapper class of the MXChInfo structure.

CDAQMXConfig Class

Class for storing the setup data. It is a wrapper class of the MXChConfigData structure.

CDAQMXDataInfo Class

Class derived from the CDAQDataInfo class. Class for storing the measured data. It is a wrapper class of the MXDataInfo structure.

CDAQMXDateTime Class

Class derived from the CDAQDateTime class. Class for storing time information data. It is a wrapper class of the MXDateTime structure.

CDAQMXDOData Class

Class for storing the DO data. It is a wrapper class of the MXDOData structure.

CDAQMXNetInfo Class

Class for storing the network information data. It is a wrapper class of the MXNetInfo structure.

CDAQMXOutputData

Class for storing the output channel data of the MX100. It is a wrapper class of the MXOutputData structure.

CDAQMXSegment Class

Class for storing the display pattern of the 7-segment LED. It is a wrapper class of the MXSegment structure.

CDAQMXStatus Class

Class for storing the MX100 status data. It is a wrapper class of the MXStatus structure.

CDAQMXSysInfo Class

Class for storing the system configuration data of the MX100. It is a wrapper class of the MXSystemInfo structure.

CDAQMXTransmit

Class for storing the transmission output data of the MX100. It is a wrapper class of the MXTransmit structure.

2.2 Correspondence between the Functions and Class/Member Functions - MX100 -

This section indicates the correspondence between the functions that the software supports and the class.

Communication Functions

Function	Class and Member Function
Connect to the MX100.	CDAQMX::open
Disconnect from the MX100.	CDAQMX::close
Set the communication timeout.	CDAQMX::setTimeOut

Note

Setting of the communication timeout is not recommended because unexpected disconnection may occur due to the conflict with the timeout time when data is retrieved.

Control Functions

Starting/Stopping the FIFO

Function	Class and Member Function
Start the FIFO	CDAQMX::startFIFO
Stop the FIFO	CDAQMX::stopFIFO
Set auto control of the FIFO	CDAQMX::autoFIFO

Other Controls

Function	FIFO	Class and Member Function
Set time information on the MX100 as the number of seconds from the reference date/time (Jan. 1, 1970).	Stop	CDAQMX::setDateTime
Turn ON/OFF data saving to the CF card (data backup).	Continue	CDAQMX::setBackup
CF write mode	Stop	CDAQMXSysInfo: setCFWriteMode
Format the CF card.	Stop	CDAQMX::formatCF
• Reconfigure the system of the unit.	Stop	CDAQMX::initSystem
• Initialize the system of the unit.	Stop	
• Reset alarms (alarm ACK) of the unit.	Continue	

The FIFO column in the table indicates the FIFO operation when the function is executed while FIFO is running.

Stop: The FIFO stops when the function is executed.

Continue: The FIFO continues even when the function is executed.

With the backup settings, the CF write mode represents a partial change of the collectively obtained data. After changing the data, it must be sent all together.

Note

If the auto control of the FIFO is enabled, the FIFO is automatically resumed when the FIFO is stopped due to an execution of a function.

Setup Functions

Collective Setup

Function		FIFO	Class and Member Function
Configure the setup data collectively		Stop	CDAQMX::setConfig
Set the DO (Digital Output) data collectively		Continue	CDAQMX::setDOData
Set the basic settings collectively		Stop	CDAQMX::setMXConfig
Set the 7-segment LED display		Continue	CDAQMX::setSegment
Transmit AO/PWM data		Continue	CDAQMX::setAOPWMData
Send transmission output data		Continue	CDAQMX::setTransmit
Set initial balance data		Stop	CDAQMX::setBalance
Set output channel data		Stop	CDAQMX::setOutput
Initial balance data	Execute	Stop	CDAQMX::runBalance
	Reset	Stop	CDAQMX::resetBalance

For a description of the FIFO column in the table, see the explanation given in “Other Controls” on the previous page. With Visual Basic, setting data cannot be handled collectively.

Setup Changes

Function	Class and Member Function
Range setting SKIP (not used)	CDAQMXConfig::setSKIP
DC voltage input	CDAQMXConfig::setVOLT
Thermocouple input	CDAQMXConfig::setTC
RTD input	CDAQMXConfig::setRTD
Digital input (DI)	CDAQMXConfig::setDI
Difference computation between channels	CDAQMXConfig::setDELTA
Remote RJC	CDAQMXConfig::setRRJC
Resistance	CDAQMXConfig::setRES
Strain	CDAQMXConfig::setSTRAIN
AO	CDAQMXConfig::setAO
PWM	CDAQMXConfig::setPWM
Set the unit name of the channel.	CDAQMXChID::setUnit
Set the channel tag.	CDAQMXChID::setTag
Set a comment for the channel.	CDAQMXChID::setComment
Set the alarm.	CDAQMXChConfig::setAlarm
Set the RJC used on the channel.	CDAQMXChConfig::setRJCType
Set the filter on the channel.	CDAQMXChConfig::setFilter
Set the burnout detection action.	CDAQMXChConfig::setBurnout
Set the alarm to be assigned to the DO channel to which an alarm output was specified. (To set a DO channel to alarm output, use the setDOType function member.)	CDAQMXChConfig::setRefAlarm
Set the scan interval.	CDAQMXConfig::setInterval
Set the temperature unit.	CDAQMXConfig::setTempUnit
Set the ID number of the unit.	CDAQMXSysInfo::setUnitNo
Set the timeout value (the time from the disconnection to the start of saving to the CF card. For the calculation method of the timeout value, see appendix 3.	CDAQMXSysInfo::setCFTimeout
Select the signal type to be assigned to the DO channel.	CDAQMXConfig::setDOType
Select the signal type to be assigned to the AO channel.	CDAQMXConfig::setAOType
Select the signal type to be assigned to the PWM ch.	CDAQMXConfig::setPWMTType
Set the output channel data type.	CDAQMXOutputData::setOutputType
Set the output value when the output data power supply is ON and an error occurs.	CDAQMXOutputData::setChoice
Set the PWM output channel pulse interval (integral multiplication factor).	CDAQMXOutputData::setPulstime
Set the signal type to be assigned to the DO channel.	CDAQMXConfig::setDOType
Change a portion of the DO data.	CDAQMXDODData::setDO
Change a portion of the AO/PWM data.	CDAQMXAOPWMData::setAOPWM
Change a portion of the initial balance data.	CDAQMXBalanceData::setBalance
Change a portion of the transmission output data.	CDAQMXTransmit::setTransmit

Note

- To change the settings, use CDAQMX::getConfig to retrieve the setup data, use the setup change function member to modify the settings, and use CDAQMX::setConfig to set the data to the MX100 collectively.
- For DO data, after modifying the contents of CDAQMXDODData with the setup change function member, use CDAQMX::setDODData to set the data on the MX100 collectively.
- For AO/PWM data, after modifying the contents of CDAQMXAOPWMDData with the setup change function member, use CDAQMX::setAOPWMDData to set the data on the MX100 collectively.
- For transmission data, after modifying the contents of CDAQMXTransmit with the setup change function member, use CDAQMX::setTransmit to set the data on the MX100 collectively.

Data Retrieval Functions**Retrieval of System Status Data and System Configuration Data**

Function	Class and Member Function
Get system status data.	CDAQMX::getStatusData
Get system configuration data.	CDAQMX::getSystemConfig

Retrieval of Setup Data

Function	Class and Member Function
Get the setup data collectively.	CDAQMX::getConfig
Get basic settings collectively.	CDAQMX::getMXConfig
Declare the retrieval of the setup data.	CDAQMX::talkConfig
Retrieves setup data other than the channel setup data.	
Get channel setup data.	CDAQMX::getChConfig
Function used to retrieve channel setup data after declaring the retrieval of setup data using the talkConfig function member.	

Retrieval of DO Data

Function	Class and Member Function
Get the DO data collectively.	CDAQMX::getDODData

Retrieval of Channel Information Data

Function	Class and Member Function
Declare the retrieval of the channel information data.	CDAQMX::talkChInfo
Get channel information data.	CDAQMX::getChInfo

Retrieval of Measured Data (Channel Designation)

Function	Class and Member Function
Get the most recent data range of the specified channel.	CDAQMX::getChDataNo
Declare the retrieval of the measured data of the specified channel. Declare the retrieval of the instantaneous values of the specified channel.	CDAQMX::talkChData
Get the time information of the specified channel for each data number.	CDAQMX::getTimeData
Get the measured data of the specified channel.	CDAQMX::getChData

Retrieval of Measured Data (FIFO Designation)

Function	Class and Member Function
Get the most-recent data range of the specified FIFO no.	CDAQMX::getFIFODataNo
Declare the retrieval of the measured data of the specified FIFO number. Declare the retrieval of the instantaneous values of the specified FIFO number.	CDAQMX::talkFIFOData
Get the time information of the specified FIFO number for each data number.	CDAQMX::getTimeData
Get the measured data of the specified FIFO number.	CDAQMX::getChData

Retrieval of Initial Balance Data

Function	Class and Member Function
Get initial balance data.	CDAQMX::getBalance

Retrieves output channel data

Function	Class and Member Function
Get output channel data.	CDAQMX::getOutput
Get AO/PWM data and transmission output data.	CDAQMX::getAOPWMData

Utilities

Function	Class and Member Function
Insert the specified user count (user-defined order information) in the next packet to be issued.	CDAQMX::setUserTime
Get the MX100-specific error that was received last through communications.	CDAQMX::getLastError
Convert the measured value into a double-precision floating point number.	CDAQMXDataInfo::toDoubleValue
Convert the measured value into string.	CDAQMXDataInfo::toStringValue
Get the alarm type string.	CDAQMXDataInfo::getAlarmName
Get the maximum length of the alarm string.	CDAQMXDataInfo::getMaxLenAlarmName
Get the version number of this API.	CDAQMX::getVersionAPI
Get the revision number of this API.	CDAQMX::getRevisionAPI
Get the error message string.	CDAQMX::getErrorMessage
Get the maximum length of the error message string.	CDAQMX::getMaxLenErrorMessage
Get the number of the parameter on which an error was detected.	CDAQMX::getItemError
Convert AO/PWM output values to output data values.	DAQMXAOPWMData::toAOPWMValue
Convert AO/PWM output data values to output values.	CDAQMXAOPWMData::toRealValue
Check the validity of data numbers.	CDAQMXStatus::isDataNo
Convert to style version.	CDAQMXSysInfo::toStyleVersion

2.3 Programming - MX100/Visual C++ -

Adding the Path of the Include File

Add the path of the include file (DAQMX.h) to the project. The method of adding the include file varies depending on the environment used.

Declaration in the Source File

Write the declaration in the source file.

```
#include "DAQMX.h"
```

Note

The include file of the common section (DAQHandler.h) is referenced from the include file described above. Thus, declaration for it is not necessary.

Library Designation

Add the library (DAQMX.lib, DAQ Handler.lib) to the project. The method of adding the include file varies depending on the environment used.

This enables the use of all classes. It also enables the use of all Visual C functions.

Retrieval of the Measured Data

Program Example 1

This program retrieves measured data.

```

////////////////////////////////////
// MX100 sample for measurement
#include <stdio.h>
#include "DAQMX.h"////////////////////////////////////
////////////////////////////////////
int main(int argc, char* argv[])
{
    int rc; //return code
    CDAQMX daqMX; //class
    int flag;
    MXDataNo startNo, endNo, dataNo;
    MXUserTime usertime;
    CDAQMXDateTime datetime;
    CDAQMXChInfo chinfo;
    CDAQMXDataInfo datainfo(NULL, &chinfo);
    //connect
    rc = daqMX.open("192.168.1.12");
    //get by FIFO
    rc = daqMX.startFIFO();
    rc = daqMX.getFIFODataNo(0, &startNo, &endNo);
    rc = daqMX.talkFIFOData(0, startNo, endNo);
    do { //date time
        rc = daqMX.getTimeData(&dataNo, datetime, &usertime,
&flag);
    } while (! (flag & DAQMX_FLAG_ENDDATA));
    do { //measured data
        rc = daqMX.getChData(&dataNo, datainfo, &flag);
    } while (! (flag & DAQMX_FLAG_ENDDATA));
    rc = daqMX.stopFIFO();
    //disconnect
    rc = daqMX.close();
    return rc;
}
////////////////////////////////////

```

Description

Overview

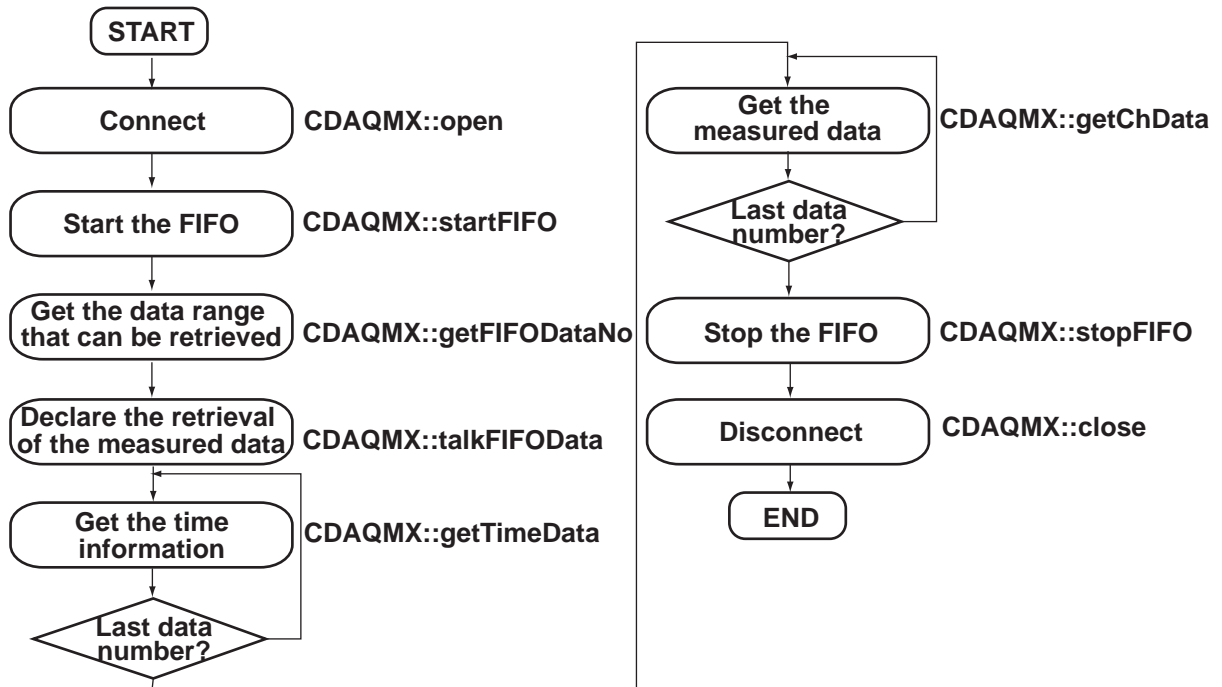
Data retrieval is possible by starting the FIFO. The range to be retrieved is specified by the FIFO number and the data number. The time stamp corresponding to the data number and the measured data are retrieved separately. The end is determined by the flag.

Include File Statement

```
#include "DAQMX.h"
```

Flow of the Process

The flow chart shown below omits the declaration section.



Communication Process

First, make a connection. After making the connection, the member functions become available. As a termination procedure, disconnect the communication.

Note

If there is no access for approximately three minutes, the MX100 drops the connection. Drop the connection if you are not accessing the MX100 for an extended time. Make the connection only when necessary.

Communication Connection

```
open("192.168.1.12")
```

The IP address of the MX100 is specified.

This statement implicitly specifies the communication constant DAQMX_COMMPORT (communication port number of the MX100).

Note

Communication can also be made when constructing the class. The connection is dropped when the class is destructed.

FIFO Start

```
startFIFO()
```

Starts the FIFO.

Retrieval of Data Range

`getFIFODataNo(0, &startNo, &endNo)`

Retrieves the range from the next data following the data retrieved last to the most recent data of the specified FIFO number using data numbers.

Talker

`talkFIFOData(0, startNo, endNo)`

Specifies the data range and declares the retrieval of the FIFO data (measured data retrieval declaration).

Retrieval of the FIFO Data Time Information

`getTimeData(&dataNo, datetime, &usertime, &flag)`

Gets the time information in the specified range in units of data numbers. The end is determined by the flag (constant `DAQMX_FLAG_ENDDATA`).

Note

“usertime” is the user-defined sequence information (user count). The value specified in advance using the `setUserTime` member function is stored.

Retrieval of FIFO Data

`getChData(&dataNo, datainfo, &flag)`

Gets the measured data in the specified range in units of channels. The end is determined by the flag (constant `DAQMX_FLAG_ENDDATA`).

FIFO Stop

`stopFIFO()`

Stops the FIFO.

Comm. cut

`closeMX(comm)`

Drops the connection.

Retrieval of Setup Data and Configuration

Program Example 2

This program executes the following three items. This program contains all three items, but each item can be written and executed separately.

- Collective retrieval of setup data
- Collective setting of setup data
- Setting a DC voltage range for the channels

```

////////////////////////////////////
// MX100 sample for configuration
#include <stdio.h>
#include "DAQMX.h"
////////////////////////////////////
int main(int argc, char* argv[])
{
    int rc; //return code
    CDAQMX daqMX; //class
    CDAQMXConfig configdata;
    //connect
    rc = daqMX.open("192.168.1.12");
    //get
    rc = daqMX.getConfig(configdata);
    //set
    rc = daqMX.setConfig(configdata);
    //range
    rc = daqMX.getConfig(configdata);
    configdata.setVOLT(1, DAQMX_RANGE_VOLT_20MV);
    rc = daqMX.setConfig(configdata);
    //disconnect
    rc = daqMX.close();
    return rc;
}
////////////////////////////////////

```

Description

Collective Retrieval of Setup Data

`getConfig(configdata)`

The setup data below can be retrieved by the collective retrieval of setup data.

- System configuration data: See the `CDAQMXSysInfo` class (section 2.5).
- Status: See the `CDAQMXStatus` class (section 2.5).
- Basic settings: See the `CDAQMXConfig` class (section 2.5).

Note

The setup data can also be retrieved using the talkConfig member function and the getChConfig member function. The talkConfig function is used to declare the retrieval of the setup data and retrieve the system configuration data, status, and network information data. Then, the getChConfig function is used to retrieve channel setup data in units of channels.

Collective Setting of Setup Data

setConfig(configdata)

The data below can be set by the collective setting of setup data.

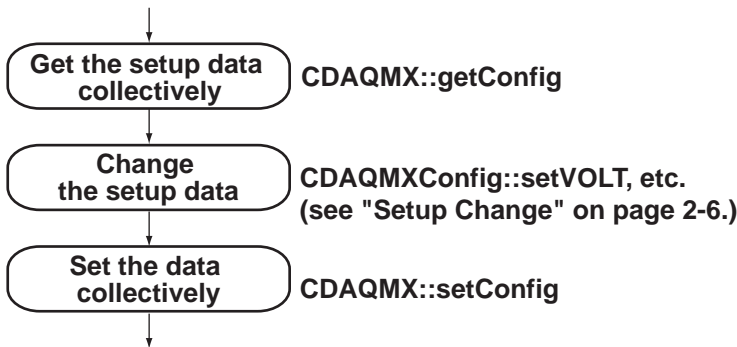
- System configuration data: See CDAQMXSysInfo class (section 2.5).
- Basic settings: See the CDAQMXConfig class (section 2.5).

Setting a DC Voltage Range for the Channels

setVOLT(1, DAQMX_RANGE_VOLT_20MV)

Sets channel number 1 to DC voltage range 20 mV. Scaling is not used.

First, the setup data is retrieved collectively. Next, the change described above is made. Then, the data is set on the MX100 collectively. Use similar steps when making changes to the settings.



Error Processing

- Most member functions return the result of the function process using an error number (0 if successful).
- The member function CDAQMX::getErrorMessage can be used to get the error message string corresponding to the error number. The member function CDAQMX::getMaxLenErrorMessage can be used to get the maximum length of the error message string.
- The member function CDAQMX::getLastError can be used to get the errors from the MX100.

2.4 Details of the MX100/DARWIN Common Class

The classes are listed in alphabetical order by the class name.

CDAQChInfo Class

This class is a base class of the channel information data.
If necessary, you can override member data manipulation with an inheritance class.
A function is provided that checks whether the instance has inherited this class.
You can override with an inheritance class.

Public Members

Construct/Destruct

CDAQChInfo	Constructs an object.
~CDAQChInfo	Destructs an object.

Member Data Manipulation

initialize	Initializes the data member.
getChType	Gets the channel type.
getChNo	Gets the channel number.
getPoint	Gets the decimal point position.
setChType	Sets the channel type.
setChNo	Sets the channel number.
setPoint	Sets the decimal point position.

Utilities

isObject	Checks an object.
----------	-------------------

Operator

operator=	Executes substitution.
-----------	------------------------

Protected Members

Data Members

m_chType	Field for storing the channel type.
m_chNo	Field for storing the channel number.
m_point	Field for storing the decimal point position.

Private Members

None.

Member Functions (Alphabetical Order)

CDAQChInfo::CDAQChInfo

Syntax

```
CDAQChInfo(int chType = 0, int chNo = 0, int point= 0);  
virtual ~CDAQChInfo(void);
```

Parameters

chType	Specify the channel type.
chNo	Specify the channel number.
point	Specify the decimal point position.

Description

Constructs or destructs an object.

When constructing, the data member is set to the specified value.

Reference

```
setChNo setChType setPoint
```

CDAQChInfo::getChNo

Syntax

```
virtual int getChNo(void);
```

Description

Gets the value of the channel number field of the data member.

Return value

Returns the channel number.

CDAQChInfo::getChType

Syntax

```
virtual int getChType(void);
```

Description

Gets the value of the channel type field of the data member.

Return value

Returns the channel type.

CDAQChInfo::getPoint

Syntax

```
virtual int getPoint(void);
```

Description

Gets the value of the decimal point position field of the data member.

Return value

Returns the decimal point position.

CDAQChInfo::initialize

Syntax

```
virtual void initialize(void);
```

Description

Initializes the data member. The default value is 0.

Reference

```
setChType setChNo setPoint
```

CDAQChInfo::isObject

Syntax

```
virtual int isObject(const char * classname = "CDAQChInfo");
```

Parameters

classname Specify the class name using a string.

Description

Checks whether the specified class name was inherited.

If parameters are omitted, checks whether it is this class.

Classes that inherit this class must be overridden in order to check their own classes.

Returns true (valid) if the class was inherited. Otherwise, returns false (invalid).

Return value

Returns a return value of 1 (true), or 0 (false).

CDAQChInfo::operator=

Syntax

```
CDAQChInfo & operator=(CDAQChInfo & cChInfo);
```

Parameters

cChInfo Specify an object for substitution.

Description

Copies the data member using the specified object information data.

Return value

Returns the reference to the object.

CDAQChInfo::setChNo

Syntax

```
virtual void setChNo(int chNo);
```

Parameters

chNo Specify the channel number.

Description

Stores the specified value in the channel number field of the data member.

CDAQChInfo::setChType

Syntax

```
virtual void setChType(int chType);
```

Parameters

chType Specify the channel type.

Description

Stores the specified value in the channel type field of the data member.

CDAQChInfo::setPoint

Syntax

```
virtual void setPoint(int point);
```

Parameters

point Specify the decimal point position.

Description

Stores the specified value in the decimal point position field of the data member

CDAQDataInfo Class

This class is a base class of the measured data.

Measured values can be retrieved by associating them with the channel information data.

If necessary, you can override member data manipulation with an inheritance class.

A function is provided that checks whether the instance has inherited this class.

You can override with an inheritance class.

Public Members

Construct/Destruct

CDAQDataInfo Constructs an object.

~CDAQDataInfo Destructs an object.

Member Data Manipulation

initialize Initializes the data member.

getValue Gets the data value.

setValue Sets the data value.

Association

getClassChInfo Gets the association with the channel information data.

setClassChInfo Sets the association with the channel information data.

Measured value Format

getDoubleValue Gets the measured value.

getStringValue Gets the measured value as a string.

toDoubleValue Generates the measured value.

toStringValue Generates the measured value as a string.

Utilities

isObject Checks an object.

Operator

operator= Executes substitution.

Protected Members

Data Members

m_value Field for storing the data value.

m_pChInfo Association with the channel information data. Field for storing the pointer to CDAQChInfo.

Private Members

None.

Member Functions (Alphabetical Order)

CDAQDataInfo::CDAQDataInfo

Syntax

```
CDAQDataInfo(int value = 0, CDAQChInfo * pcChInfo = NULL);  
virtual ~CDAQDataInfo(void);
```

Parameters

value	Specify the data value.
pcChInfo	Specify the association with the channel information data.

Description

Constructs or destructs an object.

When constructing, the data member is set to the specified value.

When destructing, the associated channel information data is not deleted.

Reference

setClassChInfo setValue

CDAQDataInfo::getClassChInfo

Syntax

```
CDAQChInfo * getClassChInfo(void);
```

Description

Gets the value of the field associated with the channel information data of the data member. Returns NULL if the value is not specified.

Return value

Returns the association with the channel information data.

CDAQDataInfo::getDoubleValue

Syntax

```
double getDoubleValue(void);
```

Description

Generates the measured value from the data value of the data member and the decimal point position associated with the channel information data.

If there is no association with the channel information data, the decimal point position is 0.

Return value

Returns the measured value as a double-precision floating number.

Reference

getClassChInfo getValue toDoubleValue
CDAQChInfo::getPoint

CDAQDataInfo::getStringValue

Syntax

```
int getStringValue(char * strValue, int lenValue);
```

Parameters

strValue Specify the field where the string is to be stored.
lenValue Specify the byte size of the field where the string is to be stored.

Description

Generates the measured value from the data value of the data member and the decimal point position associated with the channel information data.

Converts the generated measured value into a string and stores it in the specified field.

If there is no association with the channel information data, the decimal point position is 0.

The string stored to the field includes the terminator.

The return value is the length of the actual string. The return value does not include the terminator.

Return value

Returns the length of the string.

Reference

```
getClassChInfo    getValue    toStringValue  
CDAQChInfo::getPoint
```

CDAQDataInfo::getValue

Syntax

```
virtual int getValue(void);
```

Description

Gets the value of the data value field of the data member.

Return value

Returns the data value.

CDAQDataInfo::initialize

Syntax

```
virtual void initialize(void);
```

Description

Initializes the data member. The default value is 0.

The association with the channel information data is not initialized.

Reference

```
setValue
```

CDAQDataInfo::isObject

Syntax

```
virtual int isObject(const char * classname = "CDAQDataInfo");
```

Parameters

classname Specify the class name using a string.

Description

Checks whether the specified class name was inherited.

If parameters are omitted, checks whether it is this class.

Classes that inherit this class must be overridden in order to check their own classes.

Returns true (valid) if the class was inherited. Otherwise, returns false (invalid).

Return value

Returns a return value of 1 (true), or 0 (false).

CDAQDataInfo::operator=

Syntax

```
CDAQDataInfo & operator=(CDAQDataInfo & cDataInfo);
```

Parameters

cDataInfo Specify an object for substitution.

Description

Copies the data member of the specified object.

Also copies the association with the channel information data.

Return value

Returns the reference to the object.

CDAQDataInfo::setClassChInfo

Syntax

```
void setClassChInfo(CDAQChInfo * pcChInfo);
```

Parameters

pcChInfo Specify the association with the channel information data.

Description

Stores the specified value in the association field of the channel information data of the data member.

CDAQDataInfo::setValue

Syntax

```
virtual void setValue(int value);
```

Parameters

value Specify the data value.

Description

Stores the specified value in the data value field of the data member.

CDAQDataInfo::toDoubleValue

Syntax

```
static double toDoubleValue(int value, int point);
```

Parameters

value Specify the data value.

point Specify the decimal point position.

Description

Generates the measured value from the specified data value and decimal point position.

Return value

Returns the measured value as a double-precision floating number.

CDAQDataInfo::toStringValue

Syntax

```
static int toStringValue(int value, int point, char *  
strValue, int lenValue);
```

Parameters

value Specify the data value.

point Specify the decimal point position.

strValue Specify the field where the string is to be stored.

lenValue Specify the byte size of the field where the string is to be stored.

Description

Generates the measured value from the specified data value and decimal point position. Converts the generated measured value into a string and stores to the specified field. The string stored to the field includes the terminator.

The return value is the length of the actual string. The return value does not include the terminator.

Return value

Returns the length of the string.

Reference

toDoubleValue

CDAQDateTime Class

This class is a base class of time information data.
Numbers of seconds and milliseconds are passed to the data member.
If necessary, you can override member data manipulation with an inheritance class.
A function is provided that checks whether the instance has inherited this class.
You can override with an inheritance class.

Public Members

Construct/Destruct

CDAQDateTime	Constructs an object.
~CDAQDateTime	Destructs an object.

Member Data Manipulation

initialize	Initializes the data member.
getTime	Gets seconds.
getMilliSecond	Gets milliseconds.
setTime	Sets seconds.
setMilliSecond	Sets milliseconds.
setNow	Sets the current date/time.

Utilities

isObject	Checks an object.
toLocalDateTime	Converts to years, months, days, hours, minutes, and seconds according to the local time zone.

Operator

operator=	Executes substitution.
-----------	------------------------

Protected Members

Data Members

m_time	Field for storing the number of seconds from Jan. 1, 1970.
m_milli Second	Field for storing the number of milliseconds.

Private Members

None.

Member Functions (Alphabetical Order)

CDAQDateTime::CDAQDateTime

Syntax

```
CDAQDateTime(time_t time = 0, int millisecond = 0);
virtual ~CDAQDateTime(void);
```

Parameters

time	Specify seconds.
milliSecond	Specify milliseconds.

Description

Constructs or destructs an object.

When constructing, the data member is set to the specified value.

Reference

setMilliSecond setTime

CDAQDateTime::getMilliSecond

Syntax

```
virtual int getMilliSecond(void);
```

Description

Gets the value of the milliseconds field of the data member.

Return value

Returns milliseconds.

CDAQDateTime::getTime

Syntax

```
virtual time_t getTime(void);
```

Description

Gets the value of the decimal point position field of the data member.

Return value

Returns seconds.

CDAQDateTime::initialize

Syntax

```
virtual void initialize(void);
```

Description

Initializes the data member. The default value is 0.

Reference

setMillioSecond setTime

CDAQDateTime::isObject

Syntax

```
virtual int isObject(const char * classname = "CDAQDateTime");
```

Parameters

classname Specify the class name using a string.

Description

Checks whether the specified class name was inherited.

If parameters are omitted, checks whether it is this class.

Classes that inherit this class must be overridden in order to check their own classes.

Returns true (valid) if the class was inherited. Otherwise, returns false (invalid).

Return value

Returns a return value of 1 (true), or 0 (false).

CDAQDateTime::operator=

Syntax

```
CDAQDateTime & operator=(CDAQDateTime & cDateTime);
```

Parameters

cDateTime Specify an object for substitution.

Description

Copies the data member of the specified object .

Return value

Returns the reference to the object.

CDAQDateTime::setMilliSecond

Syntax

```
virtual void setMilliSecond(int milliSecond);
```

Parameters

milliSecond Specify milliseconds.

Description

Stores the specified value in the millisecond field of the data member

CDAQDateTime::setNow

Syntax

```
virtual void setNow(void);
```

Description

Gets the current data/time and stores it in the data member.

Milliseconds are set to 0.

Reference

setMilliSecond setTime

CDAQDateTime::setTime

Syntax

```
virtual void setTime(time_t time);
```

Parameters

time Specify seconds.

Description

Stores the specified value in the seconds field of the data member.

CDAQDateTime::toLocalDateTime

Syntax

```
void toLocalDateTime(int * pYear, int * pMonth, int * pDay,
int * pHour, int * pMinute, int * pSecond);
static void toLocalDateTime(time_t sectime, int * pYear, int *
pMonth, int * pDay, int * pHour, int * pMinute, int *
pSecond);
```

Parameters

sectime Specify seconds.
pYear Specify the destination where the year value is returned.
pMonth Specify the destination where the month value is returned.
pDay Specify the destination where the day value is returned.
pHour Specify the destination where the hour value is returned.
pMinute Specify the destination where the minute value is returned.
pSecond Specify the destination where the second value is returned.

Description

Converts the specified number of seconds to years, months, days, hours, minutes, and seconds according to the local time zone.

The specified number is the number of seconds from Jan. 1, 1970. Seconds are obtained from the data member for parameters with no seconds specified.

A four digit value is returned in year.

A value between 1 and 12 is returned for the month.

A value between 1 and 31 is returned for the day.

A value between 0 and 23 is returned for the hour.

A value between 0 and 59 is returned for the minutes.

A value between 0 and 59 is returned for the seconds.

Returns 0 if conversion fails.

Reference

getTime

CDAQHandler Class

This class is a base class of the handler. It provides communication functions. The communication format is TCP/IP. The communication is controlled using the communication descriptor. The communication descriptor is stored in the communication descriptor field of the data member using a pointer to the general type. The communication descriptor is constructed when the connection is made and destructed when the connection is dropped.

To change the communication format, create an inheritance class and override all communication member functions.

The data acquisition function is defined so that it can be called without knowing the model. However, this class cannot be used with only the definition. It must be overridden with the inheritance class before being implemented.

A function is provided that checks whether the instance has inherited this class. You can override with an inheritance class.

Public Members

Construct/Destruct

CDAQHandler	Constructs an object.
~CDAQHandler	Destructs an object.

Communication Functions

open	Establishes a connection.
close	Drops the connection.
sendLine	Sends string data.
receiveLine	Receives string data by lines.
setTimeout	Sets the communication timeout (setting of the communication timeout is not recommended (see section 2.2)).

Data Retrieval Functions

getData	Gets the measured data.
getChannel	Gets the channel information data.

Utilities

getVersionAPI	Gets the version number of this API.
getRevisionAPI	Gets the revision number of this API.
getErrorMessage	Gets the error message string
getMaxLenErrorMessage	Gets the maximum length of the error message string.
isObject	Checks an object.

Protected Members

Data Members

m_comm	Field for storing the communication descriptor.
m_nRemainSize	Field for storing the remaining size of the received data.

Communication Functions

send	Sends the data.
receive	Receives the data.

Utilities

receiveRemain	Receives the remaining bytes and discards them.
getVersionDLL	Gets the version number of the DLL.
getRevisionDLL	Gets the revision number of the DLL.

Private Members

None

Member Functions (Alphabetical Order)

CDAQHandler::CDAQHandler

Syntax

```
CDAQHandler(void);
CDAQHandler(const char * strAddress, unsigned int uiPort, int
* errCode = NULL);
virtual ~CDAQHandler(void);
```

Parameters

strAddress	Specify the IP address as a string.
uiPort	Specify the port number.
errCode	Specify the destination where the error number is to be returned.

Description

Constructs or destructs an object.

When constructing, the data member is initialized. The default value as a general rule is 0 (NULL). When the parameters are specified, a connection is established (open) during construction. If the return destination is specified, the error number during connection is returned.

When destructing, the data member field is released. Connection is dropped (close) when the communication descriptor exists. The error number is not returned.

Reference

close open

CDAQHandler::close

Syntax

```
virtual int close(void);
```

Description

Drops the connection. Destructs the communication descriptor.

Return value

Returns an error number.

Error:

Not connected Not connected.

Communication error An error was detected in communications.

CDAQHandler::getChannel

Syntax

```
virtual int getChannel(int chType, int chNo, CDAQChInfo &
cChInfo);
```

Parameters

chType Specify the channel type.

chNo Specify the channel number.

cChInfo Specify the destination where the channel information data
is to be returned.

Description

This function gets the channel information data by channels.

This function must be overridden with the inheritance class of each model. Returns
an error number if not overridden.

Return value

Returns an error number.

Error:

Not Support Not supported.

CDAQHandler::getData

Syntax

```
virtual int getData(int chType, int chNo, CDAQDateTime &
    cDateTime, CDAQDataInfo & cDataInfo);
```

Parameters

chType	Specify the channel type.
chNo	Specify the channel number.
cDateTime	Specify the destination where the time information data is to be returned.
cDataInfo	Specify the destination where the measured data is to be returned.

Description

This function gets the instantaneous values in units of channels.

This function must be overridden with the inheritance class of each model. Returns an error number if not overridden.

Return value

Returns an error number.

Error:

Not Support	Not supported.
-------------	----------------

CDAQHandler::getErrorMessage

Syntax

```
static const char * getErrorMessage(int errCode);
```

Parameters

errCode	Specify the error number.
---------	---------------------------

Description

Gets the error message string corresponding to the error number specified by the parameter. If the value is outside the range, the message string is set to Unknown.

Return value

Returns a pointer to the error message string.

CDAQHandler::getMaxLenErrorMessage

Syntax

```
static const int getMaxLenErrorMessage(void);
```

Description

Gets the maximum length of the error message string.

The value is the number of bytes.

The return value does not include the terminator.

Return value

Returns the length of the string.

CDAQHandler::getRevisionAPI

Syntax

```
static const int getRevisionAPI(void);
```

Description

Gets the revision number of this API.

Return value

Returns the revision number.

CDAQHandler::getRevisionDLL

Syntax

```
static const int getRevisionDLL(void);
```

Description

Gets the revision number of the DLL.

Return value

Returns the revision number.

CDAQHandler::getVersionAPI

Syntax

```
static const int getVersionAPI(void);
```

Description

Gets the version number of this API.

Return value

Returns the version number.

CDAQHandler::getVersionDLL

Syntax

```
static const int getVersionDLL(void);
```

Description

Gets the version number of the DLL.

Return value

Returns the version number.

CDAQHandler::isObject

Syntax

```
virtual int isObject(const char * classname = "CDAQHandler");
```

Parameters

classname	Specify the class name using a string.
-----------	--

Description

Checks whether the specified class name was inherited.

If parameters are omitted, checks whether it is this class.

Classes that inherit this class must be overridden in order to check their own classes.

Returns true (valid) if the class was inherited. Otherwise, returns false (invalid).

Return value

Returns a return value of 1 (true), or 0 (false).

CDAQHandler::open

Syntax

```
virtual int open(const char * strAddress, unsigned int uiPort);
```

Parameters

strAddress	Specify the IP address as a string.
uiPort	Specify the port number.

Description

Connects to the device with the IP address and port number specified by the parameters. Constructs the communication descriptor and stores it in the communication descriptor field of the data member.

If the communication descriptor already exists, it is not changed.

Return value

Returns an error number.

Error:

Creating connection is failure	Failed to construct the communication descriptor.
Connection exists already	Communication descriptor already exists.
Communication error	An error was detected in communications.

CDAQHandler::receive

Syntax

```
virtual int receive(unsigned char * bufData, int maxData, int * lenData);
```

Parameters

bufData	Specify the field where the received data is to be stored using a byte array.
maxData	Specify the byte size of the received data.
lenData	Specify the destination where the byte size of the actual data received is returned.

Description

Stores the received data in the field specified by the parameter up to the specified byte size.

Returns the byte size of the actual data received if the return destination is specified.

Return value

Returns an error number.

Error:

Not connected	Not connected.
Communication error	An error was detected in communications.

CDAQHandler::receiveLine

Syntax

```
virtual int receiveLine(char * strLine, int maxLine, int *
lenLine);
```

Parameters

strLine	Specify the field where the received string is to be stored.
maxLine	Specify the byte size of the field where the received string is to be stored.
lenLine	Specify the destination where the byte size of the actual string received is returned.

Description

Stores the received string to the field specified by the parameter until a line feed is detected or up to the specified byte size. Stores the received string excluding line feeds. Returns the byte size of the actual data received and stored if the return destination is specified.

Return value

Returns an error number.

Error:

Not connected	Not connected.
Communication Errors	An error was detected in communications.

CDAQHandler::receiveRemain

Syntax

```
int receiveRemain(void);
```

Description

If the remaining size field is greater than 0, the amount of data that is equal to the remaining size is received and discarded.

The remaining size field is set to 0.

If the remaining size field is 0 or less, the operation concludes successfully.

Return value

Returns an error number.

Reference

receive

CDAQHandler::send

Syntax

```
virtual int send(const unsigned char * bufData, int lenData);
```

Parameters

bufData	Specify the data to be sent using a byte array.
lenData	Specify the byte size of the data to be sent.

Description

Sends the amount of data to be sent specified by the parameter for the byte size.

Return value

Returns an error number.

Error:

Not connected	Not connected.
Communication error	An error was detected in communications.

CDAQHandler::sendLine

Syntax

```
virtual int sendLine(const char * strLine);
```

Parameters

strLine	Specify the string to be sent.
---------	--------------------------------

Description

Sends the specified string.

The line feed is added to the string before it is sent. Specify the string excluding the line feed.

Return value

Returns an error number.

Error:

Not connected	Not connected.
Communication error	An error was detected in communications.

2.5 Details of the MX100 Class

The classes are listed in alphabetical order by the class name.

CDAQMX Class

- CDAQHandler
 - CDAQMX

This class is derived from the CDAQHandler class.

This class provides functions for the MX100. The communication is in binary, therefore do not use primitive communication functions of the base class.

Information necessary for communications is stored internally. Send packets are generated internally from command number, user count, and communication packet version.

When retrieving data, execute the member of the retrieval declaration, then execute the member that retrieves the data number's worth of data, for each data. The presence or absence of the declaration is checked here.

A function is provided that checks whether the instance has inherited this class.

You can override with an inheritance class.

Public Members

Construct/Destruct

CDAQMX	Constructs an object.
~CDAQMX	Destructs an object.

Control Functions

Starting/Stopping the FIFO

startFIFO	Starts the FIFO.
stopFIFO	Stops the FIFO.
autoFIFO	Sets auto control of the FIFO.

Other Controls

setDateTime	Sets the time information data.
setBackup	Sets backup to the CF card.
formatCF	Formats the CF card.
initSystem	Initializes the system.
setSegment	Sets the 7-segment LED.
setDOData	Sets the DO data.
setAOPWMData	Sets AO/PWM data.
setTransmit	Sets the transmission output data.

Setup Functions

Collective Setup

setConfig	Sets the setup data.
setMXConfig	Sets the basic settings.
setOutput	Sets the basic setting output channel data.
setBalance	Sets initial balance data.
runBalance	Executes initial balancing.
resetBalance	Initializes the initial balance value.

Data Retrieval Functions

Retrieval of System Status Data and System Configuration Data

getStatusData	Gets the status.
getSystemConfig	Gets the system configuration data.

Retrieval of Setup Data

getConfig	Gets the setup data.
talkConfig	Declares the retrieval of the setup data.
getChConfig	Gets the channel setup data.
getMXConfig	Gets the basic settings.
getOutput	Gets the basic output channel data.
getBalance	Gets initial balance data.

Retrieval of Output Data

getDOData	Gets the DO data.
getAOPWMData	Gets AO/PWM data and transmission output data.

Retrieval of Channel Information Data

talkChInfo	Declares the retrieval of the channel information data.
getChInfo	Gets the channel information data.

Retrieval of the Measured Data

getChDataNo	Gets the data number of the channel.
talkChData	Declares the retrieval of the measured data according to the channel specification.
getTimeData	Gets the time information data of the measured data.
getFIFODataNo	Gets the data number of the FIFO.
talkFIFOData	Declares the retrieval of the measured data according to the FIFO specification.
getChData	Gets the measured data.
getTimeData	Gets the time information data of the measured data.

Utilities

getUserTime	Gets the user count.
setUserTime	Sets the user count.
getLastError	Gets the MX100-specific error.
getItemError	Gets the setting item number.

- **Overridden Members**

Communication Functions

open Establishes connection.

Data Retrieval Functions

getData Gets the measured data.

getChannel Gets the channel information data.

Utilities

isObject Checks an object.

- **Inherited Members**

See CDAQHandler.

close getErrorMessage getMaxLengthErrorMessage getRevisionAPI getVersionAPI

receiveLine sendLine setTimeout

Protected Members

Data Members

m_nNo	Field for storing the command number.
m_nLastError	Field for storing MX100-specific errors.
m_bAutoFIFO	Field for storing the FIFO auto control.
m_IIUserTime	Field for storing the user count.
m_nSessionNo	Field for storing the session number.
m_chFIFONo	Field for storing the FIFO number per channel.
m_chFIFOIdx	Field for storing the channel sequence number in the FIFO per channel.
m_chDataType	Field for storing the data type per channel.
m_chDeciPos	Field for storing the decimal point position per channel.
m_lastFIFODataNo	Field for storing the last data number per FIFO.
m_lastChDataNo	Field for storing the last data number per channel.
m_startChNo	Field for storing the start channel number.
m_endChNo	Field for storing the end channel number.
m_curChNo	Field for storing the current channel number.
m_startFIFOIdx	Field for storing the start channel sequence number of the FIFO.
m_endFIFOIdx	Field for storing the end channel sequence number of the FIFO.
m_curFIFOIdx	Field for storing the current channel sequence number of the FIFO.
m_startDataNo	Field for storing the start data number.
m_endDataNo	Field for storing the end data number.
m_curDataNo	Field for storing the current data number.
m_nFIFONo	Field for storing the FIFO number.

m_nDataNum	Field for storing the data number.
m_nChNum	Field for storing the number of channels.
m_nTimeNum	Field for storing the remaining number of time information data of the measured data.
m_packetVer	Field for storing the communication packet version.
m_nItemError	Field for storing the setting item number.
m_bTalkConfig	Declaration flag for retrieval of the setup data.
m_bTalkChInfo	Declaration flag for retrieval of the channel information data.
m_bTalkData	Declaration flag for retrieval of the measured data.

Communication Functions

runCommand	Executes a command.
sendPacket	Sends a packet.
receivePacket	Receives a packet.
receiveBlock	Receives a block.
runPacket	Sends/receives packets.
receiveBuffer	Receives the “size” amount of data, including the size information.

Internal Commands

nop	Executes the NOP command.
registry	Executes the registry command.

Member Data Manipulation

getNo	Gets the command number.
incCurDataNo	Increments the current data number.
incCurFIFOIdx	Increments the channel sequence number in the current FIFO.
getDataNo	Gets the data number.
searchChNo	Gets the channel number from the channel sequence number in the FIFO.
clearAttr	Initializes the data member.
clearData	Initializes the data member related to the retrieval of the measured data.
getPacketVersion	Gets the communication packet version.
clearLastDataNoCh	Initializes the last data number of each channel.
clearLastDataNoFIFO	Initializes the last data number of each FIFO.

Utilities

getVersionDLL	Gets the version number of the DLL.
getRevisionDLL	Gets the revision number of the DLL.

• Inherited Members

See CDAQHandler.
m_comm m_nRemainSize
receive receiveRemain send

Private Members

None.

Member Functions (Alphabetical Order)

CDAQMX::autoFIFO

Syntax

```
int autoFIFO(int bAuto);
```

Parameters

bAuto Specify auto control using a Boolean value.

Description

Sets auto control.

Stores the specified value in the automatic control field of the data member.

When enabled, the FIFO is started.

Return value

Returns an error number.

Reference

startFIFO

CDAQMX::CDAQMX

Syntax

```
CDAQMX(void);
CDAQMX(const char * strAddress, unsigned int uiPort =
DAQMX_COMMPORT, int * errCode = NULL);
virtual ~CDAQMX(void);
```

Parameters

strAddress Specify the IP address as a string.

uiPort Specify the port number.

errCode Specify the destination where the error number is to be returned.

Description

Constructs or destructs an object.

When constructing, the data member is initialized. The default value as a general rule is 0 (NULL). When the parameters are specified, a connection is established (open) during construction. If the return destination is specified, the error number during connection is returned.

When destructing, the data member field is released. The connection is dropped (close) when the communication descriptor exists. The error number is not returned.

Reference

clearAttr close open
CDAQHandler::CDAQHandler

CDAQMX::clearAttr

Syntax

```
void clearAttr(void);
```

Description

Initializes all the data members. The default value as a general rule is 0.

Reference

clearData

CDAQMX::clearData

Syntax

```
void clearData(int sessionNo = 0);
```

Parameters

sessionNo Specify the session number.

Description

Initializes the data member to start the retrieval of the measured data.

Stores the specified value in the session number field of the data member.

Reference

clearLastDataNoCh clearLastDataNoFIFO

CDAQMX::clearLastDataNoCh

Syntax

```
void clearLastDataNoCh(int chNo = DAQMX_CHNO_ALL);
```

Parameters

chNo Specify the channel number.

Description

Initializes the last data number field of each channel specified in the data member.

If the constant for "Specify all Channels" is specified for the channel numbers, all channels are processed.

CDAQMX::clearLastDataNoFIFO

Syntax

```
void clearLastDataNoFIFO(int fifoNo = DAQMX_FIFONO_ALL);
```

Parameters

fifoNo Specify the FIFO number.

Description

Initializes the last data number field of each FIFO in the FIFO numbers specified in the data member.

If the constant for "Specify all FIFO numbers" is specified for the FIFO numbers, all FIFOs are processed.

CDAQMX::formatCF

Syntax

```
virtual int formatCF(void);
```

Description

Formats the CF card.

The FIFO stops. If auto control is enabled, the FIFO starts after the function completes successfully.

Return value

Returns an error number.

Reference

```
getNo getPacketVersion getUserTime runCommand startFIFO
stopFIFO
```

CDAQMX::getAOPWMDData

Syntax

```
int getAOPWMDData(CDAQMXAOPWMDData & cMXAOPWMDData,
CDAQMXTransmit & cMXTransmit);
```

Parameters

cMXAOPWMDData	Specify the destination where the AO/PWM data is to be returned.
cMXTransmit	Specify the destination where the transmission output data is to be returned.

Description

Gets of AO/PWM data and transmission output data.

Return value

Returns an error number.

Reference

```
getNo getPacketVersion getUserTime runCommand
CDAQMXAOPWMDData::initialize CDAQMXAOPWMDData::setAOPWM
CDAQMXTransmit::initialize CDAQMXTransmit::setTransmit
```


CDAQMX::getBalance

Syntax

```
int getBalance(CDAQMXBalanceData & cMXBalanceData);
```

Parameters

cMXBalanceData Specify the destination where the initial balance data is to be returned.

Description

Gets initial balance data.

Gets the setup data, and stores the initial balance data portion in the specified return destination.

Return value

Returns an error number.

Reference

```
getMXConfig CDAQMXBalanceData::initialize  
CDAQMXConfig::getClassMXBalanceData
```

CDAQMX::getChannel

Syntax

```
virtual int getChannel(int chType, int chNo, CDAQChInfo &  
cChInfo);
```

Parameters

chType Specify the channel type.
chNo Specify the channel number.
cChInfo Specify the destination where the channel information data is to be returned.

Description

This function gets the channel information data by channels.

Gets the channel information data of the specified channel.

The channel type is ignored.

Return value

Returns an error number.

Reference

```
getChInfo talkChInfo
```

CDAQMX::getChConfig

Syntax

```
int getChConfig(CDAQMXChConfig & cMXChConfig, int * pFlag =
NULL);
```

Parameters

cMXChConfig	Specify the destination where the channel setup data is to be returned.
pFlag	Specify the destination where the flag is to be returned.

Description

Gets the channel information data that was declared to be retrieved using the talkConfig function in units of channels. Analyzes information and stores the data in the return destination. When the last set of data is retrieved, the flag status is set. It is also set when the function ends in error. Do not perform communications using other functions until the data retrieval is completed.

Packets differ with the style number of the main unit.

Return value

Returns an error number.

Error:

Not support	Unsupported version. Or, the sequence of execution was incorrect.
-------------	---

Reference

```
getPacketVersion receiveBlock CDAQMXChConfig::setChNo
```

CDAQMX::getChData

Syntax

```
int getChData(MXDataNo * dataNo, CDAQMXDataInfo & cMXDataInfo,
int * pFlag = NULL);
```

Parameters

dataNo	Specify the destination where the data number is to be returned.
cMXDataInfo	Specify the destination where the measured data is to be returned.
pFlag	Specify the destination where the flag is to be returned.

Description

Gets the measured data that was declared to be retrieved using the talkChData and talkFIFOData functions in units of channels. Analyzes information and stores the data in the return destination.

If an association with channel information data exists in the measured data return destination, channel information data that identifies the measured data is stored.

When the last set of data is retrieved, the flag status is set.

It is also set when the function ends in error.

Do not perform communications using other functions until the data retrieval is completed.

Return value

Returns an error number.

Reference

```
incCurFIFOIdx receiveBlock searchChNo CDAQMXChInfo::setChNo
CDAQMXChInfo::setFIFONo CDAQMXChInfo::setPoint
CDAQMXChInfo::setValid CDAQMXDataInfo::getClassMXChInfo
```

CDAQMX::getChDataNo

Syntax

```
int getChDataNo(int chNo, MXDataNo * startDataNo, MXDataNo *
endDataNo);
```

Parameters

chNo	Specify the channel number.
startDataNo	Specify the destination where the start data number is to be returned.
endDataNo	Specify the destination where the end data number is to be returned.

Description

Gets the data number of measured data that can be retrieved.

Gets the data range that can be retrieved starting from the next data after the measured data that was received last through channel designation.

Return value

Returns an error number.

Error:

Not support	The channel number is outside the range.
-------------	--

Reference

getDataNo

CDAQMX::getChInfo

Syntax

```
int getChInfo(CDAQMXChInfo & cMXChInfo, int * pFlag = NULL);
```

Parameters

cMXChInfo Specify the destination where the channel information data is to be returned.

pFlag Specify the destination where the flag is to be returned.

Description

Gets the channel information data that was declared to be retrieved using the talkChInfo function in units of channels. Analyzes information and stores the data in the return destination. When the last set of data is retrieved, the flag status is set. It is also set when the function ends in error. Do not perform communications using other functions until the data retrieval is completed.

Stores required information in the each channel information storage field of the data member.

Return value

Returns an error number.

Error:

Not support The sequence of execution was incorrect.

Reference

```
receiveBlock CDAQMXChInfo::getFIFOIndex
CDAQMXChInfo::getFIFONo CDAQMXChInfo::getPoint
CDAQMXChInfo::setChNo CDAQMXChInfo::setFIFOIndex
CDAQMXChInfo::setFIFONo
```

CDAQMX::getConfig

Syntax

```
int getConfig(CDAQMXConfig & cMXConfig);
```

Parameters

cMXConfig Specify the destination where the setup data is to be returned.

Description

Gets the setup data.

Gets the System configuration data, status, and basic settings and merges the information.

Return value

Returns an error number.

Reference

```
getMXConfig getStatusData getSystemConfig
```

CDAQMX::getData

Syntax

```
virtual int getData(int chType, int chNo, CDAQDateTime &
cDateTime, CDAQDataInfo & cDataInfo);
```

Parameters

chType	Specify the channel type.
chNo	Specify the channel number.
cDateTime	Specify the destination where the time information data is to be returned.
cDataInfo	Specify the destination where the measured data is to be returned.

Description

This function gets the instantaneous values in units of channels.
Gets the measured data of the specified channel.
The channel type is ignored.

Return value

Returns an error number.

Reference

getChData getTimeData talkChData

CDAQMX::getDataNo

Syntax

```
int getDataNo(int fifoNo, MXDataNo prevLast, MXDataNo *
startDataNo, MXDataNo * endDataNo);
```

Parameters

fifoNo	Specify the FIFO number.
prevLast	Specify the data number retrieved last.
startDataNo	Specify the destination where the start data number is to be returned.
endDataNo	Specify the destination where the end data number is to be returned.

Description

Gets the status data and calculates the range of measured data that can be retrieved. If measured data that can be retrieved does not exist, negative numbers are returned.

Return value

Returns an error number.

Error:

Not support The FIFO number is outside the range.

Reference

getStatusData
CDAQMXStatus::getNewDataNo CDAQMXStatus::getOldDataNo

CDAQMX::getDOData

Syntax

```
int getDOData(CDAQMXDOData & cMXDoData);
```

Parameters

cMXDoData Specify the destination where the DO data is to be returned.

Description

Gets the DO data.

Gets the data of all the channels collectively.

Return value

Returns an error number.

Reference

```
getNo    getPacketVersion    getUserTime    runCommand  
CDAQMXDOData::initialize    CDAQMXDOData::setDO
```

CDAQMX::getFIFODataNo

Syntax

```
int getFIFODataNo(int fifoNo, MXDataNo * startDataNo, MXDataNo  
* endDataNo);
```

Parameters

fifoNo Specify the FIFO number.

startDataNo Specify the destination where the start data number is to be returned.

endDataNo Specify the destination where the end data number is to be returned.

Description

Gets the data number of measured data that can be retrieved.

Gets the data range that can be retrieved starting from the next data after the measured data that was received last through FIFO designation.

Return value

Returns an error number.

Error:

Not support The FIFO number is outside the range.

Reference

```
getDataNo
```

CDAQMX::getItemError

Syntax

```
int getItemError(void);
```

Description

Gets the value of the setting item number field from the data member.

Return value

Returns the setting item number.

CDAQMX::getLastError

Syntax

```
int getLastError(void);
```

Description

Gets the value in the MX100-specific error field from the data member.

Return value

Returns the MX100-specific error.

CDAQMX::getMXConfig

Syntax

```
int getMXConfig(CDAQMXConfig & cMXConfig);
```

Parameters

`cMXConfig` Specify the destination where the setup data is to be returned.

Description

Gets the basic settings.

Packets differ with the style number of the MX100.

Return value

Returns an error number.

Error:

Not support Unsupported version.

Reference

```
getNo getPacketVersion getUserTime runCommand
```

CDAQMX::getNo

Syntax

```
int getNo(void);
```

Description

Gets the value in the command number field from the data member.

Increments the command number.

Return value

Returns the command number.

CDAQMX::getOutput

Syntax

```
int getOutput(CDAQMXOutputData & cMXOutputData);
```

Parameters

cMXOutputData Specify the destination where the output channel data is to be returned.

Description

Gets the basic output channel data.

Gets the setup data, and stores the output channel data portion in the specified return destination.

Return value

Returns an error number.

Reference

```
getMXConfig CDAQMXConfig::getClassMXOutputData  
CDAQMXOutputData::initialize
```

CDAQMX::getPacketVersion

Syntax

```
int getPacketVersion(void);
```

Description

Gets the value of the communication packet version field from the data member.

Return value

Returns the communication packet version.

CDAQMX::getRevisionDLL

Syntax

```
static const int getRevisionDLL(void);
```

Description

Gets the revision number of this DLL.

Return value

Returns the revision number of this DLL.

CDAQMX::getStatusData

Syntax

```
int getStatusData(CDAQMXStatus & cMXStatus);
```

Parameters

cMXStatus Specify the destination where the status data is to be returned.

Description

Gets the status data.

Packets differ with the style number of the MX100.

Return value

Returns an error number.

Error:

Not support Unsupported version.

Reference

getNo getPacketVersion getUserTime runCommand

CDAQMX::getSystemConfig

Syntax

```
int getSystemConfig(CDAQMXSysInfo & cMXSysInfo);
```

Parameters

cMXSysInfo Specify the destination where the system configuration data is to be returned.

Description

Gets the system configuration data.

Return value

Returns an error number.

Reference

getNo getPacketVersion getUserTime runCommand

CDAQMX::getTimeData

Syntax

```
int getTimeData(MXDataNo * dataNo, CDAQMXDateTime &
cMXDateTime, MXUserTime * pUserTime = NULL, int * pFlag =NULL);
```

Parameters

dataNo	Specify the destination where the data number is to be returned.
cMXDateTime	Specify the destination where the time information data is to be returned.
pUserTime	Specify the destination where the user count is to be returned.
pFlag	Specify the destination where the flag is to be returned.

Description

Gets the time information data that was declared to be retrieved using the talkChData and talkFIFOData functions in units of data numbers. Analyzes information and stores the data in the return destination.

When the last set of data is retrieved, the flag status is set. It is also set when the function ends in error.

Do not perform communications using other functions until the data retrieval is completed.

After the data is retrieved by this function member, use getChData to retrieve the data for each channel within the data number.

Return value

Returns an error number.

Error:

Not support The sequence of execution was incorrect.

Reference

```
incCurDataNo receiveBlock CDAQMXDateTime::setMilliSecond
CDAQMXDateTime::setTime
```

CDAQMX::getUserTime

Syntax

```
MXUserTime getUserTime(void);
```

Description

Gets the value in the user count field from the data member.

Return value

Returns the user count.

CDAQMX::getVersionDLL

Syntax

```
static const int getVersionDLL(void);
```

Description

Gets the version number of this DLL.

Return value

Returns the version number of this DLL.

CDAQMX::incCurDataNo

Syntax

```
MXDataNo incCurDataNo(void);
```

Description

Gets the value in the current data number field from the data member.
Increments the current data number. When the end data number is exceeded, the number is reset to the start data number.

Return value

Returns the current data number.

CDAQMX::incCurFIFOIdx

Syntax

```
int incCurFIFOIdx(void);
```

Description

Gets the value in the channel sequence number field in the current FIFO from the data member.
Increments the channel sequence number in the current FIFO. When the end channel sequence number in the FIFO is exceeded, the number is reset to the start channel sequence number in the FIFO.

Return value

Returns the channel sequence number in the current FIFO.

CDAQMX::initSystem

Syntax

```
int initSystem(int iCtrl);
```

Parameters

iCtrl Specify the system control type.

Description

Executes the operation of the specified system control type.
The FIFO stops. If auto control is enabled, the FIFO starts after the function completes successfully.
However, the alarm reset time is excluded.

Return value

Returns an error number.

Reference

```
getNo getPacketVersion getUserTime runCommand getConfig  
setMXConfig CDAQMXConfig::reconstruct
```

CDAQMX::isObject

Syntax

```
virtual int isObject(const char * classname = "CDAQMX");
```

Parameters

classname Specify the class name using a string.

Description

Checks whether the specified class name was inherited.

If parameters are omitted, checks whether it is this class.

Classes that inherit this class must be overridden in order to check their own classes.

Returns true (valid) if the class was inherited. Otherwise, returns false (invalid).

If different from this class, checks the parent class.

Return value

Returns a Boolean value.

Reference

CDAQHandler::isObject

CDAQMX::nop

Syntax

```
int nop(void);
```

Description

Executes a command.

The MX100 only returns a response.

Return value

Returns an error number.

Reference

getNo getPacketVersion getUserTime runCommand

CDAQMX::open

Syntax

```
virtual int open(const char * strAddress, unsigned int uiPort  
= DAQMX_COMMPORT);
```

Parameters

strAddress	Specify the IP address as a string.
uiPort	Specify the port number.

Description

Connects to the device with the IP address and port number specified by the parameters.

The port number can be omitted. If omitted, it is set to the MX100 communication port number.

Initializes the data member.

Executes the registry command. If execution fails, the connection is dropped.

Return value

Returns an error number.

Reference

```
clearAttr close registry  
CDAQHandler::open
```

CDAQMX::receiveBlock

Syntax

```
int receiveBlock(unsigned char * pBlock, int lenBlock);
```

Parameters

pBlock	Specify the field where the block is to be stored using a byte array.
lenBlock	Specify the byte size of the block.

Description

Receives the specified block.

Updates the remaining size field.

Return value

Returns an error number.

Error:

Not data The field size is not consistent.

Reference

```
receive
```

CDAQMX::receiveBuffer

Syntax

```
int receiveBuffer(unsigned char * pBuf, int lenBuf, int *
realLen, int * sizeBuf = NULL)
```

Parameters

pBuf	Specify the field where the data is to be stored using a byte array.
lenBuf	Specify the number of bytes of the data field.
realLen	Specify the destination where the byte size of the actual data received is returned.
sizeBuf	Specify the destination where the size information is to be returned.

Description

Receives the data including the size information. Receives and stores the amount of data specified by the size information or the specified number of bytes in the field specified by the parameter. Returns the number of bytes and size information of the actual data received if the return destination is specified.

Return value

Returns an error number.

Error:

Not acknowledge The size of the response packet is wrong.

Reference

receiveRemain receive

CDAQMX::receivePacket

Syntax

```
virtual int receivePacket(unsigned char * ackBuf, int lenAck,
int * realLen);
```

Parameters

ackBuf	Specify the field where the response packet is to be stored using a byte array.
lenAck	Specify the byte size of the response packet.
realLen	Specify the destination where the byte size that is actually received is returned.

Description

Receives the specified packet and decodes it.

If the received packet is an error packet, the MX100-specific error is stored in the MX100-specific error field.

Return value

Returns an error number.

Error:

Commands are not processed successfully
Received an error packet.

Reference

receiveBuffer

CDAQMX::registry

Syntax

```
int registry(void);
```

Description

Executes a command.

This command notifies the MX100 main unit of the PC information (host name and address).

Stores the communication packet version in the communication packet version field of the data member.

Return value

Returns an error number.

Reference

```
getNo getPacketVersion getUserTime runCommand
```

CDAQMX::resetBalance

Syntax

```
int resetBalance(CDAQMXBalanceResult & cMXBalanceResult);
```

Parameters

cMXBalanceResult	Specify the destination where the initial balance result is to be returned.
------------------	---

Description

Resets initial balancing.

Stores the reset result, initial balance result, and initial balance value in the specified return destination.

Only channels specified as valid are reset.

The FIFO stops. If auto control is enabled, the FIFO starts after the function completes successfully.

The response to this function may take five seconds or longer.

Return value

Returns an error number.

Reference

```
getNo getPacketVersion getUserTime runCommand startFIFO  
stopFIFO CDAQMXBalanceResult::getBalanceValid  
CDAQMXBalanceResult::setBalance CDAQMXBalanceResult::setResult
```

CDAQMX::runBalance

Syntax

```
int runBalance(CDAQMXBalanceResult & cMXBalanceResult);
```

Parameters

cMXBalanceResult Specify the destination where the initial balance result is to be returned.

Description

Executes initial balancing.

Stores the execution result, initial balance result, and initial balance value in the specified return destination.

Execution takes place only on channels specified as valid.

The FIFO stops. If auto control is enabled, the FIFO starts after the function completes successfully.

The response to this function may take five seconds or longer.

Return value

Returns an error number.

Reference

```
getNo    getPacketVersion    getUserTime    runCommand    startFIFO
stopFIFO    CDAQMXBalanceResult::getBalanceValid
CDAQMXBalanceResult::setBalance
CDAQMXBalanceResult::setResult
```

CDAQMX::runCommand

Syntax

```
virtual int runCommand(unsigned char * reqBuf, int lenReq,
unsigned char * ackBuf, int lenAck);
```

Parameters

reqBuf Specify the request packet using a byte array.
lenReq Specify the byte size of the request packet.
ackBuf Specify the field where the response packet is to be stored using a byte array.
lenAck Specify the byte size of the response packet.

Description

Sends the specified request packet and receives the response packet. The response packet is stored in the specified field.

Return value

Returns an error number.

Error:

Not acknowledge No response was generated for the request.

Not support The request packets were incorrect.

Not data The response packet field is insufficient.

Reference

```
receivePacket    sendPacket
```

CDAQMX::runPacket

Syntax

```
virtual int runPacket(unsigned char * reqBuf, int lenReq,  
unsigned char * ackBuf, int lenAck);
```

Parameters

reqBuf	Specify the request packet using a byte array.
lenReq	Specify the byte size of the request packet.
ackBuf	Specify the field where the response packet is to be stored using a byte array.
lenAck	Specify the byte size of the response packet.

Description

Sends the specified request packet and receives the response packet. The response packet is stored in the specified field. This is a member provided for non-standard packets. The packets must be processed on the user side.

The use of a timeout is not recommended.

Return value

Returns an error number.

Reference

`receiveBuffer send`

CDAQMX::searchChNo

Syntax

```
int searchChNo(int fifoNo, int fifoIndex);
```

Parameters

fifoNo	Specify the FIFO number.
fifoIndex	Specify the channel sequence number in the FIFO.

Description

Searches for the channel number corresponding to the specified value from the information stored in the data member.

Returns 0 if it does not exist.

Return value

Returns the channel number.

CDAQMX::sendPacket

Syntax

```
virtual int sendPacket(unsigned char * reqBuf, int lenReq);
```

Parameters

reqBuf	Specify the request packet using a byte array.
lenReq	Specify the byte size of the request packet.

Description

Encodes the specified packet and sends it.

Return value

Returns an error number.

Reference

send

CDAQMX::setAOPWMData

Syntax

```
int setAOPWMData(CDAQMXAOPWMData & cMXAOPWMData);
```

Parameters

cMXAOPWMData	Specify the AO/PWM data.
--------------	--------------------------

Description

Sets AO/PWM data.

Return value

Returns an error number.

Reference

getNo getPacketVersion getUserTime runCommand
CDAQMXAOPWMData::getAOPWMValid
CDAQMXAOPWMData::getAOPWMValue

CDAQMX::setBalance

Syntax

```
int setBalance(CDAQMXBalanceData & cMXBalanceData);
```

Parameters

cMXBalanceData Specify initial balance data.

Description

Sets initial balance data.

Gets the setup data, updates it according to the specified initial balance data, and sends the setup data.

Only channels specified as valid are updated.

Return value

Returns an error number.

Reference

```
getMXConfig setMXConfig  
CDAQMXBalanceData::getBalanceValid  
CDAQMXBalanceData::setBalance  
CDAQMXConfig::getClassMXBalanceData
```

CDAQMX::setBackup

Syntax

```
int setBackup(int bBackup);
```

Parameters

bBackup Specify backup using a Boolean value.

Description

Sets backup on the device.

Return value

Returns an error number.

Reference

```
getNo getPacketVersion getUserTime runCommand
```

CDAQMX::setBalance

Syntax

```
int setBalance(CDAQMXBalanceData & cMXBalanceData);
```

Parameters

cMXBalanceData Specifies initial balance data.

Description

Sets initial balance data.

Gets the setup data, updates it according to the specified initial balance data, and sends the setup data.

Only channels specified as valid are updated.

Return value

Returns an error number.

Reference

```
getMXConfig setMXConfig  
CDAQMXBalanceData::getBalanceValid  
CDAQMXBalanceData::setBalance  
CDAQMXConfig::getClassMXBalanceData
```

CDAQMX::setConfig

Syntax

```
int setConfig(CDAQMXConfig & cMXConfig);
```

Parameters

cMXConfig Specify the setup data.

Description

Sets the setup data.

Sets the basic settings.

Return value

Returns an error number.

Reference

```
setMXConfig
```

CDAQMX::setDateTime

Syntax

```
virtual int setDateTime(CDAQMXDateTime * pcMXDateTime = NULL);
```

Parameters

pcMXDateTime Specify the time information data.

Description

Sets time information data on the device.

If the parameter's time information data is omitted, the current date/time of the PC is used.

Milliseconds are discarded.

The FIFO stops. If auto control is enabled, the FIFO starts after the function completes successfully.

The response to this function may take one second or longer.

An error occurs if the time is a negative number.

Return value

Returns an error number.

Error:

Not data Specified value incorrect.

Reference

```
getNo getPacketVersion getUserTime  
runCommand startFIFO stopFIFO  
CDAQMXDateTime::getMilliSecond  
CDAQMXDateTime::getTime  
CDAQMXDateTime::setNow
```

CDAQMX::setDOData

Syntax

```
int setDOData(CDAQMXDOData & cMXDoData);
```

Parameters

cMXDoData Specify the DO data.

Description

Sets the DO data.

Sets the data of all the channels collectively.

Return value

Returns an error number.

Reference

```
getNo getPacketVersion getUserTime runCommand  
CDAQMXDOData::getDOONOFF CDAQMXDOData::getDOValid
```

CDAQMX::setMXConfig

Syntax

```
int setMXConfig(CDAQMXConfig & cMXConfig);
```

Parameters

cMXConfig Specify the setup data.

Description

Sets basic settings on the device.

The input data is validated before it is sent.

The FIFO stops. If auto control is enabled, the FIFO starts after the function completes successfully.

Packets differ with the style number of the main unit.

The setting item number of the validation results are stored in the setting item number field of the data member.

Return value

Returns an error number.

Error:

Not data The input data is not valid.

Not support Unsupported version.

Reference

```
getNo getPacketVersion getUserTime runCommand startFIFO
stopFIFO CDAQMXConfig::isCorrect CDAQMXConfig::getItemError
```

CDAQMX::setOutput

Syntax

```
int setOutput(CDAQMXOutputData & cMXOutputData);
```

Parameters

cMXOutputData Specify the output channel data.

Description

Sets the output channel data.

Gets the setup data, updates it according to the specified output channel data, and sends the setup data.

If the output type is changed, it will not match the channel setup data.

Return value

Returns an error number.

Reference

```
getMXConfig setMXConfig CDAQMXConfig::getClassMXOutputData
```

CDAQMX::setSegment

Syntax

```
int setSegment(int dispType, int dispTime, CDAQMXSegment &
cNewMXSegment, CDAQMXSegment & cOldMXSegment);
```

Parameters

dispType	Specify the display format.
dispTime	Specify the display time.
cNewMXSegment	Specify the display pattern.
cOldMXSegment	Specify the destination where the previous display pattern is to be returned.

Description

Sets the display of the 7-segment LED.

Stores the 7-segment LED display pattern before the change if the return destination is specified.

Return value

Returns an error number.

Reference

```
getNo getPacketVersion getUserTime runCommand
CDAQMXSegment::getPattern CDAQMXSegment::setPattern
```

CDAQMX::setTransmit

Syntax

```
int setTransmit(CDAQMXTransmit & cMXTransmit);
```

Parameters

cMXTransmit	Specify the transmission output data.
-------------	---------------------------------------

Description

Sets the transmission output data.

Return value

Returns an error number.

Reference

```
getNo getPacketVersion getUserTime runCommand
CDAQMXTransmit::getTransmit
```

CDAQMX::setUserTime

Syntax

```
void setUserTime(MXUserTime userTime);
```

Parameters

userTime Specify the user count.

Description

Stores the specified value in the user count field of the data member.

CDAQMX::startFIFO

Syntax

```
int startFIFO(void);
```

Description

Starts the FIFO.

Gets the channel information data and sets necessary information in the data member related to the retrieval of the measured data.

Return value

Returns an error number.

Reference

```
clearData   getNo   getPacketVersion   getUserTime   getChInfo  
runCommand   talkChInfo
```

CDAQMX::stopFIFO

Syntax

```
int stopFIFO(void);
```

Description

Stops the FIFO.

Stops even if auto control is enabled.

Return value

Returns an error number.

Reference

```
getNo   getPacketVersion   getUserTime   runCommand
```

CDAQMX::talkChData

Syntax

```
int talkChData(int chNo, MXDataNo startDataNo =
DAQMX_INSTANTANEOUS, MXDataNo endDataNo =
DAQMX_INSTANTANEOUS);
```

Parameters

chNo	Specify the channel number.
startDataNo	Specify the start data number.
endDataNo	Specify the end data number.

Description

Declares the retrieval of the measured data.

Gets the measured data in the specified data number range. The range that is actually retrieved is not necessarily equal to the specified range.

If the data number is omitted, the instantaneous value is retrieved.

After executing this function member, use `getTimeData` to get the time information data for each data number.

Then, use `getChData` to retrieve the data for each channel within the data number.

Return value

Returns an error number.

Reference

`getNo` `getPacketVersion` `getUserTime` `receivePacket` `sendPacket`

CDAQMX::talkChInfo

Syntax

```
int talkChInfo(int startChNo = 1, int endChNo =
DAQMX_NUMCHANNEL);
```

Parameters

startChNo	Specify the start channel number.
endChNo	Specify the end channel number.

Description

Declares the retrieval of the channel information data from the start channel number to the end channel number.

After executing this function member, use the `getChInfo` function to retrieve the data for each channel.

Return value

Returns an error number.

Reference

`getNo` `getPacketVersion` `getUserTime` `receivePacket` `sendPacket`

CDAQMX::talkConfig

Syntax

```
int talkConfig(CDAQMXSysInfo & cMXSysInfo, CDAQMXStatus &
cMXStatus, CDAQMXNetInfo & cMXNetInfo);
```

Parameters

cMXSysInfo	Specify the destination where the system configuration data is to be returned.
cMXStatus	Specify the destination where the status data is to be returned.
cMXNetInfo	Specify the destination where the network information data is to be returned.

Description

Declares the retrieval of the setup data. Gets the setup data excluding the channel setup data. After executing this function member, use the getChConfig function to retrieve the data for each channel. Within the setup data, initial balance data and output channel data is retrieved using a separately-named retrieval function.

Return value

Returns an error number.

Reference

getNo getPacketVersion getStatusData getSystemConfig
getUserTime receivePacket sendPacket

CDAQMX::talkFIFOData

Syntax

```
int talkFIFOData(int fifoNo, MXDataNo startDataNo =
DAQMX_INSTANTANEOUS, MXDataNo endDataNo =
DAQMX_INSTANTANEOUS);
```

Parameters

fifoNo	Specify the FIFO number.
startDataNo	Specify the start data number.
endDataNo	Specify the end data number.

Description

Declares the retrieval of the measured data. Gets the measured data in the specified data number range. The range that is actually retrieved is not necessarily equal to the specified range. If the data number is omitted, the instantaneous value is retrieved. After executing this function member, use getTimeData to get the time information data for each data number.

Then, use getChData to retrieve the data for each channel within the data number.

Return value

Returns an error number.

Reference

getNo getPacketVersion getUserTime receivePacket sendPacket

CDAQMXAOPWMDData Class

This class stores the AO/PWM data of the MX100.
 It is a wrapper class of the MXAOPWMDData structure.
 It is a group of AO/PWM data from all the channels.
 Each data can be accessed by PWM data number or AO data number.
 This class can be used as an interface for setting and retrieving AO/PWM data.
 This class supports the utility that converts the specified output data values and actual output values.

Public Members

Construct/Destruct

CDAQMXAOPWMDData	Constructs an object.
~CDAQMXAOPWMDData	Destructs an object.

Structure Manipulation

getMXAOPWMDData	Gets the data in a structure.
setMXAOPWMDData	Sets the data in a structure.
initMXAOPWMDData	Initializes the data in a structure.

Member Data Manipulation

initialize	Initializes the data member.
getAOPWMValid	Gets the Boolean value.
getAOPWMValue	Gets the output data value.
setAOPWM	Sets AO/PWM data.

Operator

operator=	Executes substitution.
-----------	------------------------

Utilities

toAOPWMValue	Converts the output values to output data values.
toRealValue	Converts the output data values to output values.
isObject	Checks an object.

Protected Members

Data Members

m_MXAOPWMDData	The AO/PWM data storage field.
----------------	--------------------------------

Member Access

getMXAOPWM	Gets the AO/PWM data structure for each channel.
------------	--

Private Members

None.

Member Functions

CDAQMXAOPWMDData::CDAQMXAOPWMDData

Syntax

```
CDAQMXAOPWMDData(MXAOPWMDData * pMXAOPWMDData = NULL);
virtual ~CDAQMXAOPWMDData(void);
```

Parameters

pMXAOPWMDData Specify AO/PWM data.

Description

Constructs or destructs an object.

When constructing, the data member is set to the specified value. If not specified, the data member is initialized.

Reference

setMXAOPWMDData

CDAQMXAOPWMDData::getAOPWMValid

Syntax

```
int getAOPWMValid(int aopwmNo);
```

Parameters

aopwmNo Specify the AO/PWM data number.

Description

Gets the Boolean value of the specified data number from the data member.

If it does not exist, Invalid is returned.

Return value

Returns a Boolean value.

Reference

getMXAOPWM

CDAQMXAOPWMData::getAOPWMValue

Syntax

```
int getAOPWMValue(int aopwmNo);
```

Parameters

aopwmNo Specify the AO/PWM data number.

Description

Gets the output data value of the specified data number from the data member.
Returns 0 if it does not exist.

Return value

Returns the output data value.

Reference

getMXAOPWM

CDAQMXAOPWMData::getMXAOPWM

Syntax

```
MXAOPWM * getMXAOPWM(int aopwmNo);
```

Parameters

aopwmNo Specify the AO/PWM data number.

Description

Gets the structure of the specified data number from the data member.
Returns NULL if it does not exist.

Return value

Returns a pointer to the structure.

CDAQMXAOPWMData::getMXAOPWMData

Syntax

```
void getMXAOPWMData(MXAOPWMData * pMXAOPWMData);
```

Parameters

pMXAOPWMData Specify the destination where the AO/PWM data is to be returned.

Description

Gets the data in a structure.
Stores the contents of the data member in the specified structure.

CDAQMXAOPWMData::initialize

Syntax

```
virtual void initialize(void);
```

Description

Initializes the data member. The default value as a general rule is 0.

Reference

initMXAOPWMData

CDAQMXAOPWMDData::initMXAOPWMDData

Syntax

```
static void initMXAOPWMDData(MXAOPWMDData * pMXAOPWMDData);
```

Parameters

pMXAOPWMDData Specify AO/PWM data field.

Description

Initializes the specified field.
The default value as a general rule is 0.

CDAQMXAOPWMDData::isObject

Syntax

```
virtual int isObject(const char * classname =  
"CDAQMXAOPWMDData");
```

Parameters

classname Specify the class name using a string.

Description

Checks whether the specified class name was inherited. If parameters are omitted, checks whether it is this class. Classes that inherit this class must be overridden in order to check their own classes. Returns true (valid) if the class was inherited. Otherwise, returns false (invalid).

Return value

Returns a Boolean value.

CDAQMXAOPWMDData::operator=

Syntax

```
CDAQMXAOPWMDData & operator=(CDAQMXAOPWMDData & cMXAOPWMDData);
```

Parameters

cMXAOPWMDData Specify an object for substitution.

Description

Copies the data member of the specified object.

Return value

Returns the reference to the object.

CDAQMXAOPWMData::setAOPWM

Syntax

```
void setAOPWM(int aopwmNo, int bValid, int iAOPWMValue);
```

Parameters

aopwmNo	Specify the AO/PWM data number.
bValid	Specify the Boolean value.
iAOPWMValue	Specify the output data value.

Description

Stores the specified value in the field indicated by the specified data number of the data member.

If the data number is set to the constant for “Specify All AO/PWM data numbers,” the same value is stored to all data.

Reference

getMXAOPWM

CDAQMXAOPWMData::setMXAOPWMData

Syntax

```
void setMXAOPWMData(MXAOPWMData * pMXAOPWMData);
```

Parameters

pMXAOPWMData	Specify AO/PWM data.
--------------	----------------------

Description

Sets the data in a structure.

Stores the contents of the specified structure in the data member.

If not specified, the data member is initialized.

Reference

initMXAOPWMData

CDAQMXAOPWMData::toAOPWMValue

Syntax

```
static int toAOPWMValue(double realValue, int iRangeAOPWM);
```

Parameters

realValue	Specify the output value.
iRangeAOPWM	Specify the range type of the AO or PWM range.

Description

Converts the output values to output data values according to the specified range type.

Returns 0 if the range type is invalid.

Return value

Returns the output data value.

CDAQMXAOPWMData::toRealValue

Syntax

```
static double toRealValue(int iAOPWMValue, int iRangeAOPWM);
```

Parameters

iAOPWMValue	Specify the output data value.
iRangeAOPWM	Specify the range type of the AO or PWM range.

Description

Converts the output data values to output values according to the specified range type.

Returns 0 if the range type is invalid.

Return value

Returns the output value.

CDAQMXBalanceData Class

This class stores the initial balance data of the MX100.

It is a wrapper class of the MXBalanceData structure.

It is a group of initial balance data of all the channels.

Each data can be accessed by initial balance data number. The initial balance number is the number of the strain channel.

This class can be used as an interface for setting and retrieving initial balance data.

Public Members

Construct/Destruct

CDAQMXBalanceData Constructs an object.

~CDAQMXBalanceData Destructs an object.

Structure Manipulation

getMXBalanceData Gets the data in a structure.

setMXBalanceData Sets the data in a structure.

initMXBalanceData Initializes the data in a structure.

Member Data Manipulation

initialize Initializes the data member.

getBalanceValid Gets the Boolean value.

getBalanceValue Gets the initial balance value.

setBalance Sets initial balance data.

Operator

operator= Executes substitution.

Utilities

isObject Checks an object.

Protected Members

Data Members

m_MXBalanceData Field for storing the initial balance data.

Member Access

getMXBalance channel. Gets the initial balance data structure for each channel.

Private Members

None.

Member Functions

CDAQMXBalanceData::CDAQMXBalanceData

Syntax

```
CDAQMXBalanceData(MXBalanceData * pMXBalanceData = NULL);  
virtual ~CDAQMXBalanceData(void);
```

Parameters

pMXBalanceData Specify initial balance data.

Description

Constructs or destructs an object.

When constructing, the data member is set to the specified value. If not specified, the data member is initialized.

Reference

setMXBalanceData

CDAQMXBalanceData::getBalanceValid

Syntax

```
int getBalanceValid(int balanceNo);
```

Parameters

balanceNo Specify the initial balance data number.

Description

Gets the Boolean value of the specified data number from the data member. If it does not exist, Invalid is returned.

Return value

Returns a Boolean value.

Reference

getMXBalance

CDAQMXBalanceData::getBalanceValue

Syntax

```
int getBalanceValue(int balanceNo);
```

Parameters

balanceNo Specify the initial balance data number.

Description

Gets the initial balance value of the specified data number from the data member.
Returns 0 if it does not exist.

Return value

Returns the initial balance value.

Reference

getMXBalance

CDAQMXBalanceData::getMXBalance

Syntax

```
MXBalance * getMXBalance(int balanceNo);
```

Parameters

balanceNo Specify the initial balance data number.

Description

Gets the structure of the specified data number from the data member.
Returns NULL if it does not exist.

Return value

Returns a pointer to the structure.

CDAQMXBalanceData::getMXBalanceData

Syntax

```
void getMXBalanceData(MXBalanceData * pMXBalanceData);
```

Parameters

pMXBalanceData Specify the destination where the initial balance data is to
be returned.

Description

Gets the data in a structure.
Stores the contents of the data member in the specified structure.

CDAQMXBalanceData::initialize

Syntax

```
virtual void initialize(void);
```

Description

Initializes the data member. The default value as a general rule is 0.

Reference

initMXBalanceData

CDAQMXBalanceData::isObject

Syntax

```
virtual int isObject(const char * classname =  
"CDAQMXBalanceData");
```

Parameters

classname Specify the class name using a string.

Description

Checks whether the specified class name was inherited.

If parameters are omitted, checks whether it is this class.

Classes that inherit this class must be overridden in order to check their own classes.

Returns true (valid) if the class was inherited. Otherwise, returns false (invalid).

Return value

Returns a Boolean value.

CDAQMXBalanceData::operator=

Syntax

```
CDAQMXBalanceData & operator=(CDAQMXBalanceData &  
cMXBalanceData);
```

Parameters

cMXBalanceData Specify the data to be substituted using an object.

Description

Copies the data member from the specified object.

Return value

Returns the reference to the object.

CDAQMXBalanceData::setBalance

Syntax

```
void setBalance(int balanceNo, int bValid, int iBalanceValue);
```

Parameters

balanceNo	Specify the initial balance data number.
bValid	Specify the Boolean value.
iBalanceValue	Specify the initial balance value.

Description

Stores the specified value in the field indicated by the specified data number of the data member.

If the data number is set to the constant for “Specify All initial balance data numbers,”the same value is stored to all data.

Reference

getMXBalance

CDAQMXBalanceData::setMXBalanceData

Syntax

```
void setMXBalanceData(MXBalanceData * pMXBalanceData);
```

Parameters

pMXBalanceData	Specify initial balance data.
----------------	-------------------------------

Description

Sets the data in a structure.

Stores the contents of the specified structure in the data member.

If not specified, the data member is initialized.

Reference

initMXBalanceData

CDAQMXBalanceResult::initMXBalanceData

Syntax

```
static void initMXBalanceData(MXBalanceData * pMXBalanceData);
```

Parameters

pMXBalanceData	Specify the initial balance data field.
----------------	---

Description

Initializes the specified field.

The default value as a general rule is 0.

CDAQMXBalanceResult Class

- CDAQMXBalanceData
 - CDAQMXBalanceResult

This class stores the initial balance result of the MX100.

It is a wrapper class of the MXBalanceResult structure.

It inherits the CDAQMXBalanceData class, and includes the initial balance data.

It is an group of initial balance results of all the channels.

Each data can be accessed by initial balance data number. The initial balance number is the number of the strain channel.

This class can be used as an interface for specification and results when executing or resetting initial balancing.

Public Members

Construct/Destruct

CDAQMXBalanceResult	Constructs an object.
~CDAQMXBalanceResult	Destructs an object.

Structure Manipulation

getMXBalanceResult	Gets the data in a structure.
setMXBalanceResult	Sets the data in a structure.
initMXBalanceResult	Initializes the data in a structure.

Member Data Manipulation

getResult	Gets the initial balance result.
setResult	Sets the initial balance result.

Operator

operator=	Executes substitution.
-----------	------------------------

- **Overridden Members**

Member Data Manipulation

initialize	Initializes the data member.
------------	------------------------------

Utilities

isObject	Checks an object.
----------	-------------------

Inherited Members

CDAQMXBalanceData Reference
getMXBalanceData setMXBalanceData initMXBalanceData
getBalanceValid getBalanceValue setBalance

Protected Members

Data Members

See also, "Inherited Members."

m_MXBalanceResult Field for storing the initial balance result.

Inherited Members

CDAQMXBalanceData Reference
getMXBalance m_MXBalanceData

Private Members

None.

Member Functions

CDAQMXBalanceResult::CDAQMXBalanceResult

Syntax

```
CDAQMXBalanceResult(MXBalanceData * pMXBalanceData =  
NULL, MXBalanceResult * pMXBalanceResult = NULL);  
virtual ~CDAQMXBalanceResult(void);
```

Parameters

pMXBalanceData Specify initial balance data.
pMXBalanceResult Specify the initial balance result.

Description

Constructs or destructs an object.

When constructing, the data member is set to the specified value. If not specified, the data member is initialized.

Reference

CDAQMXBalanceData::CDAQMXBalanceData
setMXBalanceResult

CDAQMXBalanceResult::getMXBalanceResult

Syntax

```
void getMXBalanceResult(MXBalanceResult * pMXBalanceResult);
```

Parameters

pMXBalanceResult Specify the destination where the initial balance result is to be returned.

Description

Gets the data in a structure.

Stores the contents of the data member in the specified structure.

CDAQMXBalanceResult::getResult

Syntax

```
int getResult(int balanceNo);
```

Parameters

balanceNo Specify the initial balance data number.

Description

Gets the initial balance result of the specified data number from the data member.

If it does not exist, Unspecified is returned.

Return value

Returns the initial balance result.

CDAQMXBalanceResult::initialize

Syntax

```
virtual void initialize(void);
```

Description

Initializes the data member.

The default value as a general rule is 0.

Reference

```
initMXBalanceResult  
CDAQMXBalanceData::initialize
```

CDAQMXBalanceResult::initMXBalanceResult

Syntax

```
static void initMXBalanceResult(MXBalanceResult *  
pMXBalanceResult);
```

Parameters

pMXBalanceResult Specify the field for the initial balance result.

Description

Initializes the specified field.

The default value as a general rule is 0.

CDAQMXBalanceResult::isObject

Syntax

```
virtual int isObject(const char * classname =  
"CDAQMXBalanceResult");
```

Parameters

classname Specify the class name using a string.

Description

Checks whether the specified class name was inherited.

If parameters are omitted, checks whether it is this class.

Classes that inherit this class must be overridden in order to check their own classes.

Returns true (valid value) if the class was inherited. Otherwise, returns false (invalid).

If different from this class, checked with the parent class.

Return value

Returns a Boolean value.

Reference

CDAQMXBalanceData::isObject

CDAQMXBalanceResult::operator=

Syntax

```
CDAQMXBalanceResult & operator=(CDAQMXBalanceResult &  
cMXBalanceResult);
```

Parameters

cMXBalanceResult Specify the data to be substituted using an object.

Description

Copies the data member from the specified object.

Return value

Returns the reference to the object.

CDAQMXBalanceResult::setMXBalanceResult

Syntax

```
void setMXBalanceResult(MXBalanceResult * pMXBalanceResult);
```

Parameters

pMXBalanceResult Specify the initial balance result.

Description

Sets the data in a structure.

Stores the contents in the structure specified in the data member.

If not specified, the data member is initialized.

Reference

initMXBalanceResult

CDAQMXBalanceResult::setResult

Syntax

```
void setResult(int balanceNo, int iResult);
```

Parameters

balanceNo Specify the initial balance data number.

iResult Specify the initial balance result.

Description

Stores the specified value in the field indicated by the specified data number of the data member.

If the data number is set to "Specify All initial balance numbers," the same value is stored in all data.

CDAQMXChConfig Class

- CDAQChInfo
 - CDAQMXChID
 - CDAQMXChConfig

This class stores the channel setup data of the MX100.

It is a wrapper class of the MXChConfig structure.

This class can be used as an interface for storing setup data by channel when retrieving setup data.

Public Members

Construct/Destruct

CDAQMXChConfig	Constructs an object.
~CDAQMXChConfig	Destructs an object.

Structure Manipulation

getMXChConfig	Gets the data in a structure.
setMXChConfig	Sets the data in a structure.
initMXChConfig	Initializes the data in a structure.

Member Data Manipulation

getSpanMin	Gets the span minimum.
getSpanMax	Gets the span maximum.
getScaleMin	Gets the scale minimum.
getScaleMax	Gets the scale maximum.
getRefChNo	Gets the reference channel number.
getFilter	Gets the filter coefficient.
getRJCType	Gets the RJC type.
getRJCVolt	Gets the RJC voltage.
getBurnout	Gets the burnout type.
isDeenergize	Gets the Boolean value of “de-energize”action of relays.
isHold	Gets the Boolean value of “hold”action of relays.
isRefAlarm	Gets the reference alarm.
setRefChNo	Sets the reference channel number.
setFilter	Set the filter coefficient.
setBurnout	Sets the burnout type.
setRJCType	Sets the RJC type.
setAlarm	Sets the alarm value.
setDeenergize	Sets the Boolean value of “de-energize”action of relays.
setHold	Sets the Boolean value of “hold”action of relays.
setRefAlarm	Sets the reference alarm.

Range Settings

setSKIP	Sets SKIP (not used).
setVOLT	Sets DC voltage input.
setTC	Sets thermocouple input.
setRTD	Sets RTD input.
setDI	Sets Digital input (DI).
setDELTA	Sets difference computation between channels.
setSpan	Sets the span.
setScalling	Sets the scale.
changeRange	Changes the range using the temperature unit type.
setRES	Sets the resistance range.
setSTRAIN	Sets the strain range.
setAO	Sets the AO range.
setPWM	Sets the PWM range.

Check

isCorrect	Checks the validity.
-----------	----------------------

Utilities

getItemError	Gets number of the parameter on which error was detected.
getRangePoint	Gets the decimal point position of the range type.
getRangeMin	Gets minimum value of the setting range of the range type.
getRangeMax	Gets maximum value of the setting range of the range type.

Operator

operator=	Executes substitution.
-----------	------------------------

- **Overridden Members**

Member Data Manipulation

initialize	Initializes the data member.
------------	------------------------------

Utilities

isObject	Checks an object.
----------	-------------------

- **Inherited Members**

See CDAQChInfo.

getChNo getPoint setChNo setPoint

CDAQMXChIDReference

getAlarmType getAlarmValueOFF getAlarmValueON getChName
 getChType getComment getKind getMXChID getRange getScale
 getTag getUnit isValid setAlarmValue setComment setChType
 setMXChID setTag setType setUnit setValid toChName toChNo
 toUnitNo

Protected Members

Data Members

<code>m_MXChConfigAIDI</code>	Field for storing AI and DI setup data.
<code>m_MXChConfigAI</code>	Field for storing AI setup data.
<code>m_MXChConfigDO</code>	Field for storing DO setup data.
<code>m_nItemError</code>	Field for storing the setting item number.

Inherited Members

See CDAQChInfo.

`m_chNo` `m_chType` `m_point`

CDAQMXChIDReference

`m_alarm` `m_comment` `m_kind` `m_range` `m_scaleType` `m_tag` `m_unit`
`m_valid`
`getMXAlarm`

Private Members

None.

Member Functions (Alphabetical Order)

CDAQMXChConfig::CDAQMXChConfig

Syntax

```
CDAQMXChConfig(MXChConfig * pMXChConfig = NULL);  
virtual ~CDAQMXChConfig(void);
```

Parameters

`pMXChConfig` Specify the channel setup data.

Description

Constructs or destructs an object.

When constructing, the specified data is stored in the data member. If not specified, the data member is initialized.

Reference

`setMXChConfig`

CDAQMXChConfig::changeRange

Syntax

```
void changeRange(int iTempUnit);
```

Parameters

iTempUnit Specify the temperature unit type.

Description

Changes the range using the temperature unit type.

The settings of the thermocouple input and RTD input are set to default values.

The span, scale, decimal point position, unit name, and reference channel number become the rated values. The alarm setting is initialized (cleared).

Reference

getRange setRTD setTC

CDAQMXChConfig::getBurnout

Syntax

```
int getBurnout(void);
```

Description

Gets the burnout value from the AI setup data field of the data member.

Return value

Returns the burnout value.

CDAQMXChConfig::getFilter

Syntax

```
int getFilter(void);
```

Description

Gets the filter coefficient value from the AI setup data field of the data member.

Return value

Returns the filter time constant.

CDAQMXChConfig::getItemError

Syntax

```
int getItemError(void);
```

Description

Gets the value of the setting item number field of the data member.

Return value

Returns the setting item number.

CDAQMXChConfig::getMXChConfig

Syntax

```
void getMXChConfig(MXChConfig * pMXChConfig);
```

Parameters

pMXChConfig Specify the destination where the channel setup data is to be returned.

Description

Gets the data in a structure.

Stores the contents of the data member in the specified structure.

Reference

getMXChID

CDAQMXChConfig::getRangeMax

Syntax

```
static int getRangeMax(int iRange, int iTempUnit =  
DAQMX_TEMPUNIT_C);
```

Parameters

iRange Specify the range type.
iTempUnit Specify the temperature unit type.

Description

Gets the maximum value of the setting range of the specified range type.

Specify the detailed range of the digital input.

Returns 0 if the designation is unknown.

The value returned is the value without the decimal places.

Return value

Returns the maximum value of the range setting.

CDAQMXChConfig::getRangeMin

Syntax

```
static int getRangeMin(int iRange, int iTempUnit =  
    DAQMX_TEMPUNIT_C);
```

Parameters

iRange Specify the range type.
iTempUnit Specify the temperature unit type.

Description

Gets the minimum value of the setting range of the specified range type.
Specify the detailed range of the digital input.
Returns 0 if the designation is unknown.
The value returned is the value without the decimal places.

Return value

Returns the minimum value of the range setting.

CDAQMXChConfig::getRangePoint

Syntax

```
static int getRangePoint(int iRange, int iTempUnit =  
    DAQMX_TEMPUNIT_C);
```

Parameters

iRange Specify the range type.
iTempUnit Specify the temperature unit type.

Description

Gets the decimal point position of the specified range type.
Specify the detailed range of the digital input.
Returns 0 if the designation is unknown.

Return value

Returns the decimal point position.

CDAQMXChConfig::getRefChNo

Syntax

```
int getRefChNo(void);
```

Description

Gets the reference channel number from the AI and DI setup data field of the data member.

Return value

Returns the reference channel number.

CDAQMXChConfig::getRJCType

Syntax

```
int getRJCType(void);
```

Description

Gets the RJC type value from the AI setup data field of the data member.

Return value

Returns the RJC type.

CDAQMXChConfig::getRJCVolt

Syntax

```
int getRJCVolt(void);
```

Description

Gets the RJC voltage value from the AI setup data field of the data member.

Return value

Returns the RJC voltage.

CDAQMXChConfig::getScaleMax

Syntax

```
int getScaleMax(void);
```

Description

Gets the scale maximum from the AI and DI setup data field of the data member.

Return value

Returns the scale maximum.

CDAQMXChConfig::getScaleMin

Syntax

```
int getScaleMin(void);
```

Description

Gets the scale minimum from the AI and DI setup data field of the data member.

Return value

Returns the scale minimum.

CDAQMXChConfig::getSpanMax

Syntax

```
int getSpanMax(void);
```

Description

Gets the maximum value of span from the AI and DI setup data field of the data member.

Return value

Returns the span maximum.

CDAQMXChConfig::getSpanMin

Syntax

```
int getSpanMin(void);
```

Description

Gets the minimum value of span from the AI and DI setup data field of the data member.

Return value

Returns the span minimum.

CDAQMXChConfig::initialize

Syntax

```
virtual void initialize(void);
```

Description

Initializes the data member. The default value as a general rule is 0.

Reference

```
CDAQMXChID::initialize
```

CDAQMXChConfig::initMXChConfig

Syntax

```
static void initMXChConfig(MXChConfig * pMXChConfig);
```

Parameters

pMXChConfig Specify the channel setup data field.

Description

Initializes the specified field.
The default value as a general rule is 0.

CDAQMXChConfig::isCorrect

Syntax

```
int isCorrect(int iTempUnit = DAQMX_TEMPUNIT_C);
```

Parameters

iTempUnit Specify the temperature unit type.

Description

Checks the validity.

Checks each setting item according to the channel type.

If an invalid value is detected, Invalid is returned.

If an invalid value is detected, the setting item number than indicates the detected location is stored in the setting item number field of the data member.

Return value

Returns a Boolean value.

Reference

getAlarmType getAlarmValueOFF getAlarmValueON getBurnout
getErrorChoice getFilter getIdleChoice getKind getPoint
getPresetValue getPulseTime getRange getRJCType getRJCVolt
getScale getScaleMax getScaleMin getSpanMax getSpanMin isValid

CDAQMXChConfig::isDeenergize

Syntax

```
int isDeenergize(void);
```

Description

Gets the de-energized action value from the DO setup data field of the data member.

The value of 0 is invalid; other values are valid.

Return value

Returns a Boolean value.

CDAQMXChConfig::isHold

Syntax

```
int isHold(void);
```

Description

Gets the hold action value from the DO setup data field of the data member.

The value of 0 is invalid; other values are valid.

Return value

Returns a Boolean value.

CDAQMXChConfig::isObject

Syntax

```
virtual int isObject(const char * classname =  
"CDAQMXChConfig");
```

Parameters

classname Specify the class name using a string.

Description

Checks whether the specified class name was inherited.

If parameters are omitted, checks whether it is this class.

Classes that inherit this class must be overridden in order to check their own classes.

Returns true (valid) if the class was inherited. Otherwise, returns false (invalid).

If different from this class, checks the parent class.

Return value

Returns a Boolean value.

Reference

CDAQMXChID::isObject

CDAQMXChConfig::isRefAlarm

Syntax

```
unsigned char isRefAlarm(int refChNo, int levelNo);
```

Parameters

refChNo Specify the reference channel number.

levelNo Specify the alarm level.

Description

Gets the specified reference alarm value from the DO setup data field of the data member.

The value of 0 is invalid; other values are valid.

Returns Invalid if the designation is outside the range.

Return value

Returns a Boolean value.

CDAQMXChConfig::operator=

Syntax

```
CDAQMXChConfig & operator=(CDAQMXChConfig & cMXChConfig);
```

Parameters

cMXChConfig Specify an object for substitution.

Description

Copies the data member of the specified object.

Return value

Returns the reference to the object.

Reference

```
getMXChConfig setMXChConfig
```

CDAQMXChConfig::setAlarm

Syntax

```
void setAlarm(int levelNo, int iAlarmType, int value, int  
histerisys = 0);
```

Parameters

levelNo	Specify the alarm level.
iAlarmType	Specify the alarm type.
value	Specify the alarm value.
histerisys	Specify the hysteresis.

Description

Stores the specified value in the alarm field of the data member.

Generates the threshold level for alarm activation (ON value) and the threshold level for alarm termination (Off value) from the alarm value and hysteresis.

If the alarm type is not allowed per the previously set channel type, it is ignored.

Reference

```
getkind setAlarmValue
```

CDAQMXChConfig::setAO

Syntax

```
void setAO(int iRangeAO);
```

Parameters

iRangeAO Specify the AO range type.

Description

Sets the specified range.

Makes the channel status valid.

The settings are set to the default values of the specified range type.

Holds the reference channel number according to the previously set channel type.

For command AO channels, the reference channel number is the “undefined reference channel number.”

Any channels other than the AO range are ignored. If AO or command AO channels are unspecified, they are ignored.

The span, scale, decimal point position, and unit name are set to the default values.

The alarm setting is initialized (cleared).

Reference

```
getKind getRefChNo setAlarm setPoint setRefChNo setScaling
setSpan setType setUnit setValid
```

CDAQMXChConfig::setBurnout

Syntax

```
void setBurnout(int iBurnout);
```

Parameters

iBurnout Specify the burnout value.

Description

Stores the specified value in the AI setup data field of the data member.

CDAQMXChConfig::setDeenergize

Syntax

```
void setDeenergize(int bDeenergize);
```

Parameters

bDeenergize Specify the de-energize action using a Boolean value.

Description

Stores the specified value in the DO setup data field of the data member.

CDAQMXChConfig::setDELTA

Syntax

```
void setDELTA(int refChNo, int iRange, int iTempUnit =
DAQMX_TEMPUNIT_C);
```

Parameters

refChNo	Specify the reference channel number.
iRange	Specify the range type.
iTempUnit	Specify the temperature unit type.

Description

Sets the specified range.

Makes the channel status valid. The settings are set to the default values of the specified range type.

The channel type is set to AI (difference between channels) or DI (difference between channels).

Specify the range type independently without using reference ranges. For the digital input (DI), it must be a digital input (DI) detailed range.

Any specification other than the input range is ignored.

The span, scale, decimal point position, unit name, and reference channel are set to the default values.

The alarm setting is initialized (cleared).

Reference

```
setAlarm setPoint setRefChNo setScaling setSpan setType
setUnit setValid
```

CDAQMXChConfig::setDI

Syntax

```
void setDI(int iRangeDI);
```

Parameters

iRangeDI	Specify the range type of the digital input (DI) detailed range.
----------	--

Description

Sets the specified range.

Makes the channel status valid. The settings are set to the default values of the specified range type.

The channel type is set to DI.

Any specification other than the digital input detailed range is ignored.

The span, scale, decimal point position, unit name, and reference channel are set to the default values.

The alarm setting is initialized (cleared).

Reference

```
setAlarm setPoint setRefChNo setScaling setSpan setType
setUnit setValid
```

CDAQMXChConfig::setFilter

Syntax

```
void setFilter(int iFilter);
```

Parameters

iFilter Specify the filter coefficient.

Description

Stores the specified value in the AI setup data field of the data member.

CDAQMXChConfig::setHold

Syntax

```
void setHold(int bHold);
```

Parameters

bHold Specify the hold action using a Boolean value.

Description

Stores the specified value in the DO setup data field of the data member.

CDAQMXChConfig::setMXChConfig

Syntax

```
void setMXChConfig(MXChConfig * pMXChConfig);
```

Parameters

pMXChConfig Specify the channel setup data.

Description

Sets the data in a structure. Stores the contents of the specified structure in the data member.

If not specified, the data member is initialized.

Reference

initialize setMXChID

CDAQMXChConfig::setPWM

Syntax

```
void setPWM(int iRangePWM);
```

Parameters

iRangePWM Specify the PWM range type.

Description

Sets the specified range.

Makes the channel status valid.

The settings are set to the default values of the specified range type.

Holds the reference channel number according to the previously set channel type.

For command PWM channels, the reference channel number is the “undefined reference channel number.”

Any specification other than the PWM range is ignored. If PWM or command PWM channels are unspecified, they are ignored.

The span, scale, decimal point position, and unit name are set to the default values.

The alarm setting is initialized (cleared).

Reference

```
getKind getRefChNo setAlarm setPoint setRefChNo setScaling  
setSpan setType setUnit setValid
```

CDAQMXChConfig::setRefAlarm

Syntax

```
void setRefAlarm(int refChNo, int levelNo, int bValid);
```

Parameters

refChNo Specify the reference channel number.

levelNo Specify the alarm level.

bValid Specify the Boolean value.

Description

Stores the specified value in the DO setup data field of the data member.

If the constant for “Specify all reference channel numbers” is specified for the channel numbers to be referenced, the value is stored for all the channels.

If the “Specify all alarm level numbers” constant is specified for the alarm level, the value is stored for all alarm levels.

CDAQMXChConfig::setRefChNo

Syntax

```
void setRefChNo(int refChNo);
```

Parameters

refChNo Specify the reference channel number.

Description

Stores the specified value in the AI and DI setup data fields of the data member.

Channel numbers other than its own range are ignored.

If no reference channels exist for the specification, the constant for “Undefined reference channel numbers” is specified.

CDAQMXChConfig::setRES

Syntax

```
void setRES(int iRangeRES);
```

Parameters

iRangeRES Specify the range type of the resistance range.

Description

Sets the specified range.

Makes the channel status valid.

The settings are set to the default values of the specified range type.

The channel type is set to AI.

Any specification other than the resistance range is ignored.

The span, scale, decimal point position, unit name, and reference channel are set to the default values.

The alarm setting is initialized (cleared).

Reference

```
setAlarm setPoint setRefChNo setScaling setSpan setType  
setUnit setValid
```

CDAQMXChConfig::setRJCType

Syntax

```
void setRJCType(int iRJCType, int volt = 0);
```

Parameters

iRJCType Specify the RJC type.

volt Specify the RJC voltage.

Description

Stores the specified value in the AI setup data field of the data member.

CDAQMXChConfig::setRTD

Syntax

```
void setRTD(int iRangeRTD, int iTempUnit = DAQMX_TEMPUNIT_C);
```

Parameters

iRangeRTD	Specify the range type of the RTD input.
iTempUnit	Specify the temperature unit type.

Description

Sets the specified range.

Makes the channel status valid. The settings are set to the default values of the specified range type.

The channel type is set to the value of AI.

Any specification other than the RTD range is ignored.

The span, scale, decimal point position, unit name, and reference channel are set to the default values. The alarm setting is initialized (cleared).

Reference

```
setAlarm setPoint setRefChNo setScalling setSpan setType  
setUnit setValid
```

CDAQMXChConfig::setScalling

Syntax

```
void setScalling(int scaleMin, int scaleMax, int scalePoint,  
int iTempUnit = DAQMX_TEMPUNIT_C);
```

Parameters

scaleMin	Specify the scale minimum.
scaleMax	Specify the scale maximum.
scalePoint	Specify the decimal point position.
iTempUnit	Specify the temperature unit type.

Description

Stores the specified value in the AI and DI setup data fields of the data member.

Sets the scale type to Linear.

Checks the value according to the range type and channel type that are already set.

If the maximum and minimum values are the same, the scale type is set to None.

The decimal point position is set to the default value.

AI (remote RJC), DO, AO, PWM, and other channel types on which the scale cannot be set are ignored.

Reference

```
getKind setPoint getRange getRangePoint setType
```

CDAQMXChConfig::setSKIP

Syntax

```
void setSKIP(void);
```

Description

Sets SKIP (not used).
Makes the channel status invalid.

Reference

setValid

CDAQMXChConfig::setSpan

Syntax

```
void setSpan(int spanMin, int spanMax, int iTempUnit =  
DAQMX_TEMPUNIT_C);
```

Parameters

spanMin	Specify the span minimum.
spanMax	Specify the span maximum.
iTempUnit	Specify the temperature unit type.

Description

Stores the specified value in the AI and DI setup data fields of the data member. Checks the value according to the range type and channel type that are already set. If the maximum and minimum values are equal, the value is not stored. If the value is outside the range, it is rounded to a valid value. If the maximum and minimum values are reversed for the AO/PWM channels, they are ignored.

Reference

getKind getRange

CDAQMXChConfig::setSTRAIN

Syntax

```
void setSTRAIN(int iRangeSTR);
```

Parameters

iRangeSTR	Specify the strain range type.
-----------	--------------------------------

Description

Sets the specified range.
Makes the channel status valid.
The settings are set to the default values of the specified range type.
The channel type is set to AI.
Any specification other than the strain range is ignored.
The span, scale, decimal point position, unit name, and reference channel are set to the default values.
The alarm setting is initialized (cleared).

Reference

setAlarm setPoint setRefChNo setScaling setSpan setType
setUnit setValid

CDAQMXChConfig::setTC

Syntax

```
void setTC(int iRangeTC, int iTempUnit = DAQMX_TEMPUNIT_C);
```

Parameters

iRangeTC Specify the range type of the thermocouple input.
iTempUnit Specify the temperature unit type.

Description

Sets the specified range.

Makes the channel status valid. The settings are set to the default values of the specified range type.

The channel type is set to the value of AI.

Any specification other than the TC range is ignored.

The span, scale, decimal point position, unit name, and reference channel are set to the default values.

The alarm setting is initialized (cleared).

Reference

```
setAlarm setPoint setRefChNo setScaling setSpan setType  
setUnit setValid
```

CDAQMXChConfig::setVOLT

Syntax

```
void setVOLT(int iRangeVOLT);
```

Parameters

iRangeVOLT Specify the range type of the DC voltage input.

Description

Sets the specified range.

Makes the channel status valid. The settings are set to the default values of the specified range type.

The channel type is set to the value of AI.

Any specification other than the DC voltage range is ignored.

The span, scale, decimal point position, unit name, and reference channel are set to the default values.

The alarm setting is initialized (cleared).

Reference

```
setAlarm setPoint setRefChNo setScaling setSpan setType  
setUnit setValid
```

CDAQMXChConfigData class

This class stores the channel setup data of all the channels of the MX100.

It is a wrapper class of the MXChConfigData structure.

This class can be used as an interface for storing setup data for all channels when retrieving setup data.

This class is a group of all channels of the CDAQMXChConfig class.

It implements any processing required for the between-channel association information.

Public Members

Construct/Destruct

CDAQMXChConfigData	Constructs an object.
~CDAQMXChConfigData	Destructs an object.

Structure Manipulation

getMXChConfigData	Gets the data in a structure.
setMXChConfigData	Sets the data in a structure.
initMXChConfigData	Initializes the data in a structure.
setMXChConfig	Sets the data in a structure for each channel.

Member Data Manipulation

initialize	Initializes the data member.
getClassMXChConfig	Gets the data for each channel.

Range Settings

setRRJC	Sets remote RJC.
changeRange	Changes the range using the temperature unit type.

Check

isCorrect	Checks the validity.
-----------	----------------------

Utilities

getItemError	Gets the number of the parameter on which an error was detected.
isObject	Checks an object.

Operator

operator=	Executes substitution.
-----------	------------------------

Protected Members

Data Members

m_cMXChConfig	Field for storing the channel setup data of all the channels.
m_pcMXChConfig	Head pointer of the field for storing the channel setup data of all the channels.
m_nItemError	Field for storing the setting item number.

Private Members

None.

Member Functions (Alphabetical Order)

CDAQMXChConfigData::CDAQMXChConfigData

Syntax

```
CDAQMXChConfigData(MXChConfigData * pMXChConfigData = NULL);  
virtual ~CDAQMXChConfigData(void);
```

Parameters

pMXChConfigData Specify the channel setup data for all channels.

Description

Constructs or destructs an object.

When constructing, the data member is set to the specified value. If not specified, the data member is initialized.

Reference

setMXChConfigData

CDAQMXChConfigData::changeRange

Syntax

```
void changeRange(int iTempUnit);
```

Parameters

iTempUnit Specify the temperature unit type.

Description

Changes the range using the temperature unit type.

The settings of the thermocouple input and RTD input are set to default values.

All channels are changed collectively.

Reference

CDAQMXChConfig::changeRange

CDAQMXChConfigData::getClassMXChConfig

Syntax

```
CDAQMXChConfig * getClassMXChConfig(int chNo);
```

Parameters

chNo Specify the channel number.

Description

Gets from the data member the channel setup data field corresponding to the specified channel number as an object.

Returns NULL if it does not exist.

Return value

Returns the reference to the object.

CDAQMXChConfigData::getItemError

Syntax

```
int getItemError(void);
```

Description

Gets the value of the setting item number field of the data member.

Return value

Returns the setting item number.

CDAQMXChConfigData::getMXChConfigData

Syntax

```
void getMXChConfigData(MXChConfigData * pMXChConfigData);
```

Parameters

pMXChConfigData Specify the destination where the channel setup data is to be returned.

Description

Gets the data in a structure. Stores the contents of the data member in the specified structure.

Reference

CDAQMXChConfig::getMXChConfig

CDAQMXChConfigData::initialize

Syntax

```
virtual void initialize(void);
```

Description

Initializes the data member. The default value as a general rule is 0.

Reference

CDAQMXChConfig::initialize

CDAQMXChConfigData::initMXChConfigData

Syntax

```
static void initMXChConfigData(MXChConfigData *  
pMXChConfigData);
```

Parameters

pMXChConfig Specify the field for the channel setup data for all channels.

Description

Initializes the specified field.

The default value as a general rule is 0.

CDAQMXChConfigData::isCorrect

Syntax

```
int isCorrect(int iTempUnit = DAQMX_TEMPUNIT_C);
```

Parameters

iTempUnit Specify the temperature unit type.

Description

Checks the validity.

All channels are checked collectively.

Checks the associations between channels.

If an invalid value is detected, Invalid is returned.

If an invalid value is detected, the setting item number that indicates the detected location is stored in the setting item number field of the data member.

Return value

Returns a Boolean value.

Reference

```
CDAQMXChConfig::getItemError CDAQMXChConfig::getKind  
CDAQMXChConfig::getRange CDAQMXChConfig::getRefChNo  
CDAQMXChConfig::isCorrect CDAQMXChConfig::isValid
```

CDAQMXChConfigData::isObject

Syntax

```
virtual int isObject(const char * classname =
"CDAQMXChConfigData");
```

Parameters

classname Specify the class name using a string.

Description

Checks whether the specified class name was inherited.

If parameters are omitted, checks whether it is this class.

Classes that inherit this class must be overridden in order to check their own classes.

Returns true (valid) if the class was inherited. Otherwise, returns false (invalid).

Return value

Returns a Boolean value.

CDAQMXChConfigData::operator=

Syntax

```
CDAQMXChConfigData & operator=(CDAQMXChConfigData &
cMXChConfigData);
```

Parameters

cMXChConfigData Specify an object for substitution.

Description

Copies the data member of the specified object.

Return value

Returns the reference to the object.

CDAQMXChConfigData::setMXChConfig

Syntax

```
void setMXChConfig(MXChConfig * pMXChConfig);
```

Parameters

pMXChConfig Specify the channel setup data.

Description

Sets the data in a structure.

Stores the contents of the specified structure to the data member field corresponding to the channel number within the specified structure. If the corresponding data member field does not exist, the function does nothing.

Reference

```
getClassMXChConfig
CDAQMXChConfig::setMXChConfig
```

CDAQMXChConfigData::setMXChConfigData

Syntax

```
void setMXChConfigData(MXChConfigData * pMXChConfigData);
```

Parameters

pMXChConfigData Specify the channel setup data for all channels.

Description

Sets the data in a structure. Stores the contents of the specified structure in the data member.

If not specified, the data member is initialized.

Reference

```
initialize  
CDAQMXChConfig::setMXChConfig
```

CDAQMXChConfigData::setRRJC

Syntax

```
void setRRJC(int chNo, int refChNo);
```

Parameters

chNo Specify the channel number.
refChNo Specify the reference channel number.

Description

Sets remote RJC.

The measurement range is set to the same range as the specified reference channel.

Copies the contents of the reference channel and overwrites the channel number, channel type, and reference channel number.

The channel type is set to AI (remote RJC).

The scale type is set to NONE.

If the channel is not for the thermocouple input, the function does nothing.

The alarm setting is initialized.

Reference

```
getClassMXChConfig  
CDAQMXChConfig::getKind CDAQMXChConfig::getRange  
CDAQMXChConfig::isValid CDAQMXChConfig::setAlarm  
CDAQMXChConfig::setChNo CDAQMXChConfig::setRefChNo  
CDAQMXChConfig::setType
```

CDAQMXChID Class

- CDAQChInfo
 - CDAQMXChID

This class stores the channel ID information of the MX100.

It is a wrapper class of the MXChID structure.

This is the common section of the channel information data and channel setup data.

Public Members

Construct/Destruct

CDAQMXChID	Constructs an object.
~CDAQMXChID	Destructs an object.
initMXChID	Initializes the data in a structure.

Structure Manipulation

getMXChID	Gets the data in a structure.
setMXChID	Sets the data in a structure.
initMXChID	Initializes the data in a structure.

Member Data Manipulation

isValid	Gets the channel status.
getKind	Gets the channel type.
getRange	Gets the range type.
getScale	Gets the scale type.
getUnit	Gets the unit name.
getTag	Gets the tag.
getComment	Gets the comment.
getAlarmType	Gets the alarm type.
getAlarmValueON	Gets the ON value.
getAlarmValueOFF	Gets the OFF value.
setValid	Sets the channel status.
setType	Sets the channel type, range type, and scale type.
setUnit	Sets the unit name.
setTag	Sets the tag.
setComment	Sets the comment.
setAlarmValue	Sets the alarm value.

Utilities

getChName	Gets the channel name.
toChName	Creates the channel name.
toChNo	Extracts the channel number from the channel name.
toUnitNo	Extracts the unit number from the channel name.

Operator

operator= Executes substitution.

Overridden Members

Member Data Manipulation

initialize Initializes the data member.

getChType Gets the channel type.

setChType Set the channel type.

Utilities

isObject Checks an object.

Inherited Members

See CDAQChInfo.

getChNo getPoint setChNo setPoint

Protected Members

Data Members

m_valid Field for storing the channel status.

m_kind Field for storing the channel type.

m_range Field for storing the range type.

m_scaleType Field for storing the scale type.

m_unit Field for storing the unit name.

m_tag Field for storing the tag.

m_comment Field for storing the comment.

m_alarm Field for storing the alarm.

Member Access

getMXAlarm Gets the alarm information structure for each alarm level.

Inherited Members

See CDAQChInfo.

m_chNo m_chType m_point

Private Members

None.

Member Functions

CDAQMXChID::CDAQMXChID

Syntax

```
CDAQMXChID(MXChID * pMXChID = NULL);
virtual ~CDAQMXChID(void);
```

Parameters

pMXChID Specify the channel ID information.

Description

Constructs or destructs an object. When constructing, the specified data is stored in the data member. If not specified, the data member is initialized.

Reference

setMXChID

CDAQMXChID::getAlarmType

Syntax

```
int getAlarmType(int levelNo);
```

Parameters

levelNo Specify the alarm level.

Description

Gets the alarm type of the specified alarm level from the alarm field of the data member.

Returns DAQMX_ALARM_NONE if it does not exist.

Return value

Returns the alarm type.

Reference

getMXAlarm

CDAQMXChID::getAlarmValueOFF

Syntax

```
int getAlarmValueOFF(int levelNo);
```

Parameters

levelNo Specify the alarm level.

Description

Gets the threshold level for alarm termination (OFF value) of the specified alarm level from the alarm field of the data member. Returns 0 if the alarm level is outside the range.

Return value

Returns the threshold level (OFF value) for alarm termination.

Reference

getMXAlarm

CDAQMXChID::getAlarmValueON

Syntax

```
int getAlarmValueON(int levelNo);
```

Parameters

levelNo Specify the alarm level.

Description

Gets the threshold level for alarm generation (ON value) of the specified alarm level from the alarm field of the data member.

Returns 0 if the alarm level is outside the range.

Return value

Returns the threshold level (On value) for alarm activation.

Reference

getMXAlarm

CDAQMXChID::getChName

Syntax

```
int getChName(int unitno = 0);
```

Parameters

unitno Specify the unit number.

Description

Creates the channel name from the channel number and specified unit number of the data member.

Return value

Returns the channel name.

Reference

getChNo toChName

CDAQMXChID::getChType

Syntax

```
virtual int getChType(void);
```

Description

Gets the channel type field from the data member.

Always returns 0, since the channel type is 0.

Return value

Returns the channel type.

CDAQMXChID::getComment

Syntax

```
const char * getComment(void);
```

Description

Gets the comment in the comment field from the data member.

Return value

Returns a pointer to the string.

CDAQMXChID::getKind

Syntax

```
int getKind(void);
```

Description

Gets the value of the channel type field from the data member.

Return value

Returns the channel type.

CDAQMXChID::getMXAlarm

Syntax

```
MXAlarm * getMXAlarm(int levelNo);
```

Parameters

levelNo Specify the alarm level.

Description

Gets the structure of the specified alarm level from the alarm field of the data member.

Returns NULL if it does not exist.

Return value

Returns a pointer to the structure.

CDAQMXChID::getMXChID

Syntax

```
void getMXChID(MXChID * pMXChID);
```

Parameters

pMXChID Specify the destination where the channel ID information is to be returned.

Description

Gets the data in a structure. Stores the contents of the data member in the specified structure.

Reference

```
getChNo getComment getKind getPoint getRange getScale getTag  
getUnit isValid
```

CDAQMXChID::getRange

Syntax

```
int getRange(void);
```

Description

Gets the range type field from the data member.

Return value

Returns the range type.

CDAQMXChID::getScale

Syntax

```
int getScale(void);
```

Description

Gets the scale type field from the data member.

Return value

Returns the scale type.

CDAQMXChID::getTag

Syntax

```
const char * getTag(void);
```

Description

Gets the tag in the tag field from the data member.

Return value

Returns a pointer to the string.

CDAQMXChID::getUnit

Syntax

```
const char * getUnit(void);
```

Description

Gets unit name of the unit name field from the data member.

Return value

Returns a pointer to the string.

CDAQMXChID::initialize

Syntax

```
virtual void initialize(void);
```

Description

Initializes the data member. The default value as a general rule is 0.

Reference

CDAQChInfo::initialize

CDAQMXChID::initMXChID

Syntax

```
static void initMXChID(MXChID * pMXChID);
```

Parameters

pMXChID Specify the channel ID information field.

Description

Initializes the specified field.

The default value as a general rule is 0.

CDAQMXChID::isObject

Syntax

```
virtual int isObject(const char * classname = "CDAQMXChID");
```

Parameters

classname Specify the class name using a string.

Description

Checks whether the specified class name was inherited.

If parameters are omitted, checks whether it is this class.

Classes that inherit this class must be overridden in order to check their own classes.

Returns true (valid) if the class was inherited. Otherwise, returns false (invalid).

If different from this class, checks the parent class.

Return value

Returns a Boolean value.

Reference

```
CDAQChInfo::isObject
```

CDAQMXChID::isValid

Syntax

```
int isValid(void);
```

Description

Gets the channel status field from the data member.

The value of 0 is invalid; other values are valid.

Return value

Returns a Boolean value.

CDAQMXChID::operator=

Syntax

```
CDAQMXChID & operator=(CDAQMXChID & cMXChID);
```

Parameters

cMXChID Specify an object for substitution.

Description

Copies the data member of the specified object.

Return value

Returns the reference to the object.

Reference

getMXChID setMXChID

CDAQMXChID::setAlarmValue

Syntax

```
void setAlarmValue(int levelNo, int iAlarmType =  
DAQMX_ALARM_NONE, int valueON = 0, int valueOFF = 0);
```

Parameters

levelNo Specify the alarm level.
iAlarmType Specify the alarm type.
valueON Specify the threshold level (ON value) for alarm activation.
valueOFF Specify the threshold level (OFF value) for alarm termination.

Description

Stores the specified value in the alarm field of the data member.

If the "Specify all alarm levels" constant is specified for the alarm level, the same value is stored for all alarm levels.

CDAQMXChID::setChType

Syntax

```
virtual void setChType(int chType);
```

Parameters

chType Specify the channel type.

Description

Stores the channel type field of the data member to the specified value.

Since the channel type is 0, this function does nothing.

CDAQMXChID::setComment

Syntax

```
void setComment(const char * strComment);
```

Parameters

strComment Specify the comment.

Description

Stores the specified value in the comment field of the data member.

CDAQMXChID::setMXChID

Syntax

```
void setMXChID(MXChID * pMXChID);
```

Parameters

pMXChID Specify the channel ID information.

Description

Sets the data in a structure. Stores the contents of the specified structure in the data member.

If not specified, the data member is initialized.

Reference

```
initialize setChNo setComment setPoint setTag setType setUnit  
setValid
```

CDAQMXChID::setTag

Syntax

```
void setTag(const char * strTag);
```

Parameters

strTag Specify the tag.

Description

Stores the specified value in the tag field of the data member.

CDAQMXChID::setType

Syntax

```
void setType(int iKind, int iRange, int iScale =  
DAQMX_SCALE_NONE);
```

Parameters

iKind	Specify the channel type.
iRange	Specify the range type.
iScale	Specify the scale type.

Description

Stores the specified values in the channel type field, range type field, and scale type field of the data member.

CDAQMXChID::setUnit

Syntax

```
void setUnit(const char * strUnit);
```

Parameters

strUnit	Specify the unit name.
---------	------------------------

Description

Stores the specified value in the unit name field of the data member.

CDAQMXChID::setValid

Syntax

```
void setValid(int bValid);
```

Parameters

bValid	Specify the Boolean value.
--------	----------------------------

Description

Stores the specified value in the channel status field of the data member.

CDAQMXChID::toChName

Syntax

```
static int toChName(int chno, int unitno = 0);
```

Parameters

chno	Specify the channel number.
unitno	Specify the unit number.

Description

Creates the channel name from the specified channel number and unit number.

Return value

Returns the channel name.

CDAQMXChID::toChNo

Syntax

```
static int toChNo(int chname);
```

Parameters

chname Specify the channel name.

Description

Separates the channel number from the specified channel name.

Return value

Returns the channel number.

CDAQMXChID::toUnitNo

Syntax

```
static int toUnitNo(int chname);
```

Parameters

chname Specify the channel name.

Description

Separates the unit number from the specified channel name.

Return value

Returns the unit number.

CDAQMXChInfo Class

- CDAQChInfo
 - CDAQMXChID
 - CDAQMXChInfo

This class stores the channel information data of the MX100.

It is a wrapper class of the MXChInfo structure.

Reference minimum and maximum values are not used.

This class can be used as an interface for storing channel information data when retrieving channel information data.

Measured data is easier to handle when associated with the measured data class.

Public Members

Construct/Destruct

CDAQMXChInfo	Constructs an object.
~CDAQMXChInfo	Destructs an object.

Structure Manipulation

getMXChInfo	Gets the data in a structure.
setMXChInfo	Sets the data in a structure.
initMXChInfo	Initializes the data in a structure.

Member Data Manipulation

getFIFONo	Gets the FIFO number.
getFIFOIndex	Gets the channel sequence number in the FIFO.
getOriginalMin	Gets the reference minimum value.*
getOriginalMax	Gets the reference maximum value.*
* The reference minimum/maximum values are currently not used.	
getDisplayMin	Gets the display minimum value.
getDisplayMax	Gets the display maximum value.
getRealMin	Gets the measurable minimum value.
getRealMax	Gets the measurable maximum value.
setFIFONo	Sets the FIFO number.
setFIFOIndex	Sets the channel sequence number in the FIFO.

Operator

operator=	Executes substitution.
-----------	------------------------

Overridden Members

Member Data Manipulation

initialize	Initializes the data member.
------------	------------------------------

Utilities

isObject	Checks an object.
----------	-------------------

Inherited Members

See CDAQChInfo.

getChNo getPoint setChNo setPoint

CDAQMXChIDReference

getAlarmType getAlarmValueOFF getAlarmValueON getChName
getChType getComment getKind getMXChID getRange getScale
getTag getUnit initMXChID isValid setAlarmValue setChType
setComment setMXChID setUnit setType setTag setValid toChName
toChNo toUnitNo

Protected Members

Data Members

m_FIFONo	Field for storing the FIFO number.
m_FIFOIndex	Field for storing the channel sequence number in the FIFO.
m_origMin	Field for storing the reference minimum value.*
m_origMax	Field for storing the reference maximum value.*
*The reference minimum/maximum values are currently not used.	
m_dispMin	Field for storing the display minimum value.
m_dispMax	Field for storing the display maximum value.
m_realMin	Field for storing the measurable minimum value of the range.
m_realMax	Field for storing the measurable maximum value of the range.

Inherited Members

See CDAQChInfo.

m_chNo m_chType m_point

CDAQMXChIDReference

m_alarm m_comment m_kind m_range m_scaleType m_tag m_unit
m_valid
getMXAlarm

Private Members

None.

Member Functions (Alphabetical Order)

CDAQMXChInfo::CDAQMXChInfo

Syntax

```
CDAQMXChInfo(MXChInfo * pMXChInfo = NULL);  
virtual ~CDAQMXChInfo(void);
```

Parameters

pMXChInfo Specify the channel information data.

Description

Constructs or destructs an object.

When constructing, the specified data is stored in the data member. If not specified, the data member is initialized.

Reference

setMXChInfo

CDAQMXChInfo::getDisplayMax

Syntax

```
double getDisplayMax(void);
```

Description

Gets value of the display maximum field from the data member.

Return value

Returns the display maximum value.

CDAQMXChInfo::getDisplayMin

Syntax

```
double getDisplayMin(void);
```

Description

Gets value of the display minimum field from the data member.

Return value

Returns the display minimum value.

CDAQMXChInfo::getFIFOIndex

Syntax

```
int getFIFOIndex(void);
```

Description

Gets the value in the channel sequence number field in the FIFO from the data member.

Return value

Returns the channel sequence number in the FIFO.

CDAQMXChInfo::getFIFONo

Syntax

```
int getFIFONo(void);
```

Description

Gets the value in the FIFO number field from the data member.

Return value

Returns the FIFO number.

CDAQMXChInfo::getMXChInfo

Syntax

```
void getMXChInfo(MXChInfo * pMXChInfo);
```

Parameters

pMXChInfo	Specify the destination where the channel information data is to be returned.
-----------	---

Description

Gets the data in a structure. Stores the contents of the data member in the specified structure.

Reference

getDisplayMax getDisplayMin getFIFOIndex getFIFONo getMXChID
getOriginalMax getOriginalMin getRealMax getRealMin

CDAQMXChInfo::getOriginalMax

Syntax

```
double getOriginalMax(void);
```

Description

Gets value of the reference maximum field from the data member.

***Reference** minimum value is not used currently.

Return value

Returns the reference maximum value.

CDAQMXChInfo::getOriginalMin

Syntax

```
double getOriginalMin(void);
```

Description

Gets value of the reference minimum field from the data member.

***Reference** minimum value is not used currently.

Return value

Returns the reference minimum value.

CDAQMXChInfo::getRealMax

Syntax

```
double getRealMax(void);
```

Description

Gets the value in the measurable maximum value field from the data member.

Return value

Returns the measurable maximum value.

CDAQMXChInfo::getRealMin

Syntax

```
double getRealMin(void);
```

Description

Gets the value in the measurable minimum value field from the data member.

Return value

Returns the measurable minimum value.

CDAQMXChInfo::initMXChInfo

Syntax

```
static void initMXChInfo(MXChInfo * pMXChInfo);
```

Parameters

pMXChInfo Specify the channel information data field.

Description

Initializes the specified field.

The default value as a general rule is 0.

CDAQMXChInfo::initialize

Syntax

```
virtual void initialize(void);
```

Description

Initializes the data member. The default value as a general rule is 0.

Reference

```
CDAQMXChID::initialize
```

CDAQMXChInfo::isObject

Syntax

```
virtual int isObject(const char * classname = "CDAQMXChInfo");
```

Parameters

classname Specify the class name using a string.

Description

Checks whether the specified class name was inherited.

If parameters are omitted, checks whether it is this class.

Classes that inherit this class must be overridden in order to check their own classes.

Returns true (valid) if the class was inherited. Otherwise, returns false (invalid).

If different from this class, checks the parent class.

Return value

Returns a Boolean value.

Reference

CDAQMXChID::isObject

CDAQMXChInfo::operator=

Syntax

```
CDAQMXChInfo & operator=(CDAQMXChInfo & cMXChInfo);
```

Parameters

cMXChInfo Specify an object for substitution.

Description

Copies the data member of the specified object.

Return value

Returns the reference to the object.

Reference

getMXChInfo setMXChInfo

CDAQMXChInfo::setFIFOIndex

Syntax

```
void setFIFOIndex(int fifoIndex);
```

Parameters

fifoIndex Specify the channel sequence number in the FIFO.

Description

Stores the specified value in the channel sequence number field in the FIFO from the data member.

CDAQMXChInfo::setFIFONo

Syntax

```
void setFIFONo(int fifoNo);
```

Parameters

fifoNo Specify the FIFO number.

Description

Stores the specified value in the FIFO number field of the data member.

CDAQMXChInfo::setMXChInfo

Syntax

```
void setMXChInfo(MXChInfo * pMXChInfo);
```

Parameters

pMXChInfo Specify the channel information data.

Description

Sets the data in a structure. Stores the contents of the specified structure in the data member.

If not specified, the data member is initialized.

Reference

```
initialize setFIFOIndex setFIFONo setMXChID
```

CDAQMXConfig Class

This class stores the setup data of the MX100.

It is a wrapper class of the MXConfigData structure.

This class can be used as an interface for storing setup data when retrieving setup data.

Public Members

Construct/Destruct

CDAQMXConfig	Constructs an object.
~CDAQMXConfig	Destructs an object.

Structure Manipulation

getMXConfigData	Gets the data in a structure.
setMXConfigData	Sets the data in a structure.
initMXConfigData	Initializes the data in a structure.

Member Data Manipulation

initialize	Initializes the data member.
getClassMXSysInfo	Gets the system configuration data.
getClassMXStatus	Gets the status.
getClassMXNetInfo	Gets the network information data.
getClassMXChConfigData	Gets the channel setup data.
getClassMXBalanceData	Gets initial balance data.
getClassMXOutputData	Gets the basic output channel data.
getClassMXChConfig	Gets individual channel setup data.
reconstruct	Reconstructs the system.
setTempUnit	Sets the temperature unit type.
setDOType	Sets the DO channel type.
setInterval	Set the interval type.
setAOType	Sets the AO channel type.
setPWMTtype	Sets the PWM channel type.

Range Settings

setSKIP	Sets SKIP (not used).
setVOLT	Sets DC voltage input.
setTC	Sets thermocouple input.
setRTD	Sets RTD input.
setDI	Sets digital input (DI).
setDELTA	Sets difference computation between channels.
setRRJC	Sets remote RJC.
setScalling	Sets the scale.

setRES	Sets the resistance range.
setSTRAIN	Sets the strain range.
setAO	Sets the AO range.
setPWM	Sets the PWM range.

Check

isCorrect	Checks the validity.
-----------	----------------------

Operator

operator=	Executes substitution.
-----------	------------------------

Utilities

getItemError	Gets the number of the parameter on which an error was detected.
isObject	Checks an object.
getSpanPoint	Gets the decimal point position of the channel.
getRangePoint	Gets the decimal point position of the range type.
getChName	Gets the channel name.
setChKind	Sets the channel type.

Protected Members

Data Members

m_cMXSysInfo	Field for storing the system configuration data.
m_cMXStatus	Field for storing the status data.
m_cMXNetInfo	Field for storing the network information data.
m_cMXChConfigData	Field for storing the channel setup data.
m_cMXBalanceData	Field for storing the initial balance data.
m_cMXOutputData	Field for storing the output channel data.
m_nItemError	Field for storing the setting item number.

Private Members

None.

Member Functions (Alphabetical Order)

CDAQMXConfig::CDAQMXConfig

Syntax

```
CDAQMXConfig(MXConfigData * pMXConfigData);
virtual ~CDAQMXConfig(void);
```

Parameters

pMXConfigData Specify the setup data.

Description

Constructs or destructs an object.

When constructing, the specified data is stored in the data member. If not specified, the data member is initialized.

Reference

setMXConfigData

CDAQMXConfig::getChName

Syntax

```
int getChName(int chNo);
```

Parameters

chNo Specify the channel number.

Description

Creates the channel name from the specified channel number and system structure data field of the data member.

Return value

Returns the channel name.

Reference

```
getClassMXChConfig
getClassMXSysInfo
CDAQMXChConfig::getChName
CDAQMXSysInfo::getUnitNo
```

CDAQMXConfig::getClassMXBalanceData

Syntax

```
CDAQMXBalanceData & getClassMXBalanceData(void);
```

Description

Gets initial balance data from the data member.

Return value

Returns the reference to the object.

CDAQMXConfig::getClassMXChConfig

Syntax

```
CDAQMXChConfig * getClassMXChConfig(int chNo);
```

Parameters

chNo Specify the channel number.

Description

Gets the channel setup data of the specified channel number.
Returns NULL if it does not exist.

Return value

Returns a pointer to the object.

Reference

```
getClassMXChConfigData  
CDAQMXChConfigData::getClassMXChConfig
```

CDAQMXConfig::getClassMXChConfigData

Syntax

```
CDAQMXChConfigData & getClassMXChConfigData(void);
```

Description

Gets the channel setup data field from the data member.

Return value

Returns a reference to the object.

CDAQMXConfig::getClassMXNetInfo

Syntax

```
CDAQMXNetInfo & getClassMXNetInfo(void);
```

Description

Gets the network information data field as a data member.

Return value

Returns a reference to the object.

CDAQMXConfig::getClassMXOutputData

Syntax

```
CDAQMXOutputData & getClassMXOutputData(void);
```

Description

Gets the output channel data field as a data member.

Return value

Returns the reference to the object.

CDAQMXConfig::getClassMXStatus

Syntax

```
CDAQMXStatus & getClassMXStatus(void);
```

Description

Gets the status field as a data member.

Return value

Returns a reference to the data member.

CDAQMXConfig::getClassMXSysInfo

Syntax

```
CDAQMXSysInfo & getClassMXSysInfo(void);
```

Description

Gets the System configuration data field as a data member.

Return value

Returns a reference to the object.

CDAQMXConfig::getItemError

Syntax

```
int getItemError(void);
```

Description

Gets the value of the setting item number field from the data member.

Return value

Returns the setting item number.

CDAQMXConfig::getMXConfigData

Syntax

```
void getMXConfigData(MXConfigData * pMXConfigData);
```

Parameters

pMXConfigData	Specify the destination where the setup data is to be returned.
---------------	---

Description

Gets the data in a structure.

Stores the contents of the data member in the specified structure.

Reference

```
DAQMXBalanceData::getMXBalanceData  
CDAQMXOutputData::getMXOutputData  
CDAQMXChConfigData::getMXChConfigData  
CDAQMXNetInfo::getMXNetInfo  
CDAQMXStatus::getMXStatus  
CDAQMXSysInfo::getMXSystemInfo
```

CDAQMXConfig::getRangePoint

Syntax

```
int getRangePoint(int iRange);
```

Parameters

iRange Specify the range type.

Description

Gets the decimal point position of the specified range type.

For the contact range, specify the contact detailed range.

Returns 0 if it does not exist.

Return value

Returns the decimal point position.

Reference

```
getClassMXSysInfo  
CDAQMXChConfig::getRangePoint  
CDAQMXSysInfo::getTempUnit
```

CDAQMXConfig::getSpanPoint

Syntax

```
int getSpanPoint(int chNo);
```

Parameters

chNo Specify the channel number.

Description

Gets the decimal point position of the range type of the specified channel number.

Returns 0 if it does not exist.

Return value

Returns the decimal point position.

CDAQMXConfig::initialize

Syntax

```
virtual void initialize(void);
```

Description

Initializes the data member. The default value as a general rule is 0.

Reference

```
CDAQMXBalanceData::initialize  
CDAQMXChConfigData::initialize  
CDAQMXNetInfo::initialize  
CDAQMXOutputData::initialize  
CDAQMXStatus::initialize  
CDAQMXSysInfo::initialize
```

CDAQMXConfig::initMXConfigData

Syntax

```
static void initMXConfigData(MXConfigData * pMXConfigData);
```

Parameters

pMXConfigData Specify the setup data field.

Description

Initializes the specified field.

The default value as a general rule is 0.

Reference

```
CDAQMXBalanceData::initMXBalanceData
CDAQMXChConfigData::initMXChConfigData
CDAQMXNetInfo::initMXNetInfo
CDAQMXOutputData::initMXOutputData
CDAQMXStatus::initMXStatus
CDAQMXSysInfo::initMXSystemInfo
```

CDAQMXConfig::isCorrect

Syntax

```
int isCorrect(void);
```

Description

Checks the validity.

Checks each setting item according to the system configuration data.

If an invalid value is detected, Invalid is returned.

If an invalid value is detected, the setting item number than indicates the detected location is stored in the setting item number field of the data member.

Return value

Returns a Boolean value.

Reference

```
getClassMXBalanceData getClassMXChConfig
getClassMXChConfigData getClassMXOutputData getClassMXSysInfo
CDAQMXBalanceData::getBalanceValid
CDAQMXBalanceData::getBalanceValue
CDAQMXChConfig::getRange
CDAQMXChConfig::getKind
CDAQMXChConfigData::getItemError
CDAQMXChConfigData::isCorrect
CDAQMXOutputData::getOutputType
CDAQMXOutputData::getPulseTime
CDAQMXSysInfo::getItemError
CDAQMXSysInfo::getModuleType
CDAQMXSysInfo::getTempUnit
CDAQMXSysInfo::isCorrect
```

CDAQMXConfig::isObject

Syntax

```
virtual int isObject(const char * classname = "CDAQMXConfig");
```

Parameters

classname Specify the class name using a string.

Description

Checks whether the specified class name was inherited.

If parameters are omitted, checks whether it is this class.

Classes that inherit this class must be overridden in order to check their own classes.

Returns true (valid) if the class was inherited. Otherwise, returns false (invalid).

Return value

Returns a Boolean value.

CDAQMXConfig::operator=

Syntax

```
CDAQMXConfig & operator=(CDAQMXConfig & cMXConfig);
```

Parameters

cMXConfig Specify an object for substitution.

Description

Copies the data member of the specified object.

Return value

Returns the reference to the object.

CDAQMXConfig::reconstruct

Syntax

```
void reconstruct(int bRealType);
```

Parameters

bRealType	Specify whether or not to set the actual module types to reconstruct the system using a Boolean.
-----------	--

Description

Reconstructs the system.

When Valid is specified, creates setup data according to the actual module types.

When Invalid is specified, creates setup data according to the current module types.

Settings are set to default values.

Reference

```
setAO setAOType setDI setDOType setPWM setPWMType setSTRAIN  
setVOLT  
CDAQMXChConfigData::initialize  
CDAQMXSysInfo::getChNum  
CDAQMXSysInfo::getModuleType  
CDAQMXSysInfo::setCFTimeout  
CDAQMXSysInfo::setCFWriteMode  
CDAQMXSysInfo::setRealModule  
CDAQMXSysInfo::setTempUnit  
CDAQMXSysInfo::setUnitNo
```

CDAQMXConfig::setAO

Syntax

```
void setAO(int chNo,int iRangeAO,int spanMin = 0,int spanMax = 0);
```

Parameters

chNo	Specify the channel number.
iRangeAO	Specify the AO range type.
spanMin	Specify the span minimum.
spanMax	Specify the span maximum.

Description

Sets the channel of the specified channel number to the specified range.

If the maximum and minimum values are the same, the span is considered omitted.

If the channel does not exist or the channel is not on the corresponding module, the setting is void.

If the constant for "Specify all channel numbers" is specified for the channel numbers, all channels are processed.

The output type of the output channel data is reset to match the specified range.

Reference

```
getClassMXChConfig getClassMXOutputData getClassMXSysInfo  
CDAQMXChConfig::setAO  
CDAQMXChConfig::setSpan  
CDAQMXOutputData::setOutputType  
CDAQMXSysInfo::getModuleType  
CDAQMXSysInfo::getTempUnit
```

CDAQMXConfig::setAOType

Syntax

```
void setAOType(int aoNo,int iKind,int refChNo =
DAQMX_REFCHNO_NONE);
```

Parameters

aoNo	Specify the AO data number.
iKind	Specify the AO type using channel type.
refChNo	Specify the reference channel number.

Description

Sets the channel type of the AO module.

If unused is specified for the channel type, the channel is set to SKIP (not used).

If AO (transmission output) is specified for the channel type, specify the channel number of the input channel for the reference channel.

If the channel does not exist, the channel is not on the AO module, or the specified type is not an AO type, the channel type is not set.

If the constant for "Specify all AO/PWM numbers" is specified for the AO data numbers, all AO channels are processed.

Each setting type for the channels is reset to match the output type of the output channel data.

Reference

```
getClassMXChConfig getClassMXOutputData getClassMXSysInfo
setSKIP
CDAQMXChConfig::setAO
CDAQMXChConfig::setRefChNo
CDAQMXChConfig::setType
CDAQMXChConfig::setValid
CDAQMXOutputData::getOutputType
CDAQMXSysInfo::getModuleType
```

CDAQMXConfig::setChKind

Syntax

```
void setChKind(int chNo,int iKind,int refChNo =  
DAQMX_REFCHNO_NONE);
```

Parameters

chNo	Specify the channel number.
iKind	Specify the channel type.
refChNo	Specify the reference channel number.

Description

Sets the channel type on the channel of the specified channel number. If the channel type is AI (difference between channels), DI (difference between channels), AI (remote RJC), AO (transmission output), or PWM (transmission output), the reference channel specification is valid. The settings for each channel are set to the default values.

Reference

```
setAOType setDELTA setDI setDOType setPWMTyp setRRJC setSKIP  
setVOLT
```

CDAQMXConfig::setDELTA

Syntax

```
void setDELTA(int chNo, int refChNo, int spanMin = 0, int  
spanMax = 0, int iRange = DAQMX_RANGE_REFERENCE);
```

Parameters

chNo	Specify the channel number.
refChNo	Specify the reference channel number.
spanMin	Specify the span minimum.
spanMax	Specify the span maximum.
iRange	Specify the range type.

Description

Sets the channel of the specified channel number to the specified range. If the maximum and minimum values are the same, the span is considered omitted. If the channel does not exist or the channel is not on the corresponding module, the setting is void. If the range type is set to "Reference Channel" the range of the channel specified by the reference channel number is applied. If the constant for "Specify all channel numbers" is specified for the channel numbers, all channels are processed. If the range type is strain, the initial balancing data is reset.

Reference

```
getClassMXBalanceData getClassMXChConfig getClassMXSysInfo  
CDAQMXBalanceData::setBalance CDAQMXChConfig::getRange  
CDAQMXChConfig::isValid CDAQMXChConfig::setDELTA  
CDAQMXChConfig::setSpan CDAQMXSysInfo::getModuleType  
CDAQMXSysInfo::getTempUnit
```

CDAQMXConfig::setDI

Syntax

```
void setDI(int chNo, int iRangeDI, int spanMin = 0, int spanMax = 0);
```

Parameters

chNo	Specify the channel number.
iRangeDI	Specify the range type of the digital input (DI).
spanMin	Specify the span minimum.
spanMax	Specify the span maximum.

Description

Sets the channel of the specified channel number to the specified range.

If the maximum and minimum values are the same, the span is considered omitted.

If the channel does not exist or the channel is not on the corresponding module, the setting is void. Sets the range type using the digital input (DI) detailed range.

If the constant for “Specify all channel numbers” is specified for the channel numbers, all channels are processed.

Reference

```
getClassMXChConfig getClassMXSysInfo
CDAQMXChConfig::setDI CDAQMXChConfig::setSpan
CDAQMXSysInfo::getModuleType CDAQMXSysInfo::getTempUnit
```

CDAQMXConfig::setDOType

Syntax

```
void setDOType(int doNo, int iKind, int bDeenergize = DAQMX_VALID_OFF, int bHold = DAQMX_VALID_OFF);
```

Parameters

doNo	Specify the data number.
iKind	Specify the DO type using channel type.
bDeenergize	Specify the de-energize action using a Boolean value.
bHold	Specify hold action using a Boolean value.

Description

Sets the channel type of the DO module.

If unused is specified for the channel type, the channel is set to SKIP (not used).

If the channel does not exist, the channel is not on the DO module, or the specified type is not a DO type, the channel type is not set. If the constant for “Specify all DO numbers” is specified for the DO data numbers, all channels are processed.

Reference

```
getClassMXChConfig setSKIP
CDAQMXChConfig::setDeenergize CDAQMXChConfig::setHold
CDAQMXChConfig::setType CDAQMXChConfig::setValid
CDAQMXSysInfo::getModuleType
```

CDAQMXConfig::setInterval

Syntax

```
void setInterval(int moduleNo, int iInterval, int iHz =
DAQMX_INTEGRAL_AUTO);
```

Parameters

moduleNo	Specify the module number.
iInterval	Specify the interval type.
iHz	Specify the type of A/D integration time.

Description

Sets the module of the specified module number to the specified value.

If the module does not exist, the value is not set.

If the constant for “Specify all module numbers” is specified for the module numbers, all modules are processed.

Reference

```
CDAQMXSysInfo::getChNum CDAQMXSysInfo::getModuleType
CDAQMXSysInfo::setModule
```

CDAQMXConfig::setMXConfigData

Syntax

```
void setMXConfigData(MXConfigData * pMXConfigData);
```

Parameters

pMXConfigData Specify the setup data.

Description

Sets the data in a structure. Stores the contents of the specified structure in the data member.

If not specified, the data member is initialized.

Reference

```
initialize
CDAQMXBalanceData::setMXBalanceData
CDAQMXChConfigData::setMXChConfigData
CDAQMXNetInfo::setMXNetInfo
CDAQMXOutputData::setMXOutputData
CDAQMXStatus::setMXStatus
CDAQMXSysInfo::setMXSystemInfo
```

CDAQMXConfig::setPWM

Syntax

```
void setPWM(int chNo,int iRangePWM,int spanMin = 0,int spanMax = 0);
```

Parameters

chNo	Specify the channel number.
iRangePWM	Specify the PWM range type.
spanMin	Specify the span minimum.
spanMax	Specify the span maximum.

Description

Sets the channel of the specified channel number to the specified range.

If the maximum and minimum values are the same, the span is considered omitted.

If the channel does not exist or the channel is not on the corresponding module, the setting is void.

If the constant for “Specify all channel numbers” is specified for the channel numbers, all channels are processed.

The output type of the output channel data is reset to match the specified range.

Reference

```
getClassMXChConfig  
getClassMXOutputData  
getClassMXSysInfo  
CDAQMXChConfig::setSpan  
CDAQMXChConfig::setPWM  
CDAQMXOutputData::setOutputType  
CDAQMXSysInfo::getModuleType  
CDAQMXSysInfo::getTempUnit
```

CDAQMXConfig::setPWMType

Syntax

```
void setPWMType(int pwmNo,int iKind,int refChNo =  
DAQMX_REFCHNO_NONE);
```

Parameters

pwmNo	Specify the PWM data number.
iKind	Specify the PWM type with the channel type.
refChNo	Specify the reference channel number.

Description

Sets the channel type of the PWM module.

If unused is specified for the channel type, the channel is set to SKIP (not used).

If PWM (transmission output) is specified for the channel type, specify the channel number of the input channel for the reference channel.

If the channel does not exist, the channel is not on the PWM module, or the specified type is not a PWM type, the channel type is not set.

If the constant for "Specify all AO/PWM numbers" is specified for the PWM data numbers, all PWM channels are processed.

Each setting type for the channels is reset to match the output type of the output channel data.

Reference

```
getClassMXChConfig getClassMXOutputData getClassMXSysInfo  
setSKIP  
CDAQMXChConfig::setPWM  
CDAQMXChConfig::setRefChNo  
CDAQMXChConfig::setType  
CDAQMXChConfig::setValid  
CDAQMXOutputData::getOutputType  
CDAQMXSysInfo::getModuleType
```

CDAQMXConfig::setRES

Syntax

```
void setRES(int chNo,int iRangeRES,int spanMin = 0,int spanMax
= 0);
```

Parameters

chNo	Specify the channel number.
iRangeRES	Specify the range type of the resistance range.
spanMin	Specify the span minimum.
spanMax	Specify the span maximum.

Description

Sets the channel of the specified channel number to the specified range. If the maximum and minimum values are the same, the span is considered omitted. If the channel does not exist or the channel is not on the corresponding module, the setting is void. If the constant for “Specify all channel numbers” is specified for the channel numbers, all channels are processed.

Reference

```
getClassMXChConfig getClassMXSysInfo
CDAQMXChConfig::setRES CDAQMXChConfig::setSpan
CDAQMXSysInfo::getModuleType CDAQMXSysInfo::getTempUnit
```

CDAQMXConfig::setRRJC

Syntax

```
void setRRJC(int chNo, int refChNo, int spanMin = 0, int
spanMax = 0);
```

Parameters

chNo	Specify the channel number.
refChNo	Specify the reference channel number.
spanMin	Specify the span minimum.
spanMax	Specify the span maximum.

Description

Sets the channel of the specified channel number to the specified range. If the maximum and minimum values are the same, the span is considered omitted. If the channel does not exist, if the range is not TC, or if the channel is not on the corresponding module, the setting is void. If the constant for “Specify all channel numbers” is specified for the channel numbers, all channels are processed.

Reference

```
getClassMXChConfig getClassMXChConfigData getClassMXSysInfo
CDAQMXChConfig::getRange CDAQMXChConfig::isValid
CDAQMXChConfig::setSpan CDAQMXChConfigData::setRRJC
CDAQMXSysInfo::getModuleType CDAQMXSysInfo::getTempUnit
```

CDAQMXConfig::setRTD

Syntax

```
void setRTD(int chNo, int iRangeRTD, int spanMin = 0, int spanMax = 0);
```

Parameters

chNo	Specify the channel number.
iRangeRTD	Specify the range type of the RTD input.
spanMin	Specify the span minimum.
spanMax	Specify the span maximum.

Description

Sets the channel of the specified channel number to the specified range. If the maximum and minimum values are the same, the span is considered omitted. If the channel does not exist or the channel is not on the corresponding module, the setting is void. If the constant for “Specify all channel numbers” is specified for the channel numbers, all channels are processed.

Reference

```
getClassMXChConfig getClassMXSysInfo CDAQMXChConfig::setRTD  
CDAQMXChConfig::setSpan CDAQMXSysInfo::getModuleType  
CDAQMXSysInfo::getTempUnit
```

CDAQMXConfig::setScale

Syntax

```
void setScaling(int chNo, int scaleMin = 0, int scaleMax = 0,  
int scalePoint = 0);
```

Parameters

chNo	Specify the channel number.
scaleMin	Specify the scale minimum.
scaleMax	Specify the scale maximum.
scalePoint	Specify the decimal point position.

Description

Sets the scale on the channel of the specified channel number. Sets the scale type to Linear. If the maximum and minimum values are the same, it is set to None. If the constant for “Specify all channel numbers” is specified for the channel numbers, all channels are processed.

Reference

```
getClassMXChConfig getClassMXSysInfo  
CDAQMXChConfig::setScale CDAQMXSysInfo::getTempUnit
```

CDAQMXConfig::setSKIP

Syntax

```
void setSKIP(int chNo);
```

Parameters

chNo Specify the channel number.

Description

Sets the channel of the specified channel number to SKIP (not used).

If the module does not exist, the value is not set.

If the constant for “Specify all channel numbers” is specified for the channel numbers, all channels are processed.

Reference

```
getClassMXChConfig CDAQMXChConfig::setSKIP
```

CDAQMXConfig::setSTRAIN

Syntax

```
void setSTRAIN(int chNo,int iRangeSTRAIN,int spanMin = 0,int spanMax = 0);
```

Parameters

chNo Specify the channel number.

iRangeSTRAIN Specify the strain range type.

spanMin Specify the span minimum.

spanMax Specify the span maximum.

Description

Sets the channel of the specified channel number to the specified range.

If the maximum and minimum values are the same, the span is considered omitted.

If the channel does not exist or the channel is not on the corresponding module, the setting is void.

If the constant for “Specify all channel numbers” is specified for the channel numbers, all channels are processed.

Resets the initial balance data.

Reference

```
getClassMXBalanceData
getClassMXChConfig
getClassMXSysInfo
CDAQMXBalanceData::setBalance
CDAQMXChConfig::setSpan
CDAQMXChConfig::setSTRAIN
CDAQMXSysInfo::getModuleType
CDAQMXSysInfo::getTempUnit
```

CDAQMXConfig::setTC

Syntax

```
void setTC(int chNo, int iRangeTC, int spanMin = 0, int spanMax = 0);
```

Parameters

chNo	Specify the channel number.
iRangeTC	Specify the range type of the thermocouple input.
spanMin	Specify the span minimum.
spanMax	Specify the span maximum.

Description

Sets the channel of the specified channel number to the specified range. If the maximum and minimum values are the same, the span is considered omitted. If the channel does not exist or the channel is not on the corresponding module, the setting is void. If the constant for "Specify all channel numbers" is specified for the channel numbers, all channels are processed.

Reference

```
getClassMXChConfig getClassMXSysInfo CDAQMXChConfig::setSpan  
CDAQMXChConfig::setTC CDAQMXSysInfo::getModuleType  
CDAQMXSysInfo::getTempUnit
```

CDAQMXConfig::setTempUnit

Syntax

```
void setTempUnit(int iTempUnit);
```

Parameters

iTempUnit	Specify the temperature unit type.
-----------	------------------------------------

Description

Changes the temperature unit type of the setup data. Stores the specified value in the System configuration data field. Changes the settings of the channels affected.

Reference

```
CDAQMXChConfigData::changeRange  
CDAQMXSysInfo::setTempUnit
```

CDAQMXConfig::setVOLT

Syntax

```
void setVOLT(int chNo,int iRangeVOLT,int spanMin = 0,int spanMax = 0);
```

Parameters

chNo	Specify the channel number.
iRangeVOLT	Specify the range type of the DC voltage input.
spanMin	Specify the span minimum.
spanMax	Specify the span maximum.

Description

Sets the channel of the specified channel number to the specified range.

If the maximum and minimum values are the same, the span is considered omitted.

If the channel does not exist or the channel is not on the corresponding module, the setting is void.

If the constant for "Specify all channel numbers" is specified for the channel numbers, all channels are processed.

Reference

```
getClassMXChConfig getClassMXSysInfo  
CDAQMXChConfig::setSpan  
CDAQMXChConfig::setVOLT  
CDAQMXSysInfo::getModuleType  
CDAQMXSysInfo::getTempUnit
```

CDAQMXDataInfo Class

- CDAQDataInfo
 - CDAQMXDataInfo

This class stores the measured data of the MX100.

It is a wrapper class of the MXDataInfo structure.

This class can be used as an interface for storing measured data when retrieving measured data.

Actual measured data can be calculated by associating with the channel information data class.

Public Members

Construct/Destruct

CDAQMXDataInfo	Constructs an object.
~CDAQMXDataInfo	Destructs an object.

Structure Manipulation

getMXDataInfo	Gets the data as a structure.
setMXDataInfo	Sets the data in a structure.
initMXDataInfo	Initializes the structure data.

Member Data Manipulation

getStatus	Gets the data status.
isAlarm	Gets the alarm value.
setStatus	Set the data status.
setAlarm	Sets the alarm value.

Association

getClassMXChInfo	Gets the association with the channel information data.
setClassMXChInfo	Sets the association with the channel information data.

Utilities

getAlarmName	Gets the name of the alarm type.
--------------	----------------------------------

Operator

operator=	Executes substitution.
-----------	------------------------

Overridden Members

Member Data Manipulation

initialize	Initializes the data member.
------------	------------------------------

Utilities

isObject	Checks an object.
----------	-------------------

Inherited Members

CDAQDataInfoReference
 getClassChInfo getDoubleValue getStringValue getValue
 setClassChInfo setValue toDoubleValue toStringValue

Protected Members

Data Members

m_dataStatus Field for storing the data status.
 m_alarm Field for storing the presence or absence of the alarm.

Inherited Members

CDAQDataInfoReference
 m_pChInfo m_value

Private Members

None.

Member Functions (Alphabetical Order)

CDAQMXDataInfo::CDAQMXDataInfo

Syntax

```
CDAQMXDataInfo(MXDataInfo * pMXDataInfo = NULL, CDAQMXChInfo *
pcMXChInfo = NULL);
virtual ~CDAQMXDataInfo(void);
```

Parameters

pMXDataInfo Specify the measured data.
 pcMXChInfo Specify the association with the channel information data.

Description

Constructs or destructs an object.

When constructing, the specified data is stored in the data member. If not specified, the data member is initialized.

Reference

setClassMXChInfo setMXDataInfo

CDAQMXDataInfo::getAlarmName

Syntax

```
static const char * getAlarmName(int iAlarmType);
```

Parameters

iAlarmType Specify the alarm type.

Description

Gets the string corresponding to the specified alarm type.
If outside the range, the string is set the same as no alarm.

Return value

Returns a pointer to the string.

CDAQMXDataInfo::getClassMXChInfo

Syntax

```
CDAQMXChInfo * getClassMXChInfo(void);
```

Description

Gets the association with the channel information data of the data member.
Returns NULL if the value is not specified.

Return value

Returns the association with the channel information data.

Reference

getClassChInfo

CDAQMXDataInfo::getMXDataInfo

Syntax

```
void getMXDataInfo(MXDataInfo * pMXDataInfo);
```

Parameters

pMXDataInfo Specify the destination where the measured data is to be returned.

Description

Gets the data in a structure. Stores the contents of the data member in the specified structure.

Reference

getStatus getValue

CDAQMXDataInfo::getStatus

Syntax

```
int getStatus(void);
```

Description

Gets the value in the data status field of the data member.

Return value

Returns the data status.

CDAQMXDataInfo::initialize

Syntax

```
void initialize(void);
```

Description

Initializes the data member.

The default value as a general rule is 0.

The association with the channel information data is not initialized.

Reference

CDAQDataInfo::initialize

CDAQMXDataInfo::initMXDataInfo

Syntax

```
static void initMXDataInfo(MXDataInfo * pMXDataInfo);
```

Parameters

pMXDataInfo Specify the measured data field.

Description

Initializes the specified field.

The default value as a general rule is 0.

CDAQMXDataInfo::isAlarm

Syntax

```
int isAlarm(int levelNo);
```

Parameters

levelNo Specify the alarm level.

Description

Gets the value of the alarm presence/absence field of the data member.

Returns the value corresponding to the specified alarm level. Returns Invalid Value(OFF) if the alarm level is outside the range.

Return value

Returns a Boolean value.

CDAQMXDataInfo::isObject

Syntax

```
virtual int isObject(const char * classname = "CDAQMXDataInfo");
```

Parameters

classname Specify the class name using a string.

Description

Checks whether the specified class name was inherited.

If parameters are omitted, checks whether it is this class.

Classes that inherit this class must be overridden in order to check their own classes.

Returns true (valid) if the class was inherited. Otherwise, returns false (invalid).

If different from this class, checks the parent class.

Return value

Returns a Boolean value.

Reference

CDAQDataInfo::isObject

CDAQMXDataInfo::operator=

Syntax

```
CDAQMXDataInfo & operator=(CDAQMXDataInfo & cMXDataInfo);
```

Parameters

cMXDataInfo Specify an object for substitution.

Description

Copies the data member of the specified object.

Also copies the association with the channel information data.

Return value

Returns the reference to the object.

Reference

getClassMXChInfo getMXDataInfo setClassMXChInfo setMXDataInfo

CDAQMXDataInfo::setAlarm

Syntax

```
void setAlarm(int levelNo, int bValid);
```

Parameters

levelNo	Specify the alarm level.
bValid	Specify the Boolean value.

Description

Sets the alarm presence/absence field of the data member to the specified value. If the alarm level is outside the range, it is not set.

CDAQMXDataInfo::setClassMXChInfo

Syntax

```
void setClassMXChInfo(CDAQMXChInfo * pcMXChInfo);
```

Parameters

pcMXChInfo	Specify a pointer to the channel information data class.
------------	--

Description

Stores the specified value to the association with the channel information data of the data member.

Reference

setClassChInfo

CDAQMXDataInfo::setMXDataInfo

Syntax

```
void setMXDataInfo(MXDataInfo * pMXDataInfo);
```

Parameters

pMXDataInfo	Specify the association with the channel information data.
-------------	--

Description

Sets the data in a structure.
Stores the contents of the specified structure in the data member.
If not specified, the data member is initialized.

Reference

initialize setStatus setValue

CDAQMXDataInfo::setStatus

Syntax

```
void setStatus(int iDataStatus);
```

Parameters

iDataStatus	Specify the data status.
-------------	--------------------------

Description

Stores the data status field of the data member to the specified value.

CDAQMXDateTime Class

- CDAQDateTime
 - CDAQMXDateTime

This class stores the time information data of the MX100.

It is a wrapper class of the MXDateTime structure.

This class can be used as an interface for storing time information data when retrieving time information data for the retrieval of measured data.

Public Members

Construct/Destruct

CDAQMXDateTime	Constructs an object.
~CDAQMXDateTime	Destructs an object.

Structure Manipulation

getMXDateTime	Gets the data in a structure.
setMXDateTime	Sets the data in a structure.
initMXDateTime	Initializes the data in a structure.

Operator

operator=	Executes substitution.
-----------	------------------------

Overridden Members

Utilities

isObject	Checks an object.
----------	-------------------

Inherited Members

CDAQDateTimeReference
 getMilliSecond getTime initialize setMilliSecond setNow setTime toLocalDateTime

Protected Members

Inherited Members

CDAQDateTimeReference
 m_milliSecond m_time

Private Members

None.

Member Functions (Alphabetical Order)

CDAQMXDateTime::CDAQMXDateTime

Syntax

```
CDAQMXDateTime(time_t time = 0, int milliSecond = 0);  
CDAQMXDateTime(MXDateTime * pMXDateTime);  
virtual ~CDAQMXDateTime(void);
```

Parameters

time	Specify seconds.
milliSecond	Specify milliseconds.
pMXDateTime	Specify the time information data.

Description

Constructs or destructs an object.

When constructing, the data member is set to the specified value. If not specified, the data member is initialized.

Reference

setMXDateTime

CDAQMXDateTime::getMXDateTime

Syntax

```
void getMXDateTime(MXDateTime * pMXDateTime);
```

Parameters

pMXDateTime	Specify the destination where the time information data is to be returned.
-------------	--

Description

Gets the data in a structure.

Stores the contents of the data member in the specified structure.

CDAQMXDateTime::initMXDateTime

Syntax

```
static void initMXDateTime(MXDateTime * pMXDateTime);
```

Parameters

pMXDateTime Specify the time information data field.

Description

Initializes the specified field.

The default value as a general rule is 0.

CDAQMXDateTime::isObject

Syntax

```
virtual int isObject(const char * classname =  
"CDAQMXDateTime");
```

Parameters

classname Specify the class name using a string.

Description

Checks whether the specified class name was inherited.

If parameters are omitted, checks whether it is this class.

Classes that inherit this class must be overridden in order to check their own classes.

Returns true (valid) if the class was inherited. Otherwise, returns false (invalid).

If different from this class, checks the parent class.

Return value

Returns a Boolean value.

Reference

CDAQDateTime::isObject

CDAQMXDateTime::operator=

Syntax

```
CDAQMXDateTime & operator=(CDAQMXDateTime & cMXDateTime);
```

Parameters

cMXDateTime Specify the data to be substituted using an object.

Description

Copies the data member from the specified object.

Return value

Returns the reference to the object.

CDAQMXDateTime::setMXDateTime

Syntax

```
void setMXDateTime(MXDateTime * pMXDateTime);
```

Parameters

pMXDateTime Specify the time information data.

Description

Sets the data in a structure.

Stores the contents of the specified structure in the data member.

If not specified, the data member is initialized.

Reference

initialize

CDAQMXDOData Class

This class stores the DO data of the MX100.

It is a wrapper class of the MXDOData structure.

It is a group of DO data of all the channels.

This class can be used as an interface for storing DO data when retrieving or setting DO data.

Public Members

Construct/Destruct

CDAQMXDOData Constructs an object.

~CDAQMXDOData Destructs an object.

Structure Manipulation

getMXDOData Gets the data in a structure.

setMXDOData Sets the data in a structure.

initMXDOData Initializes the data in a structure.

Member Data Manipulation

initialize Initializes the data member.

getDOValid Gets Boolean.

getDOONOFF Gets ON/OFF.

setDO Sets the DO data.

setDOONOFF Sets ON/OFF.

Operator

operator= Executes substitution.

Utilities

isObject Checks an object.

Protected Members

Data Members

m_MXDOData Field for storing the DO data.

Member Access

getMXDO Gets the DO data structure for each channel.

Private Members

None.

Member Functions (Alphabetical Order)

CDAQMXDOData::CDAQMXDOData

Syntax

```
CDAQMXDOData(MXDOData * pMXDOData = NULL);  
virtual ~CDAQMXDOData(void);
```

Parameters

pMXDOData Specify the DO data.

Description

Constructs or destructs an object.

When constructing, the data member is set to the specified value. If not specified, the data member is initialized.

Reference

setMXDOData

CDAQMXDOData::getDOONOFF

Syntax

```
int getDOONOFF(int doNo);
```

Parameters

doNo Specify the data number.

Description

Gets the ON/OFF value indicated by the specified DO data number from the DO data field of the data member.

If it does not exist, returns Invalid.

Return value

Returns a Boolean value.

Reference

getMXDO

CDAQMXDOData::getDOValid

Syntax

```
int getDOValid(int doNo);
```

Parameters

doNo Specify the data number.

Description

Gets the Boolean value indicated by the specified DO data number from the DO data field of the data member.

If it does not exist, returns Invalid.

Return value

Returns a Boolean value.

Reference

getMXDO

CDAQMXDOData::getMXDO

Syntax

```
MXDO * getMXDO(int doNo);
```

Parameters

doNo Specify the data number.

Description

Gets the structure indicated by the specified DO data number from the DO data field of the data member.

Returns NULL if it does not exist.

Return value

Returns a pointer to the structure.

CDAQMXDOData::getMXDOData

Syntax

```
void getMXDOData(MXDOData * pMXDOData);
```

Parameters

pMXDOData Specify the destination where the DO data is to be returned.

Description

Gets the data in a structure.

Stores the contents of the data member in the specified structure.

CDAQMXDOData::initialize

Syntax

```
virtual void initialize(void);
```

Description

Initializes the data member.

The default value as a general rule is 0.

Reference

initMXDOData

CDAQMXDOData::initMXDOData

Syntax

```
static void initMXDOData(MXDOData * pMXDOData);
```

Parameters

pMXDOData Specify the DO data field.

Description

Initializes the specified field.

The default value as a general rule is 0.

CDAQMXDOData::isObject

Syntax

```
virtual int isObject(const char * classname = "CDAQMXDOData");
```

Parameters

classname Specify the class name using a string.

Description

Checks whether the specified class name was inherited.

If parameters are omitted, checks whether it is this class.

Classes that inherit this class must be overridden in order to check their own classes.

Returns true (valid) if the class was inherited. Otherwise, returns false (invalid).

Return value

Returns a Boolean value.

CDAQMXDOData::operator=

Syntax

```
CDAQMXDOData & operator=(CDAQMXDOData & cMXDOData);
```

Parameters

cMXDOData Specify an object for substitution.

Description

Copies the data member of the specified object.

Return value

Returns the reference to the object.

CDAQMXDOData::setDO

Syntax

```
void setDO(int doNo, int bValid, int bONOFF = DAQMX_VALID_OFF);
```

Parameters

doNo Specify the data number.
bValid Specify valid or invalid using a Boolean value.
bONOFF Specify ON/OFF using a Boolean value.

Description

Stores the specified value in the field indicated by the specified DO data number in the DO data field of the data member.

If the constant for "Specify all DO numbers" is specified for the DO data number, the value is stored to all DO data.

Reference

getMXDO

CDAQMXDOData::setDOONOFF

Syntax

```
void setDOONOFF(int bONOFF);
```

Parameters

bONOFF Specify ON/OFF using a Boolean value.

Description

Changes to the specified value, the DO ON/OFF setting of all DO data whose DO data number is Valid in the DO data field of the data member.

Reference

getMXDO

CDAQMXDOData::setMXDOData

Syntax

```
void setMXDOData(MXDOData * pMXDOData);
```

Parameters

pMXDOData Specify the DO data.

Description

Sets the data in a structure. Stores the contents of the specified structure in the data member.

If not specified, the data member is initialized.

Reference

initialize

CDAQMXNetInfo Class

This class stores the network information data of the MX100.

It is a wrapper class of the MXNetInfo structure.

This class can be used as an interface for storing network information data when retrieving setup data.

Public Members

Construct/Destruct

CDAQMXNetInfo Constructs an object.

~CDAQMXNetInfo Destructs an object.

Structure Manipulation

getMXNetInfo Gets the data in a structure.

setMXNetInfo Sets the data in a structure.

initMXNetInfo Initializes the data in a structure.

Member Data Manipulation

initialize Initializes the data member.

getAddress Gets the IP address.

getPort Gets the port number.

getSubMask Gets the subnet mask.

getGateway Gets the gateway address.

getHost Gets the host name

Operator

operator= Executes substitution.

Utilities

getPart Gets parts of the IP address.

isObject Checks an object.

Protected Members

Data Members

m_MXNetInfo Field for storing the network information data.

Private Members

None.

Member Functions (Alphabetical Order)

CDAQMXNetInfo::CDAQMXNetInfo

Syntax

```
CDAQMXNetInfo(MXNetInfo * pMXNetInfo);  
virtual ~CDAQMXNetInfo(void);
```

Parameters

pMXNetInfo Specify the network information data.

Description

Constructs or destructs an object.

When constructing, the data member is set to the specified value. If not specified, the data member is initialized.

Reference

setMXNetInfo

CDAQMXNetInfo::getAddress

Syntax

```
unsigned int getAddress(void);
```

Description

Gets the IP address from the network information data field of the data member.

Return value

Returns the IP address

CDAQMXNetInfo::getGateway

Syntax

```
unsigned int getGateway(void);
```

Description

Gets the GATEWAY address from the network information data field of the data member.

Return value

Returns the gateway address

CDAQMXNetInfo::getHost

Syntax

```
const char * getHost(void);
```

Description

Gets the host name from the network information data field of the data member.

Return value

Returns a pointer to the string.

CDAQMXNetInfo::getMXNetInfo

Syntax

```
void getMXNetInfo(MXNetInfo * pMXNetInfo);
```

Parameters

pMXNetInfo Specify the destination where the network information data is to be returned.

Description

Gets the data in a structure. Stores the contents of the data member in the specified structure.

CDAQMXNetInfo::getPart

Syntax

```
static int getPart(unsigned int address, int index);
```

Parameters

address Specify the IP address.
index Specify the part position.

Description

Gets the byte value of the specified IP address where the address is divided at the specified part position.

Returns the byte value of the specified part position.

The part position is specified with the index value (starting from 0) in units of bytes.

The range is from 0 to 3.

Returns 0 if it does not exist.

Return value

Returns the byte value.

CDAQMXNetInfo::getPort

Syntax

```
unsigned int getPort(void);
```

Description

Gets the port number from the network information data field of the data member.

Return value

Returns the port number.

CDAQMXNetInfo::getSubMask

Syntax

```
unsigned int getSubMask(void);
```

Description

Gets the subnet mask from the network information data field of the data member.

Return value

Returns the subnet mask.

CDAQMXNetInfo::initialize

Syntax

```
virtual void initialize(void);
```

Description

Initializes the data member.

The default value as a general rule is 0.

Reference

initMXNetInfo

CDAQMXNetInfo::initMXNetInfo

Syntax

```
static void initMXNetInfo(MXNetInfo * pMXNetInfo);
```

Parameters

pMXNetInfo Specify the network information data field.

Description

Initializes the specified field.

The default value as a general rule is 0.

CDAQMXNetInfo::isObject

Syntax

```
virtual int isObject(const char * classname = "CDAQMXNetInfo");
```

Parameters

classname Specify the class name using a string.

Description

Checks whether the specified class name was inherited.

If parameters are omitted, checks whether it is this class.

Classes that inherit this class must be overridden in order to check their own classes.

Returns true (valid) if the class was inherited. Otherwise, returns false (invalid).

Return value

Returns a Boolean value.

CDAQMXNetInfo::operator=

Syntax

```
CDAQMXNetInfo & operator=(CDAQMXNetInfo & cMXNetInfo);
```

Parameters

cMXNetInfo Specify an object for substitution.

Description

Copies the data member of the specified object.

Return value

Returns the reference to the object.

CDAQMXNetInfo::setMXNetInfo

Syntax

```
void setMXNetInfo(MXNetInfo * pMXNetInfo);
```

Parameters

pMXNetInfo Specify the network information data.

Description

Sets the data in a structure.

Stores the contents of the specified structure in the data member.

If not specified, the data member is initialized.

Reference

initialize

CDAQMXOutput Class

This class stores the output channel data on the MX100.

It is a wrapper class of the MXOutputData structure.

It is a group of all the channels worth of output channel data.

Each data can be accessed by output channel data number.

The output channel data number is the AO/PWM data number.

This class can be used as an interface for setting and retrieving output channel data.

Public Members

Construct/Destruct

CDAQMXOutputData Constructs an object.

~CDAQMXOutputData Destructs an object.

Structure Manipulation

getMXOutputData Gets the data in a structure.

setMXOutputData Sets the data in a structure.

initMXOutputData Initializes the data in a structure.

Member Data Manipulation

initialize Initializes the data member.

getOutputType Gets the output terminal type.

getIdleChoice Gets the selected value when idling.

getErrorChoice Gets the selected value when an error occurs.

getPresetValue Gets the value if the selected value is the “specified value.”

getPulseTime Gets the integer multiple of the pulse interval.

setOutputType Sets the output type.

setChoice Sets the selected value.

setPulseTime Sets the integer multiple of the pulse interval.

Utilities

isObject Checks an object.

Operator

operator= Executes substitution.

Protected Members

Data Members

m_MXOutputData Field for storing the output channel data.

Member Access

getMXOutput Gets the output channel data structure for each channel.

Private Members

None.

Member Functions (Alphabetical Order)

CDAQMXOutputData::CDAQMXOutputData

Syntax

```
CDAQMXOutputData(MXOutputData * pMXOutputData = NULL);  
virtual ~CDAQMXOutputData(void);
```

Parameters

pMXOutputData Specify the output channel data.

Description

Constructs or destructs an object.

When constructing, the data member is set to the specified value. If the specification is omitted, the data member is initialized.

Reference

setMXOutputData

CDAQMXOutputData::getErrorChoice

Syntax

```
int getErrorChoice(int outputNo);
```

Parameters

outputNo Specify the output channel data number.

Description

Gets the selected value when an error occurs of the specified data number from the data member.

If it does not exist, "Previous value" is returned.

Return value

Returns the selected value.

Reference

getMXOutput

CDAQMXOutputData::getIdleChoice

Syntax

```
int getIdleChoice(int outputNo);
```

Parameters

outputNo Specify the output channel data number.

Description

Gets the selected value during idling of the specified data number from the data member.

If it does not exist, "Previous value" is returned.

Return value

Returns the selected value.

Reference

getMXOutput

CDAQMXOutputData::getMXOutput

Syntax

```
MXOutput * getMXOutput(int outputNo);
```

Parameters

outputNo Specify the output channel data number.

Description

Gets the structure of the specified data number from the data member.

Returns NULL if it does not exist.

Return value

Returns a pointer to the structure.

CDAQMXOutputData::getMXOutputData

Syntax

```
void getMXOutputData(MXOutputData * pMXOutputData);
```

Parameters

pMXOutputData Specify the destination where the output channel data is to be returned.

Description

Gets the data in a structure.

Stores the contents of the data member in the specified structure.

CDAQMXOutputData::getOutputType

Syntax

```
int getOutputType(int outputNo);
```

Parameters

outputNo Specify the output channel data number.

Description

Gets the output type of the specified data number from the data member.
If it does not exist, “No output” is returned.

Return value

Returns the output type.

Reference

getMXOutput

CDAQMXOutputData::getPresetValue

Syntax

```
int getPresetValue(int outputNo);
```

Parameters

outputNo Specify the output channel data number.

Description

Gets the value when the selected value that indicates the specified data number is the “Specified value” from the data member.
Returns 0 if it does not exist.

Return value

Returns the value if the selected value is the “specified value.”

Reference

getMXOutput

CDAQMXOutputData::getPulseTime

Syntax

```
int getPulseTime(int outputNo);
```

Parameters

outputNo Specify the output channel data number.

Description

Gets the integral multiple of the pulse interval of the specified data number from the data member.
Returns 1 if it does not exist.

Return value

Returns the integer multiple of the pulse interval .

Reference

getMXOutput

CDAQMXOutputData::initialize

Syntax

```
virtual void initialize(void);
```

Description

Initializes the data member.

The default value as a general rule is 0.

Reference

initMXOutputData

CDAQMXOutputData::initMXOutputData

Syntax

```
static void initMXOutputData(MXOutputData * pMXOutputData);
```

Parameters

pMXOutputData Specify the output channel data field.

Description

Initializes the specified field.

The default value as a general rule is 0.

CDAQMXOutputData::isObject

Syntax

```
virtual int isObject(const char * classname =  
"CDAQMXOutputData");
```

Parameters

classname Specify the class name using a string.

Description

Checks whether the specified class name was inherited.

If parameters are omitted, checks whether it is this class.

Classes that inherit this class must be overridden in order to check their own classes.

Returns true (valid) if the class was inherited. Otherwise, returns false (invalid).

Return value

Returns a Boolean value.

CDAQMXOutputData::operator=

Syntax

```
CDAQMXOutputData & operator=(CDAQMXOutputData &
cMXOutputData);
```

Parameters

cMXOutputData Specify an object for substitution.

Description

Copies the data member of the specified object.

Return value

Returns the reference to the object.

CDAQMXOutputData::setChoice

Syntax

```
void setChoice(int outputNo, int idleChoice, int errorChoice,
int presetValue = 0);
```

Parameters

outputNo Specify the output channel data number.
idleChoice Specify the selected value when idling.
errorChoice Specify the selected value when an error occurs.
presetValue Specify the value if the selected value is the “specified value.”

Description

Stores the specified value in the field indicated by the specified data number of the data member.

If the data number is set to “Specify All output data numbers,” the same value is stored in all data.

Reference

getMXOutput

CDAQMXOutputData::setMXOutputData

Syntax

```
void setMXOutputData(MXOutputData * pMXOutputData);
```

Parameters

pMXOutputData Specify the output channel data.

Description

Sets the data in a structure.

Stores the contents of the specified structure in the data member.

If not specified, the data member is initialized.

Reference

initMXOutputData

CDAQMXOutputData::setOutputType

Syntax

```
void setOutputType(int outputNo, int iOutput);
```

Parameters

outputNo	Specify the output channel data number.
iOutput	Specify the output type.

Description

Stores the specified value in the field indicated by the specified data number of the data member.

If the data number is set to "Specify All output data numbers," the same value is stored in all data.

Other item fields are set to default values.

Reference

```
getMXOutput setChoice setPulseTime
```

CDAQMXOutputData::setPulseTime

Syntax

```
void setPulseTime(int outputNo, int pulseTime);
```

Parameters

outputNo	Specify the output channel data number.
pulseTime	Specify the integer multiple of the pulse interval.

Description

Stores the specified value in the field indicated by the specified data number of the data member.

If the data number is set to "Specify All output data numbers," the same value is stored in all data.

Reference

```
getMXOutput
```

CDAQMXSegment Class

This class stores the display pattern of the 7-segment LED on the MX100.

It is a wrapper class of the MXSegment structure.

This class can be used as an interface for storing the display pattern of the 7-segment LED.

Public Members

Construct/Destruct

CDAQMXSegment Constructs an object.

~CDAQMXSegment Destructs an object.

Structure Manipulation

getMXSegment Gets the data in a structure.

setMXSegment Sets the data in a structure.

initMXSegment Initializes the data in a structure.

Member Data Manipulation

initialize Initializes the data member.

getPattern Gets the display pattern.

setPattern Sets the display pattern.

Operator

operator= Executes substitution.

Utilities

isObject Checks an object.

Protected Members

Data Members

m_MXSegment Field for storing the 7-segment LED.

Private Members

None.

Member Functions (Alphabetical Order)

CDAQMXSegment::CDAQMXSegment

Syntax

```
CDAQMXSegment(MXSegment * pMXSegment = NULL);
CDAQMXSegment(int pattern0, int pattern1);
virtual ~CDAQMXSegment(void);
```

Parameters

pMXSegment Specify the 7-segment LED.
 pattern0 Specify the display pattern of segment number 0.
 pattern1 Specify the display pattern of segment number 1.

Description

Constructs or destructs an object.

When constructing, the data member is set to the specified value. If not specified, the data member is initialized.

Reference

initialize setMXSegment setPattern

CDAQMXSegment::getMXSegment

Syntax

```
void getMXSegment(MXSegment * pMXSegment);
```

Parameters

pMXSegment Specify the destination where the 7-segment LED is to be returned.

Description

Gets the data in a structure. Stores the contents of the data member in the specified structure.

CDAQMXSegment::getPattern

Syntax

```
int getPattern(int segmentNo);
```

Parameters

segmentNo Specify the segment number.

Description

Gets the display pattern value of the specified segment number from the 7-segment LED field of the data member.

Returns 0 if it does not exist.

Return value

Returns the display pattern.

CDAQMXSegment::initialize

Syntax

```
virtual void initialize(void);
```

Description

Initializes the data member.

The default value as a general rule is 0.

Reference

initMXSegment

CDAQMXSegment::initMXSegment

Syntax

```
static void initMXSegment(MXSegment * pMXSegment);
```

Parameters

pMXSegment Specify the 7-segment LED field.

Description

Initializes the specified field.

The default value as a general rule is 0.

CDAQMXSegment::isObject

Syntax

```
virtual int isObject(const char * classname = "CDAQMXSegment");
```

Parameters

classname Specify the class name using a string.

Description

Checks whether the specified class name was inherited.

If parameters are omitted, checks whether it is this class.

Classes that inherit this class must be overridden in order to check their own classes.

Returns true (valid) if the class was inherited. Otherwise, returns false (invalid).

Return value

Returns a Boolean value.

CDAQMXSegment::operator=

Syntax

```
CDAQMXSegment & operator=(CDAQMXSegment & cMXSegment);
```

Parameters

cMXSegment Specify an object for substitution.

Description

Copies the data member of the specified object.

Return value

Returns the reference to the object.

CDAQMXSegment::setMXSegment

Syntax

```
void setMXSegment(MXSegment * pMXSegment);
```

Parameters

pMXSegment Specify the 7-segment LED.

Description

Sets the data in a structure. Stores the contents of the specified structure in the data member.

If not specified, the data member is initialized.

Reference

initialize

CDAQMXSegment::setPattern

Syntax

```
void setPattern(int segmentNo, int pattern);
```

Parameters

segmentNo Specify the segment number.

pattern Specify the display pattern.

Description

Stores the specified value to the field indicated by the specified segment number in the 7-segment LED field of the data member.

If the constant for “Specify all segment numbers” is specified for the segment number, the value is stored for all 7-segment LEDs.

CDAQMXStatus Class

This class stores the status data of the MX100.

It is a wrapper class of the MXStatus structure.

This class can be used as an interface for storing status data when retrieving status data and setup data.

Public Members

Construct/Destruct

CDAQMXStatus Constructs an object.

~CDAQMXStatus Destructs an object.

Structure Manipulation

getMXStatus Gets the data in a structure.

setMXStatus Sets the data in a structure.

initMXStatus Initializes the data in a structure.

Member Data Manipulation

initialize Initializes the data member.

getFIFONum Gets the valid number of FIFOs.

getFIFOStatus Gets the FIFO status value.

getInterval Gets the interval type.

getOldDataNo Gets the oldest data number.

getNewDataNo Gets the newest data number.

getCFStatus Gets the CF status type.

getCFSize Gets the size of the CF.

getCFRemain Gets the remaining size.

getUnitStatus Gets the unit status value.

getConfigCnt Gets the setup number (a sequential number that counts the setup execution).

getTimeCnt Gets the time number (a sequential number that counts the execution of time setting).

isBackup Gets the presence/absence of backup.

getTime Gets the time.

getMilliSecond Gets milliseconds.

Operator

operator= Executes substitution.

Utilities

isDataNo Checks the validity of data numbers.

isObject Checks an object.

Protected Members

Data Members

m_MXStatus Field for storing the status data.

Member Access

getMXFIFOInfo Gets the FIFO information structure for each FIFO.

Private Members

None.

Member Functions (Alphabetical Order)

CDAQMXStatus::CDAQMXStatus

Syntax

```
CDAQMXStatus(MXStatus * pMXStatus = NULL);
virtual ~CDAQMXStatus(void);
```

Parameters

pMXStatus Specify the status data.

Description

Constructs or destructs an object.

When constructing, the data member is set to the specified value. If not specified, the data member is initialized.

Reference

setMXStatus

CDAQMXStatus::getCFRemain

Syntax

```
int getCFRemain(void);
```

Description

Gets the CF remaining capacity value from the status data field of the data member. The unit is KB.

Return value

Returns the remaining size.

CDAQMXStatus::getCFSize

Syntax

```
int getCFSize(void);
```

Description

Gets the CF capacity value from the status data field of the data member.
The unit is KB.

Return value

Returns the size.

CDAQMXStatus::getCFStatus

Syntax

```
int getCFStatus(void);
```

Description

Gets the CF status type value from the status data field of the data member.

Return value

Returns the CF status type.

CDAQMXStatus::getConfigCnt

Syntax

```
int getConfigCnt(void);
```

Description

Gets the setting number value from the status data field of the data member.

Return value

Returns the setup number.

CDAQMXStatus::getDateTime

Syntax

```
void getDateTime(CDAQDateTime & cDateTime);
```

Parameters

cDateTime	Specify the destination where the time information data is to be returned.
-----------	--

Description

Stores the time information data from the status data field of the data member in the specified field.

Reference

```
getMilliSecond getTime  
CDAQDateTime::setMilliSecond  
CDAQDateTime::setTime
```

CDAQMXStatus::getFIFONum

Syntax

```
int getFIFONum(void);
```

Description

Gets the valid number of FIFO value from the status data field of the data member.

Return value

Returns the valid number of FIFOs.

CDAQMXStatus::getFIFOStatus

Syntax

```
int getFIFOStatus(int fifoNo);
```

Parameters

fifoNo Specify the FIFO number.

Description

Gets the FIFO status value of the FIFO information corresponding to the specified FIFO number from the status data field of the data member.

If it does not exist, "Unknown" is returned.

Return value

Returns the FIFO status value.

Reference

getMXFIFOInfo

CDAQMXStatus::getInterval

Syntax

```
int getInterval(int fifoNo);
```

Parameters

fifoNo Specify the FIFO number.

Description

Gets the interval type value of the FIFO information corresponding to the specified FIFO number from the status data field of the data member.

Returns 0 if it does not exist.

Return value

Returns the interval type.

Reference

getMXFIFOInfo

CDAQMXStatus::getMilliSecond

Syntax

`int getMilliSecond(void);`

Description

Gets milliseconds from the status data field of the data member.

Return value

Returns milliseconds.

CDAQMXStatus::getMXFIFOInfo

Syntax

`MXFIFOInfo * getMXFIFOInfo(int fifoNo);`

Parameters

`fifoNo` Specify the FIFO number.

Description

Gets the structure of the specified FIFO number from the status data field of the data member.

Returns NULL if it does not exist.

Return value

Returns a pointer to the structure.

CDAQMXStatus::getMXStatus

Syntax

`void getMXStatus(MXStatus * pMXStatus);`

Parameters

`pMXStatus` Specify the destination where the status data is to be returned.

Description

Gets the data in a structure.

Stores the contents of the data member in the specified structure.

CDAQMXStatus::getNewDataNo

Syntax

```
MXDataNo getNewDataNo(int fifoNo);
```

Parameters

fifoNo Specify the FIFO number.

Description

Gets the newest data number of the FIFO information corresponding to the specified FIFO number from the status data field of the data member.

Returns the constant for “Data number for instantaneous value specification” if it does not exist.

Return value

Returns the data number.

Reference

```
getMXFIFOInfo
```

CDAQMXStatus::getOldDataNo

Syntax

```
MXDataNo getOldDataNo(int fifoNo);
```

Parameters

fifoNo Specify the FIFO number.

Description

Gets the oldest data number of the FIFO information corresponding to the specified FIFO number from the status data field of the data member.

Returns the constant for “Data number for instantaneous value specification” if it does not exist.

Return value

Returns the data number.

Reference

```
getMXFIFOInfo
```

CDAQMXStatus::getTime

Syntax

```
time_t getTime(void);
```

Description

Gets the number of seconds from the status data field of the data member.

Return value

Returns seconds.

CDAQMXStatus::getTimeCnt

Syntax

```
int getTimeCnt(void);
```

Description

Gets the time number value from the status data field of the data member.

Return value

Returns the time number.

CDAQMXStatus::getUnitStatus

Syntax

```
int getUnitStatus(void);
```

Description

Gets the unit status value from the status data field of the data member.

Return value

Returns the unit status value.

CDAQMXStatus::initialize

Syntax

```
virtual void initialize(void);
```

Description

Initializes the data member.

The default value as a general rule is 0.

Reference

initMXStatus

CDAQMXStatus::initMXStatus

Syntax

```
static void initMXStatus(MXStatus * pMXStatus);
```

Parameters

pMXStatus Specify the status data field.

Description

Initializes the specified field.

The default value as a general rule is 0.

CDAQMXStatus::isBackup

Syntax

```
int isBackup(void);
```

Description

Gets the presence/absence of backup from the status data field of the data member.

Return value

Returns a Boolean value.

CDAQMXStatus::isDataNo

Syntax

```
static int isDataNo(MXDataNo dataNo);
```

Parameters

dataNo Specify the data number.

Description

Checks whether the specified data number is a valid number.

Returns "Valid" if the data number is 0 or greater.

Return value

Returns a Boolean value.

CDAQMXStatus::isObject

Syntax

```
virtual int isObject(const char * classname = "CDAQMXStatus");
```

Parameters

classname Specify the class name using a string.

Description

Checks whether the specified class name was inherited.

If parameters are omitted, checks whether it is this class.

Classes that inherit this class must be overridden in order to check their own classes.

Returns true (valid) if the class was inherited. Otherwise, returns false (invalid).

Return value

Returns a Boolean value.

CDAQMXStatus::operator=

Syntax

```
CDAQMXStatus & operator=(CDAQMXStatus & cMXStatus);
```

Parameters

cMXStatus Specify an object for substitution.

Description

Copies the data member of the specified object.

Return value

Returns the reference to the object.

CDAQMXStatus::setMXStatus

Syntax

```
void setMXStatus(MXStatus * pMXStatus);
```

Parameters

pMXStatus Specify the status data.

Description

Sets the data in a structure.

Stores the contents of the specified structure in the data member.

If not specified, the data member is initialized.

Reference

initialize

CDAQMXSysInfo Class

This class stores the System configuration data of the MX100.

It is a wrapper class of the MXSystemInfo structure.

This class can be used as an interface for storing system configuration data when retrieving system configuration data and setup data.

Public Members

Construct/Destruct

CDAQMXSysInfo	Constructs an object.
~CDAQMXSysInfo	Destructs an object.

Structure Manipulation

getMXSystemInfo	Gets the data in a structure.
setMXSystemInfo	Sets the data in a structure.
initMXSystemInfo	Initializes the data in a structure.

Member Data Manipulation

initialize	Initializes the data member.
getUnitType	Gets the unit type.
getStyle	Gets the style.
getUnitNo	Gets the unit number.
getTempUnit	Gets the temperature unit type.
getCFTimeout	Gets the timeout value.
getCFWriteMode	Gets the CF write mode.
getFrequency	Gets the power supply frequency.
getPartNo	Gets the part number.
getOption	Gets the option.
getUnitSerial	Gets the serial number of the unit.
getMAC	Gets the MAC address.
getModuleType	Gets the module type.
getChNum	Gets the number of channels.
getInterval	Gets the interval type.
getIntegral	Gets the type of A/D integration time.
getStandbyType	Gets the module type at startup.
getRealType	Gets the actual module type.
isModuleValid	Gets the Boolean value of the module.
getModuleVersion	Gets the module version.
getTerminalType	Gets the terminal type.
getFIFONo	Gets the FIFO number.
getModuleSerial	Gets the serial number of the module.
setUnitNo	Sets the unit number.

setTempUnit	Sets the temperature unit type.
setCFTimeout	Sets the timeout value.
setCFWriteMode	Sets the CF write mode.
setModule	Sets the module.
setRealModule	Changes the module to the actual module.

Check

isCorrect	Checks the validity.
-----------	----------------------

Utilities

getItemError	Gets the number of the parameter on which an error was detected.
isObject	Checks an object.

Operator

operator=	Executes substitution.
-----------	------------------------

Protected Members

Data Members

m_MXSystemInfo	Field for storing the system configuration data.
m_nItemError	Field for storing the setting item number.

Member Access

getMXModuleData	Gets the module information structure for each module.
-----------------	--

Private Members

None.

Member Functions (Alphabetical Order)

CDAQMXSysInfo::CDAQMXSysInfo

Syntax

```
CDAQMXSysInfo(MXSystemInfo * pMXSystemInfo = NULL);  
virtual ~CDAQMXSysInfo(void);
```

Parameters

pMXSystemInfo	Specify the system configuration data.
---------------	--

Description

Constructs or destructs an object.

When constructing, the data member is set to the specified value. If not specified, the data member is initialized.

Reference

setMXSystemInfo

CDAQMXSysInfo::getCFTimeout

Syntax

```
int getCFTimeout(void);
```

Description

Gets the timeout value from the System configuration data field of the data member.

Return value

Returns the timeout value.

CDAQMXSysInfo::getCFWriteMode

Syntax

```
int getCFWriteMode(void);
```

Description

Gets the CF write mode value from the System configuration data field of the data member.

Return value

Returns the CF write mode.

CDAQMXSysInfo::getChNum

Syntax

```
int getChNum(int moduleNo);
```

Parameters

moduleNo Specify the module number.

Description

Gets the number of channels of the module corresponding to the specified module number from the System configuration data field of the data member.

Returns 0 if it does not exist.

Return value

Returns the number of channels.

Reference

getMXModuleData

CDAQMXSysInfo::getFIFONo

Syntax

```
int getFIFONo(int moduleNo);
```

Parameters

moduleNo Specify the module number.

Description

Gets the FIFO number of the module corresponding to the specified module number from the System configuration data field of the data member.

Returns a negative value if it does not exist.

Return value

Returns the FIFO number.

Reference

getMXModuleData

CDAQMXSysInfo::getFrequency

Syntax

```
int getFrequency(void);
```

Description

Gets the power supply frequency from the System configuration data field of the data member.

Return value

Returns the power supply frequency.

CDAQMXSysInfo::getIntegral

Syntax

```
int getIntegral(int moduleNo);
```

Parameters

moduleNo Specify the module number.

Description

Gets the type of A/D integration time of the module corresponding to the specified module number from the System configuration data field of the data member.

If it does not exist, "Automatic" is returned.

Return value

Returns the type of AD integration time.

Reference

getMXModuleData

CDAQMXSysInfo::getInterval

Syntax

```
int getInterval(int moduleNo);
```

Parameters

moduleNo Specify the module number.

Description

Gets the interval type of the module corresponding to the specified module number from the System configuration data field of the data member.

Returns 0 if it does not exist.

Return value

Returns the interval type.

Reference

getMXModuleData

CDAQMXSysInfo::getItemError

Syntax

```
int getItemError(void);
```

Description

Gets the value of the setting item number field of the data member.

Return value

Returns the setting item number.

CDAQMXSysInfo::getMAC

Syntax

```
unsigned char getMAC(int index);
```

Parameters

index Specify the byte position.

Description

Gets the MAC address from the System configuration data field of the data member. The address is retrieved in units of bytes. Returns 0 if the value is outside the range.

The byte position is an integer starting with 0.

There is a definition for the number of MAC address elements.

Return value

Returns the byte value.

CDAQMXSysInfo::getModuleSerial

Syntax

```
const char * getModuleSerial(int moduleNo);
```

Parameters

moduleNo Specify the module number.

Description

Gets the serial number of the module corresponding to the specified module number from the System configuration data field of the data member.

Returns NULL if it does not exist.

Return value

Returns a pointer to the string.

Reference

getMXModuleData

CDAQMXSysInfo::getModuleType

Syntax

```
int getModuleType(int moduleNo);
```

Parameters

moduleNo Specify the module number.

Description

Gets the module type of the module corresponding to the specified module number from the System configuration data field of the data member.

If it does not exist, "No module" is returned.

Return value

Returns the module type.

Reference

getMXModuleData

CDAQMXSysInfo::getModuleVersion

Syntax

```
int getModuleVersion(int moduleNo);
```

Parameters

moduleNo Specify the module number.

Description

Gets the module version of the module corresponding to the specified module number from the System configuration data field of the data member.

Returns 0 if it does not exist.

Return value

Returns the module version.

Reference

getMXModuleData

CDAQMXSysInfo::getMXModuleData

Syntax

```
MXModuleData * getMXModuleData(int moduleNo);
```

Parameters

moduleNo Specify the module number.

Description

Gets the structure of the specified module number from the System configuration data field of the data member.

Returns NULL if it does not exist.

Return value

Returns a pointer to the structure.

CDAQMXSysInfo::getMXSystemInfo

Syntax

```
void getMXSystemInfo(MXSystemInfo * pMXSystemInfo);
```

Parameters

pMXSystemInfo Specify the destination where the system configuration data is to be returned.

Description

Gets the data in a structure.

Stores the contents of the data member in the specified structure.

CDAQMXSysInfo::getOption

Syntax

```
int getOption(void);
```

Description

Gets the option value from the System configuration data field of the data member.

Return value

Returns the option.

CDAQMXSysInfo::getPartNo

Syntax

```
const char * getPartNo(void);
```

Description

Gets the part number from the System configuration data field of the data member.

Return value

Returns a pointer to the string.

CDAQMXSysInfo::getRealType

Syntax

```
int getRealType(int moduleNo);
```

Parameters

moduleNo Specify the module number.

Description

Gets the actual module type of the module corresponding to the specified module number from the System configuration data field of the data member.

If it does not exist, "No module" is returned.

Return value

Returns the module type.

Reference

getMXModuleData

CDAQMXSysInfo::getStandbyType

Syntax

```
int getStandbyType(int moduleNo);
```

Parameters

moduleNo Specify the module number.

Description

Gets the module type upon startup of the module corresponding to the specified module number from the System configuration data field of the data member.

If it does not exist, "No module" is returned.

Return value

Returns the module type.

Reference

getMXModuleData

CDAQMXSysInfo::getStyle

Syntax

```
int getStyle(void);
```

Description

Gets the style from the System configuration data field of the data member.

Return value

Returns the style value.

CDAQMXSysInfo::getTempUnit

Syntax

```
int getTempUnit(void);
```

Description

Gets the temperature units from the System configuration data field of the data member.

Return value

Returns the temperature unit type.

CDAQMXSysInfo::getTerminalType

Syntax

```
int getTerminalType(int moduleNo);
```

Parameters

moduleNo Specify the module number.

Description

Gets the terminal type of the module corresponding to the specified module number from the System configuration data field of the data member.

If it does not exist, "Clamp" is returned.

Return value

Returns the terminal type.

Reference

getMXModuleData

CDAQMXSysInfo::getUnitNo

Syntax

```
int getUnitNo(void);
```

Description

Gets the unit number value from the System configuration data field of the data member.

Return value

Returns the unit number.

CDAQMXSysInfo::getUnitSerial

Syntax

```
const char * getUnitSerial(void);
```

Description

Gets the serial number of the unit from the System configuration data field of the data member.

Return value

Returns a pointer to the string.

CDAQMXSysInfo::getUnitType

Syntax

```
int getUnitType(void);
```

Description

Gets the unit type value from the System configuration data field of the data member.

Return value

Returns the unit type.

CDAQMXSysInfo::initialize

Syntax

```
virtual void initialize(void);
```

Description

Initializes the data member.

The default value as a general rule is 0.

The timeout value is set to default.

Reference

```
initMXSystemInfo setCFTimeout
```

CDAQMXSysInfo::initMXSystemInfo

Syntax

```
static void initMXSystemInfo(MXSystemInfo * pMXSystemInfo);
```

Parameters

pMXSystemInfo Specify the system configuration data field.

Description

Initializes the specified field.

The default value as a general rule is 0.

CDAQMXSysInfo::isCorrect

Syntax

```
int isCorrect(void);
```

Description

Checks the validity.

Checks each setting item.

Checks the FIFO number limit.

If an invalid value is detected, Invalid is returned.

If an invalid value is detected, the setting item number than indicates the detected location is stored in the setting item number field of the data member.

Return value

Returns a Boolean value.

Reference

```
getCFTimeout getCFWriteMode getMXModuleData getTempUnit  
getUnitNo
```

CDAQMXSysInfo::isModuleValid

Syntax

```
int isModuleValid(int moduleNo);
```

Parameters

moduleNo Specify the module number.

Description

Gets the Boolean value of the module corresponding to the specified module number from the System configuration data field of the data member.

If it does not exist, Invalid is returned.

Return value

Returns a Boolean value.

Reference

getMXModuleData

CDAQMXSysInfo::isObject

Syntax

```
virtual int isObject(const char * classname = "CDAQMXSysInfo");
```

Parameters

classname Specify the class name using a string.

Description

Checks whether the specified class name was inherited.

If parameters are omitted, checks whether it is this class.

Classes that inherit this class must be overridden in order to check their own classes.

Returns true (valid) if the class was inherited. Otherwise, returns false (invalid).

Return value

Returns a Boolean value.

CDAQMXSysInfo::operator=

Syntax

```
CDAQMXSysInfo & operator=(CDAQMXSysInfo & cMXSysInfo);
```

Parameters

cMXSysInfo Specify an object for substitution.

Description

Copies the data member of the specified object.

Return value

Returns the reference to the object.

CDAQMXSysInfo::setCFTimeout

Syntax

```
void setCFTimeout(int timeout = 60);
```

Parameters

timeout Sets the timeout value.

Description

Stores the specified value in the System configuration data field of the data member. The unit is seconds.

CDAQMXSysInfo::setCFWriteMode

Syntax

```
void setCFWriteMode(int iCFWriteMode);
```

Parameters

iCFWriteMode Specify the CF write mode.

Description

Stores the specified value in the System configuration data field of the data member.

CDAQMXSysInfo::setModule

Syntax

```
void setModule(int moduleNo, int iModuleType, int iChNum, int iInterval, int iHz = DAQMX_INTEGRAL_AUTO);
```

Parameters

moduleNo Specify the module number.
iModuleType Specify the module type.
iChNum Specify the number of channels.
iInterval Specify the interval type.
iHz Specify the type of A/D integration time.

Description

Stores the specified values in the data member's System configuration data field of the module corresponding to the specified module number. If the module does not exist, the value is not stored.

Reference

getMXModuleData

CDAQMXSysInfo::setMXSystemInfo

Syntax

```
void setMXSystemInfo(MXSystemInfo * pMXSystemInfo);
```

Parameters

pMXSystemInfo Specify the system configuration data.

Description

Sets the data in a structure.

Stores the contents of the specified structure in the data member.

If not specified, the data member is initialized.

Reference

initialize

CDAQMXSysInfo::setRealModule

Syntax

```
int setRealModule(int moduleNo);
```

Parameters

moduleNo Specify the module number.

Description

Sets the module corresponding to the specified module number to the actual module type.

Updates the module settings corresponding to the module number specified in the System configuration data field of the data member.

The settings are set to the default values of the module type.

Returns the actual module type.

Return value

Returns the module type.

Reference

getRealType setModule

CDAQMXSysInfo::setTempUnit

Syntax

```
void setTempUnit(int iTempUnit);
```

Parameters

iTempUnit Specify the temperature unit type.

Description

Stores the specified value in the System configuration data field of the data member.

CDAQMXSysInfo::setUnitNo

Syntax

```
void setUnitNo(int unitNo);
```

Parameters

unitNo Specify the unit number.

Description

Stores the specified value in the System configuration data field of the data member.

CDAQMXTransmit Class

This class stores the transmission output data of the MX100.

It is a wrapper class of the MXTransmit structure.

It is a group of transmissions statuses of all the channels.

Each data can be accessed by transmission output data number. The transmission output data number is the PWM data number or the AO data number.

This class can be used as an interface for setting and retrieving transmission output data.

Public Members

Construct/Destruct

CDAQMXTransmit	Constructs an object.
~CDAQMXTransmit	Destructs an object.

Structure Manipulation

getMXTransmit	Gets the data in a structure.
setMXTransmit	Sets the data in a structure.
initMXTransmit	Initializes the data in a structure.

Member Data Manipulation

initialize	Initializes the data member.
getTransmit	Gets the transmission status.
setTransmit	Sets the transmission status.

Operator

operator=	Executes substitution.
-----------	------------------------

Utilities

isObject	Checks an object.
----------	-------------------

Protected Members

Data Members

m_MXTransmit	Field for storing the transmission status.
--------------	--

Private Members

None.

Member Functions (Alphabetical Order)

CDAQMXTransmit::CDAQMXTransmit

Syntax

```
CDAQMXTransmit(MXTransmit * pMXTransmit = NULL);
virtual ~CDAQMXTransmit(void);
```

Parameters

pMXTransmit Specify the transmission status.

Description

Constructs or destructs an object.

When constructing, the data member is set to the specified value. If not specified, the data member is initialized.

Reference

setMXTransmit

CDAQMXTransmit::getMXTransmit

Syntax

```
void getMXTransmit(MXTransmit * pMXTransmit);
```

Parameters

pMXTransmit Specify the return destination of the transmission status.

Description

Gets the data in a structure.

Stores the contents of the data member in the specified structure.

CDAQMXTransmit::getTransmit

Syntax

```
int getTransmit(int aopwmNo);
```

Parameters

aopwmNo Specify the AO/PWM data number.

Description

Gets the transmission status of the specified data number from the data member.

If it does not exist, "Unspecified (unknown)" is returned.

Return value

Returns the transmission status.

CDAQMXTransmit::initialize

Syntax

```
virtual void initialize(void);
```

Description

Initializes the data member.

The default value as a general rule is 0.

Reference

```
initMXTransmit
```

CDAQMXTransmit::initMXTransmit

Syntax

```
static void initMXTransmit(MXTransmit * pMXTransmit);
```

Parameters

pMXTransmit Specify the transmission status field.

Description

Initializes the specified field.

The default value as a general rule is 0.

CDAQMXTransmit::isObject

Syntax

```
virtual int isObject(const char * classname =  
"CDAQMXTransmit");
```

Parameters

classname Specify the class name using a string.

Description

Checks whether the specified class name was inherited.

If parameters are omitted, checks whether it is this class.

Classes that inherit this class must be overridden in order to check their own classes.

Returns true (valid) if the class was inherited. Otherwise, returns false (invalid).

Return value

Returns a Boolean value.

CDAQMXTransmit::operator=

Syntax

```
CDAQMXTransmit & operator=(CDAQMXTransmit & cMXTransmit);
```

Parameters

cMXTransmit Specify an object for substitution.

Description

Copies the data member of the specified object.

Return value

Returns the reference to the object.

CDAQMXTransmit::setMXTransmit

Syntax

```
void setMXTransmit(MXTransmit * pMXTransmit);
```

Parameters

pMXTransmit Specify the transmission status.

Description

Sets the data in a structure.

Stores the contents of the specified structure in the data member.

If not specified, the data member is initialized.

Reference

initMXTransmit

CDAQMXTransmit::setTransmit

Syntax

```
void setTransmit(int aopwmNo, int iTransmit);
```

Parameters

aopwmNo Specify the AO/PWM data number.

iTransmit Specify the transmission status.

Description

Stores the specified value in the field indicated by the specified data number of the data member.

If the data number is set to "Specify All AO/PWM data numbers" the same value is stored in all data.

3.1 Functions and Their Functionalities - MX100/ Visual C -

This section indicates the correspondence between the functionalities that the API supports and the C functions.

Communication Functions

Function	Function
Connect to the MX100.	openMX
Disconnect from the MX100.	closeMX
Set the communication timeout.	setTimeoutMX

Note

Setting of the communication timeout is not recommended because unexpected disconnection may occur due to the conflict with the timeout time when data is retrieved.

Control Functions

Starting/Stopping the FIFO

Function	Function
Start the FIFO	startFIFOMX
Stop the FIFO	stopFIFOMX
Set auto control of the FIFO.	autoFIFOMX

Other Controls

Function	FIFO	Function
Set the number of seconds from the reference date/time (Jan. 1, 1970) on the MX100.	Stop	setDateTimeMX
Set the MX100 to the current date/time.	Stop	setDateTimeNowMX
Turn ON/OFF data saving to the CF card (data backup).	Continue	setBackupMX
Format the CF card.	Stop	formatCFMX
• Reconfigure the system of the unit.	Stop	initSystemMX
• Initialize the system of the unit.	Stop	
• Reset alarms (alarm ACK) of the unit.	Continue	
Set the 7-segment LED display.	Continue	setSegmentMX

The FIFO column in the table indicates the FIFO operation when the function is executed while FIFO is running.

Stop: Stops the FIFO when the function is executed.

Continue: Continues the FIFO even when the function is executed.

With the backup settings, if the CF write mode is changed, the FIFO stops. The CF write mode for backup settings performs a setting change (modifies collectively acquired data and sends it collectively).

Note

If the auto control of the FIFO is enabled, the FIFO is automatically resumed when the FIFO is stopped due to an execution of a function.

Functionality

Collective Setup

Function	FIFO	Function
Configure the setup data collectively.	Stop	setConfigDataMX
Set the setup data (system setup data).	Stop	setSystemConfigMX
Configure the setup data (channel setup data).	Stop	setChConfigMX
Set the DO (Digital Output) data collectively.	Continue	setDODataMX
Send AO/PWM data collectively.	Continue	setAOPWMDataMX
Send transmission output data	Continue	setTransmitMX

For a description of the FIFO column in the table, see the explanation given in “Other Controls” on the previous page. If an invalid data error occurs during entry of setup data, the last detected item is saved as a setup item number. It can be retrieved with a utility. It can be retrieved with a utility.

Individual Setup

Function	FIFO	Function
Set initial balance data	Stop	setBalanceMX
Set output channel data	Stop	setOutputMX
Initial balance data	Execute	runBalanceMX
	Reset	resetBalanceMX

Setup Change

Function	FIFO	Function
Range Settings	SKIP (not used)	setSKIPMX
DC voltage input	Stop	setVOLTMX
Thermocouple input	Stop	setTCMX
RTD input	Stop	setRTDMX
Digital input (DI)	Stop	setDIMX
Difference computation between channels	Stop	setDELTAMX
Remote RJC	Stop	setRRJCMX
Resistance input	Stop	setRESMX
Strain	Stop	setSTRAINMX
AO	Stop	setAOMX
PWM	Stop	setPWMMX
Set the unit name of the channel.	Stop	setScalingUnitMX
Set the channel tag.	Stop	setTagMX

Function	FIFO	Function
Set a comment for the channel.	Stop	setCommentMX
Set the alarm.	Stop	setAlarmMX
Set the RJC used on the channel.	Stop	setRJCTypeMX
Set the filter.	Stop	setFilterMX
Set the burnout detection action.	Stop	setBurnoutMX
Set the alarm to be assigned to the DO channel (To set a DO channel to alarm output, use the setDOTypeMX function.)	Stop	setRefAlarmMX
Set the scan interval.	Stop	setIntervalMX
Set the temperature unit.	Stop	setTempUnitMX
Set the ID number of the unit.	Stop	setUnitNoMX
Set the timeout value (the time from the disconnection to the start of saving to the CF card). For the calculation method of the timeout value, see appendix 3.	Stop	setSystemTimeoutMX
Set the signal type to be assigned to the DO channel.	Stop	setDOTypeMX
Set the AO channel type.	Stop	setAOTypeMX
Set the PWM channel type.	Stop	setPWMTTypeMX
Output channel data	Output types	Stop setOutputTypeMX
	Selected value	Stop setChoiceMX
	Pulse interval integer multiple	Stop setPulseTimeMX
Change a portion of the DO data.	Continue	changeDODataMX
Change a portion of the AO/PWM data changeAOPWMDataMX	Continue	
Change a portion of the initial balance data	Continue	changeBalanceMX
Change a portion of the transmission output data	Continue	changeTransmitMX

For a description of the FIFO column in the table, see the explanation given in “Other Controls” on the previous page. If an invalid data error occurs, the last detected item is saved as a setting item number. It can be retrieved with a utility.

Data Retrieval Functions

Retrieval of System Status Data and System Configuration Data

Function	Function
Get system status data.	getStatusDataMX
Get system configuration data.	getSystemConfigMX

Retrieval of Setup Data

Function	Function
Get the setup data collectively	getConfigDataMX
Declare the retrieval of the setup data.	talkConfigMX
Retrieves setup data other than the channel setup data.	
Get channel setup data.	getChConfigMX
Function used to retrieve channel setup data after declaring the retrieval of setup data using the talkConfigMX function.	

Retrieval of DO Data

Function	Function
Get the DO data collectively.	getDODataMX

Retrieval of AO/PWM Data and Transmission Output Data

Function	Function
Get AO/PWM data and transmission output data collectively.	getAOPWMDataMX

Retrieval of Channel Information Data

Function	Function
Declare the retrieval of the channel information data.	talkChInfoMX
Get channel information data collectively.	getChInfoMX

Retrieval of Measured Data (Channel Designation)

Function	Function
Get the most recent data range of the specified channel.	getChDataNoMX
Declare the retrieval of the measured data of the specified channel.	talkChDataMX
Declare the retrieval of the instantaneous values of the specified channel.	talkChDataInstMX
Get the time information of the specified channel for each data number.	getTimeDataMX
Get the measured data of the specified channel.	getChDataMX

Retrieval of Measured Data (FIFO Designation)

Function	Function
Get the most recent data range of the specified FIFO number.	getFIFODataNoMX
Declare the retrieval of the measured data of the specified FIFO number.	talkFIFODataMX
Declare the retrieval of the instantaneous values of the specified FIFO number.	talkFIFODataInstMX
Get the time information of the specified FIFO number for each data number.	getTimeDataMX
Get the measured data of the specified FIFO number.	getChDataMX

Measured data retrieval is possible only when the FIFO is running.

Retrieval of Initial Balance Data

Function	Function
Get initial balance data collectively.	getBalanceMX

Retrieve Output Channel Data

Function	Function
Get output channel data collectively.	getOutputMX

Utilities

Function	Function
Insert the specified user count (user-defined order information) in the next packet to be issued.	setUserTimeMX
Get the MX100-specific error that was received last through communications.	getLastErrorMX
Convert the measured value into double-precision floating point number.	toDoubleValueMX
Convert the measured value into string.	toStringValueMX
Get the alarm type string.	toAlarmNameMX getAlarmNameMX
Get the version number of this API.	getVersionAPIMX
Get the revision number of this API.	getRevisionAPIMX
Get the error message string.	getErrorMessageMX toErrorMessageMX
Get the maximum length of the error message string.	getMaxLenErrorMessageMX
Create data number equal to increment (specified data number) + (specified value).	incrementDataNoMX
Create data number equal to (specified data number).	decrementDataNoMX
Compare the two specified data numbers.	compareDataNoMX
Convert the time information to date and time.	toDateTimeMX
Get the number of the parameter on which an error was detected.	getItemErrorMX
Get the maximum length of the alarm string.	getMaxLenAlarmNameMX
AO/PWM Convert the output values to output data values.	toAOPWMValueMX
Convert the output data values to output values.	toRealValueMX
Check the validity of data numbers.	isDataNoMX
Convert to style version.	toStyleVersionMX

3.2 Program - MX100/Visual C -

Adding the Path to the Include File

Add the path of the include file (DAQMX.h) to the project. The method of adding the include file varies depending on the environment used.

Declaration in the Source File

Write the declaration in the source file.

```
#include "DAQMX.h"
```

Note

The include file of the common section (DAQHandler.h) is referenced from the include file described above. Thus, declaration for it is not necessary.

Load Library Statement

The statement below is added so that the executable module (.dll) of the API can link to the process.

The executable module (.dll) of the API is mapped within the address space (LoadLibrary). Next, the address of the export function in the executable module is retrieved (GetProcAddress).

The callback type of the function pointer is the function name with a prefix "DLL" added and converted to uppercase. It is defined in the include file of the API.

```
HMODULE pDll = LoadLibrary("DAQMX");  
DLLOPENMX openMX = (DLLOPENMX)GetProcAddress(pDll, "openMX");
```

Retrieval of the Measured Data

Program Example 1

This program retrieves measured data.

```

////////////////////////////////////
// MX100 sample for measurement
#include <stdio.h>
#include "DAQMX.h"
////////////////////////////////////
int main(int argc, char* argv[])
{
    int rc; //return code
    DAQMX comm; //discriptor
    int flag;
    MXDataNo startNo, endNo, dataNo;
    MXUserTime usertime;
    MXDateTime datetime;
    MXChInfo chinfo;
    MXDataInfo datainfo;
#ifdef WIN32
    HMODULE pDll; //DLL handle
    //callback
    DLLOPENMX openMX;
    DLLCLOSEMX closeMX;
    DLLSTARTFIFOMX startFIFOMX;
    DLLSTOPFIFOMX stopFIFOMX;
    DLLGETFIFODATANOMX getFIFODataNoMX;
    DLLTALKFIFODATAMX talkFIFODataMX;
    DLLGETTIMEDATAMX getTimeDataMX;
    DLLGETCHDATAMX getChDataMX;
    //load
    pDll = LoadLibrary("DAQMX");
    //get address
    openMX = (DLLOPENMX)GetProcAddress(pDll, "openMX");
    closeMX = (DLLCLOSEMX)GetProcAddress(pDll, "closeMX");
    startFIFOMX = (DLLSTARTFIFOMX)GetProcAddress(pDll,
"startFIFOMX");
    stopFIFOMX = (DLLSTOPFIFOMX)GetProcAddress(pDll,
"stopFIFOMX");
    getFIFODataNoMX = (DLLGETFIFODATANOMX)GetProcAddress(pDll,
"getFIFODataNoMX");
    talkFIFODataMX = (DLLTALKFIFODATAMX)GetProcAddress(pDll,
"talkFIFODataMX");
    getTimeDataMX = (DLLGETTIMEDATAMX)GetProcAddress(pDll,
"getTimeDataMX");
    getChDataMX = (DLLGETCHDATAMX)GetProcAddress(pDll,
"getChDataMX");
#endif //WIN32

```

```
//connect
comm = openMX("192.168.1.12" &rc);
//get by FIFO
rc = startFIFOMX(comm);
rc = getFIFODataNoMX(comm, 0, &startNo, &endNo);
rc = talkFIFODataMX(comm, 0, startNo, endNo);
do { //date time
    rc = getTimeDataMX(comm, &dataNo, &datetime, &usertime,
&flag);
} while (!(flag & DAQMX_FLAG_ENDDATA));
do { //measured data
    rc = getChDataMX(comm, &dataNo, &chinfo, &datainfo,
&flag);
} while (!(flag & DAQMX_FLAG_ENDDATA));
rc = stopFIFOMX(comm);
//disconenct
rc = closeMX(comm);
#ifdef WIN32
    FreeLibrary(pDll);
#endif
return rc;
}
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
```

Description

Overview

Data retrieval is possible by starting the FIFO. The range to be retrieved is specified by the FIFO number and the data number. The time stamp corresponding to the data number and the measured data are retrieved separately. The end is determined by the flag.

Include File Statement

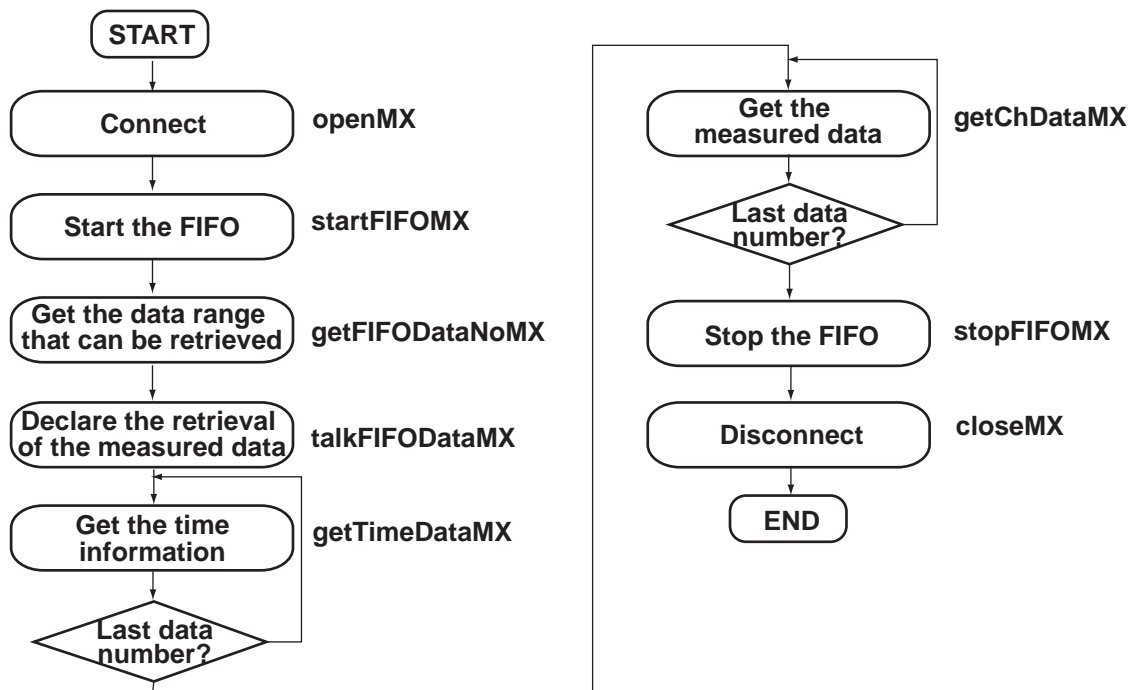
```
#include "DAQMX.h"
```

Load Library Statement

The load library statement is from #ifdef WIN32 to #endif //WIN32. A callback type (such as DLLOPENMX) is used.

Flow of the Process

The flow chart shown below omits the declaration section.



Communication Process

First, make a connection. After making the connection, the functions become available. As a termination procedure, disconnect the communication.

Note

- If there is no access for approximately three minutes, the MX100 drops the connection. Drop the connection if you are not accessing the MX100 for an extended time. Make the connection only when necessary.
- If you want to keep the connection open, run status retrieval.

Communication Connection

```
openMX("192.168.1.12" &rc)
```

The IP address of the MX100 is specified.

This statement implicitly specifies the communication constant for the communication port number of the MX100.

FIFO Start

```
startFIFOMX(comm)
```

Starts the FIFO.

Retrieval of Data Range

```
getFIFODataNoMX(comm, 0, &startNo, &endNo)
```

Retrieves the range from the next data following the data retrieved last to the most recent data of the specified FIFO number using data numbers.

Talker

`talkFIFODataMX(comm, 0, startNo, endNo)`

Specifies the data range and declares the retrieval of the FIFO data (measured data retrieval declaration).

Retrieval of the FIFO Data Time Information

`getTimeDataMX(comm, &dataNo, &datetime, &usertime, &flag)`

Gets the time information in the specified range in units of data numbers.

The end is determined by the flag status of "end data."

Note

usertime is the user-defined sequence information (user count). The value specified in advance using the `setUserTimeMX` function is returned.

Retrieval of FIFO Data

`getChDataMX(comm, &dataNo, &chinfo, &datainfo, &flag)`

Gets the measured data in the specified range in units of channels.

The end is determined by the flag status of "end data."

FIFO Stop

`stopFIFOMX(comm)`

Stops the FIFO.

Comm. cut

`closeMX(comm)`

Drops the connection.

Retrieval of Setup Data and Configuration

Program Example 2

This program executes the following three items. This program contains all four items, but each item can be written and executed separately.

- Get the setup data collectively
- Configure the setup data collectively.
- Set a DC voltage range to the channel

```

////////////////////////////////////
// MX100 sample for configuration
#include <stdio.h>
#include "DAQMX.h"
////////////////////////////////////
int main(int argc, char* argv[])
{
    int rc; //return code
    DAQMX comm; //discriptor
    //data
    MXConfigData configdata;
#ifdef WIN32
    HMODULE pDll; //DLL handle
    //callback
    DLLOPENMX openMX;
    DLLCLOSEMX closeMX;
    DLLGETCONFIGDATAMX getConfigDataMX;
    DLLSETCONFIGDATAMX setConfigDataMX;
    DLLSETVOLTMX setVOLTMX;
    //laod
    pDll = LoadLibrary("DAQMX");
    //get address
    openMX = (DLLOPENMX)GetProcAddress(pDll, "openMX");
    closeMX = (DLLCLOSEMX)GetProcAddress(pDll, "closeMX");
    getConfigDataMX = (DLLGETCONFIGDATAMX)GetProcAddress(pDll,
"getConfigDataMX");
    setConfigDataMX = (DLLSETCONFIGDATAMX)GetProcAddress(pDll,
"setConfigDataMX");
    setVOLTMX = (DLLSETVOLTMX)GetProcAddress(pDll, "setVOLTMX");
#endif //WIN32
    //connect
    comm = openMX("192.168.1.12" &rc);
    //get
    rc = getConfigDataMX(comm, &configdata);
    //set
    rc = setConfigDataMX(comm, &configdata);
    //range
    rc = setVOLTMX(comm, DAQMX_RANGE_VOLT_20MV, 1, 1, 0, 0, 0,
0, 0);
}

```

```
//disconnect
rc = closeMX(comm);
#ifdef WIN32
FreeLibrary(pDll);
#endif
return rc;
}
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
```

Description

Collective Retrieval of Setup Data

```
getConfigDataMX(comm, &configdata)
```

The setup data below can be retrieved by the collective retrieval of setup data. For details on the items that can be retrieved, see section 6.4, "MX100 Types."

- System configuration data: Stored in the MXSystemInfo structure.
- Status: Stored in the MXStatus structure.
- Basic settings: Basic settings of the system.

Note

The setup data can also be retrieved using the talkConfigMX function and the getChConfigMX function. The talkConfigMX function is used to declare the retrieval of the setup data and retrieve the system configuration data, status, and network information data. Then, the getChConfigMX function is used to retrieve channel setup data in units of channels.

Collective Setting of Setup Data

```
setConfigDataMX(comm, &configdata)
```

The data below can be set by the collective setting of setup data. For details on the items that can be set, see section 6.4, "MX100 Types."

- System configuration data: Contents of the MXSystemInfo structure.
- Basic settings: Basic settings of the system.

Setting a DC voltage range to the channel

```
setVOLTMX(comm, DAQMX_RANGE_VOLT_20MV, 1, 1, 0, 0, 0, 0, 0)
```

Sets channel number 1 to DC voltage range 20 mV. Scaling is not used.

Error Processing

- Most functions return the result of the function process using an error number (0 if successful).
- The function getErrorMessageMX can be used to get the error message string corresponding to the error number. The function getMaxLenErrorMessageMX can be used to get the maximum length of the error message string.
- The function getLastErrorMX can be used to get the errors from the MX100.
- If an invalid data error occurs in the settings, the setting item number of the detected error is retrieved by the function.

4.1 Functions and Their Functionalities - MX100/ Visual Basic -

This section indicates the correspondence between the functionalities that the API supports and the Visual Basic functions.

Communication Functions

Function	Function
Connect to the MX100.	openMX
Disconnect from the MX100.	closeMX
Set the communication timeout.	setTimeoutMX

Note

Setting of the communication timeout is not recommended because unexpected disconnection may occur due to the conflict with the timeout time when data is retrieved.

Control Functions

Starting/Stopping the FIFO

Function	Function
Start the FIFO	startFIFOMX
Stop the FIFO	stopFIFOMX
Set auto control of the FIFO.	autoFIFOMX

Other Controls

Function	FIFO	Function
Set the number of seconds from the reference date/time (Jan. 1, 1970) on the MX100.	Stop	setDateTimeMX
Set the MX100 to the current date/time.	Stop	setDateTimeNowMX
Turn ON/OFF data saving to the CF card (data backup).	Continue	setBackupMX
Format the CF card.	Stop	formatCFMX
• Reconfigure the system of the unit.	Stop	initSystemMX
• Initialize the system of the unit.	Stop	
• Reset alarms (alarm ACK) of the unit.	Continue	
Set the 7-segment LED display.	Continue	setSegmentMX

The FIFO column in the table indicates the FIFO operation when the function is executed while FIFO is running.

Stop: Stops the FIFO when the function is executed.

Continue: Continues the FIFO even when the function is executed.

With the backup settings, if the CF write mode is changed, the FIFO stops. The CF write mode for backup settings performs a setting change (modifies collectively acquired data and sends it collectively).

Note

If the auto control of the FIFO is enabled, the FIFO is automatically resumed when the FIFO is stopped due to an execution of a function.

Functionality

Collective Setup

Function	FIFO	Function
Set the setup data (system setup data).	Stop	setSystemConfigMX
Configure the setup data (channel setup data).	Stop	setChConfigMX
Set the DO (Digital Output) data collectively.	Continue	setDODataMX
Transmit AO/PWM data	Continue	setAOPWMDataMX
Send transmission Output Data	Continue	setTransmitMX

For a description of the FIFO column in the table, see the explanation given in “Other Controls” on the previous page. Setup data cannot be handled collectively. If an invalid data error occurs, the last detected item is saved as a setting item number. It can be retrieved with a utility.

Individual Setup

Function	FIFO	Function
Set initial balance data	Stop	setBalanceMX
Set output channel data	Stop	setOutputMX
Initial balance data	Execute	runBalanceMX
	Reset	resetBalanceMX

Setup Change

Function	FIFO	Function
Range	SKIP (not used)	setSKIPMX
Settings	DC voltage input	setVOLTMX
	Thermocouple input	setTCMX
	RTD input	setRTDMX
	Digital input (DI)	setDIMX
	Difference computation between channels	setDELTAMX
	Remote RJC	setRRJCMX
	Resistance	setRESMX
	Strain	setSTRAINMX
	AO	setAOMX
	PWM	setPWMMX
Set the unit name of the channel.	Stop	setScalingUnitMX
Set the channel tag.	Stop	setTagMX
Set a comment for the channel.	Stop	setCommentMX
Set the alarm.	Stop	setAlarmMX
Set the RJC used on the channel.	Stop	setRJCTypeMX

Function	FIFO	Function	
Set the filter.	Stop	setFilterMX	
Set the burnout detection action.	Stop	setBurnoutMX	
Set the alarm to be assigned to the DO channel to which an alarm output was specified. (To set a DO channel to alarm output, use the setDOTypeMX function.)	Stop	setRefAlarmMX	
Set the scan interval.	Stop	setIntervalMX	
Set the temperature unit.	Stop	setTempUnitMX	
Set the ID number of the unit.	Stop	setUnitNoMX	
Set the timeout value (the time from the disconnection to the start of saving to the CF card. For the calculation method of the timeout value, see appendix 3.	Stop	setSystemTimeoutMX	
Set the signal type to be assigned to the DO channel.	Stop	setDOTypeMX	
Sets the AO channel type.	Stop	setAOTypeMX	
Set the PWM type for the channel	Stop	setPWMTypeMX	
Output channel data	Output types	Stop	setOutputTypeMX
	Selected value	Stop	setChoiceMX
	Pulse interval integer multiple	Stop	setPulseTimeMX
Change a portion of the DO data.	Continue	changeDODataMX	
Change a portion of the AO/PWM data	Continue	changeAOPWMDataMX	
Change a portion of the initial balance data	Continue	changeBalanceMX	
Change a portion of the transmission output data	Continue	changeTransmitMX	

Automatically sent individually. The FIFO stops.

For a description of the FIFO column in the table, see the explanation given in “Other Controls” on the previous page.

Data Retrieval Functions

Retrieval of System Status Data and System Configuration Data

Function	Function
Get system status data.	getStatusDataMX
Get system configuration data.	getSystemConfigMX

Retrieval of Setup Data

Function	Function
Declare the retrieval of the setup data.	talkConfigMX
Retrieve setup data other than the channel setup data.	
Get channel setup data.	getChConfigMX
Function used to retrieve channel setup data after declaring the retrieval of setup data using the talkConfigMX function.	

Retrieval of DO data

Function	Function
Get the DO data collectively.	getDODataMX

Retrieval of AO/PWM Data and Transmission Output Data

Function	Function
Get AO/PWM data and transmission output data collectively.	getAOPWMDataMX

Retrieval of Channel Information Data

Function	Function
Declare the retrieval of the channel information data.	talkChInfoMX
Get channel information data	getChInfoMX

Retrieval of Measured Data (Channel Designation)

Function	Function
Get the most recent data range of the specified channel.	getChDataNoMX
Declare the retrieval of the measured data of the specified channel.	talkChDataVBMX
Declare the retrieval of the instantaneous values of the specified channel.	talkChDataInstMX
Get the time information of the specified channel for each data number.	getTimeDataMX
Get the measured data of the specified channel.	getChDataMX

Retrieval of Measured Data (FIFO Designation)

Function	Function
Get the most recent data range of the specified FIFO number.	getFIFODataNoMX
Declare the retrieval of the measured data of the specified FIFO number.	talkFIFODataVBMX
Declare the retrieval of the instantaneous values of the specified FIFO number.	talkFIFODataInstMX
Get the time information of the specified FIFO number for each data number.	getTimeDataMX
Get the measured data of the specified FIFO number.	getChDataMX

Measured data retrieval is possible only when the FIFO is running.

Initial Balance Data

Function	Function
Get initial balance data collectively	getBalanceMX

Retrieve output channel data

Function	Function
Get output channel data collectively	getOutputMX

Utilities

Function	Function
Insert the specified user count (user-defined order information) in the next packet to be issued.	setUserTimeVBMX
Get the MX100-specific error that was received last through communications.	getLastErrorMX
Convert the measured value into double-precision floating point number.	toDoubleValueMX
Convert the measured value into a string.	toStringValueMX
Get the alarm type string.	toAlarmNameMX
Get the version number of this API.	getVersionAPIMX
Get the revision number of this API.	getRevisionAPIMX
Get the error message string.	toErrorMessageMX
Get the maximum length of the error message string.	getMaxLenErrorMessageMX
Create data number equal to (specified data number) + (specified value).	incrementDataNoMX
Create data number equal to (specified data number) - (specified value).	decrementDataNoMX
Compare the two specified data numbers.	compareDataNoMX
Convert the time information to date and time.	toDateTimeMX
Get the number of the parameter on which an error was detected	getItemErrorMX
Get the maximum length of the alarm string.	getMaxLenAlarmNameMX
AO/PWM Convert the output values to output data values.	toAOPWMValueMX
Convert the output data values to output values.	toRealValueMX
Check the validity of data numbers.	isDataNoVBMX
Convert to style version.	toStyleVersionMX

4.2 Programming - MX100/Visual Basic -

Declaration of Types, Functions, and Constants

To use types, functions, and constants for Visual Basic, they must be declared in advance. The following methods of declaration statements are available.

Statement of All Declarations

Adding the standard module library file for Visual Basic (DAQMX.bas) to the project is equivalent to declaring all types, functions, and constants.

Statement of Selective Declarations

The API Viewer that comes with Visual Studio can be used to copy the declaration statements of arbitrary types, functions, and constants. Load the text file for the API Viewer (DAQMX.txt) on the API Viewer to use this function.

For a description of how to use the API Viewer, read the operation manual for Visual Studio.

Writing Declarations Directly

Below is an example of a declaration statement.

```
Public Declare Function openMX Lib "DAQMX" (ByVal strAddress As String, ByRef errorCode As Long) As Long
```


Retrieval of the Measured Data

Program Example 1

This program retrieves measured data.

```
Public Function main()
Dim startNo As MXDataNo
Dim endNo As MXDataNo
Dim dataNo As MXDataNo
Dim usertime As MXUserTime
Dim datetime As MXDateTime
Dim chinfo As MXChInfo
Dim datainfo As MXDataInfo
'connect
host = "192.168.1.12"comm = openMX(host, rc)
'get FIFO
rc = startFIFOMX(comm)
rc = getFIFODataNoMX(comm, 0, startNo, endNo)
rc = talkFIFODataVBMX(comm, 0, startNo, endNo)
Do
    rc = getTimeDataMX(comm, dataNo, datetime, usertime, flag)
Loop While (flag And DAQMX_FLAG_ENDDATA) = 0
Do
    rc = getChDataMX(comm, dataNo, chinfo, datainfo, flag)
Loop While (flag And DAQMX_FLAG_ENDDATA) = 0
rc = stopFIFOMX(comm)
'disconnect
rc = closeMX(comm)
End Function
```

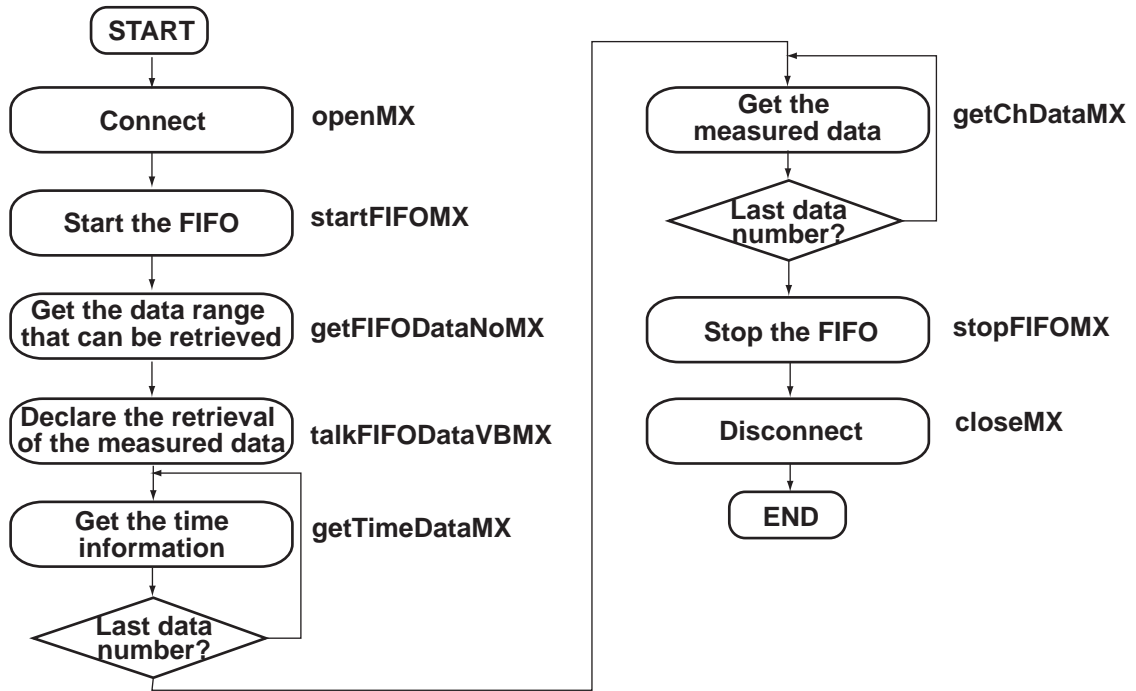
Description

Overview

Data retrieval is possible by starting the FIFO. The range to be retrieved is specified by the FIFO number and the data number. The time stamp corresponding to the data number and the measured data are retrieved separately. The end is determined by the flag.

Flow of the Process

The flow chart shown below omits the declaration section.



Communication Process

First, make a connection. After making the connection, the functions become available. As a termination procedure, disconnect the communication.

Note

If there is no access for approximately three minutes, the MX100 drops the connection. Drop the connection if you are not accessing the MX100 for an extended time. Make the connection only when necessary. If you want to keep the connection open, run status retrieval.

Communication Connection

```
openMX(host, rc)
```

The IP address of the MX100 is specified.

This statement implicitly specifies the communication constant for the communication port number of the MX100.

FIFO Start

```
startFIFOMX(comm)
```

Starts the FIFO.

Retrieval of Data Range

```
getFIFODataNoMX(comm, 0, startNo, endNo)
```

Retrieves the range from the next data following the data retrieved last to the most recent data of the specified FIFO number using data numbers.

Talker

```
talkFIFODataVBMX(comm, 0, startNo, endNo)
```

Specifies the data range and declares the retrieval of the FIFO data (measured data retrieval declaration).

Retrieval of the FIFO Data Time Information

```
getTimeDataMX(comm, dataNo, datetime, usertime, flag)
```

Gets the time information in the specified range in units of data numbers.

The end is determined by the flag status of "end data."

Note

usertime is the user-defined sequence information (user count). The value specified in advance using the setUserTimeVBMX function is returned.

Retrieval of FIFO Data

```
getChDataMX(comm, dataNo, chinfo, datainfo, flag)
```

Gets the measured data in the specified range in units of channels.

The end is determined by the flag status of "end data."

FIFO Stop

```
stopFIFOMX(comm)
```

Stops the FIFO.

Comm. cut

```
closeMX(comm)
```

Drops the connection.

Retrieval of Setup Data and Configuration

Program Example 2

This program executes the following four items. This program contains all four items, but each item can be written and executed separately.

- Get setup data.
- Set setup data other than channel setup data collectively.
- Set the channel setup data
- Set a DC voltage range to the channel

```
Public Function main()  
Dim sysinfo As MXSystemInfo  
Dim status As MXStatus  
Dim netinfo As MXNetInfo  
Dim chconfig As MXChConfig  
'connect  
host = "192.168.1.12" comm = openMX(host, rc)  
'get  
rc = talkConfigMX(comm, sysinfo, status, netinfo)  
Do  
    rc = getChConfigMX(comm, chconfig, flag)  
Loop While (flag And DAQMX_FLAG_ENDDATA) = 0  
'set  
rc = setSystemConfigMX(comm, sysinfo)  
rc = setChConfigMX(comm, chconfig)  
'range  
rc = setVOLTMX(comm, DAQMX_RANGE_VOLT_20MV, 1, 1, 0, 0, 0, 0,  
0)  
'disconnect  
rc = closeMX(comm)  
End Function
```

Description

Retrieval of Setup Data

```
talkConfigMX(comm, sysinfo, status, netinfo)
```

Declares the retrieval of the setup data.

Gets the setup data (system configuration data, status, and network information data) excluding the channel setup data.

The system configuration data, the status, and the network information data are stored in the MXSystemInfo, MXStatus, and MXNetInfo structures, respectively.

```
getChConfigMX(comm, chconfig, flag)
```

Gets the channel setup data for each channel.

The end is determined by the flag status of "end data."

Collective Setting of Setup Data Other Than Channel Setup Data

```
setSystemConfigMX(comm, sysinfo)
```

The contents of the specified MXSystemInfo structure are assigned to the MX100.

Setting of the Channel Setup Data

```
setChConfigMX(comm, chconfig)
```

The contents of the specified MXChConfig structure are assigned to the MX100.

Setting a DC Voltage Range to the Channel

```
setVOLTMX(comm, DAQMX_RANGE_VOLT_20MV, 1, 1, 0, 0, 0, 0, 0)
```

Sets channel number 1 to DC voltage range 20 mV. Scaling is not used.

Error Processing

- Most functions return the result of the function process using an error number (0 if successful).
- The function toErrorMessageMX can be used to get the error message string corresponding to the error number. The function getMaxLenErrorMessageMX can be used to get the maximum length of the error message string.
- The function getLastErrorMX can be used to get the errors from the MX100.
- If an invalid data error occurs in the settings, the setting item number of the detected error is retrieved by the function.

5.1 Details of Functions - MX100 (Visual C/Visual Basic) -

This section describes the MX100 functions that are used in C and Visual Basic. The functions are listed in alphabetical order by the function name.

For details on constants and types, see chapter 6.

For information about the MX100, see appendix 1.

Most functions return an error number as a return value. Error number 0 is returned if there is no error.

autoFIFOMX

Syntax

```
int autoFIFOMX(DAQMX daqmx, int bAuto);
```

Declaration

```
Public Declare Function autoFIFOMX Lib "DAQMX" (ByVal daqmx As Long, ByVal bAuto As Long) As Long
```

Parameters

daqmx	Specify the device descriptor.
bAuto	Specify the boolean value.

Description

Sets auto control of the FIFO.

Return value

Returns an error number.

Error:

Not descriptor No device descriptor.

Reference

CDAQMX::autoFIFO

changeAOPWMDataMX

Syntax

```
int changeAOPWMDataMX(MXAOPWMData * pMXAOPWMData, int aopwmNo,
int bValid, int iAOPWMValue);
```

Declaration

```
Public Declare Function changeAOPWMDataMX Lib "DAQMX" (ByRef
pMXAOPWMData As MXAOPWMData, ByVal aopwmNo As Long, ByVal
bValid As Long, ByVal iAOPWMValue As Long) As Long
```

Parameters

cMXAOPWMData	Specify AO/PWM data.
aopwmNo	Specify the AO/PWM data number.
bValid	Specify the boolean value.
iAOPWMValue	Specify the output data value.

Description

Changes the data of the specified AO/PWM data.

- If the constant for “Specify all AO/PWM numbers” is specified for the AO data numbers, all AO data are changed.
- If the AO/PWM data number is outside of the range and invalid, the specification is discarded.
- The output data specifies the changed value of the actually output value.
- Completes normally.

Return value

Returns an error number.

Reference

CDAQMXAOPWMData::getMXAOPWMData

CDAQMXAOPWMData::setAOPWM

changeBalanceMX

Syntax

```
int changeBalanceMX(MXBalanceData * pMXBalanceData, int balanceNo, int bValid, int iValue);
```

Declaration

```
Public Declare Function changeBalanceMX Lib "DAQMX" (ByRef pMXBalanceData As MXBalanceData, ByVal balanceNo As Long, ByVal bValid As Long, ByVal iValue As Long) As Long
```

Parameters

pMXBalanceData	Specify initial balance data.
balanceNo	Specify the initial balance data number.
bValid	Specify the boolean value.
iValue	Specify the initial balance value.

Description

Changes the data of the specified initial balance data.

- If the constant for “Specify all initial balance numbers” is specified for the initial balance data numbers, all data are changed.
- If the initial balance data number is outside of the range and invalid, the specification is discarded.
- Completes normally.

Return value

Returns an error number.

Reference

```
CDAQMXBalanceData::getMXBalanceData  
CDAQMXBalanceData::setBalance
```

changeDODataMX

Syntax

```
int changeDODataMX(MXDOData * pMXDOData, int doNo, int bValid,  
int bONOFF);
```

Declaration

```
Public Declare Function changeDODataMX Lib "DAQMX" (ByRef  
pMXDOData As MXDOData, ByVal doNo As Long, ByVal bValid As  
Long, ByVal bONOFF As Long) As Long
```

Parameters

pMXDOData	Specify the DO data.
doNo	Specify the data number.
bValid	Specify the boolean value.
bONOFF	Specify the boolean value.

Description

Changes the data of the DO data number of the specified DO data.

- When DO data is transmitted, the value specified by parameter bONOFF is output for the DO data that is specified as “valid” by parameter bValid.
- If the DO data number is invalid, the specification is discarded.
- Completes normally.

Return value

Returns an error number.

Reference

```
CDAQMXDOData::setDO  
CDAQMXDOData::CDAQMXDOData
```

changeTransmitMX

Syntax

```
int changeTransmitMX(MXTransmit * pMXTransmit, int aopwmNo,  
int iTrans);
```

Declaration

```
Public Declare Function changeTransmitMX Lib "DAQMX" (ByRef  
pMXTransmit As MXTransmit, ByVal pwmNo As Long, ByVal iTrans  
As Long) As Long
```

Parameters

pMXTransmit	Specify the transmission output data.
aopwmNo	Specify the AO/PWM data number.
iTrans	Specify the transmission status.

Description

Changes the data of the specified transmission output data.

- If the constant for “Specify all AO/PWM data numbers” is specified for the AO/PWM data numbers, all data are changed.
- If the AO/PWM data number is outside of the range and invalid, the specification is discarded.
- Completes normally.

Return value

Returns an error number.

Reference

```
CDAQMXTransmit::getMXTransmit  
CDAQMXTransmit::setTransmit
```

closeMX

Syntax

```
int closeMX(DAQMX daqmx);
```

Declaration

```
Public Declare Function closeMX Lib "DAQMX" (ByVal daqmx As Long) As Long
```

Parameters

daqmx Specify the device descriptor.

Description

Disconnects the communication using the specified device descriptor.

When the communication is disconnected, the value of the device descriptor is meaningless. Do not use the value as a device descriptor.

Return value

Returns an error number.

Error:

Not descriptor No device descriptor.

Reference

CDAQMX::close

compareDataNoMX

Syntax

```
int compareDataNoMX(MXDataNo * prevDataNo, MXDataNo *  
nextDataNo);
```

Declaration

```
Public Declare Function compareDataNoMX Lib "DAQMX" (ByRef  
prevDataNo As MXDataNo, ByRef nextDataNo As MXDataNo) As Long
```

Parameters

prevDataNo Specify the data number (previous).
nextDataNo Specify the data number (next).

Description

Compares the specified data numbers.

- The data number is 64 bits while the return value is 32 bits. Therefore, the returned value is not the difference between the two.
- In the case of Visual C, if NULL is specified for a parameter, an indefinite value is returned.

Return value

Returns 0 if the data numbers are the same.

Returns a positive number if the data number (previous) is less than the data number (next).

Returns a negative number if the data number (previous) is greater than the data number (next).

decrementDataNoMX

Syntax

```
void decrementDataNoMX(MXDataNo * dataNo, int decrement);
```

Declaration

```
Public Declare Sub decrementDataNoMX Lib "DAQMX" (ByRef dataNo  
As MXDataNo, ByVal decrement As Long)
```

Parameters

dataNo	Specify the data number.
decrement	Specify the value to decrement by.

Description

Decrements the specified data number by the specified amount. The field of the specified data number is changed.

formatCFMX

Syntax

```
int formatCFMX(DAQMX daqmx);
```

Declaration

```
Public Declare Function formatCFMX Lib "DAQMX" (ByVal daqmx As Long) As Long
```

Parameters

daqmx Specify the device descriptor.

Description

Formats the CF (Compact Flash) card.

- The response to this function may take several seconds or longer. The time required varies depending on the medium used.

Return value

Returns an error number.

Error:

Not descriptor No device descriptor.

Reference

CDAQMX::formatCF

getAlarmNameMX **[Visual C only]**

Syntax

```
const char * getAlarmNameMX(int iAlarmType);
```

Parameters

iAlarmType Specify the alarm type.

Description

Gets the string corresponding to the specified alarm type.

- In Visual Basic, use the toAlarmNameMX function.

Return value

Returns a pointer to the string.

Reference

CDAQMXDataInfo::getAlarmName

getAOPWMDataMX

Syntax

```
int getAOPWMDataMX(DAQMX daqmx, MXAOPWMData * pMXAOPWMData,  
MXTransmit * pMXTransmit);
```

Declaration

```
Public Declare Function getAOPWMDataMX Lib "DAQMX" (ByVal daqmx  
As Long, ByRef pMXAOPWMData As MXAOPWMData, ByRef pMXTransmit  
As MXTransmit) As Long
```

Parameters

daqmx	Specify the device descriptor.
pMXAOPWMData	Specify the destination where the AO/PWM data is to be returned.
pMXTransmit is	Specify the destination where the transmission output data is to be returned.

Description

Gets AO/PWM data and transmission output data collectively.

- Stores retrieved data if the return destination is specified.

Return value

Returns an error number.

Error:

Not descriptor	No device descriptor.
----------------	-----------------------

Reference

```
CDAQMX::getAOPWMData  
CDAQMXAOPWMData::getMXAOPWMData  
CDAQMXTransmit::getMXTransmit
```

getBalanceMX

Syntax

```
int getBalanceMX(DAQMX daqmx, MXBalanceData * pMXBalanceData);
```

Declaration

```
Public Declare Function getBalanceMX Lib "DAQMX" (ByVal daqmx  
As Long, ByRef pMXBalanceData As MXBalanceData) As Long
```

Parameters

daqmx	Specify the device descriptor.
pMXBalanceData	Specify the destination where the initial balance data is to be returned.

Description

Gets initial balance data collectively.

- Stores initial balance data to the specified location if the return destination is specified.

Return value

Returns an error number.

Error:

Not descriptor	No device descriptor.
----------------	-----------------------

Reference

CDAQMX::getBalance

CDAQMXBalanceData::getMXBalanceData

getChConfigMX

Syntax

```
int getChConfigMX(DAQMX daqmx, MXChConfig * pMXChConfig, int * pFlag);
```

Declaration

```
Public Declare Function getChConfigMX Lib "DAQMX" (ByVal daqmx As Long, ByRef pMXChConfig As MXChConfig, ByRef pFlag As Long) As Long
```

Parameters

daqmx	Specify the device descriptor.
pMXChConfig	Specify the destination where the channel setup data is to be returned.
pFlag	Specify the destination where the flag is to be returned.

Description

Gets the channel setup data that was declared to be retrieved using the talkConfigMX function in units of channels.

- When the last set of data is retrieved, the flag status is set.
- Do not perform communications using other functions until the data retrieval is completed. Other functions may not operate properly.

Return value

Returns an error number.

Error:

Not descriptor No device descriptor.

Reference

CDAQMX::getChConfig
CDAQMXChConfig::getMXChConfig

getChDataMX

Syntax

```
int getChDataMX(DAQMX daqmx, MXDataNo * pMXDataNo, MXChInfo *
pMXChInfo, MXDataInfo * pMXDataInfo, int * pFlag);
```

Declaration

```
Public Declare Function getChDataMX Lib "DAQMX" (ByVal daqmx As
Long, ByRef pMXDataNo As MXDataNo, ByRef pMXChInfo As
MXChInfo, ByRef pMXDataInfo As MXDataInfo, ByRef pFlag As
Long) As Long
```

Parameters

daqmx	Specify the device descriptor.
pMXDataNo	Specify the destination where the data number is to be returned.
pMXChInfo	Specify the destination where the channel information data is to be returned using the MXChInfo structure.
pMXDataInfo	Specify the destination where the measured data is to be returned using the MXDataInfo structure.
pFlag	Specify the destination where the flag is to be returned.

Description

After declaring with a data retrieval start function (talkChDataMX, talkFIFODataMX, etc.), retrieve time information data with the getTimeDataMX function and then retrieve the measured data, data by data.

- When the last set of data is retrieved, the flag status is set.
- Do not perform communications using other functions until the data retrieval is completed. Other functions may not operate properly.

Return value

Returns an error number.

Error:

Not descriptor No device descriptor.

Reference

```
CDAQMX::getChData
CDAQMXChInfo::getMXChInfo
CDAQMXDataInfo::getMXDataInfo
```

getChDataNoMX

Syntax

```
int getChDataNoMX(DAQMX daqmx, int chNo, MXDataNo *  
startDataNo, MXDataNo * endDataNo);
```

Declaration

```
Public Declare Function getChDataNoMX Lib "DAQMX" (ByVal daqmx  
As Long, ByVal chNo As Long, ByRef startDataNo As MXDataNo,  
ByRef endDataNo As MXDataNo) As Long
```

Parameters

daqmx	Specify the device descriptor.
chNo	Specify the channel number.
startDataNo	Specify the destination where the start data number is to be returned.
endDataNo	Specify the destination where the end data number is to be returned.

Description

Gets the range of measured data that can be retrieved at the specified channel.

- The start data number is set to the next number following the data number that was retrieved last.
- If the measured data that must be retrieved does not exist, the negative of each data number is returned.

Example:

If data numbers from 10 to 49 exist on the main unit and the last retrieved data number was 29, a start data number of 30 and an end data number of 49 is returned.

If the last retrieved data number was 49, the negative of each data number is returned, and the function ends normally.

- For the validity check of the returned data, see the isDataNoMX function.

Return value

Returns an error number.

Error:

Not descriptor No device descriptor.

Reference

CDAQMX::getChDataNo

getChInfoMX

Syntax

```
int getChInfoMX(DAQMX daqmx, MXChInfo * pMXChInfo, int *
pFlag);
```

Declaration

```
Public Declare Function getChInfoMX Lib "DAQMX" (ByVal daqmx As
Long, ByRef pMXChInfo As MXChInfo, ByRef pFlag As Long) As
Long
```

Parameters

daqmx	Specify the device descriptor.
pMXChInfo	Specify the destination where the channel information data is to be returned.
pFlag	Specify the destination where the flag is to be returned.

Description

Gets the channel information data that was declared to be retrieved using the talkChInfoMX function in units of channels.

- When the last set of data is retrieved, the flag status is set.
- Do not perform communications using other functions until the data retrieval is completed. Other functions may not operate properly.

Return value

Returns an error number.

Error:

Not descriptor No device descriptor.

Reference

CDAQMX::getChInfo

getConfigDataMX [Visual C only]

Syntax

```
int getConfigDataMX(DAQMX daqmx, MXConfigData *  
pMXConfigData);
```

Parameters

daqmx Specify the device descriptor.
pMXConfigData Specify the destination where the setup data is to be returned.

Description

Gets the setup data collectively.

- Stores setup data in the specified location if the return destination is specified.
- In the case of Visual Basic, retrieve data collectively for each data structure individually. Or, retrieve them one after another.

Return value

Returns an error number.

Error:

Not descriptor No device descriptor.

Reference

CDAQMX::getConfig

CDAQMXConfig::getMXConfigData

getDODataMX

Syntax

```
int getDODataMX(DAQMX daqmx, MXDOData * pMXDOData);
```

Declaration

```
Public Declare Function getDODataMX Lib "DAQMX" (ByVal daqmx As Long, ByRef pMXDOData As MXDOData) As Long
```

Parameters

daqmx Specify the device descriptor.
pMXDOData Specify the destination where the DO data is to be returned.

Description

Gets the DO data collectively.

Stores DO data in the specified location if the return destination is specified.

Return value

Returns an error number.

Error:

Not descriptor No device descriptor.

Reference

CDAQMX::getDOData

CDAQMXDOData::getMXDOData

getErrorMessageMX

[Visual C only]

Syntax

```
const char * getErrorMessage(int errorCode);
```

Parameters

errorCode Specify the error number.

Description

Gets the error message string corresponding to the error number.

In Visual Basic, use the toErrorMessageMX function.

Return value

Returns the length of the string.

Reference

CDAQMX::getErrorMessage

getFIFODataNoMX

Syntax

```
int getFIFODataNoMX(DAQMX daqmx, int fifoNo, MXDataNo *
startDataNo, MXDataNo * endDataNo);
```

Declaration

```
Public Declare Function getFIFODataNoMX Lib "DAQMX" (ByVal
daqmx As Long, ByVal fifoNo As Long, ByRef startDataNo As
MXDataNo, ByRef endDataNo As MXDataNo) As Long
```

Parameters

daqmx	Specify the device descriptor.
fifoNo	Specify the FIFO number.
startDataNo	Specify the destination where the start data number is to be returned.
endDataNo	Specify the destination where the end data number is to be returned.

Description

Gets the range of measured data that can be retrieved at the specified FIFO number.

The start data number is set to the next number following the data number that was retrieved last.

- The start data number is set to the next number following the data number that was retrieved last.
- If the measured data that must be retrieved does not exist, the negative of each data number is returned.

Example:

If data numbers from 10 to 49 exist on the main unit and the last retrieved data number was 29, a start data number of 30 and an end data number of 49 is returned.

If the last retrieved data number was 49, the negative of each data number is returned, and the function ends normally.

- For the validity check of the returned data, see the isDataNoMX function.

Return value

Returns an error number.

Error:

Not descriptor No device descriptor.

Reference

CDAQMX::getFIFODataNo

getItemErrorMX

Syntax

```
int getItemErrorMX(DAQMX daqmx, int * itemErr);
```

Declaration

```
Public Declare Function getItemErrorMX Lib "DAQMX" (ByVal daqmx  
As Long, ByRef itemErr As Long) As Long
```

Parameters

daqmx	Specify the device descriptor.
itemErr	Specify the destination where the setting item number is to be returned.

Description

Gets the number of the parameter on which an error was last detected.

- Returns the setting item number to the specified return destination.

Return value

Returns an error number.

Error:

Not descriptor	No device descriptor.
----------------	-----------------------

Reference

CDAQMX::getItemError

getLastErrorMX

Syntax

```
int getLastErrorMX(DAQMX daqmx, int * lastErr);
```

Declaration

```
Public Declare Function getLastErrorMX Lib "DAQMX" (ByVal daqmx  
As Long, ByRef lastErr As Long) As Long
```

Parameters

daqmx	Specify the device descriptor.
lastErr	Specify the destination where the MX100-specific error is to be returned.

Description

Gets the MX100-specific error that was received last through communications.

Return value

Returns an error number.

Error:

Not descriptor	No device descriptor.
----------------	-----------------------

Reference

getMaxLenAlarmNameMX

Syntax

```
int getMaxLenAlarmNameMX(void);
```

Declaration

```
Public Declare Function getMaxLenAlarmNameMX Lib "DAQMX"() As  
Long
```

Description

Gets the maximum length of the alarm type.

- The return value does not include the terminator.

Return value

Returns the length of the string.

Reference

CDAQMXDataInfo::getMaxLenAlarmName

getMaxLenErrorMessageMX

Syntax

```
int getMaxLenErrorMessageMX(void);
```

Declaration

```
Public Declare Function getMaxLenErrorMessageMX Lib "DAQMX"()  
As Long
```

Description

Gets the maximum length of the error message string.

- The return value does not include the terminator.

Return value

Returns the length of the string.

Reference

CDAQMX::getMaxLenErrorMessage

getOutputMX

Syntax

```
int getOutputMX(DAQMX daqmx, MXOutputData * pMXOutputData);
```

Declaration

```
Public Declare Function getOutputMX Lib "DAQMX" (ByVal daqmx As Long, ByRef pMXOutputData As MXOutputData) As Long
```

Parameters

daqmx	Specify the device descriptor.
pMXOutputData	Specify the destination where the output channel data is to be returned.

Description

Gets the output channel data.

- Stores retrieved data if the return destination is specified.

Return value

Returns an error number.

Error:

Not descriptor	No device descriptor.
----------------	-----------------------

Reference

```
CDAQMX::getMXConfig  
CDAQMXConfig::getClassMXOutputData  
CDAQMXOutputData::getMXOutputData  
CDAQMXOutputData::setMXOutputData
```

getRevisionAPIMX

Syntax

```
const int getRevisionAPIMX(void);
```

Declaration

```
Public Declare Function getRevisionAPIMX Lib "DAQMX"() As Long
```

Description

Gets the revision number of this API.

Return value

Returns the revision number of this API as an integer value.

Reference

```
CDAQMX::getRevisionAPIMX
```


getStatusDataMX

Syntax

```
int getStatusDataMX(DAQMX daqmx, MXStatus * pMXStatus);
```

Declaration

```
Public Declare Function getStatusDataMX Lib "DAQMX" (ByVal  
daqmx As Long, ByRef pMXStatus As MXStatus) As Long
```

Parameters

daqmx Specify the device descriptor.
pMXStatus Specify the destination where the status is to be returned.

Description

Gets the status.

- Stores the status in the specified location if the return destination is specified.

Return value

Returns an error number.

Error:

Not descriptor No device descriptor.

Reference

CDAQMX::getStatusData

CDAQMXStatus::getMXStatus

getSystemConfigMX

Syntax

```
int getSystemConfigMX(DAQMX daqmx, MXSystemInfo * pSysInfo);
```

Declaration

```
Public Declare Function getSystemConfigMX Lib "DAQMX" (ByVal daqmx As Long, ByRef pSysInfo As MXSystemInfo) As Long
```

Parameters

daqmx	Specify the device descriptor.
pSysInfo	Specify the destination where the system configuration data is to be returned.

Description

Gets the system configuration data.

- Stores system configuration data in the specified location if the return destination is specified.

Return value

Returns an error number.

Error:

Not descriptor No device descriptor.

Reference

CDAQMX::getSystemConfig

CDAQMXSysInfo::getMXSystemInfo

getTimeDataMX

Syntax

```
int getTimeDataMX(DAQMX daqmx, MXDataNo * pMXDataNo,  
MXDateTime * pMXDateTime, MXUserTime * pMXUserTime, int *  
pFlag);
```

Declaration

```
Public Declare Function getTimeDataMX Lib "DAQMX" (ByVal daqmx  
As Long, ByRef pMXDataNo As MXDataNo, ByRef pMXDateTime As  
MXDateTime, ByRef pMXUserTime As MXUserTime, ByRef pFlag As  
Long) As Long
```

Parameters

daqmx	Specify the device descriptor.
pMXDataNo	Specify the destination where the data number is to be returned.
pMXDateTime	Specify the destination where the time information data is to be returned.
pMXUserTime	Specify the destination where the user count is to be returned.
pFlag	Specify the destination where the flag is to be returned.

Description

Gets the time information data that was declared to be retrieved using a data retrieval start function (the talkChDataMX or talkFIFODataMX functions) one data number at a time.

- When the last set of data is retrieved, the flag status is set.
- Do not perform communications using other functions until the data retrieval is completed. Other functions may not operate properly.

Return value

Returns an error number.

Error:

Not descriptor No device descriptor.

Reference

CDAQMX::getTimeData
CDAQMXDateTime::getMXDateTime

getVersionAPIMX

Syntax

```
const int getVersionAPIMX(void);
```

Declaration

```
Public Declare Function getVersionAPIMX Lib "DAQMX"() As Long
```

Description

Gets the version number of this API.

Return value

Returns the version number of this API as an integer value.

Reference

CDAQMX::getVersionAPI

incrementDataNoMX

Syntax

```
void incrementDataNoMX(MXDataNo * dataNo, int increment);
```

Declaration

```
Public Declare Sub incrementDataNoMX Lib "DAQMX" (ByRef dataNo  
As MXDataNo, ByVal increment As Long)
```

Parameters

dataNo	Specify the data number.
increment	Specify the value to increment by.

Description

Increments the specified data number by the specified amount. The field of the specified data number is changed.

initSystemMX

Syntax

```
int initSystemMX(DAQMX daqmx, int iCtrl);
```

Declaration

```
Public Declare Function initSystemMX Lib "DAQMX" (ByVal daqmx  
As Long, ByVal iCtrl As Long) As Long
```

Parameters

daqmx	Specify the device descriptor.
iCtrl	Specify the system control type.

Description

Executes the specified system control.

Return value

Returns an error number.

Error:

Not descriptor No device descriptor.

Reference

CDAQMX::initSystem

isDataNoMX [Visual C only]

Syntax

```
int isDataNoMX(MXDataNo dataNo);
```

Parameters

dataNo Specify the data number.

Description

Checks whether the specified data number is a valid number.

- Returns “Valid” if the data number is 0 or greater.

Return value

Returns a boolean value.

Reference

CDAQMXStatus::isDataNo

isDataNoVBMX

Syntax

```
int isDataNoVBMX(MXDataNo * dataNo);
```

Declaration

```
Public Declare Function isDataNoVBMX Lib "DAQMX" (ByRef dataNo  
As MXDataNo) As Long
```

Parameters

dataNo Specify the data number.

Description

Checks whether the specified data number is a valid number.

- Except for reference specification, the data number specification is the same as the isDataNoMX function.
- In Visual C, if NULL is specified for a data number, Invalid is returned.

Return value

Returns a boolean value.

Reference

isDataNoMX

openMX

Syntax

```
DAQMX openMX(const char * strAddress, int * errorCode);
```

Declaration

```
Public Declare Function openMX Lib "DAQMX" (ByVal strAddress As String, ByRef errorCode As Long) As Long
```

Parameters

strAddress	Specify the IP address as a string.
errorCode	Specify the destination where the error number is to be returned.

Description

Connects to the device with the IP address specified by the parameters.

- Creates a device descriptor and returns the value as a return value.
- Stores the error number if the return destination is specified.
- If unsuccessful, returns NULL in Visual C or 0 in Visual Basic.

Return value

Returns the device descriptor.

Error:

Creating descriptor is failure	Failed to create the device descriptor.
--------------------------------	---

Reference

CDAQMX::open

resetBalanceMX

Syntax

```
int resetBalanceMX(DAQMX daqmx, MXBalanceData *
pMXBalanceData, MXBalanceResult * pMXBalanceResult);
```

Declaration

```
Public Declare Function resetBalanceMX Lib "DAQMX" (ByVal daqmx
As Long, ByRef pMXBalanceData As MXBalanceData, ByRef
pMXBalanceResult As MXBalanceResult) As Long
```

Parameters

daqmx	Specify the device descriptor.
pMXBalanceData	Specify initial balance data.
pMXBalanceResult	Specify the destination where the initial balance result is to be returned.

Description

Initializes the initial balance value.

- Executed on channels whose initial balance data items are set to Valid.
- The initial balance value is returned to the initial balance data. Also, the channel-by-channel initial balance results are returned to the specified return destination. When this occurs, only the data from channels whose initial balance data items are set to Valid is overwritten.
- The response to this function may take five seconds or longer.

Return value

Returns an error number.

Error:

Not descriptor	No device descriptor.
----------------	-----------------------

Reference

```
CDAQMX::restBalance
CDAQMXBalanceResult::getMXBalanceData
CDAQMXBalanceResult::getMXBalanceResult
```

runBalanceMX

Syntax

```
int runBalanceMX(DAQMX daqmx, MXBalanceData * pMXBalanceData,
MXBalanceResult * pMXBalanceResult);
```

Declaration

```
Public Declare Function runBalanceMX Lib "DAQMX" (ByVal daqmx
As Long, ByRef pMXBalanceData As MXBalanceData, ByRef
pMXBalanceResult As MXBalanceResult) As Long
```

Parameters

daqmx	Specify the device descriptor.
pMXBalanceData	Specify initial balance data.
pMXBalanceResult	Specify the destination where the initial balance result is to be returned.

Description

Executes initial balancing.

- Executed on channels whose initial balance data items are set to Valid.
- The initial balance value is returned to the initial balance data. Also, the channel-by-channel initial balance results are returned to the specified return destination. When this occurs, only the data from channels whose initial balance data items are set to Valid is overwritten.
- The response to this function may take five seconds or longer.

Return value

Returns an error number.

Error:

Not descriptor	No device descriptor.
----------------	-----------------------

Reference

```
CDAQMX::runBalance
CDAQMXBalanceResult::getMXBalanceData
CDAQMXBalanceResult::getMXBalanceResult
```

setAlarmMX

Syntax

```
int setAlarmMX(DAQMX daqmx, int levelNo, int startChNo, int
endChNo, int iAlarmType, int value, int histerisys);
```

Declaration

```
Public Declare Function setAlarmMX Lib "DAQMX" (ByVal daqmx As
Long, ByVal levelNo As Long, ByVal startChNo As Long, ByVal
endChNo As Long, ByVal iAlarmType As Long, ByVal value As
Long, ByVal histerisys As Long) As Long
```

Parameters

daqmx	Specify the device descriptor.
levelNo	Specify the alarm level.
startChNo	Specify the start channel number.
endChNo	Specify the end channel number.
iAlarmType	Specify the alarm type.
value	Specify the alarm value.
histerisys	Specify the hysteresis.

Description

Sets the alarm to the specified channel range.

The setup data is retrieved collectively, changed, and sent collectively.

Return value

Returns an error number.

Error:

Not descriptor No device descriptor.

Reference

```
CDAQMX::getMXConfig    CDAQMX::setMXConfig
CDAQMXChConfig::setAlarm
CDAQMXConfig::getClassMXChConfig
```

setAOMX

Syntax

```
int setAOMX(DAQMX daqmx, int iRangeAO, int startChNo, int endChNo, int spanMin, int spanMax);
```

Declaration

```
Public Declare Function setAOMX Lib "DAQMX" (ByVal daqmx As Long, ByVal iRangeAO As Long, ByVal startChNo As Long, ByVal endChNo As Long, ByVal spanMin As Long, ByVal spanMax As Long) As Long
```

Parameters

daqmx	Specify the device descriptor.
iRangeAO	Specify the AO range type from the range type.
startChNo	Specify the start channel number.
endChNo	Specify the end channel number.
spanMin	Specify the span minimum.
spanMax	Specify the span maximum.

Description

Set the specified channel range to the AO range.

- If the minimum and maximum values of the span are equal, the span is considered to be omitted.
- The setup data is retrieved collectively, changed, and sent collectively.
- Channels from unsupported modules are ignored.
- Output channel data is also changed.

Return value

Returns an error number.

Error:

Not descriptor No device descriptor.

Reference

```
CDAQMX::getMXConfig  
CDAQMX::setMXConfig  
CDAQMXConfig::setAO
```

setAOPWMDataMX

Syntax

```
int setAOPWMDataMX(DAQMX daqmx, MXAOPWMData * pMXAOPWMData);
```

Declaration

```
Public Declare Function setAOPWMDataMX Lib "DAQMX" (ByVal daqmx  
As Long, ByRef pMXAOPWMData As MXAOPWMData) As Long
```

Parameters

daqmx	Specify the device descriptor.
pMXAOPWMData	Specify the AO/PWM data to output.

Description

Sends AO/PWM data collectively.

- Changes the output of command AO and command PWM channels.
- You can create data to send by changing the AO/PWM data of a data operation function.
- The values specified for channels whose AO/PWM data is set to Valid are output.

Return value

Returns an error number.

Error:

Not descriptor	No device descriptor.
----------------	-----------------------

Reference

CDAQMX::setAOPWMData

setAOTypeMX

Syntax

```
int setAOTypeMX(DAQMX daqmx, int aoNo, int iKind, int  
refChNo);
```

Declaration

```
Public Declare Function setAOTypeMX Lib "DAQMX" (ByVal daqmx As  
Long, ByVal aoNo As Long, ByVal iKind As Long, ByVal refChNo  
As Long) As Long
```

Parameters

daqmx	Specify the device descriptor.
aoNo	Specify the AO data number.
iKind	Specify the type of AO channel using channel type.
refChNo	Specify the reference channel using a channel number.

Description

Sets the channel corresponding to the AO data number to the specified channel type.

- Channels must be on the AO module.
- The channel type that can be specified is either AO or command AO.
- The reference channel number must be an input channel.
- When specifying command AO, the reference channel number is ignored.
- The setup data is retrieved collectively, changed, and sent collectively.

Return value

Returns an error number.

Error:

Not descriptor No device descriptor.

Reference

```
CDAQMX::getMXConfig  
CDAQMX::setMXConfig  
CDAQMXConfig::setAOType
```

setBackupMX

Syntax

```
int setBackupMX(DAQMX daqmx, int bBackup, int iCFWriteMode);
```

Declaration

```
Public Declare Function setBackupMX Lib "DAQMX" (ByVal daqmx As Long, ByVal bBackup As Long, ByVal iCFWriteMode As Long) As Long
```

Parameters

daqmx	Specify the device descriptor.
bBackup	Specify the boolean value.
iCFWriteMode	Specify the CF write mode.

Description

Sets backup (saving of data to the CF card).

- Set the CF write mode only when different from the current setting.
- When the CF write mode is set, the FIFO stops.

Return value

Returns an error number.

Error:

Not descriptor	No device descriptor.
----------------	-----------------------

Reference

```
CDAQMX::getMXConfig  
CDAQMX::setBackup  
CDAQMX::setMXConfig  
CDAQMXConfig::getClassMXStatus  
CDAQMXStatus::getCFWriteMode  
CDAQMXStatus::setCFWriteMode
```


setBalanceMX

Syntax

```
int setBalanceMX(DAQMX daqmx, MXBalanceData * pMXBalanceData);
```

Declaration

```
Public Declare Function setBalanceMX Lib "DAQMX" (ByVal daqmx  
As Long, ByRef pMXBalanceData As MXBalanceData) As Long
```

Parameters

daqmx	Specify the device descriptor.
pMXBalanceData	Specify initial balance data.

Description

Sets initial balance data.

- Writes the initial balance data of channels whose initial balance data items are set to Valid.
- The setup data is retrieved collectively, changed, and sent collectively.

Return value

Returns an error number.

Error:

Not descriptor	No device descriptor.
----------------	-----------------------

Reference

CDAQMX::setBalance

setBurnoutMX

Syntax

```
int setBurnoutMX(DAQMX daqmx, int iBurnout, int startChNo, int endChNo);
```

Declaration

```
Public Declare Function setBurnoutMX Lib "DAQMX" (ByVal daqmx As Long, ByVal iBurnout As Long, ByVal startChNo As Long, ByVal endChNo As Long) As Long
```

Parameters

daqmx	Specify the device descriptor.
iBurnout	Specify the burnout type.
startChNo	Specify the start channel number.
endChNo	Specify the end channel number.

Description

Sets the burnout type to the specified channel range.

- The setup data is retrieved collectively, changed, and sent collectively.

Return value

Returns an error number.

Error:

Not descriptor No device descriptor.

Reference

```
CDAQMX::getMXConfig  
CDAQMX::setMXConfig  
CDAQMXChConfig::setBurnout  
CDAQMXConfig::getClassMXChConfig
```

setChConfigMX

Syntax

```
int setChConfigMX(DAQMX daqmx, MXChConfig * pMXChConfig);
```

Declaration

```
Public Declare Function setChConfigMX Lib "DAQMX" (ByVal daqmx  
As Long, ByRef pMXChConfig As MXChConfig) As Long
```

Parameters

daqmx Specify the device descriptor.
pMXChConfig Specify the channel setup data.

Description

Sets the channel setup data.

- The setup data is retrieved collectively, changed, and sent collectively.

Return value

Returns an error number.

Error:

Not descriptor No device descriptor.

Reference

```
CDAQMX::getMXConfig  
CDAQMX::setMXConfig  
CDAQMXChConfigData::setMXChConfig  
CDAQMXConfig::getClassMXChConfigData
```

setChoiceMX

Syntax

```
int setChoiceMX(DAQMX daqmx, int startChNo, int endChNo, int
idleChoice, int errorChoice, int presetValue);
```

Declaration

```
Public Declare Function setChoiceMX Lib "DAQMX" (ByVal daqmx As
Long, ByVal startChNo As Long, ByVal endChNo As Long, ByVal
idleChoice As Long, ByVal errorChoice As Long, ByVal
presetValue As Long) As Long
```

Parameters

daqmx	Specify the device descriptor.
startChNo	Specify the start channel number.
endChNo	Specify the end channel number.
idleChoice	Specify the selected value when idling.
errorChoice	Specify the selected value when an error occurs.
presetValue	Specify the output value if the selected value is the "specified value."

Description

Sets the output during idling and when errors occur on the output channel data.

- The user specified output values are specified with integers in the same manner as the data value and span.
- The setup data is retrieved collectively, changed, and sent collectively.

Return value

Returns an error number.

Error:

Not descriptor	No device descriptor.
----------------	-----------------------

Reference

```
CDAQMX::getMXConfig
CDAQMX::setMXConfig
CDAQMXChConfig::setChoice
```

setCommentMX

Syntax

```
int setCommentMX(DAQMX daqmx, const char * strComment, int startChNo, int endChNo);
```

Declaration

```
Public Declare Function setCommentMX Lib "DAQMX" (ByVal daqmx As Long, ByVal strComment As String, ByVal startChNo As Long, ByVal endChNo As Long) As Long
```

Parameters

daqmx	Specify the device descriptor.
strComment	Specify the comment.
startChNo	Specify the start channel number.
endChNo	Specify the end channel number.

Description

Sets the comment to the specified channel range.

- The setup data is retrieved collectively, changed, and sent collectively.

Return value

Returns an error number.

Error:

Not descriptor No device descriptor.

Reference

```
CDAQMX::getMXConfig  
CDAQMX::setMXConfig  
CDAQMXChConfig::setComment  
CDAQMXConfig::getClassMXChConfig
```

setConfigDataMX

Syntax

```
int setConfigDataMX(DAQMX daqmx, MXConfigData *  
pMXConfigData);
```

Parameters

daqmx Specify the device descriptor.
pMXConfigData Specify the setup data.

Description

Sends the setup data collectively.

- In Visual Basic, set each data structure individually.

Return value

Returns an error number.

Error:

Not descriptor No device descriptor.

Reference

CDAQMX::setMXConfig

setDateTimeMX

Syntax

```
int setDateTimeMX(DAQMX daqmx, MXDateTime * pMXDateTime);
```

Declaration

```
Public Declare Function setDateTimeMX Lib "DAQMX" (ByVal daqmx  
As Long, ByRef pMXDateTime As MXDateTime) As Long
```

Parameters

daqmx Specify the device descriptor.
pMXDateTime Specify the time information data.

Description

Sets time information data on the device.

- Milliseconds are discarded.
- In Visual C, if the parameter's time information data is set to NULL, the current date/time of the PC is used.
- The response to this function may take one second or longer.

Return value

Returns an error number.

Error:

Not descriptor No device descriptor.

Reference

CDAQMX::setDateTime

setDateTimeNowMX

Syntax

```
int setDateTimeNowMX(DAQMX daqmx);
```

Declaration

```
Public Declare Function setDateTimeNowMX Lib "DAQMX" (ByVal  
daqmx As Long) As Long
```

Parameters

daqmx Specify the device descriptor.

Description

Sets the current date/time.

Return value

Returns an error number.

Reference

setDateTimeMX

setDELTAMX

Syntax

```
int setDELTAMX(DAQMx daqmx, int refChNo, int startChNo, int
endChNo, int spanMin, int spanMax, int scaleMin, int scaleMax,
int scalePoint, int iRange);
```

Declaration

```
Public Declare Function setDELTAMX Lib "DAQMX" (ByVal daqmx As
Long, ByVal refChNo As Long, ByVal startChNo As Long, ByVal
endChNo As Long, ByVal spanMin As Long, ByVal spanMax As Long,
ByVal scaleMin As Long, ByVal scaleMax As Long, ByVal
scalePoint As Long, ByVal iRange As Long) As Long
```

Parameters

daqmx	Specify the device descriptor.
refChNo	Specify the reference channel using a channel number.
startChNo	Specify the start channel number.
endChNo	Specify the end channel number.
spanMin	Specify the span minimum.
spanMax	Specify the span maximum.
scaleMin	Specify the scale minimum.
scaleMax	Specify the scale maximum.
scalePoint	Specify the decimal point position for scaling.
iRange	Specify the range type for the input of the target channel.

Description

- Sets the specified channel range to difference computation.
- If the minimum and maximum values of the span are equal, the span is considered to be omitted.
- If the minimum and maximum values of the scale are equal, the scale is considered to be omitted.
- The setup data is retrieved collectively, changed, and sent collectively.
- Channels from unsupported modules are ignored.
- The alarm setting is initialized.
- If, when specifying the input range for the target channel, "Reference range" is specified for the range type, the same range as the reference channel is used for the measurement range of the target channel.

Return value

Returns an error number.

Error:

Not descriptor No device descriptor.

Reference

```
CDAQMX::getMXConfig
CDAQMX::setMXConfig
CDAQMXConfig::setDELTA
CDAQMXConfig::setScale
```

setDIMX

Syntax

```
int setDIMX(DAQMX daqmx, int iRangeDI, int startChNo, int
endChNo, int spanMin, int spanMax, int scaleMin, int scaleMax,
int scalePoint);
```

Declaration

```
Public Declare Function setDIMX Lib "DAQMX" (ByVal daqmx As
Long, ByVal iRangeDI As Long, ByVal startChNo As Long, ByVal
endChNo As Long, ByVal spanMin As Long, ByVal spanMax As Long,
ByVal scaleMin As Long, ByVal scaleMax As Long, ByVal
scalePoint As Long) As Long
```

Parameters

daqmx	Specify the device descriptor.
iRangeDI	Specify the range type of the digital input (DI).
startChNo	Specify the start channel number.
endChNo	Specify the end channel number.
spanMin	Specify the span minimum.
spanMax	Specify the span maximum.
scaleMin	Specify the scale minimum.
scaleMax	Specify the scale maximum.
scalePoint	Specify the decimal point position for scaling.

Description

Sets the specified channel range to digital input (DI).

- If the minimum and maximum values of the span are equal, the span is considered to be omitted.
- If the minimum and maximum values of the scale are equal, the scale is considered to be omitted.
- The setup data is retrieved collectively, changed, and sent collectively.
- Channels from unsupported modules are ignored.
- The alarm setting is initialized.

Return value

Returns an error number.

Error:

Not descriptor No device descriptor.

Reference

```
CDAQMX::getMXConfig
CDAQMX::setMXConfig
CDAQMXConfig::setDI
CDAQMXConfig::setScale
```

setDODataMX

Syntax

```
int setDODataMX(DAQMX daqmx, MXDOData * pMXDOData);
```

Declaration

```
Public Declare Function setDODataMX Lib "DAQMX" (ByVal daqmx As Long, ByRef pMXDOData As MXDOData) As Long
```

Parameters

daqmx Specify the device descriptor.
pMXDOData Specify the DO data.

Description

Sends the DO data collectively.

- Changes the output of command DO channels.
- You can create data to send by changing the DO data of a data operation function.
- The ON/OFF values specified as "Valid" within the DO data are output.

Return value

Returns an error number.

Error:

Not descriptor No device descriptor.

Reference

CDAQMX::setDOData

setDOTypeMX

Syntax

```
int setDOTypeMX(DAQMX daqmx, int doNo, int iKind, int
bDeenergize, int bHold);
```

Declaration

```
Public Declare Function setDOTypeMX Lib "DAQMX" (ByVal daqmx As
Long, ByVal doNo As Long, ByVal iKind As Long, ByVal
bDeenergize As Long, ByVal bHold As Long) As Long
```

Parameters

daqmx	Specify the device descriptor.
doNo	Specify the data number.
iKind	Specify the DO channel type.
bDeenergize	Specify de-energize using a Boolean value.
bHold	Specify hold using a Boolean value.

Description

Sets the channel corresponding to the DO data number to the specified channel type.

- Channels are on the DO module, and the type must be set to DO (alarm output, command DO, system fail, or system error).
- Set the reference alarm using the setRefAlarmMX function.
- The setup data is retrieved collectively, changed, and sent collectively.

Return value

Returns an error number.

Error:

Not descriptor No device descriptor.

Reference

```
CDAQMX::getMXConfig
CDAQMX::setMXConfig
CDAQMXConfig::setDOType
```

setFilterMX

Syntax

```
int setFilterMX(DAQMX daqmx, int iFilter, int startChNo, int endChNo);
```

Declaration

```
Public Declare Function setFilterMX Lib "DAQMX" (ByVal daqmx As Long, ByVal iFilter As Long, ByVal startChNo As Long, ByVal endChNo As Long) As Long
```

Parameters

daqmx	Specify the device descriptor.
iFilter	Specify the filter coefficient.
startChNo	Specify the start channel number.
endChNo	Specify the end channel number.

Description

Sets the filter coefficient to the specified channel range.

- The setup data is retrieved collectively, changed, and sent collectively.

Return value

Returns an error number.

Error:

Not descriptor No device descriptor.

Reference

```
CDAQMX::getMXConfig  
CDAQMX::setMXConfig  
CDAQMXChConfig::setFilter  
CDAQMXConfig::getClassMXChConfig
```

setIntervalMX

Syntax

```
int setIntervalMX(DAQMX daqmx, int moduleNo, int iInterval,
int iHz);
```

Declaration

```
Public Declare Function setIntervalMX Lib "DAQMX" (ByVal daqmx
As Long, ByVal moduleNo As Long, ByVal iInterval As Long,
ByVal iHz As Long) As Long
```

Parameters

daqmx	Specify the device descriptor.
moduleNo	Specify the module number.
iInterval	Specify the interval type.
iHz	Specify the type of A/D integration time.

Description

Sets the module of the specified module number to the specified value.

- The setup data is retrieved collectively, changed, and sent collectively.
- If the constant for "Specify all module numbers" is specified for the module numbers, all modules are set.

Return value

Returns an error number.

Error:

Not descriptor No device descriptor.

Reference

```
CDAQMX::getMXConfig
CDAQMX::setMXConfig
CDAQMXConfig::setInterval
```

setOutputMX

Syntax

```
int setOutputMX(DAQMX daqmx, MXOutputData * pMXOutputData);
```

Declaration

```
Public Declare Function setOutputMX Lib "DAQMX" (ByVal daqmx As Long, ByRef pMXOutputData As MXOutputData) As Long
```

Parameters

daqmx	Specify the device descriptor.
pMXOutputData	Specify the output channel data.

Description

Sets the output channel data.

- The setup data is retrieved collectively, changed, and sent collectively.

Return value

Returns an error number.

Error:

Not descriptor	No device descriptor.
----------------	-----------------------

Reference

```
CDAQMX::getMXConfig  
CDAQMX::setMXConfig  
CDAQMXConfig::getClassMXOutputData  
CDAQMXOutputData::setMXOutputData
```

setOutputTypeMX

Syntax

```
int setOutputTypeMX(DAQMX daqmx, int iOutput, int startChNo,  
int endChNo);
```

Declaration

```
Public Declare Function setOutputTypeMX Lib "DAQMX" (ByVal  
daqmx As Long, ByVal iOutput As Long, ByVal startChNo As Long,  
ByVal endChNo As Long) As Long
```

Parameters

daqmx	Specify the device descriptor.
iOutput	Specify the output type.
startChNo	Specify the start channel number.
endChNo	Specify the end channel number.

Description

Sets the output type of the output channel data.

- Each item is set to the default value.
- The settings of the corresponding channels are also changed.
- The setup data is retrieved collectively, changed, and sent collectively.

Return value

Returns an error number.

Error

Not descriptor No device descriptor.

Reference

CDAQMX::getMXConfig

CDAQMX::setMXConfig

CDAQMXConfig::setAO

CDAQMXConfig::setPWM

setPulseTimeMX

Syntax

```
int setPulseTimeMX(DAQMX daqmx, int pulseTime, int startChNo,  
int endChNo);
```

Declaration

```
Public Declare Function setPulseTimeMX Lib "DAQMX" (ByVal daqmx  
As Long, ByVal pulseTime As Long, ByVal startChNo As Long,  
ByVal endChNo As Long) As Long
```

Parameters

daqmx	Specify the device descriptor.
pulseTime	Specify the integer multiple of the pulse interval.
startChNo	Specify the start channel number.
endChNo	Specify the end channel number.

Description

Sets the integral multiple of the pulse interval of the output channel data.

- The setup data is retrieved collectively, changed, and sent collectively.

Return value

Returns an error number.

Error:

Not descriptor	No device descriptor.
----------------	-----------------------

Reference

```
CDAQMX::getMXConfig  
CDAQMX::setMXConfig  
CDAQMXChConfig::setPulseTime
```

setPWMMX

Syntax

```
int setPWMMX(DAQMX daqmx, int iRangePWM, int startChNo, int
endChNo, int spanMin, int spanMax);
```

Declaration

```
Public Declare Function setPWMMX Lib "DAQMX" (ByVal daqmx As
Long, ByVal iRangePWM As Long, ByVal startChNo As Long, ByVal
endChNo As Long, ByVal spanMin As Long, ByVal spanMax As Long)
As Long
```

Parameters

daqmx	Specify the device descriptor.
iRangePWM	Specify the PWM range type from the range type.
startChNo	Specify the start channel number.
endChNo	Specify the end channel number.
spanMin	Specify the span minimum.
spanMax	Specify the span maximum.

Description

Sets the specified channel range to the PWM range.

- If the minimum and maximum values of the span are equal, the span is considered to be omitted.
- The setup data is retrieved collectively, changed, and sent collectively.
- Channels from unsupported modules are ignored.
- Output channel data is also changed.

Return value

Returns an error number.

Error:

Not descriptor No device descriptor.

Reference

```
CDAQMX::getMXConfig
CDAQMX::setMXConfig
CDAQMXConfig::setPWM
```

setPWMTypeMX

Syntax

```
int setPWMTypeMX(DAQMX daqmx, int pwmNo, int iKind, int  
refChNo);
```

Declaration

```
Public Declare Function setPWMTypeMX Lib "DAQMX" (ByVal daqmx  
As Long, ByVal pwmNo As Long, ByVal iKind As Long, ByVal  
refChNo As Long) As Long
```

Parameters

daqmx	Specify the device descriptor.
pwmNo	Specify the PWM data number.
iKind	Specify the type of PWM channel using a channel type.
refChNo	Specify the reference channel using a channel number.

Description

Sets the channel corresponding to the PWM data number to the specified channel type.

- Channels must be on the PWM module.
- The channel type that can be specified is either PWM or command PWM.
- The reference channel number must be an input channel.
- When specifying command PWM, the reference channel number is ignored.
- The setup data is retrieved collectively, changed, and sent collectively.

Return value

Returns an error number.

Error:

Not descriptor No device descriptor.

Reference

```
CDAQMX::getMXConfig  
CDAQMX::setMXConfig  
CDAQMXConfig::setPWMType
```

setRefAlarmMX

Syntax

```
int setRefAlarmMX(DAQMX daqmx, int refChNo, int startChNo, int
endChNo, int levelNo, int bValid);
```

Declaration

```
Public Declare Function setRefAlarmMX Lib "DAQMX" (ByVal daqmx
As Long, ByVal refChNo As Long, ByVal startChNo As Long, ByVal
endChNo As Long, ByVal levelNo As Long, ByVal bValid As Long)
As Long
```

Parameters

daqmx	Specify the device descriptor.
refChNo	Specify the reference channel using a channel number.
startChNo	Specify the start channel number.
endChNo	Specify the end channel number.
levelNo	Specify the alarm level.
bValid	Specify the boolean value.

Description

Sets the reference alarm to the specified channel range.

- The setup data is retrieved collectively, changed, and sent collectively.
- If the constant for “Specify all reference channel numbers” is specified for the reference channel, all channels are processed.
- If the constant for “Specify all alarm level numbers” is specified for the alarm level, all alarm levels of the reference channel are processed.

Return value

Returns an error number.

Error:

Not descriptor No device descriptor.

Reference

```
CDAQMX::getMXConfig
CDAQMX::setMXConfig
CDAQMXChConfig::setRefAlarm
CDAQMXConfig::getClassMXChConfig
```

setRESMX

Syntax

```
int setRESMX(DAQMX daqmx, int iRangeRES, int startChNo, int endChNo, int spanMin, int spanMax, int scaleMin, int scaleMax, int scalePoint);
```

Declaration

```
Public Declare Function setRESMX Lib "DAQMX" (ByVal daqmx As Long, ByVal iRangeRES As Long, ByVal startChNo As Long, ByVal endChNo As Long, ByVal spanMin As Long, ByVal spanMax As Long, ByVal scaleMin As Long, ByVal scaleMax As Long, ByVal scalePoint As Long) As Long
```

Parameters

daqmx	Specify the device descriptor.
iRangeRES	Specify the resistance range type from the range type.
startChNo	Specify the start channel number.
endChNo	Specify the end channel number.
spanMin	Specify the span minimum.
spanMax	Specify the span maximum.
scaleMin	Specify the scale minimum.
scaleMax	Specify the scale maximum.
scalePoint	Specify the decimal point position for scaling.

Description

Sets the specified channel range to the resistance range.

- If the minimum and maximum values of the span are equal, the span is considered to be omitted.
- If the minimum and maximum values of the scale are equal, the scale is considered to be omitted.
- The setup data is retrieved collectively, changed, and sent collectively.
- Channels from unsupported modules are ignored.
- The alarm setting is initialized.

Return value

Returns an error number.

Error

Not descriptor No device descriptor.

Reference

```
CDAQMX::getMXConfig  
CDAQMX::setMXConfig  
CDAQMXConfig::setRES  
CDAQMXConfig::setScalePoint
```

setRJCTypeMX

Syntax

```
int setRJCTypeMX(DAQMX daqmx, int iRJCType, int startChNo, int
endChNo, int volt);
```

Declaration

```
Public Declare Function setRJCTypeMX Lib "DAQMX" (ByVal daqmx
As Long, ByVal iRJCType As Long, ByVal startChNo As Long,
ByVal endChNo As Long, ByVal volt As Long) As Long
```

Parameters

daqmx	Specify the device descriptor.
iRJCType	Specify the RJC type.
startChNo	Specify the start channel number.
endChNo	Specify the end channel number.
volt	Specify the RJC voltage.

Description

Sets the RJC items to the specified channel range.

- The setup data is retrieved collectively, changed, and sent collectively.

Return value

Returns an error number.

Error:

Not descriptor No device descriptor.

Reference

```
CDAQMX::getMXConfig    CDAQMX::setMXConfig
CDAQMXChConfig::setRJCType
CDAQMXConfig::getClassMXChConfig
```

setRRJCMX

Syntax

```
int setRRJCMX(DAQMX daqmx, int refChNo, int startChNo, int endChNo, int spanMin, int spanMax);
```

Declaration

```
Public Declare Function setRRJCMX Lib "DAQMX" (ByVal daqmx As Long, ByVal refChNo As Long, ByVal startChNo As Long, ByVal endChNo As Long, ByVal spanMin As Long, ByVal spanMax As Long) As Long
```

Parameters

daqmx	Specify the device descriptor.
refChNo	Specify the reference channel using a channel number.
startChNo	Specify the start channel number.
endChNo	Specify the end channel number.
spanMin	Specify the span minimum.
spanMax	Specify the span maximum.

Description

Sets remote RJC to the specified channel range.

- If the minimum and maximum values of the span are equal, the span is considered to be omitted.
- The setup data is retrieved collectively, changed, and sent collectively.
- Channels from unsupported modules are ignored.
- The alarm setting is initialized.

Return value

Returns an error number.

Error:

Not descriptor No device descriptor.

Reference

```
CDAQMX::getMXConfig  
CDAQMX::setMXConfig  
CDAQMXConfig::setRRJC
```

setRTDMX

Syntax

```
int setRTDMX(DAQMX daqmx, int iRangeRTD, int startChNo, int
endChNo, int spanMin, int spanMax, int scaleMin, int scaleMax,
int scalePoint);
```

Declaration

```
Public Declare Function setRTDMX Lib "DAQMX" (ByVal daqmx As
Long, ByVal iRangeRTD As Long, ByVal startChNo As Long, ByVal
endChNo As Long, ByVal spanMin As Long, ByVal spanMax As Long,
ByVal scaleMin As Long, ByVal scaleMax As Long, ByVal
scalePoint As Long) As Long
```

Parameters

daqmx	Specify the device descriptor.
iRangeRTD	Specify the range type of the RTD input.
startChNo	Specify the start channel number.
endChNo	Specify the end channel number.
spanMin	Specify the span minimum.
spanMax	Specify the span maximum.
scaleMin	Specify the scale minimum.
scaleMax	Specify the scale maximum.
scalePoint	Specify the decimal point position for scaling.

Description

Sets the specified channel range to RTD input.

- If the minimum and maximum values of the span are equal, the span is considered to be omitted.
- If the minimum and maximum values of the scale are equal, the scale is considered to be omitted.
- The setup data is retrieved collectively, changed, and sent collectively.
- Channels from unsupported modules are ignored.
- The alarm setting is initialized.

Return value

Returns an error number.

Error:

Not descriptor No device descriptor.

Reference

```
CDAQMX::getMXConfig
CDAQMX::setMXConfig
CDAQMXConfig::setRTD
CDAQMXConfig::setScale
```

setScallingUnitMX

Syntax

```
int setScallingUnitMX(DAQMX daqmx, const char * strUnit, int startChNo, int endChNo);
```

Declaration

```
Public Declare Function setScallingUnitMX Lib "DAQMX" (ByVal daqmx As Long, ByVal strUnit As String, ByVal startChNo As Long, ByVal endChNo As Long) As Long
```

Parameters

daqmx	Specify the device descriptor.
strUnit	Specify the unit name.
startChNo	Specify the start channel number.
endChNo	Specify the end channel number.

Description

Sets the unit name to the specified channel range.

- The setup data is retrieved collectively, changed, and sent collectively.

Return value

Returns an error number.

Error:

Not descriptor No device descriptor.

Reference

```
CDAQMX::getMXConfig  
CDAQMX::setMXConfig  
CDAQMXChConfig::setUnit  
CDAQMXConfig::getClassMXChConfig
```

setSegmentMX

Syntax

```
int setSegmentMX(DAQMX daqmx, int dispType, int dispTime,
MXSegment * newSegment, MXSegment * oldSegment);
```

Declaration

```
Public Declare Function setSegmentMX Lib "DAQMX" (ByVal daqmx
As Long, ByVal iDispType As Long, ByVal dispTime As Long,
ByRef newSegment As MXSegment, ByRef oldSegment As MXSegment)
As Long
```

Parameters

daqmx	Specify the device descriptor.
iDispType	Specify the display type.
dispTime	Specify the display time.
newSegment	Specify the display pattern.
oldSegment	Specify the destination where the previous display pattern is to be returned.

Description

Sets the display of the 7-segment LED.

- Stores the 7-segment LED display pattern before the change if the return destination is specified.

Return value

Returns an error number.

Error:

Not descriptor No device descriptor.

Reference

CDAQMX::setSegment

CDAQMXSegment::getMXSegment

setSKIPMX

Syntax

```
int setSKIPMX(DAQMX daqmx, int startChNo, int endChNo);
```

Declaration

```
Public Declare Function setSKIPMX Lib "DAQMX" (ByVal daqmx As Long, ByVal startChNo As Long, ByVal endChNo As Long) As Long
```

Parameters

daqmx	Specify the device descriptor.
startChNo	Specify the start channel number.
endChNo	Specify the end channel number.

Description

Sets the specified channel range to SKIP (not used).

- The setup data is retrieved collectively, changed, and sent collectively.
- When set to an unused channel, the channel settings on the main unit are lost.

Return value

Returns an error number.

Error:

Not descriptor No device descriptor.

Reference

```
CDAQMX::getMXConfig  
CDAQMX::setMXConfig  
CDAQMXConfig::setSKIP
```

setSTRAINMX

Syntax

```
int setSTRAINMX(DAQMX daqmx, int iRangeSTRAIN, int startChNo,
int endChNo, int spanMin, int spanMax, int scaleMin, int
scaleMax, int scalePoint);
```

Declaration

```
Public Declare Function setSTRAINMX Lib "DAQMX" (ByVal daqmx As
Long, ByVal iRangeSTRAIN As Long, ByVal startChNo As Long,
ByVal endChNo As Long, ByVal spanMin As Long, ByVal spanMax As
Long, ByVal scaleMin As Long, ByVal scaleMax As Long, ByVal
scalePoint As Long) As Long
```

Parameters

daqmx	Specify the device descriptor.
iRangeSTRAIN	Specify the strain range from the range type.
startChNo	Specify the start channel number.
endChNo	Specify the end channel number.
spanMin	Specify the span minimum.
spanMax	Specify the span maximum.
scaleMin	Specify the scale minimum.
scaleMax	Specify the scale maximum.
scalePoint	Specify the decimal point position for scaling.

Description

Sets the specified channel range to the strain range.

- If the minimum and maximum values of the span are equal, the span is considered to be omitted.
- If the minimum and maximum values of the scale are equal, the span is considered to be omitted.
- The setup data is retrieved collectively, changed, and sent collectively.
- Channel from unsupported modules are ignored.
- The alarm setting is initialized.

Return value

Returns an error number.

Error:

Not descriptor	No device descriptor.
----------------	-----------------------

Reference

```
CDAQMX::getMXConfig
CDAQMX::setMXConfig
CDAQMXConfig::setScaleing
CDAQMXConfig::setSTRAIN
```

setSystemConfigMX

Syntax

```
int setSystemConfigMX(DAQMX daqmx, MXSystemInfo *  
pMXSystemInfo);
```

Declaration

```
Public Declare Function setSystemConfigMX Lib "DAQMX" (ByVal  
daqmx As Long, ByVal pMXSystemInfo As MXSystemInfo) As Long
```

Parameters

daqmx Specify the device descriptor.
pMXSystemInfo Specify the system configuration data.

Description

Sets the system configuration data within the setup data.

- The setup data is retrieved collectively, changed, and sent collectively.

Return value

Returns an error number.

Error:

Not descriptor No device descriptor.

Reference

```
CDAQMX::getMXConfig  
CDAQMX::setMXConfig  
CDAQMXConfig::getClassMXSysInfo  
CDAQMXSysInfo::setMXSystemInfo
```

setSystemTimeoutMX

Syntax

```
int setSystemTimeoutMX(DAQMX daqmx, int timeout);
```

Declaration

```
Public Declare Function setSystemTimeoutMX Lib "DAQMX" (ByVal daqmx As Long, ByVal timeout As Long) As Long
```

Parameters

daqmx	Specify the device descriptor.
timeout	Specify the timeout value in numbers of seconds.

Description

Sets the timeout value.

- The setup data is retrieved collectively, changed, and sent collectively.

Return value

Returns an error number.

Error:

Not descriptor No device descriptor.

Reference

```
CDAQMX::getMXConfig  
CDAQMX::setMXConfig  
CDAQMXConfig::getClassMXSysInfo  
CDAQMXSysInfo::setCFTimeout
```

setTagMX

Syntax

```
int setTagMX(DAQMX daqmx, const char * strTag, int startChNo,
int endChNo);
```

Declaration

```
Public Declare Function setTagMX Lib "DAQMX" (ByVal daqmx As
Long, ByVal strTag As String, ByVal startChNo As Long, ByVal
endChNo As Long) As Long
```

Parameters

daqmx	Specify the device descriptor.
strTag	Specify the tag.
startChNo	Specify the start channel number.
endChNo	Specify the end channel number.

Description

Sets the tag to the specified channel range.

- The setup data is retrieved collectively, changed, and sent collectively.

Return value

Returns an error number.

Error:

Not descriptor No device descriptor.

Reference

```
CDAQMX::getMXConfig
CDAQMX::setMXConfig
CDAQMXChConfig::setTag
CDAQMXConfig::getClassMXChConfig
```

setTCMX

Syntax

```
int setTCMX(DAQMX daqmx, int iRangeTC, int startChNo, int
endChNo, int spanMin, int spanMax, int scaleMin, int scaleMax,
int scalePoint);
```

Declaration

```
Public Declare Function setTCMX Lib "DAQMX" (ByVal daqmx As
Long, ByVal iRangeTC As Long, ByVal startChNo As Long, ByVal
endChNo As Long, ByVal spanMin As Long, ByVal spanMax As Long,
ByVal scaleMin As Long, ByVal scaleMax As Long, ByVal
scalePoint As Long) As Long
```

Parameters

daqmx	Specify the device descriptor.
iRangeTC	Specify the range type of the thermocouple input.
startChNo	Specify the start channel number.
endChNo	Specify the end channel number.
spanMin	Specify the span minimum.
spanMax	Specify the span maximum.
scaleMin	Specify the scale minimum.
scaleMax	Specify the scale maximum.
scalePoint	Specify the decimal point position for scaling.

Description

Sets the specified channel range to thermocouple input.

- If the minimum and maximum values of the span are equal, the span is considered to be omitted.
- If the minimum and maximum values of the scale are equal, the scale is considered to be omitted.
- The setup data is retrieved collectively, changed, and sent collectively.
- Channels from unsupported modules are ignored.
- The alarm setting is initialized.

Return value

Returns an error number.

Error:

Not descriptor No device descriptor.

Reference

```
CDAQMX::getMXConfig
CDAQMX::setMXConfig
CDAQMXConfig::setScale
CDAQMXConfig::setTC
```


setTempUnitMX

Syntax

```
int setTempUnitMX(DAQMX daqmx, int iTempUnit);
```

Declaration

```
Public Declare Function setTempUnitMX Lib "DAQMX" (ByVal daqmx  
As Long, ByVal iTempUnit As Long) As Long
```

Parameters

daqmx Specify the device descriptor.
iTempUnit Specify the temperature unit type.

Description

Sets the temperature unit type.

- The setup data is retrieved collectively, changed, and sent collectively.
- The channel settings of the thermocouple and RTD ranges are initialized.

Return value

Returns an error number.

Error:

Not descriptor No device descriptor.

Reference

```
CDAQMX::getMXConfig  
CDAQMX::setMXConfig  
CDAQMXConfig::setTempUnit
```

setTimeoutMX

Syntax

```
int setTimeoutMX(DAQMX daqmx, int seconds);
```

Declaration

```
Public Declare Function setTimeoutMX Lib "DAQMX" (ByVal daqmx  
As Long, ByVal seconds As Long) As Long
```

Parameters

daqmx	Specify the device descriptor.
seconds	Specify the communication timeout value in units of seconds.

Description

Sets a timeout for the communication with the device.

- If a negative value is specified, the timeout is discarded.
- Use of this function is not recommended (see section 3.1 or 4.1).

Return value

Returns an error number.

Error:

Not descriptor	No device descriptor.
----------------	-----------------------

Reference

CDAQMX::setTimeout

setTransmitMX

Syntax

```
int setTransmitMX(DAQMX daqmx, MXTransmit • pMXTransmit);
```

Declaration

```
Public Declare Function setTransmitMX Lib "DAQMX" (ByVal daqmx  
As Long, ByRef pMXTransmit As MXTransmit) As Long
```

Parameters

daqmx	Specify the device descriptor.
pMXTransmit	Specify the transmission output data.

Description

Sends the transmission output data collectively.

- Changes to the condition in which transmission output (AO and PWM) channels are specified.
- You can create data to send by changing the transmission output data of a data operation function.
- Specifications other than for the transmission output channels are ignored.
- If channels specified for output start are already outputting, output continues.

Return value

Returns an error number.

Error:

Not descriptor	No device descriptor.
----------------	-----------------------

Reference

CDAQMX::setTransmit

setUnitNoMX

Syntax

```
int setUnitNoMX(DAQMX daqmx, int unitNo);
```

Declaration

```
Public Declare Function setUnitNoMX Lib "DAQMX" (ByVal daqmx As Long, ByVal unitNo As Long) As Long
```

Parameters

daqmx	Specify the device descriptor.
unitNo	Specify the unit number.

Description

Sets the unit number.

- The setup data is retrieved collectively, changed, and sent collectively.

Return value

Returns an error number.

Error:

Not descriptor No device descriptor.

Reference

```
CDAQMX::getMXConfig  
CDAQMX::setMXConfig  
CDAQMXConfig::getClassMXSysInfo  
CDAQMXSysInfo::setUnitNo
```

setUserTimeMX [Visual C Only]

Syntax

```
int setUserTimeMX(DAQMX daqmx, MXUserTime userTime);
```

Parameters

daqmx Specify the device descriptor.
userTime Specify the user count.

Description

Sets the user count.

- Inserts the specified user count in the next packet to be issued.
- In Visual Basic, use the setUserTimeVBMX function.

Return value

Returns an error number.

Error:

Not descriptor No device descriptor.

Reference

CDAQMX::setUserTime

setUserTimeVBMX

Syntax

```
int setUserTimeVBMX(DAQMX daqmx, MXUserTime * userTime);
```

Declaration

```
Public Declare Function setUserTimeVBMX Lib "DAQMX" (ByVal daqmx As Long, ByRef userTime As MXUserTime) As Long
```

Parameters

daqmx	Specify the device descriptor.
userTime	Specify the user count.

Description

Sets the user count.

- Except for reference specification, the user count specification is the same as the setUserTimeMX function.
- In Visual C, if NULL is specified for user count, it is considered to be 0.

Return value

Returns an error number.

Reference

setUserTimeMX

setVOLTMX

Syntax

```
int setVOLTMX(DAQMx daqmx, int iRangeVOLT, int startChNo, int endChNo, int spanMin, int spanMax, int scaleMin, int scaleMax, int scalePoint);
```

Declaration

```
Public Declare Function setVOLTMX Lib "DAQMX" (ByVal daqmx As Long, ByVal iRangeVOLT As Long, ByVal startChNo As Long, ByVal endChNo As Long, ByVal spanMin As Long, ByVal spanMax As Long, ByVal scaleMin As Long, ByVal scaleMax As Long, ByVal scalePoint As Long) As Long
```

Parameters

daqmx	Specify the device descriptor.
iRangeVOLT	Specify the range type of the DC voltage input.
startChNo	Specify the start channel number.
endChNo	Specify the end channel number.
spanMin	Specify the span minimum.
spanMax	Specify the span maximum.
scaleMin	Specify the scale minimum.
scaleMax	Specify the scale maximum.
scalePoint	Specify the decimal point position for scaling.

Description

Sets the specified channel range to DC voltage input.

- If the minimum and maximum values of the span are equal, the span is considered to be omitted.
- If the minimum and maximum values of the scale are equal, the scale is considered to be omitted.
- The setup data is retrieved collectively, changed, and sent collectively.
- Channel from unsupported modules are ignored.
- The alarm setting is initialized.

Return value

Returns an error number.

Error:

Not descriptor No device descriptor.

Reference

```
CDAQMX::getMXConfig  
CDAQMX::setMXConfig  
CDAQMXConfig::setScale  
CDAQMXConfig::setVOLT
```

startFIFOMX

Syntax

```
int startFIFOMX(DAQMX daqmx);
```

Declaration

```
Public Declare Function startFIFOMX Lib "DAQMX" (ByVal daqmx As Long) As Long
```

Parameters

daqmx Specify the device descriptor.

Description

Starts the FIFO.

- If the FIFO is already started, it is continued.
- It may take time from the start of the FIFO until the measured data can be retrieved. The time required varies depending on the measurement interval.

Return value

Returns an error number.

Error:

Not descriptor No device descriptor.

Reference

CDAQMX::startFIFO

stopFIFOMX

Syntax

```
int stopFIFOMX(DAQMX daqmx);
```

Declaration

```
Public Declare Function stopFIFOMX Lib "DAQMX" (ByVal daqmx As Long) As Long
```

Parameters

daqmx Specify the device descriptor.

Description

Stops the FIFO.

Return value

Returns an error number.

Error:

Not descriptor No device descriptor.

Reference

CDAQMX::stopFIFO

talkChDataInstMX

Syntax

```
int talkChDataInstMX(DAQMX daqmx, int chNo);
```

Declaration

```
Public Declare Function talkChDataInstMX Lib "DAQMX" (ByVal daqmx As Long, ByVal chNo As Long) As Long
```

Parameters

daqmx	Specify the device descriptor.
chNo	Specify the channel number.

Description

Declares the retrieval of the measured data of instantaneous values of the specified channel number.

- For a description of the retrieval operation, see the talkChDataMX function.

Return value

Returns an error number.

Reference

talkChDataMX

talkChDataMX [Visual C only]

Syntax

```
int talkChDataMX(DAQMX daqmx, int chNo, MXDataNo startDataNo,
MXDataNo endDataNo);
```

Parameters

daqmx	Specify the device descriptor.
chNo	Specify the channel number.
startDataNo	Specify the start data number.
endDataNo	Specify the end data number.

Description

Declares the retrieval of the measured data of the specified channel number.

- Specify the range to be retrieved using the start and end data numbers.
- When retrieving instantaneous values, specify the constant for “Data number for specifying instantaneous values” for the start/end data number. Or, use the talkChDataInstMX function.
- After executing this function, use the getTimeDataMX function to retrieve the data times for all the data points. Then, use the getChDataMX function to retrieve the measured data of all the data points.
- In Visual Basic, use the talkChDataVBMX function.

Return value

Returns an error number.

Error:

Not descriptor No device descriptor.

Reference

CDAQMX::talkChData

talkChDataVBMX

Syntax

```
int talkChDataVBMX(DAQMX daqmx, int chNo, MXDataNo *  
startDataNo, MXDataNo * endDataNo);
```

Declaration

```
Public Declare Function talkChDataVBMX Lib "DAQMX" (ByVal daqmx  
As Long, ByVal chNo As Long, ByRef startDataNo As MXDataNo,  
ByRef endDataNo As MXDataNo) As Long
```

Parameters

daqmx	Specify the device descriptor.
chNo	Specify the channel number.
startDataNo	Specify the start data number.
endDataNo	Specify the end data number.

Description

Declares the retrieval of the measured data of the specified channel number.

- Except for reference specification, the data number specification is the same as the talkChDataMX function.
- In Visual C, if NULL is specified for a data number, it is considered to be the constant for "Data number for specifying instantaneous values."
- For a description of the retrieval operation, see the talkChDataMX function.

Return value

Returns an error number.

Reference

talkChDataMX

talkChInfoMX

Syntax

```
int talkChInfoMX(DAQMX daqmx, int startChNo, int endChNo);
```

Declaration

```
Public Declare Function talkChInfoMX Lib "DAQMX" (ByVal daqmx  
As Long, ByVal startChNo As Long, ByVal endChNo As Long) As  
Long
```

Parameters

daqmx	Specify the device descriptor.
startChNo	Specify the start channel number.
endChNo	Specify the end channel number.

Description

Declares the retrieval of the channel information data.

- After executing this function, use the getChInfoMX function to retrieve the data for each channel.

Return value

Returns an error number.

Error:

Not descriptor No device descriptor.

Reference

CDAQMX::talkChInfo

talkConfigMX

Syntax

```
int talkConfigMX(DAQMX daqmx, MXSystemInfo * pMXSystemInfo,
MXStatus * pMXStatus, MXNetInfo * pMXNetInfo);
```

Declaration

```
Public Declare Function talkConfigMX Lib "DAQMX" (ByVal daqmx
As Long, ByRef pMXSystemInfo As MXSystemInfo, ByRef pMXStatus
As MXStatus, ByRef pMXNetInfo As MXNetInfo) As Long
```

Parameters

daqmx	Specify the device descriptor.
pMXSystemInfo	Specify the destination where the system configuration data is to be returned.
pMXStatus	Specify the destination where the status is to be returned.
pMXNetInfo	Specify the destination where the network information data is to be returned.

Description

Declares the retrieval of the setup data.

- Stores the system configuration data, status, and network information of the setup data in the specified return destination.
- After executing this function, use the getChConfigMX function to retrieve the channel setup data of all the channels.
- Within the setup data, initial balance data and output channel data is retrieved using a separately-named retrieval function.

Return value

Returns an error number.

Error:

Not descriptor No device descriptor.

Reference

```
CDAQMX::talkConfig
CDAQMXNetInfo::getMXNetInfo
CDAQMXStatus::getMXStatus
CDAQMXSysInfo::getMXSystemInfo
```

talkFIFODataInstMX

Syntax

```
int talkFIFODataInstMX(DAQMx daqmx, int fifoNo);
```

Declaration

```
Public Declare Function talkFIFODataInstMX Lib "DAQMX" (ByVal daqmx As Long, ByVal fifoNo As Long) As Long
```

Parameters

daqmx	Specify the device descriptor.
fifoNo	Specify the FIFO number.

Description

Declares the retrieval of the measured data of instantaneous values of the specified FIFO number.

- For a description of the retrieval operation, see the talkChDataMX function.

Return value

Returns an error number.

Reference

talkFIFODataMX

talkFIFODataMX [Visual C Only]

Syntax

```
int talkFIFODataMX(DAQMX daqmx, int fifoNo, MXDataNo
startDataNo, MXDataNo endDataNo);
```

Parameters

daqmx	Specify the device descriptor.
fifoNo	Specify the FIFO number.
startDataNo	Specify the start data number.
endDataNo	Specify the end data number.

Description

Declares the retrieval of the measured data of the specified FIFO number.

- Specify the range to be retrieved using the start and end data numbers.
- When retrieving instantaneous values, specify the constant for “Data number for specifying instantaneous values” for the start/end data number. Or, use the talkFIFODataInstMX function.
- After executing this function, use the getTimeDataMX function to retrieve the data times for all the data points. Then, use the getChDataMX function to retrieve the measured data of all the data points.
- In Visual Basic, use the talkFIFODataVBMX function.

Return value

Returns an error number.

Error:

Not descriptor No device descriptor.

Reference

CDAQMX::talkFIFOData

talkFIFODataVBMX

Syntax

```
int talkFIFODataVBMX(DAQMX daqmx, int fifoNo, MXDataNo *  
startDataNo, MXDataNo * endDataNo);
```

Declaration

```
Public Declare Function talkFIFODataVBMX Lib "DAQMX" (ByVal  
daqmx As Long, ByVal fifoNo As Long, ByRef startDataNo As  
MXDataNo, ByRef endDataNo As MXDataNo) As Long
```

Parameters

daqmx	Specify the device descriptor.
fifoNo	Specify the FIFO number.
startDataNo	Specify the start data number.
endDataNo	Specify the end data number.

Description

Declares the retrieval of the measured data of the specified FIFO number.

- Except for reference specification, the data number specification is the same as the talkFIFODataMX function.
- In Visual C, if NULL is specified for a data number, it is considered to be the constant for "Data number for specifying instantaneous values."
- For a description of the retrieval operation, see the talkFIFODataMX function.

Return value

Returns an error number.

Reference

talkFIFODataMX

toAlarmNameMX

Syntax

```
int toAlarmNameMX(int iAlarmType, char * strAlarm, int lenAlarm);
```

Declaration

```
Public Declare Function toAlarmNameMX Lib "DAQMX" (ByVal iAlarmType As Long, ByVal strAlarm As String, ByVal lenAlarm As Long) As Long
```

Parameters

iAlarmType	Specify the alarm type.
strAlarm	Specify the field where the string is to be stored.
lenAlarm	Specify the byte size of the field where the string is to be stored.

Description

Stores the string corresponding to the specified alarm type to the specified field.

- The string stored in the field includes the terminator (NULL).
- The return value is the length of the actual string. The return value does not include the terminator.

Return value

Returns the length of the string.

Reference

CDAQMXDataInfo::getAlarmName

toAOPWMValueMX

Syntax

```
int toAOPWMValueMX(double realValue, int iRangeAOPWM);
```

Declaration

```
Public Declare Function toAOPWMValueMX Lib "DAQMX" (ByVal  
realValue As Double, ByVal iRangeAOPWM As Long) As Long
```

Parameters

realValue	Specify the actual output value.
iRangeAOPWM	Specify the range type.

Description

Converts the actual output values to AO/PWM data output data values according to the specified range type.

- Valid range types are AO and PWM.
- Returns 0 if the value is unknown.

Return value

Returns the output data value.

Reference

CDAQMXAOPWMData::toAOPWMValue

toDateTimeMX

Syntax

```
void toDateTimeMX(MXDateTime * pMXDateTime, int * pYear, int *  
pMonth, int * pDay, int * pHour, int * pMinute, int *  
pSecond);
```

Declaration

```
Public Declare Sub toDateTimeMX Lib "DAQMX" (ByRef pMXDateTime  
As MXDateTime, ByRef pYear As Long, ByRef pMonth As Long,  
ByRef pDay As Long, ByRef pHour As Long, ByRef pMinute As  
Long, ByRef pSecond As Long)
```

Parameters

pMXDateTime	Specify the time information.
pYear	Specify the destination where the year value is returned.
pMonth	Specify the destination where the month value is returned.
pDay	Specify the destination where the day value is returned.
pHour	Specify the destination where the hour value is returned.
pMinute	Specify the destination where the minute value is returned.
pSecond	Specify the destination where the second value is returned.

Description

Converts the number of seconds with respect to Jan. 1, 1970 in the specified time information into year, month, day, hour, minute, and second values.

- A four digit value is returned for the year. A value between 1 and 12 is returned for the month. A value between 1 and 31 is returned for the day. A value between 0 and 23 is returned for the hour. A value between 0 and 59 is returned for the minute. A value between 0 and 59 is returned for the second.

Reference

CDAQDateTime::toLocalDateTime

toDoubleValueMX

Syntax

```
double toDoubleValueMX(int dataValue, int point);
```

Declaration

```
Public Declare Function toDoubleValueMX Lib "DAQMX" (ByVal  
dataValue As Long, ByVal point As Long) As Double
```

Parameters

dataValue	Specify the data value.
point	Specify the decimal point position.

Description

Generates the measured value from the specified data value and decimal point position.

Return value

Returns the measured value as a double-precision floating number.

Reference

CDAQDataInfo::toDoubleValue

toErrorMessageMX

Syntax

```
int toErrorMessageMX(int errCode, char * errStr, int errLen);
```

Declaration

```
Public Declare Function toErrorMessageMX Lib "DAQMX" (ByVal  
errCode As Long, ByVal errStr As String, ByVal errLen As Long)  
As Long
```

Parameters

errCode	Specify the error number.
errStr	Specify the field where the string is to be stored.
errLen	Specify the byte size of the field where the string is to be stored.

Description

Stores the error message string corresponding to the error number in the specified field.

- The string stored in the field includes the terminator (NULL).
- The return value is the length of the actual string. The return value does not include the terminator.

Return value

Returns the length of the string.

Reference

getErrorMessageMX

toRealValueMX

Syntax

```
double toRealValueMX(int iAOPWMValue, int iRangeAOPWM);
```

Declaration

```
Public Declare Function toRealValueMX Lib "DAQMX" (ByVal  
iAOPWMValue As Long, ByVal iRangeAOPWM As Long) As Double
```

Parameters

iAOPWMValue	Specify the output data value.
iRangeAOPWM	Specify the range type.

Description

Converts the output data of AO/PWM data to actual output values according to the specified range type.

- Valid range types are AO and PWM.
- Returns 0 if the value is unknown.

Return value

Returns the actual output value.

Reference

CDAQMXAOPWMData::toRealValue

toStringValueMX

Syntax

```
int toStringValueMX(int dataValue, int point, char * strValue,  
int lenValue);
```

Declaration

```
Public Declare Function toStringValueMX Lib "DAQMX" (ByVal  
dataValue As Long, ByVal point As Long, ByVal strValue As  
String, ByVal lenValue As Long) As Long
```

Parameters

dataValue	Specify the data value.
point	Specify the decimal point position.
strValue	Specify the field where the string is to be stored.
lenValue	Specify the byte size of the field where the string is to be stored.

Description

Generates the measured value from the specified data value and decimal point position.

- Converts the generated measured value into a string and stores it in the specified field.
- The string stored in the field includes the terminator (NULL).
- The return value is the length of the actual string. The return value does not include the terminator.

Return value

Returns the length of the string.

Reference

CDAQDataInfo::toStringValue

toStyleVersionMX

Syntax

```
int toStyleVersionMX(int style)
```

Declaration

```
Public Declare Function toStyleVersionMX Lib "DAQMX" (ByVal  
style As Long) As Long
```

Parameters

style Specify the style.

Description

Gets the version number from the system configuration style.

Return value

Returns the style version.

Reference

CDAQMXSysInfo::toStyleVersion

6.1 Overview of the MX100 Constants

The API provides the following types of constants. The constants are common to Visual C++, Visual C, and Visual Basic.

Type	Description	Page
Communication constants	Communication port number of the MX100	6-3
Enumeration constants	Number of modules, etc.	6-3
Maximum values	Maximum length of the tag string, etc.	6-3
Constants	Data numbers for specifying the instantaneous values, etc.	6-4
Boolean value	Valid (ON) setting or Invalid (OFF) setting	6-4
Flag statuses	Identifies the last data set when data is retrieved	6-4
Data status values	Status of the measured data	6-4
Alarm types	Upper-limit alarm, etc.	6-5
System control types	System control operation	6-5
Channel kinds	Universal input, digital input, etc.	6-5
Scale types	No scaling or linear scale	6-6
Module types	4-CH Universal Input, etc.	6-6
Channel numbers	4 or 10	6-6
Interval types	10 ms to 60000 ms	6-6
Filter time constants	Input filter time constant	6-7
RJC types	Internal RJC or external RJC	6-7
Burnout types	Off/Up/Down	6-7
Unit types	MX100	6-7
Terminal types	Screw terminal or clamp terminal	6-7
A/D integration time types	Auto, 50 Hz or 60 Hz	6-7
Temperature unit types	Celsius or Fahrenheit	6-8
CF write modes	Data write mode to the CF card	6-8
CF status types	CF status	6-8
Unit status values	Unit status	6-8
FIFO status values	FIFO status	6-8
Display format values	Display format of the 7-segment LED	6-8
Output types	Output range types	6-9
Selection values	Output value selections	6-9
Transmission statuses	Transmission output statuses	6-9
Initial balance results	Result of execution of initial balancing	6-9
Options	Presence/absence of options	6-9
Reference ranges	See the measurement range of the channels undergoing difference between channels computation for the measurement range of the reference channel.	6-9
DC voltage range types	20 mV, etc.	6-10
TC range types	Type R, etc.	6-10

6.1 Overview of the MX100 Constants

Type	Description	Page
RTD (1 mA) range types	Pt100, etc.	6-11
RTD (2 mA) range types	Pt100, etc.	6-13
RTD (other) range	Pt500, Pt1000	6-14
Resistance ranges	20 Ω , 200 Ω , or 2 k Ω (0.25 mA)	6-14
Digital Input (DI) ranges	Level or contact input	6-14
Digital input (DI) detailed ranges	Contact input of the 4-CH Universal Input Module, etc.	6-14
Strain ranges	2000 μ strain, 20000 μ strain, or 200000 μ strain	6-15
AO ranges	V output or mA output	6-15
PWM ranges	PWM output resolution 1 ms or 10 ms	6-15

6.2 MX100 Constants

This section describes the mnemonic for, and the meaning of the constants. For the details on the MX100 functions, see the relevant user's manual.

Communication Constant

Mnemonic	Description
DAQMX_COMMPORT	Communication port number of the MX100.

Enumeration Constants

Mnemonic	Description
DAQMX_NUMMODULE	The number of modules.
DAQMX_NUMCHANNEL	The number of channels.
DAQMX_NUMDO	The number of DO data sets.
DAQMX_NUMFIFO	The number of FIFOs.
DAQMX_NUMALARM	The number of alarms.
DAQMX_NUMSEGMENT	The number of 7-segment LEDs.
DAQMX_NUMMACADDR	The number of MAC address elements (byte count).
DAQMX_NUMAOPWM	The number of AO/PWM data.
DAQMX_NUMBALANCE	Number of initial balance data.
DAQMX_NUMOUTPUT	Number of output channel data.

Maximum Values

Mnemonic	Description
DAQMX_MAXHOSTNAMELEN	Maximum length of the host name string.
DAQMX_MAXUNITLEN	Maximum length of the unit name string.
DAQMX_MAXTAGLEN	Maximum length of the tag string.
DAQMX_MAXCOMMENTLEN	Maximum length of the comment string.
DAQMX_MAXSERIALLEN	Maximum length of the MX100 serial number string.
DAQMX_MAXPARTNOLEN	Maximum length of the part number (firmware part number) string.
DAQMX_MAXDECIMALPOINT	Maximum value of the decimal point position.
DAQMX_MAXDISPTIME	Maximum value of the 7-segment LED display time.
DAQMX_MAXPULSETIME	Maximum value of the integer multiple of the pulse interval.

The maximum length of the string does not include the terminator (NULL).

Constants

Mnemonic	Description
DAQMX_INSTANTANEOUS	Data number when specifying the retrieval of the instantaneous value.
DAQMX_REFCHNO_ALL	Designation of all reference channel numbers.
DAQMX_LEVELNO_ALL	Designation of all alarm level numbers.
DAQMX_DONO_ALL	Designation of all DO numbers.
DAQMX_SEGMENTNO_ALL	Designation of all segment numbers of the 7-segment LED.
DAQMX_CHNO_ALL	Designation of all channel numbers.
DAQMX_MODULENO_ALL	Designation of all module numbers.
DAQMX_FIFONO_ALL	Designation of all FIFO numbers.
DAQMX_AOPWMNO_ALL	Designation of all AO/PWM data numbers.
DAQMX_BALANCENO_ALL	Designation of all initial balance numbers.
DAQMX_OUTPUTNO_ALL	Designation of all output data numbers.
DAQMX_REFCHNO_NONE	Undefined reference channel numbers.

Boolean Value (Valid/Invalid Value)

Mnemonic	Description
DAQMX_VALID_OFF	Invalid (OFF) value.
DAQMX_VALID_ON	Valid (ON) value.

Flag Statuses

Can be synthesized using logical OR operators.

Mnemonic	Description
DAQMX_FLAG_OFF	All OFF.
DAQMX_FLAG_ENDDATA	The data retrieved in units of channels or data numbers is the last data set.

Data Status Values

Mnemonic	Description
DAQMX_DATA_UNKNOWN	Unknown.
DAQMX_DATA_NORMAL	Normal.
DAQMX_DATA_PLUSOVER	Positive overrange.
DAQMX_DATA_MINUSOVER	Negative overrange.
DAQMX_DATA_SKIP	SKIP (not used).
DAQMX_DATA_ILLEGAL	Illegal data status.
DAQMX_DATA_NODATA	No data status.
DAQMX_DATA_LACK	Data dropout status
DAQMX_DATA_INVALID	Invalid status.

Alarm Types

◇ indicates a space.

Mnemonic	Description	String
DAQMX_ALARM_NONE	Alarm OFF	◇◇
DAQMX_ALARM_UPPER	Upper limit alarm	H◇
DAQMX_ALARM_LOWER	Lower limit alarm	L◇
DAQMX_ALARM_UPDIFF	Difference upper limit alarm	dH
DAQMX_ALARM_LOWDIFF	Difference lower limit alarm	dL

System Control Types

Mnemonic	Description
DAQMX_SYSTEM_RECONSTRUCT	System reconstruction
DAQMX_SYSTEM_INITOPE	System initialization
DAQMX_SYSTEM_RESEALARM	Alarm reset (alarm acknowledge)

Channel Types

Mnemonic	Description
DAQMX_CHKIND_NONE	Not used
DAQMX_CHKIND_AI	AI*
DAQMX_CHKIND_AIDIFF	AI* (difference between channels computation designation)
DAQMX_CHKIND_AIRJC	AI* (remote RJC channel)
DAQMX_CHKIND_DI	DI*
DAQMX_CHKIND_DIDIFF	DI* (difference between channel computation designation)
DAQMX_CHKIND_DO	DO* (alarm output designation)
DAQMX_CHKIND_DOCOM	DO* (command DO designation)
DAQMX_CHKIND_DOFAIL	DO* (system failure output designation)
DAQMX_CHKIND_DOERR	DO* (system error output designation)
DAQMX_CHKIND_AO	AO* (transmission output)
DAQMX_CHKIND_AOCOM	AO* (command AO)
DAQMX_CHKIND_PWM	PWM* (transmission output)
DAQMX_CHKIND_PWMCOM	PWM* (command PWM)

- * AI: Analog input, DC voltage input, TC input, etc.
- AO: Analog output, analog output module
- DI: Digital input, digital input module
- DO: Digital output, digital output module
- PWM: Pulse Width Modulation, PWM output module

For input channels, the channel types of the input channel is set using the range setting function.

For output channels, the channel type is set using the channel setting function.

Scale Types

Mnemonic	Description
DAQMX_SCALE_NONE	No scale
DAQMX_SCALE_LINER	Linear scale

Module Types

Mnemonic	Description
DAQMX_MODULE_NONE	None
DAQMX_MODULE_MX110UNVH04	4-CH, High-Speed Universal Input Module
DAQMX_MODULE_MX110UNVM10	10-CH, Medium-Speed Universal Input Module
DAQMX_MODULE_MX115D05H10	10-CH, High-Speed Digital Input Module
DAQMX_MODULE_MX125MKCM10	10-CH, Medium-Speed Digital Output Module
DAQMX_MODULE_MX110V4RM06	Six-Channel, Medium-Speed Four-Wire RTD Resistance Input Module
DAQMX_MODULE_MX112NDIM04	4-CH Strain Input Module (NDIS)
DAQMX_MODULE_MX110V4RM06	4-CH Strain Input Module (350Ω)
DAQMX_MODULE_MX112B12M04	4-CH Strain Input Module (20Ω)
DAQMX_MODULE_MX115D24H10	10-CH, High-Speed Digital Input Module (DC24 V)
DAQMX_MODULE_MX120VAOM08	8-CH, Medium-Speed Analog Output Module
DAQMX_MODULE_MX120PWMM08	8-CH, Medium-Speed PWM Output Module

Channel Numbers

Mnemonic	Description
DAQMX_CHNUM_0	0
DAQMX_CHNUM_4	4
DAQMX_CHNUM_6	6
DAQMX_CHNUM_8	8
DAQMX_CHNUM_10	10

Interval Types

Mnemonic	Description
DAQMX_INTERVAL_10	10 msec
DAQMX_INTERVAL_50	50 msec
DAQMX_INTERVAL_100	100 msec
DAQMX_INTERVAL_200	200 msec
DAQMX_INTERVAL_500	500 msec
DAQMX_INTERVAL_1000	1000 msec
DAQMX_INTERVAL_2000	2000 msec
DAQMX_INTERVAL_5000	5000 msec
DAQMX_INTERVAL_10000	10000 msec
DAQMX_INTERVAL_20000	20000 msec
DAQMX_INTERVAL_30000	30000 msec
DAQMX_INTERVAL_60000	60000 msec

Filter Coefficient

Mnemonic	Description
DAQMX_FILTER_0	Coefficient 0
DAQMX_FILTER_5	Coefficient 5
DAQMX_FILTER_10	Coefficient 10
DAQMX_FILTER_20	Coefficient 20
DAQMX_FILTER_25	Coefficient 25
DAQMX_FILTER_40	Coefficient 40
DAQMX_FILTER_50	Coefficient 50
DAQMX_FILTER_100	Coefficient 100

RJC Types

Mnemonic	Description
DAQMX_RJC_INTERNAL	RJC function of the MX100
DAQMX_RJC_EXTERNAL	External RJC function

Burnout Types

Mnemonic	Description
DAQMX_BURNOUT_OFF	No burnout detection function
DAQMX_BURNOUT_UP	Display +OVER when burnout is detected
DAQMX_BURNOUT_DOWN	Display .OVER when burnout is detected

Unit Type Logic

Can be synthesized using OR computations

Mnemonic	Description
DAQMX_UNITTYPE_NONE	Unknown
DAQMX_UNITTYPE_MX100	MX100

Terminal Types

Mnemonic	Description
DAQMX_TERMINAL_SCREW	Screw terminal
DAQMX_TERMINAL_CLAMP	Clamp terminal
DAQMX_TERMINAL_NDIS	NDIS

A/D Integration Time Type

Mnemonic	Description
DAQMX_INTEGRAL_AUTO	Auto (MX100 automatically sets 50 or 60 Hz)
DAQMX_INTEGRAL_50HZ	50 Hz
DAQMX_INTEGRAL_60HZ	60 Hz

Temperature Unit Types

Mnemonic	Description
DAQMX_TEMPUNIT_C	°C
DAQMX_TEMPUNIT_F	°F

CF Write Modes

Mnemonic	Description
DAQMX_CFWRITEMODE_ONCE	No overwriting (stops writing when there is no more free space)
DAQMX_CFWRITEMODE_FIFO	Repeat (overwrite from the oldest data)

CF Status Types

Can be synthesized using logical OR computations

Mnemonic	Description
DAQMX_CFSTATUS_NONE	All OFF
DAQMX_CFSTATUS_EXIST	Presence or absence
DAQMX_CFSTATUS_USE	CF card is usable.
DAQMX_CFSTATUS_FORMAT	CF card is being formatted.

Unit Status Values

Mnemonic	Description
DAQMX_UNITSTAT_NONE	Unknown
DAQMX_UNITSTAT_INIT	Initializing
DAQMX_UNITSTAT_STOP	Stopped
DAQMX_UNITSTAT_RUN	Measuring
DAQMX_UNITSTAT_BACKUP	Measuring (backing up)

FIFO Status Values

Mnemonic	Description
DAQMX_FIFOSTAT_NONE	Unknown
DAQMX_FIFOSTAT_INIT	Initializing
DAQMX_FIFOSTAT_STOP	Stopped
DAQMX_FIFOSTAT_RUN	Measuring
DAQMX_FIFOSTAT_BACKUP	Measuring (backing up)

Display Format Values

Mnemonic	Description
DAQMX_DISPTYPE_NONE	Undefined
DAQMX_DISPTYPE_ON	ON
DAQMX_DISPTYPE_BLINK	Blinking

Output Types

Mnemonic	Description	Setting range
DAQMX_OUTPUT_NONE	No output	
DAQMX_OUTPUT_AO_10V	V output	-11.000 to 1.000 V
DAQMX_OUTPUT_AO_20MA	mA output	0 to 2.000 mA
DAQMX_OUTPUT_PWM_1MS	PWM output resolution 1 ms	0 to 00.000 %
DAQMX_OUTPUT_PWM_10MS	PWM output resolution 10 ms	0 to 00.000 %

Corresponds to the output range type.

Selected Values

Mnemonic	Description
DAQMX_CHOICE_PREV	Previous value
DAQMX_CHOICE_PRESET	Specified time

Transmission Statuses

Mnemonic	Description
DAQMX_TRANSMIT_NONE	No specification (unknown)
DAQMX_TRANSMIT_RUN	Output start (outputting)
DAQMX_TRANSMIT_STOP	Output stop

Initial balance Results

Mnemonic	Description
DAQMX_BALANCE_NONE	No specification
DAQMX_BALANCE_DONE	Concluded successfully
DAQMX_BALANCE_NG	Out of range
DAQMX_BALANCE_ERROR	Error

Options

Mnemonic	Description
DAQMX_OPTION_NONE	No option
DAQMX_OPTION_DS	Dual Save (/DS option)

Range Type

Reference range

Mnemonic	Description
DAQMX_RANGE_REFERENCE	Measurement range of the reference channel.

If this constant is specified as the measurement range of the difference computation channel, the measurement range of the difference computation channel is set to the same range as the measurement range of the reference channel.

The reference range is used for specification when you want to set the same range as the reference channels to be referenced in difference computation and other calculations.

DC Voltage Range Types

Mnemonic	Description	Setting range
DAQMX_RANGE_VOLT_20MV	20 mV	-20.000 to 20.000 mV
DAQMX_RANGE_VOLT_60MV	60 mV	-60.00 to 60.00 mV
DAQMX_RANGE_VOLT_200MV	200 mV	-200.00 to 200.00 mV
DAQMX_RANGE_VOLT_2V	2 V	-2.0000-2.0000 V
DAQMX_RANGE_VOLT_6V	6 V	-6.000-6.000 V
DAQMX_RANGE_VOLT_20V	20 V	-20.000 to 20.000 V
DAQMX_RANGE_VOLT_100V	100 V	-100.00 to 100.00 V
DAQMX_RANGE_VOLT_60MVH	60 mV:High resolution	0.000 to 60.000 mV
DAQMX_RANGE_VOLT_1V	1 V	-10000 to 1.0000 V
DAQMX_RANGE_VOLT_6VH	6 V:High resolution	0.0000 to 6.0000 V

TC Range Types

Mnemonic	Description	Setting range
DAQMX_RANGE_TC_R	R	0.0 to 1760.0°C 32 to 3200°F
DAQMX_RANGE_TC_S	S	0.0 to 1760.0°C 32 to 3200°F
DAQMX_RANGE_TC_B	B	0.0 to 1820.0°C 32 to 3308°F
DAQMX_RANGE_TC_K	K	-200.0 to 1370.0°C -328 to 2498°F
DAQMX_RANGE_TC_E	E	-200.0 to 800.0°C -328.0 to 1472.0°F
DAQMX_RANGE_TC_J	J	-200.0 to 1100.0°C -328.0 to 2012.0°F
DAQMX_RANGE_TC_T	T	-200.0 to 400.0°C -328.0 to 752.0°F
DAQMX_RANGE_TC_N	N	0.0 to 1300.0°C 32 to 2372°F
DAQMX_RANGE_TC_W	W	0.0 to 2315.0°C 32 to 4199°F
DAQMX_RANGE_TC_L	L	-200.0 to 900.0°C -328.0 to 1652.0°F
DAQMX_RANGE_TC_U	U	-200.0 to 400.0°C -328.0 to 752.0°F
DAQMX_RANGE_TC_KP	KpAu7Fe	0.0 to 300.0K 0.0 to 300.0K
DAQMX_RANGE_TC_PL	PLATINEL	0.0 to 1400.0°C 32 to 2552°F
DAQMX_RANGE_TC_PR	PR40-20	0.0 to 1900.0°C 32 to 3452°F
DAQMX_RANGE_TC_NNM	NiNiMo	0.0 to 1310.0°C 32 to 2390°F
DAQMX_RANGE_TC_WR	WRe3-25	0.0 to 2400.0°C 32 to 4352°F
DAQMX_RANGE_TC_WWR	W/WRe26	0.0 to 2400.0°C 32 to 4352°F
DAQMX_RANGE_TC_AWG	Type-N(AWG14)	0.0 to 1300.0°C 32 to 2372°F

RTD (1 mA) Range Types

Mnemonic	Description	Setting range
DAQMX_RANGE_RTD_1MAPT	Pt100	-200.0 to 600.0°C -328.0 to 1112.0°F
DAQMX_RANGE_RTD_1MAJPT	JPt100	-200.0 to 550.0°C -328.0 to 1022.0°F
DAQMX_RANGE_RTD_1MAPTH	Pt100:high resolution	-140.00 to 150.00°C -220.0 to 302.0°F
DAQMX_RANGE_RTD_1MAJPTH	JPt100:high resolution	-140.00 to 150.00°C -220.0 to 302.0°F
DAQMX_RANGE_RTD_1MANIS	Ni100:SAMA	-200.0 to 250.0°C -328.0 to 482.0°F
DAQMX_RANGE_RTD_1MANID	Ni100:DIN	-60.0 to 180.0°C -76.0 to 356.0°F
DAQMX_RANGE_RTD_1MANI120	Ni120	-70.0 to 200.0°C -94.0 to 392.0°F
DAQMX_RANGE_RTD_1MAPT50	Pt50	-200.0 to 550.0°C -328.0 to 1022.0°F
DAQMX_RANGE_RTD_1MACU10GE	Cu10:GE	-200.0 to 300.0°C -328.0 to 572.0°F
DAQMX_RANGE_RTD_1MACU10LN	Cu10:L&N	-200.0 to 300.0°C -328.0 to 572.0°F
DAQMX_RANGE_RTD_1MACU10WEED	Cu10:WEED	-200.0 to 300.0°C -328.0 to 572.0°F
DAQMX_RANGE_RTD_1MACU10BAILEY	Cu10:BAILEY	-200.0 to 300.0°C -328.0 to 572.0°F
DAQMX_RANGE_RTD_1MAJ263B	J263*B	0.0 to 300.0 K 0.0 to 300.0 K
DAQMX_RANGE_RTD_1MACU10A392	Cu10 at 20°C a=0.00392	-200.0 to 300.0°C -328.0 to 572.0°F
DAQMX_RANGE_RTD_1MACU10A393	Cu10 at 20°C a=0.00393	-200.0 to 300.0°C -328.0 to 572.0°F
DAQMX_RANGE_RTD_1MACU25	Cu25 at 0°C a=0.00425	-200.0 to 300.0°C -328.0 to 572.0°F
DAQMX_RANGE_RTD_1MACU53	Cu53 at 0°C a=0.00426035	-50.0 to 150.0°C 58.0 to 302.0°F
DAQMX_RANGE_RTD_1MACU100	Cu100 at 0°C a=0.00425	-50.0 to 150.0°C -58.0 to 302.0°F

6.2 MX100 Constants

Mnemonic	Description	Setting range
DAQMX_RANGE_VOLT_200MV	Pt25	-200.0 to 550.0°C -328.0 to 1022.0°F
DAQMX_RANGE_RTD_1MACU10GEH	Cu10:GE: high resolution	-200.0 to 300.0°C -328.0 to 572.0°F
DAQMX_RANGE_RTD_1MACU10LNH	Cu10:L&N: high resolution	-500.0 to 500.0°C -328.0 to 572.0°F
DAQMX_RANGE_RTD_1MACU10WEEDH	Cu10:WEED: high resolution	-500.0 to 500.0°C -328.0 to 572.0°F
DAQMX_RANGE_RTD_1MACU10BAILEYH	Cu10:BAILEY: high resolution	-500.0 to 500.0°C -328.0 to 572.0°F
DAQMX_RANGE_RTD_1MAPTN	Pt100: high noise resistance	-800.0 to 800.0°C -328.0 to 1112.0°F
DAQMX_RANGE_RTD_1MAJPTN	Jpt100: high noise resistance	-750.0 to 750.0°C -328.0 to 1022.0°F

RTD (2 mA) Range Types

Mnemonic	Description	Setting range
DAQMX_RANGE_RTD_2MAPT	Pt100	-200.0 to 250.0°C -328.0 to 482.0°F
DAQMX_RANGE_RTD_2MAJPT	JPt100	-200.0 to 250.0°C -328.0 to 482.0°F
DAQMX_RANGE_RTD_2MAPTH	Pt100: high resolution	-140.00 to 150.00°C -220.0 to 302.0°F
DAQMX_RANGE_RTD_2MAJPTH	JPt100: high resolution	-140.00 to 150.00°C -220.0 to 302.0°F
DAQMX_RANGE_RTD_2MAPT50	Pt50	-200.0 to 550.0°C -328.0 to 1022.0°F
DAQMX_RANGE_RTD_2MACU10GE	CU10:GE	-200.0 to 300.0°C -328.0 to 572.0°F
DAQMX_RANGE_RTD_2MACU10LN	Cu10:L&N	-200.0 to 300.0°C -328.0 to 572.0°F
DAQMX_RANGE_RTD_2MACU10WEED	Cu10:WEED	-200.0 to 300.0°C -328.0 to 572.0°F
DAQMX_RANGE_RTD_2MACU10BAILEY	Cu10:BAILEY	-200.0 to 300.0°C -328.0 to 572.0°F
DAQMX_RANGE_RTD_2MAJ263B	J263*B	0.0-300.0K 0.0 to 300.0K
DAQMX_RANGE_RTD_2MACU10A392	Cu10 at 20°C a=0.00392	-200.0 to 300.0°C -328.0 to 572.0°F
DAQMX_RANGE_RTD_2MACU10A393	Cu10 at 20°C a=0.00393	-200.0 to 300.0°C -328.0 to 572.0°F
DAQMX_RANGE_RTD_2MACU25	Cu25 at 0°C a=0.00425	-200.0 to 300.0°C -328.0 to 572.0°F
DAQMX_RANGE_RTD_2MACU53	Cu53 at 0°C a=0.00426035	-50.0 to 150.0°C -58.0 to 302.0°F
DAQMX_RANGE_RTD_2MACU100	Cu100 at 0°C a=0.00425	-50.0 to 150.0°C -58.0 to 302.0°F
DAQMX_RANGE_RTD_2MAPT25	Pt25	-200.0 to 550.0°C -328.0 to 1022.0°F
DAQMX_RANGE_RTD_2MACU10GEH	CU10:GE: high resolution	-200.0 to 300.0°C -328.0 to 572.0°F
DAQMX_RANGE_RTD_2MACU10LNH	Cu10:L&N: high resolution	-200.0 to 300.0°C -328.0 to 572.0°F
DAQMX_RANGE_RTD_2MACU10WEEDH	Cu10:WEED: high resolution	-200.0 to 300.0°C -328.0 to 572.0°F

6.2 MX100 Constants

Mnemonic	Description	Setting range
DAQMX_RANGE_RTD_2MACU10BAILEYH	Cu10:BAILEY: high resolution	–200.0 to 300.0°C –328.0 to 572.0°F
DAQMX_RANGE_RTD_2MAPTN	Pt100: high noise resistance	–200.0 to 250.0°C –328.0 to 482.0°F
DAQMX_RANGE_RTD_2MAJPTN	Jpt100: high noise resistance	–200.0 to 250.0°C –328.0 to 482.0°F

RTD (Other) Ranges

Mnemonic	Description	Setting range
DAQMX_RANGE_RTD_025MAPT500	0.25 mA Pt500	–200.0 to 600.0 ° –328.0 to 1112.0°F
DAQMX_RANGE_RTD_025MAPT1K	0.25 mA Pt1000	–200.0 to 600.0 ° –328.0 to 1112.0°F

Resistance Ranges

Mnemonic	Description	Setting range
DAQMX_RANGE_RES_20	20 Ω	0 to 20.000
DAQMX_RANGE_RES_200	200 Ω	0 to 200.00
DAQMX_RANGE_RES_2K	2kΩ (0.25mA)	0 to 2000.0

Digital Input (DI) Range Types

Mnemonic	Description	Setting range
DAQMX_RANGE_DI_LEVEL	Level	0: Less than 2.4 V, 1: Greater than or equal to 2.4 V
DAQMX_RANGE_DI_CONTACT	Contact input	0:open, 1:close

Digital Input (DI) Detailed Range Types

Mnemonic	Description
DAQMX_RANGE_DI_LEVEL_AI	DI/Level of the universal input module
DAQMX_RANGE_DI_CONTACT_AI4	DI/Contact input of the 4-CH, Universal Input Module
DAQMX_RANGE_DI_CONTACT_AI10	DI/Contact input of the 10-CH, Universal Input Module
DAQMX_RANGE_DI_LEVEL_DI	DI/Level of the digital input module
DAQMX_RANGE_DI_CONTACT_DI	DI/Contact input of the digital input module
DAQMX_RANGE_DI_LEVEL_DI5V* DI 5V	DI/contact input of the digital input module (5 V)
DAQMX_RANGE_DI_LEVEL_DI24V DI 24V	DI/contact input of the digital input module (24 V)

* Separate name for DAQMX_RANGE_DI_LEVEL_DI. Defined to differentiate from 24 V.

Strain Ranges

Mnemonic	Description	Setting range
DAQMX_RANGE_STRAIN_2K	2000 μ STR	-2000.0 to 2000.0 \pm 563200000
DAQMX_RANGE_STRAIN_20K	20000 μ STR	-20000 to 20000 \pm 56320000
DAQMX_RANGE_STRAIN_200K	200000 μ STR	-200000 to 200000 \pm 5632000

AO Ranges

Mnemonic	Description	Setting range
DAQMX_RANGE_AO_10V	V output	-10.000 to 10.000 V
DAQMX_RANGE_AO_20MA	mA output	4.000 to 20.000 mA

PWM Ranges

Mnemonic	Description	Setting range
DAQMX_RANGE_PWM_1MS	PWM output resolution 1 ms	0 to 100.000 %
DAQMX_RANGE_PWM_10MS	PWM output resolution 10 ms	0 to 100.000 %

6.3 MX100 Setting Item Numbers

These numbers are added to each item of the setup data structure to unify them. When sending the setup data, the numbers show the detected location when an error occurs during a consistency check.

A definition file is available.

List of Setting Items

Each item field of the MXConfigData structure is numbered.

Definitions are given in the files below. Load the definitions as needed.

- DAQMXItems.h
- DAQMXItems.bas
- DAQMXItems.txt
- DAQMXItems.cs
- DAQMXItems.vb

Fixed descriptions such as channel number items that have no particular meaning when read out are not numbered.

If an error occurs, check the values of settings.

The character string is used if the value of the setting item number is converted to an item name. It is for the extension API only.

In Visual C#, it is defined as a constant of the DAQMXItems class.

Systems

Mnemonic	Value	Location	Description	Error	String
DAQMX_ITEM_NONE	-1	-	Unknown	-	-
DAQMX_ITEM_NETHOST	0	aNetInfo. aHost	Host name	-	Host Name
DAQMX_ITEM_NETADDRESS	1	aNetInfo. aAddress	IP Address	-	Address
DAQMX_ITEM_NETPORT	2	aNetInfo. aPort	Port number	-	Port No
DAQMX_ITEM_NETSUBMASK	3	aNetInfo. aSubMask	Subnet mask	-	Submask
DAQMX_ITEM_NETGATEWAY	4	aNetInfo. aGateway	GATEWAY Address	-	Gateway
DAQMX_ITEM_UNITTYPE	5	aSystemInfo. aUnit.aType	Unit type	-	Unit Type
DAQMX_ITEM_UNITSTYLE	6	aSystemInfo. aUnit.aStyle	Style	-	Unit Style
DAQMX_ITEM_UNITNO	7	aSystemInfo. aUnit.aNo	Unit Number	Value is out-of-range	Unit No
DAQMX_ITEM_UNITTEMP	8	aSystemInfo. aUnit.aTempUnit	Temperature unit type	Value is out-of-range	Temperature Unit

Mnemonic	Value	Location	Description	Error	String
DAQMX_ITEM_UNITCFTIMEOUT	9	aSystemInfo. aUnit.aCFTimeout	Timeout value	Value does not match equation. See "Timeout value calculation"	CF Timeout (s)
DAQMX_ITEM_UNITCFWRITEMODE	10	aSystemInfo. aUnit.aCFWriteMode	CF write mode	Value is out of range.	CF WriteMode
DAQMX_ITEM_UNITFREQUENCY	11	aSystemInfo. aUnit.aFrequency	Power supply frequency	-	Frequency
DAQMX_ITEM_UNITPARTNO	12	aSystemInfo. aUnit.aPartNo	Part number	-	Part No
DAQMX_ITEM_UNITOPTION	13	aSystemInfo. aUnit.aProduct.aOption	Options	-	Unit Option
DAQMX_ITEM_UNITSERIAL	14	aSystemInfo. aUnit.aProduct.aSerial	Serial number	-	Unit Serial
DAQMX_ITEM_UNITMAC	15	aSystemInfo. aUnit.aProduct.aMAC	MAC address	-	Mac Address
DAQMX_ITEM_UNITSTATUS	16	aStatus. aUnitStatus	Unit status value	-	Unit Status
DAQMX_ITEM_CNTCONFIG	17	aStatus. aConfigCnt	Setup number	-	Count Config
DAQMX_ITEM_CNTTIME	18	aStatus. aTimeCnt	Time number	-	Count Time
DAQMX_ITEM_FIFONUM	19	aStatus. aFIFOum	Valid number of FIFOs	-	FIFO Num
DAQMX_ITEM_BACKUP	20	aStatus. aBackup	Presence or absence of backup	-	Backup
DAQMX_ITEM_CFSTATUS	21	aStatus. aCFInfo.aStatus	CF status type	-	CF Status
DAQMX_ITEM_CFSIZE	22	aStatus. aCFInfo.aSize	Capacity	-	CF Size (KB)
DAQMX_ITEM_CFREMAIN	23	aStatus. aCFInfo.aRemain	Remaining capacity	-	CF Remain (KB)
DAQMX_ITEM_STATUSTIME	2304	aStatus. aDateTime.aTime	No. of seconds	-	Status Time
DAQMX_ITEM_STATUSMSEC	2305	aStatus. aDateTime.aMilliSecond	Milliseconds	-	Status Millisecond

No string exists for Unknown items.

FIFO

Mnemonic	Value	Location	Description	String
DAQMX_ITEM_FIFOSTATUS	24	aStatus.aFIFOInfo[]. aStatus	FIFO status value	FIFO0 Status
DAQMX_ITEM_FIFOINTERVAL	28	aStatus.aFIFOInfo[]. aInterval	Interval types	FIFO0 Interval (msec)
DAQMX_ITEM_FIFOLDNO	32	aStatus.aFIFOInfo[]. aOldNo	Oldest data number	FIFO0 Old No
DAQMX_ITEM_FIFONEWNO	36	aStatus.aFIFOInfo[]. aNewNo	Newest data number	FIFO0 New No

The actual values are the values in the table with the FIFO number (index) added. In the string FIFO0, the FIFO number is included.

6.3 MX100 Setting Item Numbers

Modules

Mnemonic	Value	Location	Description	Error	String
DAQMX_ITEM_MODULETYPE	40	aSystemInfo .aModule[]. aType	Module type	Value is out of range.	Module0 Type
DAQMX_ITEM_MODULECHNUM	48	aSystemInfo .aModule[]. aChNum	Number of channels	Value is out of range.	Module0 Channel Num
DAQMX_ITEM_MODULEINTERVAL	56	aSystemInfo .aModule[]. aInterval	Interval types	Value is out of range. FIFO exceeded.	Module0 Interval (msec)
DAQMX_ITEM_MODULEINTEGRAL	64	aSystemInfo .aModule[]. aIntegralTime	AD integration time type	Value is out of range.	Module0 Integral
DAQMX_ITEM_MODULESTANDBY	72	aSystemInfo .aModule[]. aStandbyType	Module type at startup	-	Module0 Standby Type
DAQMX_ITEM_MODULEREALTYPE	80	aSystemInfo .aModule[]. aRealType	Actual module type	-	Module0 Real Type
DAQMX_ITEM_MODULESTATUS	88	aSystemInfo .aModule[]. aStatus	Module valid/invalid	-	Module0 Status
DAQMX_ITEM_MODULEVERSION	96	aSystemInfo .aModule[]. aVersion	Module version	-	Module0 Version
DAQMX_ITEM_MODULETERMINAL	104	aSystemInfo .aModule[]. aTerminalType	Terminal type	-	Module0 Terminal
DAQMX_ITEM_MODULEFIFONO	112	aSystemInfo .aModule[]. aFIFONo	FIFO number	-	Module0 FIFO No
DAQMX_ITEM_MODULESERIAL	120	aSystemInfo .aModule[]. aProduct.aSerial	Serial number	-	Module0 Serial

The actual values are the values in the table with the module number (index) added. In the string Module0, the module number is included.

Channels

Mnemonic	Value	Location	Description	Error	String
DAQMX_ITEM_CHVALID	128	aChConfigData. aChConfig[]. aChID.aValid	Channel status	Does not exist, but valid.	Channel00 Valid
DAQMX_ITEM_CHPOINT	192	aChConfigData. aChConfig[]. aChID.aPoint	Decimal point position	Value is out of range.	Channel00 Point
DAQMX_ITEM_CHKIND	256	aChConfigData. aChConfig[]. aChID.aKind	Channel kind	Value is out of range.	Channel00 Kind
DAQMX_ITEM_CHRANGE	320	aChConfigData. aChConfig[]. aChID.aRange	Range type	Value is out of range.	Channel00 Range
DAQMX_ITEM_CHSCALE	384	aChConfigData. aChConfig[]. aChID.aScaleType	Scale type	Value is out of range.	Channel00 Scale
DAQMX_ITEM_CHUNIT	338	aChConfigData. aChConfig[]. aChID.aUnit	Unit name	-	Channel00 Unit
DAQMX_ITEM_CHTAG	512	aChConfigData. aChConfig[]. aChID.aTag	Tag	-	Channel00 Tag
DAQMX_ITEM_CHCOMMENT	576	aChConfigData. aChConfig[]. aChID.aComment	Comment	-	Channel00 Comment

6.3 MX100 Setting Item Numbers

Mnemonic	Value	Location	Description	Error	String
DAQMX_ITEM_ALARMTYPE	640	aChConfigData. aChConfig[]. aChID.aAlarm[].aType	Alarm type	Value is out of range.	Channel00 Alarm0 Type
DAQMX_ITEM_ALARMON	768	aChConfigData. aChConfig[]. aChID.aAlarm[].aON	On value	Value is out of range.	Channel00 Alarm0 On
DAQMX_ITEM_ALARMOFF	896	aChConfigData. aChConfig[]. aChID.aAlarm[].aOFF	Off value	Value is out of range.	Channel00 Alarm0 Off
DAQMX_ITEM_CHSPANMIN	1024	aChConfigData. aChConfig[]. aAIDl.aSpanMin	Span minimum	Value is out of range.	Channel00 Span Min
DAQMX_ITEM_CHSPANMAX	1088	aChConfigData. aChConfig[]. aAIDl.aSpanMax	Span maximum	Value is out of range.	Channel00 Span Max
DAQMX_ITEM_CHSCALEMIN	1152	aChConfigData. aChConfig[]. aAIDl.aScaleMin	Scale minimum	Value is out of range.	Channel00 Scale Min
DAQMX_ITEM_CHSCALEMAX	1216	aChConfigData. aChConfig[]. aAIDl.aScaleMax	Scale maximum	Value is out of range.	Channel00 Scale Max
DAQMX_ITEM_CHREFCHNO	1280	aChConfigData. aChConfig[]. aAIDl.aRefChNo	Reference channel number	Value is out of range. Channels to be are not input channels	Channel00 Reference ChannelNo
DAQMX_ITEM_CHFILTER	1344	aChConfigData. aChConfig[]. aAl.aFilter	Filter coefficient	Value is out of range.	Channel00 Filter
DAQMX_ITEM_CHRJCTYPE	1408	aChConfigData. aChConfig[]. aAl.aRJCTYPE	RJC type	Value is out of range.	Channel00 RJC Type
DAQMX_ITEM_CHRJCVOLT	1472	aChConfigData. aChConfig[]. aAl.aRJCVolt	RJC voltage	Value is out of range.	Channel00 RJC Volt
DAQMX_ITEM_CHBURNOUT	1536	aChConfigData. aChConfig[]. aAl. aBurnout	Burnout	Value is out of range.	Channel00 Burnout
DAQMX_ITEM_CHDEENERGIZE	1600	aChConfigData. aChConfig[]. aDO.aDeenergize	De-energize	-	Channel00 Deenergize
DAQMX_ITEM_CHREFALARM	1664	aChConfigData. aChConfig[]. aDO.aRefAlarm[*][]	Reference alarm	-	Channel00 Alarm0 Reference
DAQMX_ITEM_CHHOLD	1792	aChConfigData. aChConfig[]. aDO.aHold	Hold	-	Channel00 Hold
DAQMX_ITEM_CHOUTPUTTYPE	1856	aOutputData. aOutput[]. aType	Output types	Value is out of range. Range type does not match.	Channel00 Output Type
DAQMX_ITEM_CHIDLECHOICE	1920	aOutputData. aOutput[]. aIdleChoice	Value selected when idle	Value is out of range.	Channel00 Idle Choice
DAQMX_ITEM_CHERRCHOICE	1984	aOutputData. aOutput[]. aErrorChoice	Value selected during error	Value is out of range.	Channel00 Error Choice
DAQMX_ITEM_CHPRESETVALUE	2048	aOutputData. aOutput[]. aPresetValue	Value when the selection value is the specified value.	Value is out of range.	Channel00 Preset Value
DAQMX_ITEM_CHPULSETIME	2112	aOutputData. aOutput[]. aPulseTime	Pulse interval integer multiple	Value is out of range.	Channel00 Pulse Time

6.3 MX100 Setting Item Numbers

Mnemonic	Value	Location	Description	Error	String
DAQMX_ITEM_CHBALANCEVALID	2176	aBalanceData. aBalance[]. aValid	Valid/invalid	-	Channel00 Balance Valid
DAQMX_ITEM_CHBALANCEVALUE	2240	aBalanceData. aBalance[]. aValue	Initial balance value	Value is out of range.	Channel00 Balance Value

The relationship between the maximum and minimum values of span and scale are checked, so the value that expresses the minimum value is set as the detected position.

In the string Channel00, the channel number is included. The 0 in the string Alarm0 is the alarm level number.

The actual values are the values in the table with the channel number (index) added.

In the case of alarms, the alarm level number (index) is also added to the 7th bit to the left. It is the value in which the channel number (index) is added to the values in the table below.

Alarm

Mnemonic	Alarm Level	Value	Description
DAQMX_ITEM_ALARMTYPE	1	640	Alarm type of channel number 1, alarm level 1
	2	704	Alarm type of channel number 1, alarm level 2
DAQMX_ITEM_ALARMON	1	768	ON value of channel number 1, alarm level 1
	2	832	ON value of channel number 1, alarm level 2
DAQMX_ITEM_ALARMOFF	1	896	OFF value of channel number 1, alarm level 1
	2	960	OFF value of channel number 1, alarm level 2
DAQMX_ITEM_CHREFALARM	1	1664	The channel number is 1, and the alarm level of all reference channels is 1 Expresses multiple reference channels together.
	2	1728	The channel number is 1, and the alarm level of all reference channels is 2 Expresses multiple reference channels together.

Character Strings

Notations for characters strings in item contents are as follows.

- The valid/invalid value (Boolean) is notated as OFF or ON.
- The string itself is shown as-is.
- Values are written as decimals. However, hexadecimal is used to notate significant bits when present.
- The alarm levels of reference channels are expressed as valid channel numbers (01, 02, etc.).
- The choices are given by the name of the string. They are expressed as character strings, abbreviated as much as possible while retaining their usefulness for identifying mnemonics for constants.

Ranges

Mnemonic	Description
DAQMX_ITEM_ALL_START	First number of all items
DAQMX_ITEM_ALL_END_R1	End number of all items for style version 1
DAQMX_ITEM_ALL_END_R2	End number of all items for style version 2
DAQMX_ITEM_ALL_END	End number of all items for most recent style version

Masks

Mnemonic	Description
DAQMX_MASK_BYMODULE	Module number mask
DAQMX_MASK_BYCHANNEL	Channel number mask
DAQMX_MASK_BYFIFO	FIFO number mask
DAQMX_MASK_BYALARM	Alarm level number mask

Note that the alarm level number mask shows the position of the 7th bit from the end.

6.4 Overview of the MX100 Types

The data types below are provided.

Type	Description	Page
DAQMX	Device descriptor	6-25
DAQINT64	64-bit data type definition. Can be used in Visual C/Visual C++.	6-25
MXINT64	A union that can support 64 bit data in Visual Basic and Visual C/Visual C++	6-25
MXDataNo	An alias for DAQINT64. This corresponds to MXINT64 of Visual Basic. For data number of measured data.	6-25
MXUserTime	An alias for DAQINT64. In Visual Basic, it is an alias of MXIN64. For user count.	6-25
MXDateTime	Time information structure.	6-25
MXAlarm	Alarm setup information structure.	6-26
MXDataInfo	Measured data structure.	6-26
MXChConfigAID	Structure for information related to AI and DI channels.	6-26
MXChConfigAI	Structure for information related to AI channels.	6-27
MXChConfigDO	Structure for information related to DO channels.	6-27
MXChID	Channel structure.	6-28
MXChConfig	Channel setup information structure.	6-28
MXChConfigData	Channel setup information structure for every channel.	6-29
MXChInfo	Channel information data structure.	6-29
MXProductInfo	MX product information structure	6-30
MXUnitData	Unit information structure.	6-30
MXModuleData	Module information structure.	6-31
MXSystemInfo	System configuration data structure.	6-32
MXCFInfo	CF information structure.	6-32
MXFIFOInfo	FIFO structure.	6-32
MXStatus	Status structure.	6-33
MXNetInfo	Ethernet connection information structure.	6-33
MXBalance	Structure for the initial balance Boolean.	6-34
MXBalanceData	Structure for initial balance data.	6-34
MXBalanceResult	Structure for the initial balance execution result.	6-34
MXOutput	Structure for the output value.	6-34
MXOutputData	Structure for output channel data.	6-34
MXConfigData	Setup data structure.	6-35
MXDO	DO setup structure.	6-35
MXDOData	DO data structure.	6-35
MXSegment	Display pattern structure for each 7-segment LED.	6-35
MXAOPWM	Structure for pulse output.	6-35

Type	Description	Page
MXAOPWMData	Structure for the specified number of AO/PWM data.	6-36
MXTransmit	Structure for the specified number of transmission statuses.	6-36

Type	Description
Callback type	Add prefix "DLL" to the function name and write in uppercase. Example: callback type of the openMX function: DLLOPENMX

The callback type is used to link the executable module (.dll) when using Visual C.

6.5 MX100 Types

Explanation on the Description

Visual C/Visual C++, and Visual Basic Types

Indicates the type name in Visual C/Visual C++ and Visual Basic.

Types without signs in Visual C/Visual C++ become types with signs in Visual Basic.

Number of array elements for Visual C/Visual C++ types is omitted.

Retrieval/Change

The markings below are used to indicate items that can be retrieved, those that the user can set, and so on.

System

Data retrieved using the “retrieval of system configuration data” function.

R: Item that can be retrieved.

F: Item that is set within the function.

Basic Settings

Data included in the basic settings within the setup data. Basic settings can be retrieved using the “collective retrieval of setup data” function.

AI represents analog input channels (such as the DC voltage input channels on the universal input module or the 4-wire RTD resistance input module). DI

represents digital input channels (DI channel of the universal input module or 4-wire RTD resistance input module, or the DI channel of the digital input module).

DO represents digital output (DO of the digital output module), AO represents analog output channels, and PWM represents PWM output channels.

R: Item that can be retrieved.

C: Item that the user can change.

F: Item that is set within the function.

Channel Information

Data retrieved using the “retrieval of the channel information data” function.

R: Item that can be retrieved.

Data Retrieval

Data retrieved using the “retrieval of the measured data” function.

F: Item that is set within the function.

Status

Data retrieved using the “retrieval of the status” function.

R: Item that can be retrieved.

Terminology

In the explanation of types, terminology representing MX100 functions is used.

Terminology related to the MX100 are explained in appendix 1.

Detailed Explanation of Types

DAQMX

Type for storing the device descriptor.

It is an alias for the int type in C/C++ and the Long type in Visual Basic.

DAQINT64

64-bit data type definition.

Can be used in C/C++. This type cannot be used in Visual Basic.

MXINT64

MXINT64 structure

Visual C/ Visual C++ Type	Name	Description	VB Type
struct	unsigned int aVB aLow	Corresponds to VB Lower	Long
	unsigned int aHigh	Corresponds to VB Upper	Long
DAQINT64	aC	Corresponds to C/C++ All	-

Structure for supporting 64 bit data in Visual C/Visual C++ and Visual Basic. The corresponding parts for Visual C/Visual C++ and for Visual Basic are the same. Cannot be used in Visual Basic. The following parts that correspond to Visual Basic are defined by data type.

Parts Corresponding to Visual Basic

Type	Name	Description	Visual Basic Type
unsigned int	aLow	Upper	Long
unsigned int	aHigh	Lower	Long

MXDataNo

For the data number.

In Visual C/Visual C++, it is an alias of DAQINT64.

In VB, it is the same as the part of MXINT64 that corresponds to Visual Basic.

MXUserTime

For user count.

In Visual C/Visual C++, it is an alias of DAQINT64.

In VB, it is the same as the part of MXINT64 that corresponds to Visual Basic.

MXDateTime

MXDateTime structure

Visual C/ Visual C++ Type	Name	Description	Visual Basic Type
time_t	aTime	The number of seconds from Jan. 1, 1970.	Long
int	aMilliSecond	Millisecond value.	Long

Time information data structure.

Visual C++: The wrapper class is CDAQMXDateTime.

MXAlarm

MXAlarm structure

Visual C/ Visual C++ Type	Name	Description	Visual Basic Type
int	aType	Alarm Type	Long
int	aReserve	Not used	Long
int	aON	ON value (threshold level for alarm activation)	Long
int	aOFF	OFF value (threshold level for alarm termination)	Long

Alarm setup information structure. The ON and OFF values must be converted to values corresponding to the measurement range by applying the decimal point position.

MXDataInfo

MXDataInfo structure

Visual C/ Visual C++ Type	Name	Description	Visual Basic Type
int	aValue	Data Value	Long
int	aStatus	Data status values	Long
int []	aAlarm	Alarms (presence or absence)	(1 To 2) As Long
int []	align	Not used (for consistency with the DARWIN)	(3 To 4) As Long

Measured data structure.

Visual C++: The wrapper class is CDAQMXDataInfo.

MXChConfigAIDI

MXChConfigAIDI structure

Visual C/ Visual C++ Type	Name	Description	Visual Basic Type
int	aSpanMin	Span minimum	Long
int	aSpanMax	Span maximum	Long
int	aScaleMin	Scale minimum	Long
int	aScaleMax	Scale maximum	Long
int	aRefChNo	Reference channel number	Long
int	aReserve	Not used	Long

Retrieval/Change

Name	Description	Basic Settings			
		AI	DI	AO	PWM
aSpanMin	Span minimum	RC	RC	RC	RC
aSpanMax	Span maximum	RC	RC	RC	RC
aScaleMin	Scale minimum	RC	RC		
aScaleMax	Scale maximum	RC	RC		
aRefChNo	Reference channel number	RC	RC	RC	RC

Setup data structure for AI channels and DI channels.

MXChConfigAI

MXChConfigAI structure

Visual C/C++ Type	Name	Description	Visual Basic Type
int	aFilter	Filter coefficient	Long
int	aRJCType	RJC type	Long
int	aRJCVolt	RJC Voltage	Long
int	aBurnout	Burnout	Long

Retrieval/Change

Name	Description	Basic Settings AI
aFilter	Filter time constant	RC
aRJCType	RJC type	RC
aRJCVolt	RJC Voltage	RC
aBurnout	Burnout	RC

Setup data structure for AI channels.

MXChConfigDO

MXChConfigDO structure

Visual C/ Visual C++ Type	Name	Description	Visual Basic Type
int	aDeenergize	De-energize (presence or absence)	Long
int	aHold	Hold (presence or absence)	Long
unsigned char [] []	aRefAlarm	Reference alarms	(1 To 2, 1 (presence or absence) To 60) As Byte

* The order of two-dimensional arrays are opposite between Visual C/Visual C++ and VB.

Retrieval/Change

Name	Description	Basic Settings DO
aDeenergize	De-energize (presence or absence)	RC
aHold	Hold (presence or absence)	RC
aRefAlarm	Reference alarms (presence or absence)	RC

Setup data structure for DO channels.

MXChID

MXChID structure

Visual C/ Visual C++ Type	Name	Description	Visual Basic Type
int	aChNo	Channel number	Long
int	aPoint	Decimal point position	Long
int	aValid	Channel status (presence/absence)	Long
int	aKind	Channel kind	Long
int	aRange	Range type	Long
int	aScaleType	Scale type	Long
char []	aUnit	Unit name	String * DAQMX_MAXUNITLEN
char	align1	Not used	(0 To 1) As Byte
char []	aTag	Tag	String * DAQMX_MAXTAGLEN
(None)	aNULL	Not used (terminator, for Visual Basic)	Byte
char []	aComment	Comment	String * DAQMX_MAXCOMMENTLEN
char	align2	Not used	(0 To 1) As Byte
MXAlarm []	aAlarm	Alarms	(1 To 2) As MXAlarm

Retrieval/Change

Name	Description	Channel Information	Basic settings					Data Retrieval
			AI	DI	DO	AO	PWM	
aChNo	Channel number	R	F	F	F	F	F	R
aPoint	Decimal Point Position	R	RC	RC		RC	RC	F
aValid	Channel Status (presence/absence)	F	RC	RC	RC	RC	RC	F
aKind	Channel kind	R	RC	RC	RC	RC	RC	
aRange	Range type	R	RC	RC	RC	RC	RC	
aScaleType	Scale type	R	RC	RC	RC	RC	RC	
aUnit	Unit Name	R	RC	RC	RC	RC	RC	
aTag	Tag	R	RC	RC	RC	RC	RC	
aComment	Comment	R	RC	RC	RC	RC	RC	
aAlarm	Alarms	R	RC	RC				

Channel ID information structure.

MXChConfig

MXChConfig structure

Visual C/ Visual C++ Type	Name	Description	Visual Basic Type
MXChID	aChID	Channel ID information	MXChID
MXChConfigAIDI	aAIDI	Setup data of the AI and DI	MXChConfigAIDI
MXChConfigAI	aAI	AI setup data	MXChConfigAI
MXChConfigDO	aDO	DO setup data	MXChConfigDO

Retrieval/Change

Name	Description	Basic Settings				
		AI	DI	DO	AO	PWM
aChID	Channel ID information	RC	RC	RC	RC	RC
aAIDI	Setup data of the AI and DI	RC	RC		RC	RC
aAI	AI setup data	RC				
aDO	DO setup data	RC				

Channel setup information structure.

MXChConfigData

MXChConfigData structure

Visual C/ Visual C++ Type	Name	Description	Visual Basic Type
MXChConfig []	aChConfig	Channel setup information	(None)

Structure for storing the setup information of all the channels.

Cannot be used in Visual Basic. Field for each channel must be allocated.

MXChInfo

MXChInfo structure

Visual C/ Visual C++ Type	Name	Description	Visual Basic Type
MXChID	aChID	Channel ID information	MXChID
int	aFIFONo	FIFO Number	Long
int	aFIFOIndex	Channel sequence number in the FIFO	Long
double	aOrigMin	Reference minimum value (not used)	Double
double	aOrigMax	Reference maximum value (not used)	Double
double	aDispMin	Display minimum value	Double
double	aDispMax	Display maximum value	Double
double	aRealMin	Measurable minimum value	Double
double	aRealMax	Measurable maximum value	Double

Retrieval/Change

Name	Description	Channel Information	Data Retrieval
aChID	Channel ID information	R	F
aFIFONo	FIFO Number	R	F
aFIFOIndex	Channel sequence number in the FIFO	R	F
aOrigMin	Reference minimum value (not used)	R	
aOrigMax	Reference maximum value (not used)	R	
aDispMin	Display minimum value	R	
aDispMax	Display maximum value	R	
aRealMin	Measurable minimum value	R	
aRealMax	Measurable maximum value	R	

Channel information data structure.

Visual C++: The wrapper class is CDAQMXChInfo.

MXProductInfo

MXProductInfo structure

Visual C/ Visual C++Type	Name	Description	Visual Basic Type
int	aOption	Options (Unit information only)*	Long
int	aCheck	Not used	Long
char []	aSerial	Serial number	String * DAQMX_MAXSERIALLEN
(None)	aNULL	Unused (serial number terminator, for Visual Basic)	Byte
unsigned char []	aMAC	MAC Address (Unit information only)*	(0 To 5) As Byte

* (Unit information only): Information only for the unit, not available for the module.

Retrieval/Change

Name	Description	System	Basic Settings
aOption	Option (unit information only)*	R	R
aSerial	Serial number	R	R
aMAC	MAC address (unit information only)*	R	R

MX product information structure

MXUnitData

MXUnitData structure

Visual C/ Visual C++Type	Name	Description	Visual Basic Type
int	aType	Unit type	Long
int	aStyle	Style	Long
int	aNo	Unit number	Long
int	aTempUnit	Temperature unit type	Long
int	aCFTimeout	Timeout value	Long
int	aCFWriteMode	CF write mode	Long
int	aFrequency	Power supply frequency	Long
int	aReserve	Not used	Long
char []	aPartNo	Part number	String * DAQMX_MAXPARTNOLEN
(None)	aNULL	Unused (part number terminator, for Visual Basic)	Byte
MXProductInfo	aProduct	Product information	MXProductInfo

Retrieval/Change

Name	Description	System	Basic Settings
aType	Unit type	R	R
aStyle	Style	R	R
aNo	Unit Number	R	RC
aTempUnit	Temperature unit type		RC
aCFTimeout	Timeout Value		RC
aCFWriteMode	CF write mode		RC
aFrequency	Power supply frequency	R	
aPartNo	Part number	R	
aProduct	Product information	R	R

Unit information structure.

MXModuleData

MXModuleData structure

Visual C/ Visual C++Type	Name	Description	Visual Basic Type
int	aType	Module type	Long
int	aChNum	Number of channels	Long
int	aInterval	Scan interval (msec)	Long
int	aIntegralTime	A/D integration time type	Long
int	aStandbyType	Module type at startup	Long
int	aRealType	Actual module type	Long
int	aStatus	Module valid/invalid	Long
int	aVersion	Module version	Long
int	aTerminalType	Terminal type	Long
int	aFIFONo	FIFO Number	Long
MXProductInfo	aProduct	Product information	MXProductInfo

Retrieval/Change

Name	Description	System	Basic Settings
aType	Module type	R	RC
aChNum	Number of channels	R	RC
iaInterval	Scan interval (msec)		RC
iaIntegralTime	A/D integration time type		RC
aStandbyType	Module type at startup	R	
aRealType	Actual module type	R	
aStatus	Module valid/invalid	F	
aVersion	Module version	R	
aTerminalType	Terminal type	R	
aFIFONo	FIFO Number	R	
aProduct	Product information	R	

Module information structure.

MXSystemInfo

MXSystemInfo structure

Visual C/ Visual C++ Type	Name	Description	Visual Basic Type
MXUnitData	aUnit	Unit information	MXUnitData
MXModuleData []	aModule	Array for specification of module information	(0 To 5) As MXModuleData

Retrieval/Change

Name	Description	System	Basic Settings
aUnit	Unit information	R	R
aModule	Array of module information	R	R

System configuration data structure.

Visual C++: The wrapper class is CDAQMXSysInfo.

MXCFInfo

MXCFInfo structure

Visual C/ Visual C++ Type	Name	Description	Visual Basic Type
int	aStatus	CF status type	Long
int	aSize	Size (KB)	Long
int	aRemain	Remaining size (KB)	Long
int	aReserve	Not used	Long

CF information structure.

MXFIFOInfo

MXFIFOInfo structure

Visual C/ Visual C++ Type	Name	Description	Visual Basic Type
int	aNo	FIFO number	Long
int	aStatus	FIFO status value	Long
int	aInterval	Scan interval (msec)	Long
int	aReserve	Not used	Long
MXDataNo	aOldNo	Oldest data number	MXDataNo
MXDataNo	aNewNo	Newest data number	MXDataNo

FIFO information structure.

MXStatus

MXStatus structure

Visual C/ Visual C++ Type	Name	Description	Visual Basic Type
int	aUnitStatus	Unit status value	Long
int	aConfigCnt	Setup number	Long
int	aTimeCnt	Time number	Long
int	aFIFONum	Valid number of FIFOs	Long
int	aBackup	Presence/absence of backup	Long
int	aReserve	Not used	Long
MXCFInfo	aCFInfo	CF status information	MXCFInfo
MXFIFOInfo []	aFIFOInfo	FIFO information	(0 To 2) As MXFIFOInfo
MXDateTime	aDateTime	Status return time	MXDateTime

Retrieval/Change

Name	Description	Status	Basic Settings
aUnitStatus	Unit status value	R	
aConfigCnt	Setup number	R	R
aTimeCnt	Time number	R	R
aFIFONum	Valid number of FIFOs	R	
aBackup	Presence/absence of backup		R
aCFInfo	CF status information	R	
aFIFOInfo	FIFO information	R	
aDateTime	Status return time	R	

Status structure.

Visual C++: The wrapper class is CDAQMXStatus.

MXNetInfo

MXNetInfo structure

Visual C/ Visual C++ Type	Name	Description	Visual Basic Type
unsigned int	aAddress	IP Address	Long
unsigned int	aPort	Port number	Long
unsigned int	aSubMask	Subnet Mask	Long
unsigned int	aGateway	GATEWAY address	Long
char []	aHost	Host name	String * DAQMX_MAXHOSTNAMELEN
char []	align	Not used	(0 To 7) As Byte

Retrieval/Change

Name	Description	Basic Settings
aAddress	IP Address	R
aPort	Port number	R
aSubMask	Subnet Mask	R
aGateway	GATEWAY address	R
aHost	Host name	R

Structure related to Ethernet communication settings.

MXBalance

MXBalance structure

Visual C/ Visual C++ Type	Name	Description	Visual Basic Type
int	aValid	Valid/invalid	Long
int	aValue	Initial balance value	Long

Retrieval/Change

Name	Description	Basic Settings
aValid	Valid/invalid	F
aValue	Initial balance value	RC

MXBalanceData

Visual C/C++ Type	Name	Description	Visual Basic Type
MXBalance []	aBalance	Specified no. of initial balance data	(1 TO 60) As MXBalance

MXBalanceResult

MXBalanceResult structure

Visual C/C++ Type	Name	Description	Visual Basic Type
int []	aResult	Specified no. of (1 To 60) As initial balance results	Long

MXOutput

MXOutput structure

Visual C/C++ Type	Name	Description	VB Type
int	aType	Output type	Long
int	aldleChoice	Value selected when idle	Long
int	aErrorChoice	Value selected during error	Long
int	aPresetValue	Value if the selection value is the "specified value."	Long
int	aPulseTime	Pulse interval integer multiple	Long
int	aReserve	Not used	Long

Retrieval/Change

Name	Description	Basic Settings AO	Basic SetPWM
aType	Output types	RC	RC
aldleChoice	Value selected when idle	RC	RC
aErrorChoice	Value selected during error	RC	RC
aPresetValue	Value if the selection value is the "specified value."	RC	RC
aPulseTime	Pulse interval integer multiple		RC

MXOutputData

MXOutputData structure

Visual C/ C++ Type	Name	Description	Visual Basic Type
MXOutput []	aOutput	Specified no. of output	(1 TO 60) As channel data MXOutput

MXConfigData

MXConfigData structure

Visual C/C++ Type	Name	Description	Visual Basic Type
MXSystemInfo	aSystemInfo	System configuration data	(None)
MXStatus	aStatusInfo	Status	(None)
MXNetInfo	aNetInfo	Network information data	(None)
MXChConfigData	aChConfigData	Channel setup data	(None)
MXBalanceData	aBalanceData	Initial balance data	(None)
MXOutputData	aOutputData	Output channel data	(None)

Setup data structure.

Cannot be used in Visual Basic. Field must be allocated individually.

Visual C++: The wrapper class is CDAQMXConfig.

MXDO

MXDO structure

Visual C/C++ Type	Name	Description	Visual Basic Type
int	aValid	Valid/invalid (presence/absence)	Long
int	aONOFF	ON/OFF (valid/invalid)	Long

DO channel structure.

MXDOData

MXDOData structure

Visual C/C++ Type	Name	Description	Visual Basic Type
MXDO []	aDO	DO data	(1 To 60) As MXDO

DO data structure.

MXSegment

MXSegment structure

Visual C/C++ Type	Name	Description	Visual Basic Type
int []	aPattern	For each 7-segment LED display pattern	(0 To 1) As Long

7-segment LED display structure.

MXAOPWM

MXAOPWM structure

Visual C/C++ Type	Name	Description	Visual Basic Type
int	aValid	Valid/invalid (presence/absence)	Long
int	aValue	Output data value	Long

MXAOPWMData

MXAOPWMData structure

Visual C/C++ Type	Name	Description	Visual Basic Type
MXAOPWM []	aAOPWM	Number's worth of AO/PWM data	(1 To 60) As MXAOPWM

MXTransmit

MXTransmit structure

Visual C/C++ Type	Name	Description	Visual Basic Type
int []	aTrans	Number's worth of transmission status	(1 To 060) As Long

7.1 DARWIN Class

The API consists of the MX100/DARWIN Common Class and the class dedicated to the DARWIN as shown in the figure below. For the details on the MX100/DARWIN Common Class, see section 2.4.

- CDAQChInfo
 - CDAQDARWINChInfo
 - CDAQDARWINSysInfo
 - CDAQDataInfo
 - CDAQDARWINDataInfo
 - CDAQDateTime
 - CDAQDARWINDateTime
 - CDAQHandler
 - CDAQDARWIN
- : Class common to the MX100 and the DARWIN.
 • : Class dedicated to the MX100.

CDAQChInfo Class

Base class for storing the channel information data.

CDAQDARWINChInfo Class

Class derived from the CDAQChInfo class. Stores the channel information data in the DarwinChInfo structure.

CDAQDARWINSysInfo Class

Class for storing the system configuration data in the DarwinSystemInfo structure.

CDAQDataInfo Class

Base class for storing the measured data.

CDAQDARWINDataInfo Class

Class derived from the CDAQDataInfo class. Stores the measured data.

CDAQDateTime Class

Base class for storing time information.

CDAQDARWINDateTime Class

Class derived from the CDAQDateTime class. Stores the time information.

CDAQHandler Class

Handler base class for performing communications with the instrument (MX100/DARWIN).

CDAQDARWIN Class

Class derived from the CDAQHandler class. Provides communication functions common to the DARWIN series.

Note

Data type and retrieval method

The retrieval of the DARWIN data is handled by a dedicated class. The setup data is not handled by a dedicated class, since the data is retrieved at the line level.

7.2 Correspondence between the Functions and Class/Member Functions - DARWIN -

This section indicates the correspondence between the functions that the API supports and the class.

Note

This API provides a portion of the functions common to the DARWIN series instruments. Model-specific functions, setup functions of the setup mode, and A/D calibration functions are not implemented. The functions can be added by using the commands of the DARWIN communication function.

The word “command” in the table signifies the command of the DARWIN communication function. For the details on the command, see the Communication Interface User’s Manual.

Communication Functions

Function	Class and Member Function
Connect to DARWIN.	CDAQDARWIN::open
Disconnect from DARWIN.	CDAQDARWIN::close
Send data line by line. Used when controlling the data transmission in a special way.	CDAQDARWIN::sendLine
Receive data line by line. Used when controlling the data reception in a special way.	CDAQDARWIN::receiveLine
Receive data in units of bytes. Used when controlling the data reception in a special way.	CDAQDARWIN::receiveByte
Send the command and receive the response. Used when implementing function commands.	CDAQDARWIN::runCommand
Get the status byte. Send the status byte output command and receive the response.	CDAQDARWIN::getStatusByte
Send a trigger command (ESC T), and receive the response. Used when implementing a new talker function.	CDAQDARWIN::sendTrigger
Set the communication timeout.	CDAQDARWIN::setTimeOut

Note

Setting of the communication timeout is not recommended because unexpected disconnection may occur due to the conflict with the timeout time when data is retrieved.

Control Functions

Function	Command	Class and Member Function
Switch the setting mode.	DS	CDAQDARWIN::transMode
System reconfiguration	RS	CDAQDARWIN::initSystem
RAM clear (Initialize the operation mode setup parameter.)	RC	
Alarm reset	AR	
Date/Time setting	SD	CDAQDARWIN::setDateTime
Calculation start/stop	EX	CDAQDARWIN::compute
Report start/stop	DR	CDAQDARWIN::reporting
Finalize setup mode	XE	CDAQDARWIN::establish

Setup Functions

Function	Command	Class and Member Function
Range Settings		
SKIP (not used)	SR	CDAQDARWIN::setSKIP
DC voltage input	SR	CDAQDARWIN::setVOLT
Thermocouple input	SR	CDAQDARWIN::setTC
RTD input	SR	CDAQDARWIN::setRTD
Contact input (DI)	SR	CDAQDARWIN::setDI
Difference computation between channels	SR	CDAQDARWIN::setDELTA
Remote RJC	SR	CDAQDARWIN::setRRJC
DC current	SR	CDAQDARWIN::setMA
Strain	SR	CDAQDARWIN::setSTRAIN
Pulse	SR	CDAQDARWIN::setPULSE
Power monitor	SR	CDAQDARWIN::setPOWER
Set the scale unit	SN	CDAQDARWIN::setScallingUnit
Set the alarm	SA	CDAQDARWIN::setAlarm

Data Retrieval Functions

Function	Command	Class and Member Function
Get system configuration data.	TS, CF	DAQDARWIN::getSystemConfig
Declare the retrieval of the channel information data.	TS, LF	CDAQDARWIN::talkChInfo
Get channel information data.		CDAQDARWIN::getChInfo
Declare the retrieval of the measured data (ASCII code).	TS, FM	CDAQDARWIN::talkDataByASCII
Get the measured data (ASCII code).		CDAQDARWIN::getChDataByASCII
Declare the retrieval of the measured data (binary code).	TS, FM	CDAQDARWIN::talkDataByBinary
Get the measured data (binary code).		CDAQDARWIN::getChDataByBinary
Declare the retrieval of the setup data (operation mode).	TS, LF	CDAQDARWIN::talkOperationData

7.2 Correspondence between the Functions and Class/Member Functions - DARWIN -

Function	Command	Class and Member Function
Get the setup data (operation mode).		CDAQDARWIN::getSetDataByLine
Declare the retrieval of the setup data (setup mode).	TS, LF	CDAQDARWIN::talkSetupData
Get the setup data (setup mode).		CDAQDARWIN::getSetDataByLine
Declare the retrieval of the setup data (A/D calibration mode).	TS, LF	CDAQDARWIN::talkCalibrationData
Get the setup data (A/D calibration)		getSetDataByLineDARWIN
Retrieves the report status	TS, RF	CDAQDARWIN::getReportStatus

Utilities

Function	Class and Member Function
Convert the measured value into double-precision floating point number.	CDAQDARWINDataInfo::toDoubleValue
Convert the measured value into string.	CDAQDARWINDataInfo::toStringValue
Alarm	
Get the alarm type string.	CDAQDARWINDataInfo::getAlarmName
Get the maximum length of the alarm string.	CDAQDARWINDataInfo::getMaxLenAlarmName
Get the version number of this API.	CDAQDARWIN::getVersionAPI
Get the revision number of this API.	CDAQDARWIN::getRevisionAPI
Get the error message string.	CDAQDARWIN::getErrorMessage
Get the maximum length of the error message string.	CDAQDARWIN::getMaxLenErrorMessage

Implementing Function Commands

Function commands can be implemented by using the DARWIN communication function commands. Below are the DARWIN communication function commands that can be used.

- All communication commands for the DA100 Data Acquisition Unit
- All communication commands for the DC100 Data Collector
- All communication commands for the DR130, DR231, DR232, DR241, and DR242 Hybrid Recorders.

7.3 Programming - DARWIN/Visual C ++ -

Adding the Path to the Include File

Add the path of the include file (DAQDARWIN.h) to the project. The method of adding the include file varies depending on the environment used.

Declaration in the Source File

Write the declaration in the source file.

```
#include "DAQDARWIN.h"
```

Note

The include file of the common section (DAQHandler.h) is referenced from the include file described above. Thus, declaration for it is not necessary.

Library Designation

Add the library (DAQDARWIN.lib, DAQHandler.lib) to the project. The method of adding the include file varies depending on the environment used.

This enables the use of all classes. It also enables the use of all Visual C functions.

Retrieval of the Measured Data

Program Example 1

This program retrieves measured data.

```
////////////////////////////////////  
// DARWIN sample for measurement  
#include <stdio.h>  
#include "DAQDARWIN.h"  
////////////////////////////////////  
int main(int argc, char* argv[])  
{  
    int rc; //return code  
    CDAQDARWIN daqdarwin; //class  
    int flag;  
    CDAQDARWINDateTime datetime;  
    CDAQDARWINChInfo chinfo;  
    CDAQDARWINDataInfo datainfo(NULL, &chinfo);  
    //connect  
    rc = daqdarwin.open("192.168.1.11");  
    //get  
    rc = daqdarwin.talkDataByBinary(0, 1, 0, 2, datetime);  
    do { //measured data  
        rc = daqdarwin.getChDataByBinary(datainfo, &flag);  
    } while (! (flag & DAQDARWIN_FLAG_ENDDATA));  
    //disconenct  
    rc = daqdarwin.close();  
    return rc;  
}  
////////////////////////////////////
```

Description

Overview

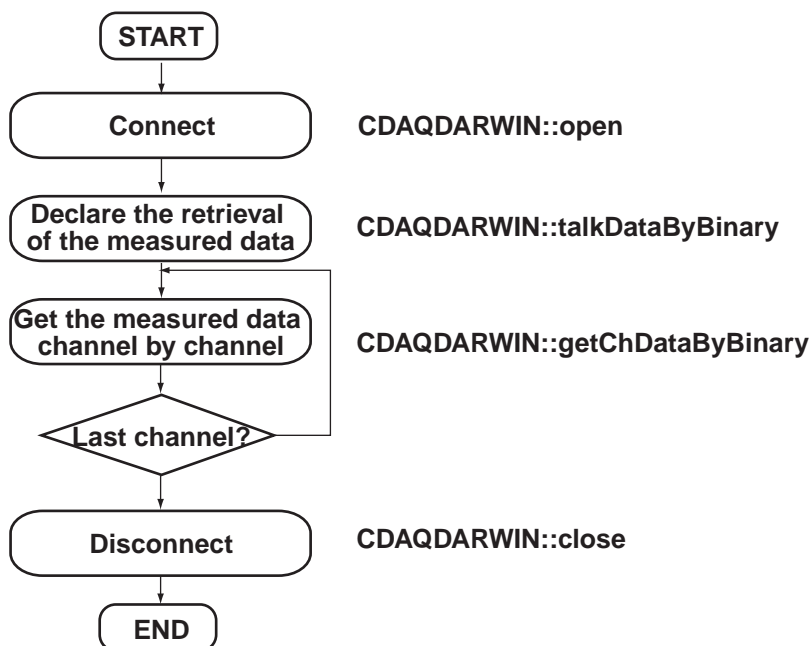
When retrieving data, the talker is executed first, and then data retrieval is executed in units of channels or lines. The end is determined by the flag.

Include File Statement

```
#include "DAQDARWIN.h"
```

Flow of the Process

The flow chart shown below omits the declaration section.



Communication Process

First, make a connection. After making the connection, the member functions become available. As a termination procedure, disconnect the communication.

Communication Connection

```
open("192.168.1.11")
```

The IP address of the DARWIN is specified. The communication report specifies the communication constant "DAQDARWIN report number."

Note

Communication can also be made when constructing the class. Connection is dropped when the class is destructed.

Talker

```
talkDataByBinary(0, 1, 0, 2, datetime)
```

Sends the retrieval request of the measured data of channels 1 and 2 of subunit number 0 and retrieves the time information (declares the retrieval of the measured data).

Retrieval of the Measured Data

```
getChDataByBinary(datainfo, &flag)
```

Gets the measured data channel by channel. It is repeated up to the specified channel. The end is determined by the flag status of "end data."

Communication Disconnection

```
close()
```

Drops the connection.

Retrieval of Setup Data and Configuration

Program Example 2

This program executes the following two items. This program contains both items, but each item can be written and executed separately.

- Retrieval of setup data
- Setting a DC voltage range to the channel

```
////////////////////////////////////  
// DARWIN sample for configuration  
#include <stdio.h>  
#include "DAQDARWIN.h"////////////////////////////////////  
////////////////////////////////////  
int main(int argc, char* argv[])  
{  
    int rc; //return code  
    CDAQDARWIN daqdarwin; //class  
    int flag;  
    char line[BUFSIZ];  
    int len;  
    //connect  
    rc = daqdarwin.open("192.168.1.11");  
    //get  
    rc = daqdarwin.talkOperationData(0, 1, 0, 2);  
    do {  
        rc = daqdarwin.getSetDataByLine(line, BUFSIZ, &len,  
&flag);  
    } while (! (flag & DAQDARWIN_FLAG_ENDDATA));  
    //range  
    rc = daqdarwin.setVOLT(DAQDARWIN_RANGE_VOLT_20MV, 0, 1, 2,  
0, 0, 0, 0, 0);  
    //disconnect  
    rc = daqdarwin.close();  
    return rc;  
}  
////////////////////////////////////
```

Description

Talker

```
talkOperationData(0, 1, 0, 2)
```

Specifies the type of setup data to be retrieved (setup data of the operation mode) and the channel range (channels 1 and 2 of subunit number 0).

Retrieval of Setup Data

```
getSetDataByLine(line, BUFSIZ, &len, &flag)
```

Gets the output by the talker function line by line.

The end is determined by the flag status of "end data."

Setting a DC voltage range to the channel

```
setVOLT(DAQDARWIN_RANGE_VOLT_20MV, 0, 1, 2, 0, 0, 0, 0, 0)
```

Sets the measurement range of channels 1 and 2 of subunit number 0 to 20 mV.

The scaling function is not used.

The constant 20mV is used to specify the range type.

Implementing Function Commands

Program Example 3

This program switches DARWIN to operation mode. The program executes the DS command of DARWIN communication function.

```

////////////////////////////////////
// DARWIN sample for command
#include <stdio.h>
#include "DAQDARWIN.h"
////////////////////////////////////
int main(int argc, char* argv[])
{
    int rc; //return code
    CDAQDARWIN daqdarwin; //class
    char line[BUFSIZ];
    //connect
    rc = daqdarwin.open("192.168.1.11");
    //run
    sprintf(line, "DS%d" DAQDARWIN_MODE_OPE);
    rc = daqdarwin.runCommand(line);
    //disconnect
    rc = daqdarwin.close();
    return rc;
}
////////////////////////////////////

```

Description

Creating the Message

```
sprintf(line, "DS%d" DAQDARWIN_MODE_OPE);
```

Stores the DS0 (switch to operation mode) command message of the DARWIN communication function in the line array.

The constant "operation mode" is used to specify operation mode.

Sending Messages

```
runCommand(line)
```

Sends the command message and receives the response. This member function adds a terminator to the message and sends it.

Implementing the Talker Function

Program Example 4

This program retrieves the system configuration data. The program executes the TS and CF commands of the DARWIN communication function.

```

////////////////////////////////////
// DARWIN sample for talker
#include <stdio.h>
#include "DAQDARWIN.h"
////////////////////////////////////
int main(int argc, char* argv[])
{
    int rc; //return code
    CDAQDARWIN daqdarwin; //class
    char line[BUFSIZ];
    int len;
    //connect
    rc = daqdarwin.open("912.168.1.11");
    //talker
    sprintf(line, "TS%d" DAQDARWIN_TALK_SYSINFODATA);
    rc = daqdarwin.runCommand(line);
    rc = daqdarwin.sendTrigger();
    rc = daqdarwin.sendLine("CF0");
    do {
        rc = daqdarwin.receiveLine(line, BUFSIZ, &len);
    } while ((rc == 0) && (line[0] != '\0'));
    //disconnect
    rc = daqdarwin.close();
    return rc;
}
////////////////////////////////////

```


Description

Talker

```
sprintf(line, "TS%d" DAQDARWIN_TALK_SYSINFODATA);
```

Stores the TS5 (declares the retrieval of the system configuration data) command message of the DARWIN communication function to line.

The constant "system configuration data output" is used to specify the output of the system configuration data.

```
runCommand(line)
```

Sends the message and receives the response. The number of bytes of the message is not specified (omitted). This member function adds a terminator to the message and sends it.

```
sendTrigger()
```

Sends a trigger (device trigger).

Designation of System Configuration Output Format

```
sendLine("CF0"
```

Sends the CF0 communication function command (specifies the module information that has been configured for the system). This member function adds a terminator to the message and sends it.

Data Retrieval

```
receiveLine(line, BUFSIZ, &len)
```

Gets the system configuration data line by line. The program ends when an end mark (E) is returned.

Note

The receiveLine member function simply receives the data. The user must write statements for determining the end of the data.

Error Processing

- Most member functions return the result of the function process using an error number.
- The member function `getErrorMessage` can be used to get the error message string corresponding to the error number. The member function `getMaxLenErrorMessage` can be used to get the maximum length of the error message string.

7.4 Details of the DARWIN Class

The classes are listed in alphabetical order by the class name.

CDAQDARWIN Class

- CDAQHandler
 - CDAQDARWIN

This class is derived from the CDAQHandler class. It provides functions common to the DARWIN series such as communications, data retrieval, and measurement range setting.

Public Members

Construct/Destruct

CDAQDARWIN	Constructs an object.
~CDAQDARWIN	Destructs an object.

Communication Functions

runCommand	Sends the message and receives the response.
getStatusByte	Gets the status byte.
sendTrigger	Sends a trigger.
receiveByte	Receives the binary data by lines.

Control Functions

setDateTime	Sets the date/time.
transMode	Switches the setting mode.
initSystem	Initializes the system.
establish	Establishes setup mode.
compute	Starts/stops computation.
reporting	Starts/stops reporting.

Data Retrieval Functions

getSystemConfig	Gets the system configuration data.
talkChInfo	Declares the retrieval of the channel information data.
getChInfo	Gets the channel information data.
talkDataByASCII	Declares the retrieval of the measured data in ASCII format.
getChDataByASCII	Gets the measured data in ASCII format.
talkDataByBinary	Declares the retrieval of the measured data in binary format.

getChDataByBinary	Gets the measured data in binary format.
talkOperationData	Declares the retrieval of the setup data of operation mode.
talkSetupData	Declares the retrieval of the setup data of setup mode.
talkCalibrationData	Declares the retrieval of the setup data of A/D calibration mode.
getSetDataByLine	Gets the setup data.
getReportStatus	Gets the report status.

Setup Functions

setSKIP	Sets SKIP (not used).
setVOLT	Sets DC voltage input.
setTC	Sets thermocouple input.
setRTD	Sets RTD input.
setDI	Sets contact input (DI).
setDELTA	Sets difference computation between channels.
setRRJC	Sets remote RJC.
setScalingUnit	Sets the scaling unit.
setAlarm	Sets the alarm value.
setMA	Sets the DC current range.
setSTRAIN	Sets the strain input.
setPULSE	Sets the pulse input.
setPOWER	Sets the power monitor.

- **Overridden Members**

Communication Functions

open	Establishes connection.
------	-------------------------

Data Retrieval Functions

getData	Gets the measured data.
getChannel	Gets the channel information data.

Utilities

isObject	Checks an object
----------	------------------

- **Inherited Members**

See CDAQHandler.

close getErrorMessage getMaxLenErrorMessage
getRevisionAPI getVersionAPI receiveLine sendLine setTimeout

Protected Members

Communication Functions

startTalker Starts the talker function.

Utilities

checkAck Checks the response.
 getVersionDLL Gets the version number of the DLL.
 getRevisionDLL Gets the revision number of the DLL.

Inherited Members

See CDAQHandler.
 m_comm m_nRemainSize receive receiveRemain send

Private Members

None

Member Functions (Alphabetical Order)

CDAQDARWIN::CDAQDARWIN

Syntax

```
CDAQDARWIN(void);
CDAQDARWIN(const char * strAddress, unsigned int uiPort =
DAQDARWIN_COMMPORT, int * errCode = NULL);
virtual ~CDAQDARWIN(void);
```

Parameters

strAddress Specify the IP address as a string.
 uiPort Specify the port number.
 errCode Specify the destination where the error number is to be returned.

Description

Constructs or destructs an object.
 When constructing, the data member is initialized. When the parameters are specified, a connection is established (open) during construction. If the return destination is specified, the error number during connection is returned.
 When destructing, the data member field is released. The connection is dropped (close) when the communication descriptor exists. The error number is not returned.

Reference

CDAQHandler::CDAQHandler

CDAQDARWIN::checkAck

Syntax

```
int checkAck(const char * strAck, int lenAck);
```

Parameters

strAck	Specify the response using a string.
lenAck	Specify the byte size of the response.

Description

Checks the string specified by the parameter as a response and returns the result.

Return value

Returns an error number.

Error:

Success	Responds with "processing executed successfully."
Commands are not processed successfully	
	Responds with "processing not executed successfully."
Not acknowledge	No response.

CDAQDARWIN::compute

Syntax

```
int compute(int iCompute);
```

Parameters

iCompute	Specify the computation.
----------	--------------------------

Description

Executes the specified computation.

This function executes the EX command of the communication interface.

It creates and sends the command and receives the response.

It is only valid with the optional computation function, or when the pulse module is installed.

Return value

Returns an error number.

Reference

runCommand

CDAQDARWIN::establish

Syntax

```
int establish(int iSetup = DAQDARWIN_SETUP_ABORT);
```

Parameters

iSetup Specifies establishment of setup.

Description

Executes the specified setup establishment.

Executes the “Communication interface”EX command.

Creates and sends the command and receives the response.

It is valid in setup mode.

Return value

Returns an error number.

Reference

runCommand

CDAQDARWIN::getChannel

Syntax

```
virtual int getChannel(int chType, int chNo, CDAQChInfo &
cChInfo);
```

Parameters

chType Specify the channel type.

chNo Specify the channel number.

cChInfo Specify the destination where the channel information data is to
be returned.

Description

This function gets the channel information data by channels.

Gets the channel information data of the specified channel.

Return value

Returns an error number.

Reference

getChInfo talkChInfo

CDAQDARWIN::getChDataByASCII

Syntax

```
int getChDataByASCII(CDAQDARWINDataInfo & cDARWINDataInfo, int * pFlag);
```

Parameters

cDARWINDataInfo	Specify the destination where the measured data is to be returned.
pFlag	Specify the destination where the flag is to be returned.

Description

Gets the output of each channel using the talker function declared by `ttalkDataByASCII`. Analyzes received information channel by channel and stores it in the return destination. When the last set of data is retrieved, the flag status is set. It is also set when the function ends in error. Do not perform communications using other functions until the data retrieval is completed. Other functions may not operate properly.

Return value

Returns an error number.

Reference

`checkAck receiveLine CDAQDARWINDataInfo::setLine`

CDAQDARWIN::getChDataByBinary

Syntax

```
int getChDataByBinary(CDAQDARWINDataInfo & cDARWINDataInfo, int * pFlag);
```

Parameters

cDARWINDataInfo	Specify the destination where the measured data is to be returned.
pFlag	Specify the destination where the flag is to be returned.

Description

Gets the output of each channel using the talker function declared by `talkDataByBinary`. 6 bytes are received for measurement channels, and 8 bytes for computation channels. Analyzes received information channel by channel and stores it in the return destination. Updates the remaining size of the data member. When the last set of data is retrieved, the flag status is set. It is also set when the function ends in error. Do not perform communications using other functions until the data retrieval is completed. Other functions may not operate properly.

Return value

Returns an error number.

Error:

Not data The output number of bytes did not satisfy the size requirement.

Reference

`receive CDAQDARWINDataInfo::setByte`

CDAQDARWIN::getChInfo

Syntax

```
int getChInfo(CDAQDARWINChInfo & cDARWINChInfo, int * pFlag);
```

Parameters

cDARWINChInfo	Specify the destination where the channel information data is to be returned.
pFlag	Specify the destination where the flag is to be returned.

Description

Gets the channel information data output by channel using the talker function declared by talkChInfo. Analyzes received information channel by channel and stores it in the return destination.

When the last set of data is retrieved, the flag status is set. It is also set when the function ends in error.

Do not perform communications using other functions until the data retrieval is completed. Other functions may not operate properly.

Return value

Returns an error number.

Reference

```
checkAck receiveLine
CDAQDARWINChInfo::setLine
```

CDAQDARWIN::getData

Syntax

```
virtual int getData(int chType, int chNo, CDAQDateTime &
cDateTime, CDAQDataInfo & cDataInfo);
```

Parameters

chType	Specify the channel type.
chNo	Specify the channel number.
cDateTime	Specify the destination where the time information data is to be returned.
cDataInfo	Specify the destination where the measured data is to be returned.

Description

This function gets the instantaneous values in units of channels.

Gets the measured data of the specified channel in binary code.

Return value

Returns an error number.

Reference

```
getChDataByBinary talkDataByBinary
```


CDAQDARWIN::getReportStatus

Syntax

```
int getReportStatus(int * pReportStatus);
```

Parameters

pReportStatus Specify the destination where the report status is to be returned.

Description

Gets the report status.

The function executes the data retrieval declaration as a talker function and the data output in succession.

Stores the report status in the return destination.

This function executes the TS and RF commands of the DARWIN communication function.

It creates and sends the command and receives the data.

Return value

Returns an error number.

Reference

```
receive send startTalker
```

CDAQDARWIN::getRevisionDLL

Syntax

```
static const int getRevisionDLL(void);
```

Description

Gets the revision number of this DLL.

Return value

Returns the revision number of this DLL.

CDAQDARWIN::getSetDataByLine

Syntax

```
int getSetDataByLine(char * strLine, int maxLine, int *
lenLine, int * pFlag);
```

Parameters

strLine	Specify the field where the string received by lines is to be stored.
maxLine	Specify the byte size of the field where the string received by lines is to be stored.
lenLine	Specify the destination where the byte size of the actual string received is returned.
pFlag	Specify the destination where the flag is to be returned.

Description

Gets the output by line using the talker functions declared by talkOperationData, talkSetupData, and talkCalibrationData.

If the return destination is specified, the flag status is set when the last set of data is retrieved. It is also set when the function ends in error.

Do not perform communications using other functions until the data retrieval is completed. Other functions may not operate properly.

Return value

Returns an error number.

Reference

receiveLine

CDAQDARWIN::getStatusByte

Syntax

```
virtual int getStatusByte(int * pStatusByte);
```

Parameters

pStatusByte	Specify the destination where the status byte is to be returned.
-------------	--

Description

Gets the status byte.

Stores the status byte as an integer to the return destination if the return destination is specified.

Sends the status byte output command (ESC S) and receives the output.

Return value

Returns an error number.

Reference

checkAck receiveLine send

CDAQDARWIN::getSystemConfig

Syntax

```
int getSystemConfig(CDAQDARWINSysInfo & cDARWINSysInfo);
```

Parameters

cDARWINSysInfo Specify the destination where the system configuration data is to be returned.

Description

Gets the system configuration data. The function executes the data retrieval declaration as a talker function and the data output.

Stores the scan interval and the system configuration data to the return destination.

This function executes the TS and CF commands of the DARWIN communication function.

It initializes the return destination, creates and sends the command, and receives the data.

Return value

Returns an error number.

Reference

```
checkAck receiveLine send startTalker  
CDAQDARWINSysInfo::initialize CDAQDARWINSysInfo::setLine
```

CDAQDARWIN::getVersionDLL

Syntax

```
static const int getVersionDLL(void);
```

Description

Gets the version number of this DLL.

Return value

Returns the version number of this DLL.

CDAQDARWIN::initSystem

Syntax

```
int initSystem(int iCtrl);
```

Parameters

iCtrl Specify the system control type.

Description

Executes the operation of the specified system control type.

This function executes the RS, RC, or AR command of the DARWIN communication function.

It creates and sends the command and receives the response.

Return value

Returns an error number.

Error:

Not support Specified value is out of range.

Reference

```
runCommand
```

CDAQDARWIN::isObject

Syntax

```
virtual int isObject(const char * classname = "CDAQDARWIN");
```

Parameters

classname Specify the class name using a string.

Description

Checks whether the specified class name was inherited.

If parameters are omitted, checks whether it is this class.

Classes that inherit this class must be overridden in order to check their own classes.

Returns true (valid) if the class was inherited. Otherwise, returns false (invalid).

If different from this class, checks the parent class.

Return value

Returns a boolean value.

Reference

CDAQHandler::isObject

CDAQDARWIN::open

Syntax

```
virtual int open(const char * strAddress, unsigned int uiPort);
```

Parameters

strAddress Specify the IP address as a string.

uiPort Specify the port number.

Description

Connects to the device with the IP address and port number specified by the parameters.

The port number can be omitted. If omitted, it is set to the DARWIN communication port number.

Return value

Returns an error number.

Reference

CDAQHandler::open

CDAQDARWIN::receiveByte

Syntax

```
virtual int receiveByte(unsigned char * byteData, int maxData  
= 1, int * lenData = NULL);
```

Parameters

byteData	Specify the field where the received data is to be stored using a byte array.
maxData	Specify the byte size of the received data.
lenData	Specify the destination where the byte size of the actual data received is returned.

Description

Stores the received data to the field specified by the parameter up to the specified byte size.

Returns the byte size of the actual data received if the return destination is specified.

Return value

Returns an error number.

Reference

receive

CDAQDARWIN::reporting

Syntax

```
int reporting(int iReportRun);
```

Parameters

iReportRun	Specify the report execution type.
------------	------------------------------------

Description

Executes the specified report execution type.

This function executes the DR command of the DARWIN communication function..

It creates and sends the command and receives the response.

It is only valid with the optional computation function, or when the pulse module is installed.

Return value

Returns an error number.

Reference

runCommand

CDAQDARWIN::runCommand

Syntax

```
virtual int runCommand(const char * strCmd)
```

Parameters

strCmd Specify the command message to be sent.

Description

Sends the specified command message and receives the response.

This function adds a terminator to the command message at the time of transmission. Therefore, do not include the terminator in the command message.

This function does not support simultaneous transmission of multiple commands or command messages that include the terminator.

Like the data output request command of the talker function, does not support commands that do not send responses.

The terminator of the string is the NULL character.

Return value

Returns an error number.

Reference

checkAck receiveLine sendLine

CDAQDARWIN::sendTrigger

Syntax

```
virtual int sendTrigger(void);
```

Description

Sends a trigger command (ESC T), and receives the response.

Return value

Returns an error number.

Reference

runCommand

CDAQDARWIN::setAlarm

Syntax

```
int setAlarm(int levelNo, int chType, int startChNo, int endChNo = 0, int iAlarmType = DAQDARWIN_ALARM_NONE, int value = 0, int relayType = 0, int relayNo = 0);
```

Parameters

levelNo	Specify the alarm level.
chType	Specify the channel type.
startChNo	Specify the start channel number.
endChNo	Specify the end channel number.
iAlarmType	Specify the alarm type using the alarm type value.
value	Specify the alarm value.
relayType	Specify the relay type.
relayNo	Specify the relay number.

Description

Sets the specified alarm (alarm level and alarm type) and alarm value to the channels in the specified channel range (specified by channel type, start channel number, and end channel number).

When the relay number is less than or equal to 0, the relay is turned OFF.

This function executes the SA command of the DARWIN communication function.

It creates and sends the command and receives the response.

Return value

Returns an error number.

Reference

runCommand
CDAQDARWINChInfo::toChRange
CDAQDARWINDataInfo::getAlarmName
CDAQDARWINSysInfo::toRelayName

CDAQDARWIN::setDateTime

Syntax

```
int setDateTime(CDAQDARWINDateTime * pcDARWINDateTime = NULL);
```

Parameters

pcDARWINDateTime Specify the time information data.

Description

Sets time information data on the device.

If NULL is specified, the current date/time of the PC is used.

This function executes the SD command of the DARWIN communication function.

It creates and sends the command and receives the response.

Return value

Returns an error number.

Reference

```
runCommand
CDAQDARWINDateTime::setNow CDAQDARWINDateTime::toString
```

CDAQDARWIN::setDELTA

Syntax

```
int setDELTA(int refChNo, int chType, int startChNo, int
endChNo = 0, int spanMin = 0, int spanMax = 0);
```

Parameters

refChNo	Specify the channel number of the reference channel.
chType	Specify the channel type of the measurement channel.
startChNo	Specify the start channel number.
endChNo	Specify the end channel number.
spanMin	Specify the left value of the span.
spanMax	Specify the right value of the span.

Description

Sets difference computation with respect to the specified reference channel to the measurement channels in the specified channel range (specified by channel type, start channel number, and end channel number).

This function executes the SR command of the DARWIN communication function.

It creates and sends the command and receives the response.

If the left and right values are the same, the span designation is considered omitted.

Return value

Returns an error number.

Reference

```
runCommand
CDAQDARWINChInfo::toChRange
```

CDAQDARWIN::setDI

Syntax

```
int setDI(int iRangeDI, int chType, int startChNo, int endChNo  
= 0, int spanMin = 0, int spanMax = 0, int scaleMin = 0, int  
scaleMax = 0, int scalePoint = 0);
```

Parameters

iRangeDI	Specify the range type of contact input (DI).
chType	Specify the channel type of the measurement channel.
startChNo	Specify the start channel number.
endChNo	Specify the end channel number.
spanMin	Specify the left value of the span.
spanMax	Specify the right value of the span.
scaleMin	Specify the left value of the scale.
scaleMax	Specify the right value of the scale.
scalePoint	Specify the decimal point position for scaling.

Description

Sets the specified measurement range to the measurement channels in the specified channel range (specified by channel type, start channel number, and end channel number).

This function executes the SR command of the DARWIN communication function. It creates and sends the command and receives the response.

If the left and right values are the same, the span or scale designation is considered omitted.

Return value

Returns an error number.

Reference

runCommand
CDAQDARWINChInfo::toChRange

CDAQDARWIN::setMA

Syntax

```
int setMA(int iRangeMA, int chType, int startChNo, int endChNo
= 0, int spanMin = 0, int spanMax = 0, int scaleMin = 0, int
scaleMax = 0, int scalePoint = 0);
```

Parameters

iRangeMA	Specify the DC current range.
chType	Specify the channel type of the measurement channel.
startChNo	Specify the start channel number.
endChNo	Specify the end channel number.
spanMin	Specify the left value of the span.
spanMax	Specify the right value of the span.
scaleMin	Specify the left value of the scale.
scaleMax	Specify the right value of the scale.
scalePoint	Specify the decimal point position for scaling.

Description

Sets the specified DC current range to the measurement channels in the specified channel range (specified by channel type, start channel number, and end channel number).

This function executes the SR command of the DARWIN communication function. It creates and sends the command and receives the response.

If the left and right values are the same, the span or scale designation is considered omitted.

Return value

Returns an error number.

Reference

runCommand CDAQDARWINChInfo::toChRange

CDAQDARWIN::setPOWER

Syntax

```
int setPOWER(int iRangePOWER, int chType, int chNo, int iItem  
= DAQDARWIN_POWERITEM_P1, int iWire = DAQDARWIN_WIRE_1PH2W,  
int spanMin = 0, int spanMax = 0, int scaleMin = 0, int  
scaleMax = 0, int scalePoint = 0);
```

Parameters

iRangeVOLT	Specify the power monitor range.
chType	Specify the channel type of the measurement channel.
chNo	Specify the channel number.
iItem	Specify the power measurement parameter.
iWire	Specify the power connection method.
spanMin	Specify the left value of the span.
spanMax	Specify the right value of the span.
scaleMin	Specify the left value of the scale.
scaleMax	Specify the right value of the scale.
scalePoint	Specify the decimal point position for scaling.

Description

Sets the specified power monitor range on the measurement channels of the specified channel (channel type, channel number).

This function executes the SR command of the DARWIN communication function.

It creates and sends the command and receives the response.

If the left and right values are the same, the span or scale designation is considered omitted.

Return value

Returns an error number.

Reference

runCommand CDAQDARWINChInfo::toChRange

CDAQDARWIN::setPULSE

Syntax

```
int setPULSE(int iRangePULSE, int chType, int startChNo, int
endChNo = 0, int spanMin = 0, int spanMax = 0, int scaleMin =
0, int scaleMax = 0, int scalePoint = 0, int bFilter =
DAQDARWIN_VALID_OFF);
```

Parameters

iRangePULSE	Specify the pulse range.
chType	Specify the channel type of the measurement channel.
startChNo	Specify the start channel number.
endChNo	Specify the end channel number.
spanMin	Specify the left value of the span.
spanMax	Specify the right value of the span.
scaleMin	Specify the left value of the scale.
scaleMax	Specify the right value of the scale.
scalePoint	Specify the decimal point position for scaling.
bFilter	Specify ON/OFF for the filter using a boolean value.

Description

Sets the specified pulse range to the measurement channels in the specified channel range (specified by channel type, start channel number, and end channel number).

This function executes the SR command of the DARWIN communication function. It creates and sends the command and receives the response.

If the left and right values are the same, the span or scale designation is considered omitted.

Return value

Returns an error number.

Reference

runCommand CDAQDARWINChInfo::toChRange

CDAQDARWIN::setRRJC

Syntax

```
int setRRJC(int refChNo, int chType, int startChNo, int  
endChNo = 0, int spanMin = 0, int spanMax = 0);
```

Parameters

refChNo	Specify the channel number of the reference channel.
chType	Specify the channel type of the measurement channel.
startChNo	Specify the start channel number.
endChNo	Specify the end channel number.
spanMin	Specify the left value of the span.
spanMax	Specify the right value of the span.

Description

Sets remote RJC with respect to the specified reference channel to the measurement channels in the specified channel range (specified by channel type, start channel number, and end channel number).

This function executes the SR command of the DARWIN communication function. It creates and sends the command and receives the response.

If the left and right values are the same, the span designation is considered omitted.

Return value

Returns an error number.

Reference

runCommand
CDAQDARWINChInfo::toChRange

CDAQDARWIN::setRTD

Syntax

```
int setRTD(int iRangeRTD, int chType, int startChNo, int
endChNo = 0, int spanMin = 0, int spanMax = 0, int scaleMin =
0, int scaleMax = 0, int scalePoint = 0);
```

Parameters

iRangeRTD	Specify the range type of the RTD input.
chType	Specify the channel type of the measurement channel.
startChNo	Specify the start channel number.
endChNo	Specify the end channel number.
spanMin	Specify the left value of the span.
spanMax	Specify the right value of the span.
scaleMin	Specify the left value of the scale.
scaleMax	Specify the right value of the scale.
scalePoint	Specify the decimal point position for scaling.

Description

Sets the specified RTD range to the measurement channels in the specified channel range (specified by channel type, start channel number, and end channel number).

This function executes the SR command of the DARWIN communication function.

It creates and sends the command and receives the response.

If the left and right values are the same, the span or scale designation is considered omitted.

Return value

Returns an error number.

Reference

runCommand
CDAQDARWINChInfo::toChRange

CDAQDARWIN::setScalingUnit

Syntax

```
int setScalingUnit(const char * strUnit, int chType, int startChNo, int endChNo = 0);
```

Parameters

strUnit	Specify the unit name using a string.
chType	Specify the channel type.
startChNo	Specify the start channel number.
endChNo	Specify the end channel number.

Description

Sets the specified unit to the channels in the specified channel range (specified by channel type, start channel number, and end channel number).

This function executes the SN command of the DARWIN communication function. It creates and sends the command and receives the response.

Return value

Returns an error number.

Error:

Not support	The string is out of range. Not specified, or exceeds the maximum length.
-------------	---

Reference

runCommand
CDAQDARWINChInfo::toChRange

CDAQDARWIN::setSKIP

Syntax

```
int setSKIP(int chType, int startChNo, int endChNo = 0);
```

Parameters

chType	Specify the channel type of the measurement channel.
startChNo	Specify the start channel number.
endChNo	Specify the end channel number.

Description

Sets the measurement channels in the specified channel range (specified by channel type, start channel number, and end channel number) to SKIP (not used).

This function executes the SR command of the DARWIN communication function. It creates and sends the command and receives the response.

Return value

Returns an error number.

Reference

runCommand
CDAQDARWINChInfo::toChRange

CDAQDARWIN::setSTRAIN

Syntax

```
int setSTRAIN(int iRangeSTRAIN, int chType, int startChNo, int
endChNo = 0, int spanMin = 0, int spanMax = 0, int scaleMin =
0, int scaleMax = 0, int scalePoint = 0);
```

Parameters

iRangeSTRAIN	Specify the strain input range.
chType	Specify the channel type of the measurement channel.
startChNo	Specify the start channel number.
endChNo	Specify the end channel number.
spanMin	Specify the left value of the span.
spanMax	Specify the right value of the span.
scaleMin	Specify the left value of the scale.
scaleMax	Specify the right value of the scale.
scalePoint	Specify the decimal point position for scaling.

Description

Sets the specified strain input range to the measurement channels in the specified channel range (specified by channel type, start channel number, and end channel number).

This function executes the SR command of the DARWIN communication function.

It creates and sends the command and receives the response.

If the left and right values are the same, the span or scale designation is considered omitted.

Return value

Returns an error number.

Reference

runCommand CDAQDARWINChInfo::toChRange

CDAQDARWIN::setTC

Syntax

```
int setTC(int iRangeTC, int chType, int startChNo, int endChNo  
= 0, int spanMin = 0, int spanMax = 0, int scaleMin = 0, int  
scaleMax = 0, int scalePoint = 0);
```

Parameters

iRangeTC	Specify the range type of the thermocouple input.
chType	Specify the channel type of the measurement channel.
startChNo	Specify the start channel number.
endChNo	Specify the end channel number.
spanMin	Specify the left value of the span.
spanMax	Specify the right value of the span.
scaleMin	Specify the left value of the scale.
scaleMax	Specify the right value of the scale.
scalePoint	Specify the decimal point position for scaling.

Description

Sets the specified thermocouple range to the measurement channels in the specified channel range (specified by channel type, start channel number, and end channel number).

This function executes the SR command of the DARWIN communication function. It creates and sends the command and receives the response.

If the left and right values are the same, the span or scale designation is considered omitted.

Return value

Returns an error number.

Reference

runCommand
CDAQDARWINChInfo::toChRange

CDAQDARWIN::setVOLT

Syntax

```
int setVOLT(int iRangeVOLT, int chType, int startChNo, int
endChNo = 0, int spanMin = 0, int spanMax = 0, int scaleMin =
0, int scaleMax = 0, int scalePoint = 0);
```

Parameters

iRangeVOLT	Specify the range type of the DC voltage input.
chType	Specify the channel type of the measurement channel.
startChNo	Specify the start channel number.
endChNo	Specify the end channel number.
spanMin	Specify the left value of the span.
spanMax	Specify the right value of the span.
scaleMin	Specify the left value of the scale.
scaleMax	Specify the right value of the scale.
scalePoint	Specify the decimal point position for scaling.

Description

Sets the specified DC voltage range to the measurement channels in the specified channel range (specified by channel type, start channel number, and end channel number). This function executes the SR command of the DARWIN communication function. It creates and sends the command and receives the response. If the left and right values are the same, the span or scale designation is considered omitted.

Return value

Returns an error number.

Reference

runCommand CDAQDARWINChInfo::toChRange

CDAQDARWIN::startTalker

Syntax

```
virtual int startTalker(int iTalk);
```

Parameters

iTalk	Specify the talker function type.
-------	-----------------------------------

Description

Executes the start procedure for using the talker function. This function executes the TS command of the DARWIN communication function and sends a trigger. After executing this function, carry out data retrieval corresponding to the specified talker function.

Return value

Returns an error number.

Reference

runCommand sendTrigger

CDAQDARWIN::talkCalibrationData

Syntax

```
int talkCalibrationData(int startChType, int startChNo, int endChType, int endChNo);
```

Parameters

startChType	Specify the start channel type.
startChNo	Specify the start channel number.
endChType	Specify the end channel type.
endChNo	Specify the end channel number.

Description

Executes the declaration for retrieving setup data of the setup mode from the start channel (start channel type and start channel number) to the end channel (end channel type and end channel number). The setting mode must be switched to calibration in advance. This function executes the TS and LF commands of the DARWIN communication function. After executing this function, use the getSetDataByLine function to retrieve the data by lines.

Return value

Returns an error number.

Reference

```
send startTalker CDAQDARWINChInfo::toChName
```

CDAQDARWIN::talkChInfo

Syntax

```
int talkChInfo(int startChType, int startChNo, int endChType, int endChNo);
```

Parameters

startChType	Specify the start channel type.
startChNo	Specify the start channel number.
endChType	Specify the end channel type.
endChNo	Specify the end channel number.

Description

Executes the declaration for retrieving channel information data from the start channel (start channel type and start channel number) to the end channel (end channel type and end channel number). This function executes the TS and LF commands of the DARWIN communication function. It creates and sends the command. After executing this function, use the getChInfo function to retrieve the data for each channel.

Return value

Returns an error number.

Reference

```
send startTalker CDAQDARWINChInfo::toChName
```

CDAQDARWIN::talkDataByASCII

Syntax

```
int talkDataByASCII(int startChType, int startChNo, int
endChType, int endChNo, CDAQDARWINDateTime & cDARWINDateTime);
```

Parameters

startChType	Specify the start channel type.
startChNo	Specify the start channel number.
endChType	Specify the end channel type.
endChNo	Specify the end channel number.
cDARWINDateTime	Specify the destination where the time information data is to be returned.

Description

Executes the declaration for retrieving measured data in ASCII format from the start channel (start channel type and start channel number) to the end channel (end channel type and end channel number).

Stores the time information data of the measured data to the specified return destination.

Specify measurement channels and computation channels separately. The start channel type is used to distinguish measurement channels and computation channels.

This function executes the TS and FM commands of the DARWIN communication function.

After executing this function, use the getChDataByASCII function to retrieve the data for each channel.

Return value

Returns an error number.

Reference

```
checkAck receiveLine send startTalker
CDAQDARWINChInfo::toChName
CDAQDARWINDateTime::setLine
```

CDAQDARWIN::talkDataByBinary

Syntax

```
int talkDataByBinary(int startChType, int startChNo, int
endChType, int endChNo, CDAQDARWINDateTime & cDARWINDateTime);
```

Parameters

startChType	Specify the start channel type.
startChNo	Specify the start channel number.
endChType	Specify the end channel type.
endChNo	Specify the end channel number.
cDARWINDateTime	Specify the destination where the time information data is to be returned.

Description

Executes the declaration for retrieving measured data in binary format from the start channel (start channel type and start channel number) to the end channel (end channel type and end channel number).

Stores the time information data of the measured data to the specified return destination.

Specify measurement channels and computation channels separately. The start channel type is used to distinguish measurement channels and computation channels.

Set the byte output order to MSB first.

Stores the remaining size (output byte size) in the remaining size field of the data member.

This function executes the BO, TS, and FM commands of the DARWIN communication function.

After executing this function, use the getChDataByBinary function to retrieve the data for each channel.

Return value

Returns an error number.

Error:

Not data	The output number of bytes did not satisfy the size requirement.
----------	--

Reference

```
checkAck receive runCommand send startTalker
CDAQDARWINChInfo::toChName
CDAQDARWINDateTime::setByte
```

CDAQDARWIN::talkOperationData

Syntax

```
int talkOperationData(int startChType, int startChNo, int
endChType, int endChNo);
```

Parameters

startChType	Specify the start channel type.
startChNo	Specify the start channel number.
endChType	Specify the end channel type.
endChNo	Specify the end channel number.

Description

Executes the declaration for retrieving setup data of the operation mode from the start channel (start channel type and start channel number) to the end channel (end channel type and end channel number). This function executes the TS and LF commands of the DARWIN communication function. After executing this function, use the `getSetDataByLine` function to retrieve the data line by line.

Return value

Returns an error number.

Reference

```
send startTalker
CDAQDARWINChInfo::toChName
```

CDAQDARWIN::talkSetupData

Syntax

```
int talkSetupData(int startChType, int startChNo, int
endChType, int endChNo);
```

Parameters

startChType	Specify the start channel type.
startChNo	Specify the start channel number.
endChType	Specify the end channel type.
endChNo	Specify the end channel number.

Description

Executes the declaration for retrieving setup data of the setup mode from the start channel (start channel type and start channel number) to the end channel (end channel type and end channel number). This function executes the TS and LF commands of the DARWIN communication function. After executing this function, use the `getSetDataByLine` function to retrieve the data line by line.

Return value

Returns an error number.

Reference

```
send startTalker
CDAQDARWINChInfo::toChName
```

CDAQDARWIN::transMode

Syntax

```
int transMode(int iMode);
```

Parameters

iMode Specify the mode.

Description

Switches to the specified mode.

This function executes the DS command of the DARWIN communication function.

It creates and sends the command and receives the response.

Return value

Returns an error number.

Reference

runCommand

CDAQDARWINChInfo Class

- CDAQChInfo
 - CDAQDARWINChInfo

This class is derived from the CDAQChInfo class.

It is a class for storing the channel information data of the DARWIN series.

It is a wrapper class of the DarwinChInfo structure.

It stores the following types of data that is retrieved when the channel information data of the talker function is retrieved.

- Channel type
- Channel number
- Decimal point position
- Channel status
- Unit name

Member functions are provided that analyze and store the output format strings below.

S1S2CCCUUUUUU,P

Public Members

Construct/Destruct

CDAQDARWINChInfo	Constructs an object.
~CDAQDARWINChInfo	Destructs an object.

Structure Manipulation

getDarwinChInfo	Gets the data in a structure.
setDarwinChInfo	Sets the data in a structure.
initDarwinChInfo	Initializes the data in a structure.

Member Data Manipulation

getChStatus	Gets the channel status.
getUnit	Gets the unit name.
setChStatus	Sets the channel status.
setUnit	Sets the unit name.

Output Format Manipulation

setLine	Analyzes the data in the output format (line format) of the channel information data and stores the information to the data member.
---------	---

Utilities

getChName	Gets the channel as a string.
toChName	Converts the channel into a string.
toChRange	Converts the channel range into a string.
getStatusName	Gets the status value as a string.
toStatus	Converts a string or data value into a data status value.
toFlag	Converts a character to a flag.
toChType	Converts a character to a channel type.

Operator

operator=	Executes substitution.
-----------	------------------------

Overridden Members

Member Data Manipulation

initialize	Initializes the data member.
------------	------------------------------

Utilities

isObject	Checks an object.
----------	-------------------

Inherited Members

See CDAQChInfo.

getChNo getPoint getChType setChNo setChType setPoint

Protected Members

Data Members

m_chStatus	Field for storing the channel status.
m_strUnit	Field for storing the unit name.

Inherited Members

See CDAQChInfo.

m_chNo m_chType m_point

Private Members

None

Member Functions (Alphabetical Order)

CDAQDARWINChInfo::CDAQDARWINChInfo

Syntax

```
CDAQDARWINChInfo(DarwinChInfo * pDarwinChInfo = NULL);
CDAQDARWINChInfo(int chType, int chNo, int point, const char *
strUnit, int iStatus = DAQDARWIN_DATA_UNKNOWN);
virtual ~CDAQDARWINChInfo(void);
```

Parameters

pDarwinChInfo	Specify the channel information data using a structure.
chType	Specify the channel type.
chNo	Specify the channel number.
point	Specify the decimal point position.
strUnit	Specify the unit name.
iStatus	Specify the channel status.

Description

Constructs or destructs an object.

When constructing, the specified data is stored in the data member. If not specified, the data member is initialized.

Reference

```
setChStatus setDarwinChInfo setUnit
CDAQChInfo::CDAQChInfo
```

CDAQDARWINChInfo::getChName

Syntax

```
int getChName(char * strName, int lenName);
```

Parameters

strName	Specify the field where the string is to be stored.
lenName	Specify the byte size of the field where the string is to be stored.

Description

Creates a channel name from the value of the channel type field and channel number field of the data member as a string and stores the string in the specified field.

Return value

Returns the length of the created string.

Reference

```
getChNo getChType toChName
```

CDAQDARWINChInfo::getChStatus

Syntax

```
int getChStatus(void);
```

Description

Gets the value of the channel status field from the data member.

Return value

Returns the channel status.

CDAQDARWINChInfo::getDarwinChInfo

Syntax

```
void getDarwinChInfo(DarwinChInfo * pDarwinChInfo);
```

Parameters

pDarwinChInfo Specify the destination where the channel information data is to be returned.

Description

Gets the data in a structure.

Stores the contents of the data member in the specified structure.

Reference

getChNo getChStatus getChType getPoint getUnit

CDAQDARWINChInfo::getStatusName

Syntax

```
static const char * getStatusName(int iStatus);
```

Parameters

iStatus Specify the data status value.

Description

Gets the string corresponding to the specified data status value.

If outside the range, the string is set to "iUnknown"

Return value

Returns a pointer to the string.

CDAQDARWINChInfo::getUnit

Syntax

```
const char * getUnit(void);
```

Description

Gets unit name of the unit name field from the data member.

Return value

Returns a pointer to the string.

CDAQDARWINChInfo::initDarwinChInfo

Syntax

```
static void initDarwinChInfo(DarwinChInfo * pDarwinChInfo);
```

Parameters

pDarwinChInfo Specify the channel information data field.

Description

Initializes the specified field.
The default value as a general rule is 0.

CDAQDARWINChInfo::initialize

Syntax

```
virtual void initialize(void);
```

Description

Initializes the data member. The default value is 0.

Reference

```
setChStatus CDAQChInfo::initialize
```

CDAQDARWINChInfo::isObject

Syntax

```
virtual int isObject(const char * classname =  
"CDAQDARWINChInfo");
```

Parameters

classname Specify the class name using a string.

Description

Checks whether the specified class name was inherited.
If parameters are omitted, checks whether it is this class.
Classes that inherit this class must be overridden in order to check their own classes.
Returns true (valid) if the class was inherited. Otherwise, returns false (invalid).
If different from this class, checks the parent class.

Return value

Returns a boolean value.

Reference

```
CDAQChInfo::isObject
```

CDAQDARWINChInfo::operator=

Syntax

```
CDAQDARWINChInfo & operator=(CDAQDARWINChInfo &
cDARWINChInfo);
```

Parameters

cDARWINChInfo Specify an object for substitution.

Description

Copies the data member of the specified object.

Return value

Returns the reference to the object.

CDAQDARWINChInfo::setChStatus

Syntax

```
void setChStatus(int iStatus);
```

Parameters

iStatus Specify the channel status.

Description

Stores the channel status field of the data member to the specified value.

CDAQDARWINChInfo::setDarwinChInfo

Syntax

```
void setDarwinChInfo(DarwinChInfo * pDarwinChInfo);
```

Parameters

pDarwinChInfo Specify the channel information data.

Description

Sets the data in a structure.

Stores the contents of the specified structure in the data member.

If not specified, the data member is initialized.

Reference

```
initialize setChNo setChStatus setChType setPoint setUnit
```

CDAQDARWINChInfo::setLine

Syntax

```
int setLine(const char * strLine, int lenLine, int * pFlag);
```

Parameters

strLine	Specify the line using a string.
lenLine	Specify the byte size of the line.
pFlag	Specify the destination where the flag is to be returned.

Description

Analyzes the specified line and stores the information in the data member.
The line format is the output format of the channel information data.

Return value

Returns an error number.

Error:

Not data	The input data is too short. Or, the string incorrect.
----------	--

Reference

```
setChNo setChStatus setChType setPoint setUnit toChType toFlag  
toStatus
```

CDAQDARWINChInfo::setUnit

Syntax

```
void setUnit(const char * strUnit)
```

Parameters

strUnit	Specify the unit name.
---------	------------------------

Description

Stores the specified value in the unit name field of the data member.

CDAQDARWINChInfo::toChName

Syntax

```
static int toChName(int chType, int chNo, char * strName, int lenName);
```

Parameters

chType	Specify the channel type.
chNo	Specify the channel number.
strName	Specify the field where the string is to be stored.
lenName	Specify the byte size of the field where the string is to be stored.

Description

Creates a channel name from the specified channel type and channel number as a string and stores the string in the specified field.

For example, if the channel type is 0 and the channel number is 1, the string is set to "01"
In addition to the channel/relay types defined as constants, subunit numbers are also channel/relay types. The subunit number is an integer between 0 and 5.

Return value

Returns the length of the created string.

CDAQDARWINChInfo::toChRange

Syntax

```
static int toChRange(int chType, int startChNo, int endChNo, char * strName, int lenName);
```

Parameters

chType	Specify the channel type.
startChNo	Specify the start channel number.
endChNo	Specify the end channel number.
strName	Specify the field where the string is to be stored.
lenName	Specify the byte size of the field where the string is to be stored.

Description

Creates a name of the channel range (channel type, start channel number, and end channel number) as a string and stores the string in the specified field.

If the end channel number is less than or equal to the start channel number, it is considered a single channel specified by the start channel number.

For example, if the channel type is 0, the start channel number is 1, and the end channel number is 2, the string is set to "01-02"

In addition to the channel/relay types defined as constants, subunit numbers are also channel/relay types. The subunit number is an integer between 0 and 5.

Return value

Returns the length of the created string.

Reference

toChName

CDAQDARWINChInfo::toChType

Syntax

```
static int toChType(char cType);
```

Parameters

cType Specify characters.

Description

Converts the specified value into a channel type.

If outside the range, the value is set to 0.

One character of the channel/relay type is converted to the channel type value.

Return value

Returns the channel type.

CDAQDARWINChInfo::toFlag

Syntax

```
static int toFlag(char cFlag);
```

Parameters

cFlag Specify characters.

Description

Converts the specified value into a flag value.

If the specified character is "E" the value is set to "end data."

If outside the range, the string is set to "All OFF." The specified string corresponds to the output format of S2.

Return value

Returns the flag.

CDAQDARWINChInfo::toStatus

Syntax

```
static int toStatus(char cStatus);  
static int toStatus(int value);
```

Parameters

cStatus	Specify characters.
value	Specify 2 bytes of the data value as an integer.

Description

Converts the specified value into a data status value.

If a string is specified and is outside the range, the data status value "Unknown" is returned.

If a data value is specified and is outside the range, "Normal" is returned.

If a string is specified and an overrange is specified, the data status value "Positive overrange" is returned. The data value of the data status for computation channels has 4 bytes. This data consists of two same 2-byte data. Use this 2-byte data when specifying a data value.

The character specified corresponds to the output format of S1.

For characters (spaces) that are read in for instantaneous value data, returns the "instantaneous value data loading communication status."

When specifying data values, values other than abnormal data values are set as out of range, and "completed normally" is returned.

Return value

Returns the data status value.

CDAQDARWINDataInfo Class

- CDAQDataInfo
 - CDAQDARWINDataInfo

This class is derived from the CDAQDataInfo class. It stores the data in units of channels that is retrieved through the retrieval of the measured data by the talker function.

This class consists of the association with the channel information data and the measured data.

The stored information varies depending on the code (ASCII or binary) when the measured data of the talker function is retrieved. For details, see section 11.4.

If the reference to the channel information data is set, the retrieved channel information data is stored.

When storing the measured data through the talker function, the following member functions corresponding to the code are used.

Code	Member Functions
ASCII code	setLine
Binary code	setByte

In the case of ASCII codes, member functions are provided that analyze and store the output format strings below.

S1S2A1A1A2A2A3A3A4A4UUUUUCC_±DDDDDE-E

In the case of binary codes, member functions are provided that analyze and store 6 bytes (for measurement channels) or 8 bytes (for computation channels) of data.

Data that does not include alarm data is not supported.

For channel information data, you can retrieve the CDAQDARWINChInfo class using the member access function member. This is the same as the channel information data class from the receiving of the channel information data by the talker function.

Public Members

Construct/Destruct

CDAQDARWINDataInfo	Constructs an object.
~CDAQDARWINDataInfo	Destructs an object.

Structure Manipulation

getDarwinDataInfo	Gets the data in a structure.
setDarwinDataInfo	Sets the data in a structure.
initDarwinDataInfo	Initializes the data in a structure.

Member Data Manipulation

<code>getStatus</code>	Gets the data status.
<code>getAlarm</code>	Gets the alarm value.
<code>setStatus</code>	Set the data status.
<code>setAlarm</code>	Sets the alarm value.

Output Format Manipulation

<code>setLine</code>	Stores the measured data from the string.
<code>setByte</code>	Stores the measured data from the byte array.

Association

<code>getClassDARWINChInfo</code>	Gets the association with the channel information data.
<code>setClassDARWINChInfo</code>	Sets the association with the channel information data.

Utilities

<code>getAlarmName</code>	Gets the name of the alarm type.
<code>toAlarmType</code>	Converts the string into an alarm type.
<code>getMaxLenAlarmName</code>	Gets the maximum length of the alarm type name.

Operator

<code>operator=</code>	Executes substitution.
------------------------	------------------------

• Overridden Members

Member Data Manipulation

<code>initialize</code>	Initializes the data member.
-------------------------	------------------------------

Utilities

<code>isObject</code>	Checks an object.
-----------------------	-------------------

Inherited Members

See `CDAQDataInfo`

`getClassChInfo` `getDoubleValue` `getStringValue` `getValue`
`setClassChInfo` `setValue` `toDoubleValue` `toStringValue`

Protected Members

Data Members

<code>m_dataStatus</code>	Field for storing the data status.
<code>m_alarm</code>	Field for storing the presence or absence of the alarm.

Inherited Members

See `CDAQDataInfo`

`m_pChInfo` `m_value`

Private Members

None

Member Functions (Alphabetical Order)

CDAQDARWINDataInfo::CDAQDARWINDataInfo

Syntax

```
CDAQDARWINDataInfo(DarwinDataInfo * pDarwinDataInfo = NULL,
CDAQDARWINChInfo * pcDARWINChInfo = NULL);
virtual ~CDAQDARWINDataInfo(void);
```

Parameters

pDarwinDataInfo Specify the measured data.
pcDARWINChInfo Specify the association with the channel information data.

Description

Constructs or destructs an object.

When constructing, the specified data is stored in the data member. If not specified, the data member is initialized.

Reference

```
setClassDARWINChInfo setDarwinDataInfo
CDAQDataInfo::CDAQDataInfo
```

CDAQDARWINDataInfo::getAlarm

Syntax

```
int getAlarm(int levelNo);
```

Parameters

levelNo Specify the alarm level.

Description

Gets the value of the alarm presence/absence field of the data member.

Returns the value corresponding to the specified alarm level.

Returns "No alarm" if the alarm level is outside the range.

The alarm value is the value of the alarm type.

Return value

Returns the presence or absence of the alarm.

CDAQDARWINDataInfo::getAlarmName

Syntax

```
static const char * getAlarmName(int iAlarmType);
```

Parameters

iAlarmType Specify the alarm type.

Description

Gets the string corresponding to the specified alarm type.
If outside the range, the string is set the same as no alarm.
The string is aligned to the left. The string is padded with spaces.

Return value

Returns a pointer to the string.

CDAQDARWINDataInfo::getClassDARWINChInfo

Syntax

```
CDAQDARWINChInfo * getClassDARWINChInfo(void);
```

Description

Gets the association with the channel information data from the data member.

Return value

Returns the association with the channel information data.

Reference

getClassChInfo

CDAQDARWINDataInfo::getDarwinDataInfo

Syntax

```
void getDarwinDataInfo(DarwinDataInfo * pDarwinDataInfo);
```

Parameters

pDarwinDataInfo Specify the destination where the measured data is to be returned.

Description

Gets the data in a structure. Stores the contents of the data member in the specified structure.

Reference

getAlarm getStatus getValue

CDAQDARWINDataInfo::getMaxLenAlarmName

Syntax

```
static int getMaxLenAlarmName(void);
```

Description

Gets the maximum length of the alarm type string.
The return value does not include the terminator.

Return value

Returns the maximum length of the string in numbers of bytes.

CDAQDARWINDataInfo::getStatus

Syntax

```
int getStatus(void);
```

Description

Gets the data status field from the data member.

Return value

Returns the data status.

CDAQDARWINDataInfo::initDarwinDataInfo

Syntax

```
static void initDarwinDataInfo(DarwinDataInfo *  
pDarwinDataInfo);
```

Parameters

pDarwinDataInfo Specify the measured data field.

Description

Initializes the specified field.
The default value as a general rule is 0.

CDAQDARWINDataInfo::initialize

Syntax

```
void initialize(void);
```

Description

Initializes the data member. The default value as a general rule is 0.
Data status field is set to "Unknown."
Presence/absence of alarm field is set to "No alarm."

Reference

CDAQDataInfo::initialize

CDAQDARWINDataInfo::isObject

Syntax

```
virtual int isObject(const char * classname =  
"CDAQDARWINDataInfo");
```

Parameters

classname Specify the class name using a string.

Description

Checks whether the specified class name was inherited.

If parameters are omitted, checks whether it is this class.

Classes that inherit this class must be overridden in order to check their own classes.

Returns true (valid) if the class was inherited. Otherwise, returns false (invalid).

If different from this class, checks the parent class.

Return value

Returns a boolean value.

Reference

CDAQDataInfo::isObject

CDAQDARWINDataInfo::operator=

Syntax

```
CDAQDARWINDataInfo & operator=(CDAQDARWINDataInfo &  
cDARWINDataInfo);
```

Parameters

cDARWINDataInfo Specify an object for substitution.

Description

Copies the data member of the specified object .

Return value

Returns the reference to the object.

CDAQDARWINDataInfo::setAlarm

Syntax

```
void setAlarm(int levelNo, int iAlarmType);
```

Parameters

level Specify the alarm level.

iAlarmType Specify the alarm type.

Description

Stores the specified value in the alarm presence/absence field of the data member.

If the alarm level is outside the range, does nothing.

CDAQDARWINDataInfo::setByte

Syntax

```
int setByte(const unsigned char pByte[], int numByte);
```

Parameters

pByte	Specify the head pointer of the byte array.
numByte	Specify the byte size of the byte array.

Description

Analyzes the specified byte array and stores the information in the data member. The byte array type is the output format of the measured data in binary code. If an association with the channel information data exists, the information that can be retrieved from the specified byte array is stored in the channel information data field.

Return value

Returns an error number.

Error:

Not data	The input data is too short.
----------	------------------------------

Reference

```
getClassDARWINChInfo setAlarm setStatus setValue
CDAQDARWINChInfo::setChNo CDAQDARWINChInfo::setChType
CDAQDARWINChInfo::toStatus
```

CDAQDARWINDataInfo::setClassDARWINChInfo

Syntax

```
void setClassDARWINChInfo(CDAQDARWINChInfo * pcDARWINChInfo);
```

Description

Sets the association with the channel information data of the data member.

Reference

```
setClassChInfo
```

CDAQDARWINDataInfo::setDarwinDataInfo

Syntax

```
void setDarwinDataInfo(DarwinDataInfo * pDarwinDataInfo);
```

Parameters

pDarwinDataInfo	Specify the measured data.
-----------------	----------------------------

Description

Sets the data in a structure. Stores the contents of the specified structure in the data member.

If not specified, the data member is initialized.

Reference

```
initialize setAlarm setStatus setValue
```

CDAQDARWINDataInfo::setLine

Syntax

```
int setLine(const char * strLine, int lenLine, int * pFlag);
```

Parameters

strLine	Specify the line using a string.
lenLine	Specify the byte size of the line.
pFlag	Specify the destination where the flag is to be returned.

Description

Analyzes the specified line and stores the information in the data member. The line format is the output format of the measured data in ASCII code. If an association with the channel information data exists, the information that can be retrieved from the specified line is stored in the channel information data field.

Return value

Returns an error number.
Error:
Not data The input data is too short. Or, the string incorrect.

Reference

```
getClassDARWINChInfo getStatus setAlarm setStatus setValue  
toAlarmType CDAQDARWINChInfo::setChNo  
CDAQDARWINChInfo::setChType CDAQDARWINChInfo::setPoint  
CDAQDARWINChInfo::setUnit CDAQDARWINChInfo::toChType  
CDAQDARWINChInfo::toFlag CDAQDARWINChInfo::toStatus
```

CDAQDARWINDataInfo::setStatus

Syntax

```
void setStatus(int iDataStatus);
```

Parameters

iDataStatus	Specify the data status.
-------------	--------------------------

Description

Stores the data status field of the data member to the specified value.

CDAQDARWINDataInfo::toAlarmType

Syntax

```
static int toAlarmType(const char * strAlarm);
```

Parameters

strAlarm	Specify the name of the alarm type.
----------	-------------------------------------

Description

Converts the specified string into an alarm type.
Returns iNo alarm if the outside the range.

Return value

Returns the alarm type.

CDAQDARWINDateTime Class

- CDAQDateTime
 - CDAQDARWINDateTime

This class is derived from the CDAQDateTime class.

Class for storing time information data of the DARWIN series.

It is a wrapper class of the DarwinDateTime structure.

Stores the time information data that is retrieved when the measured data through the talker function is retrieved.

When storing the measured data through the talker function, the following member functions corresponding to the code are used.

Code	Member Functions
ASCII code	setLine
Binary code	setByte

In the case of ASCII codes, member functions are provided that analyze and store the output format strings below.

DATEYYMMDD
TIMEhhmmss

In the case of binary codes, member functions are provided that analyze and store 6 bytes or 8 bytes (loading instantaneous value data) of data.

Public Members

Construct/Destruct

CDAQDARWINDateTime	Constructs an object.
~CDAQDARWINDateTime	Destructs an object.

Structure Manipulation

getDarwinDateTime	Gets the data in a structure.
setDarwinDateTime	Sets the data in a structure.
initDarwinDateTime	Initializes the data in a structure.

Member Data Manipulation

getYear	Gets the year.
getMonth	Gets the month.
getDay	Gets the day.
getHour	Gets the hours.
getMinute	Gets the minutes.
getSecond	Gets the seconds.
getFullYear	Gets the 4-digit year.

Output Format Manipulation

setLine Stores the time information data from the string.
setByte Stores the time information data from the byte
array.

Utilities

toString Converts the time information data into a string.

Operator

operator= Executes substitution.

Overridden Members

Member Data Manipulation

initialize Initializes the data member.
setNow Sets the current date/time.

Utilities

iisObject Checks an object.

Inherited Members

See CDAQDateTime
getMilliSecond getTime setMilliSecond setTime toLocalDateTime

Protected Members

Data Members

m_DarwinDateTime Field for storing the time information data.

Conversion

toDateTime Converts the data member.

Inherited Members

See CDAQDateTime
m_milliSecond m_time

Private Members

None

Member Functions (Alphabetical Order)

CDAQDARWINDateTime::CDAQDARWINDateTime

Syntax

```
CDAQDARWINDateTime(DarwinDateTime * pDarwinDateTime = NULL);
CDAQDARWINDateTime(int iYaer, int iMonth, int iDay, int iHour
= 0, int iMinute = 0, int iSecond = 0);
virtual ~CDAQDARWINDateTime(void);
```

Parameters

pDarwinDateTime	Specify the time information data.
iYaer	Specify the last two digits of the year.
iMonth	Specify the month.
iDay	Specify the day.
iHour	Specify the hours.
iMinute	Specify the minutes.
iSecond	Specify the seconds.

Description

Constructs or destructs an object. When constructing, the specified data is stored in the data member. If not specified, the data member is initialized.

Reference

```
initialize setDarwinDateTime toDateTime
CDAQDateTime::CDAQDateTime
```

CDAQDARWINDateTime::getDarwinDateTime

Syntax

```
void getDarwinDateTime(DarwinDateTime * pDarwinDateTime);
```

Parameters

pDarwinDateTime	Specify the destination where the time information data is to be returned.
-----------------	--

Description

Gets the data in a structure. Stores the contents of the data member in the specified structure.

CDAQDARWINDateTime::getDay

Syntax

```
int getDay(void);
```

Description

Gets the day from the data member.

Return value

Returns the day.

CDAQDARWINDateTime::getFullYear

Syntax

```
int getFullYear(void);
```

Description

Gets the year from the data member.
Corrects the last 2 digits and returns a 4 digit value.

Return value

Returns the year.

Reference

getYear

CDAQDARWINDateTime::getHour

Syntax

```
int getHour(void);
```

Description

Gets the hours from the data member.

Return value

Returns the hours.

CDAQDARWINDateTime::getMinute

Syntax

```
int getMinute(void);
```

Description

Gets the minutes from the data member.

Return value

Returns the minutes.

CDAQDARWINDateTime::getMonth

Syntax

```
int getMonth(void);
```

Description

Gets the month from the data member.

Return value

Returns the month.

CDAQDARWINDateTime::getSecond

Syntax

```
int getSecond(void);
```

Description

Gets the seconds from the data member.

Return value

Returns the seconds.

CDAQDARWINDateTime::getYear

Syntax

```
int getYear(void);
```

Description

Gets the year from the data member.

Returns the last two digits.

Return value

Returns the year.

CDAQDARWINDateTime::initDarwinDateTime

Syntax

```
static void initDarwinDateTime(DarwinDateTime *  
pDarwinDateTime);
```

Parameters

pDarwinDateTime Specify the time information field.

Description

Initializes the specified field.

The default value as a general rule is 0.

CDAQDARWINDateTime::initialize

Syntax

```
virtual void initialize(void);
```

Description

Initializes the data member. The default value as a general rule is 0.

Reference

```
initDarwinDateTime CDAQDateTime::initialize
```

CDAQDARWINDateTime::isObject

Syntax

```
virtual int isObject(const char * classname =  
"CDAQDARWINDateTime");
```

Parameters

classname Specify the class name using a string.

Description

Checks whether the specified class name was inherited.

If parameters are omitted, checks whether it is this class.

Classes that inherit this class must be overridden in order to check their own classes.

Returns true (valid) if the class was inherited. Otherwise, returns false (invalid).

If different from this class, checks the parent class.

Return value

Returns a boolean value.

Reference

CDAQDateTime::isObject

CDAQDARWINDateTime::operator=

Syntax

```
CDAQDARWINDateTime & operator=(CDAQDARWINDateTime &  
cDARWINDateTime);
```

Parameters

cDARWINDateTime Specify an object for substitution.

Description

Copies the data member of the specified object .

Return value

Returns the reference to the object.

Reference

toDateTime

CDAQDARWINDateTime::setByte

Syntax

```
int setByte(const unsigned char pByte[], int numByte);
```

Parameters

pByte	Specify the head pointer of the byte array.
numByte	Specify the byte size of the byte array.

Description

Analyzes the specified byte array and stores the information in the data member. The format in byte array is the date/time section of the output format of the measured data in binary code.

If the specification is more than 6 bytes, the 7th byte is interpreted as milliseconds. In this case, a value of 0.1 seconds is converted to milliseconds.

Return value

Returns an error number.

Error:

Not data	The input data is too short.
----------	------------------------------

Reference

toDateTime

CDAQDARWINDateTime::setDarwinDateTime

Syntax

```
void setDarwinDateTime(DarwinDateTime * pDarwinDateTime);
```

Parameters

pDarwinDateTime	Specify the time information data.
-----------------	------------------------------------

Description

Sets the data in a structure. Stores the contents of the specified structure in the data member.

If not specified, the data member is initialized.

Reference

initialize toDateTime

CDAQDARWINDateTime::setLine

Syntax

```
int setLine(const char * strLine, int lenLine);
```

Parameters

strLine	Specify the line using a string.
lenLine	Specify the byte size of the line.

Description

Analyzes the specified line and stores the information in the data member. The line format is the first two lines of the output format of the measured data in ASCII code.

Specify the date and time in separate lines.

If the front of the line is D, it is interpreted as YYMMDD format.

If the front of the line is T, it is interpreted as HHMMSS format.

Otherwise, it is interpreted as YY-MM-DD hh:mm:ss.

Return value

Returns an error number.

Error:

Not data	The input data is too short. Or, the string incorrect.
----------	--

Reference

toDateTime

CDAQDARWINDateTime::setNow

Syntax

```
void setNow(void);
```

Description

Gets the current data/time and stores it in the data member.

Reference

initialize
CDAQDateTime::setNow

CDAQDARWINDateTime::toDateTime

Syntax

```
void toDateTime(void);
```

Description

Converts the contents of the structure of the time information data field of the data member structure into seconds elapsed since Jan. 1, 1970, and stores the result in the seconds field.

Years less than 70 are corrected by adding 2000.

Reference

getDay getHour getMinute getMonth getSecond getYear
setMilliSecond setTime

CDAQDARWINDateTime::toString

Syntax

```
int toString(char * strDateTime, int lenDateTime);
```

Parameters

strDateTime Specify the field where the string is to be stored.

lenDateTime Specify the byte size of the field where the string is to be stored.

Description

Converts the time information data from the data member into a string and stores to the specified field.

The format is YY/MM/DD, hh:mm:ss. This is the parameter of the SD command of the DARWIN communication function..

Return value

Returns the length of the created string.

CDAQDARWINSysInfo Class

This class stores the system configuration data and the scan interval of the DARWIN series.

It is a wrapper class of the DarwinSystemInfo structure.

Stores the data that is retrieved when the system configuration data through the talker function is retrieved.

This class can be used as an interface for retrieving system configuration data when retrieving system configuration data.

Public Members

Construct/Destruct

CDAQDARWINSysInfo	Constructs an object.
~CDAQDARWINSysInfo	Destructs an object.

Structure Manipulation

getDarwinSystemInfo	Gets the data in a structure.
setDarwinSystemInfo	Sets the data in a structure.
initDarwinSystemInfo	Initializes the data in a structure.

Member Data Manipulation

initialize	Initializes the data member.
getInterval	Gets the scan interval.
isExist	Checks the presence or absence of the unit.
getModuleName	Gets the module name.
setLine	Stores the system configuration data from the string.
getModuleCode	Gets the internal code of the module.

Utilities

toRelayName	Converts the relay value into a string.
isObject	Checks an object.

Operator

operator=	Executes substitution.
-----------	------------------------

Protected Members

Data Members

m_nInterval	Field for storing the scan interval.
m_systemInfo	Field for storing the system configuration data.

Member Access

getDarwinUnitInfo	Gets the unit information structure.
getDarwinModuleInfo	Gets the module information structure.

Private Members

None

Member Functions (Alphabetical Order)**CDAQDARWINSysInfo::CDAQDARWINSysInfo****Syntax**

```
CDAQDARWINSysInfo(double interval = 0.0, DarwinSystemInfo *
pDarwinSystemInfo = NULL);
virtual ~CDAQDARWINSysInfo(void);
```

Parameters

interval	Specify the scan interval.
pDarwinSystemInfo	Specify the system configuration data.

Description

Constructs or destructs an object.

When constructing, the specified data is stored in the data member. If not specified, the data member is initialized.

Reference

setDarwinSystemInfo

CDAQDARWINSysInfo::getDarwinModuleInfo**Syntax**

```
DarwinModuleInfo * getDarwinModuleInfo(int unitNo, int
slotNo);
```

Parameters

unitNo	Specify the unit number.
slotNo	Specify the slot number.

Description

Gets the module information field from the system data field of the data member corresponding to the specified unit number and slot number.

Returns NULL if the value is outside the range.

Return value

Returns a pointer to the structure.

CDAQDARWINSysInfo::getDarwinSystemInfo

Syntax

```
void getDarwinSystemInfo(DarwinSystemInfo *  
pDarwinSystemInfo);
```

Parameters

pDarwinSystemInfo Specify the destination where the system configuration data is to be returned.

Description

Gets the data in a structure. Stores the contents of the data member in the specified structure.

CDAQDARWINSysInfo::getDarwinUnitInfo

Syntax

```
DarwinUnitInfo * getDarwinUnitInfo(int unitNo);
```

Parameters

unitNo Specify the unit number.

Description

Gets the unit information field from the system data field of the data member corresponding to the specified unit number.
Returns NULL if the value is outside the range.

Return value

Returns a pointer to the structure.

CDAQDARWINSysInfo::getInterval

Syntax

```
double getInterval(void);
```

Description

Gets value of the measurement interval field from the data member.

Return value

Returns the scan interval.

CDAQDARWINSysInfo::getModuleCode

Syntax

```
int getModuleCode(int unitNo, int slotNo);
```

Parameters

unitNo	Specify the unit number.
slotNo	Specify the slot number.

Description

Gets the structure of the specified module internal code structure from the System configuration data field of the data member. Returns 0 if it does not exist.

Return value

Returns the internal code.

Reference

getDarwinModuleInfo

CDAQDARWINSysInfo::getModuleName

Syntax

```
const char * getModuleName(int unitNo, int slotNo);
```

Parameters

unitNo	Specify the unit number.
slotNo	Specify the slot number.

Description

Gets the name of the specified module from the data member.
Returns NULL if it does not exist.

Return value

Returns a pointer to the string.

Reference

getDarwinModuleInfo

CDAQDARWINSysInfo::initDarwinSystemInfo

Syntax

```
static void initDarwinSystemInfo(DarwinSystemInfo *  
pDarwinSystemInfo);
```

Parameters

pDarwinSystemInfo	Specify the system configuration data field.
-------------------	--

Description

Initializes the specified field.
The default value as a general rule is 0.

CDAQDARWINSysInfo::initialize

Syntax

```
void initialize(void);
```

Description

Initializes the data member. The default value as a general rule is 0.
The scan interval field is not initialized.

Reference

```
initDarwinSystemInfo
```

CDAQDARWINSysInfo::isExist

Syntax

```
int isExist(int unitNo);
```

Parameters

unitNo Specify the unit number.

Description

Checks the validity of the specified unit.
If it does not exist, returns Invalid value.

Return value

Returns a boolean value.

Reference

```
getDarwinUnitInfo
```

CDAQDARWINSysInfo::isObject

Syntax

```
virtual int isObject(const char * classname =  
"CDAQDARWINSysInfo");
```

Parameters

classname Specify the class name using a string.

Description

Checks whether the specified class name was inherited.
If parameters are omitted, checks whether it is this class.
Classes that inherit this class must be overridden in order to check their own
classes.
Returns true (valid) if the class was inherited. Otherwise, returns false (invalid).

Return value

Returns a boolean value.

CDAQDARWINSysInfo::operator=

Syntax

```
CDAQDARWINSysInfo & operator=(CDAQDARWINSysInfo &
cDARWINSysInfo);
```

Parameters

cDARWINSysInfo Specify an object for substitution.

Description

Copies the data member of the specified object.

Return value

Returns the reference to the object.

CDAQDARWINSysInfo::setDarwinSystemInfo

Syntax

```
void setDarwinSystemInfo(DarwinSystemInfo *
pDarwinSystemInfo);
```

Parameters

pDarwinSystemInfo Specify the system configuration data.

Description

Sets the data in a structure. Stores the contents of the specified structure in the data member.

If not specified, the data member is initialized.

Reference

initialize

CDAQDARWINSysInfo::setLine

Syntax

```
int setLine(const char * strLine, int lenLine, int * pFlag);
```

Parameters

strLine	Specify the line using a string.
lenLine	Specify the byte size of the line.
pFlag	Specify the destination where the flag is to be returned.

Description

Analyzes the specified line and stores the information in the data member. The line format is the output format of the system configuration data. If the front of the line is M, it is interpreted as measurement interval format. If the front of the line is E, it is interpreted as the last line. If the front of the line is I, it is interpreted as the main unit. Otherwise, it is interpreted as a subunit.

Return value

Returns an error number.
Error:
Not data The input data is too short. Or, the string incorrect.

Reference

getDarwinUnitInfo

CDAQDARWINSysInfo::toRelayName

Syntax

```
static int toRelayName(int relayType, int relayNo, char * strName, int lenName);
```

Parameters

relayType	Specify the relay type.
relayNo	Specify the relay number.
strName	Specify the field where the string is to be stored.
lenName	Specify the byte size of the field where the string is to be stored.

Description

Creates a relay name from the specified relay type and relay number as a string and stores the string in the specified field. If the relay number is 0, the string is set to "iOFF"

Return value

Returns the length of the created string.

Reference

CDAQDARWINChInfo::toChName

8.1 Functions and Their Functionalities - DARWIN/ Visual C -

This section indicates the correspondence between the functionalities that the API supports and the C functions.

Note

This API provides a portion of the functions common to the DARWIN series instruments. Model-specific functions, setup functions of the setup mode, and A/D calibration functions are not implemented. The functions can be added by using the commands of the DARWIN communication function.

The word “command” in the table signifies the command of the DARWIN communication function. For the details on the command, see the Communication Interface User’s Manual.

Communication Functions

Function	Function
Connect to DARWIN.	openDARWIN
Disconnect from DARWIN.	closeDARWIN
Send data line by line.	sendLineDARWIN
Used when controlling the data reception in a special way.	
Receive data line by line.	receiveLineDARWIN
Used when controlling the data reception in a special way.	
Receive data by bytes.	receiveByteDARWIN
Used when controlling the data reception in a special way.	
Send the command and receive the response.	runCommandDARWIN
Used when implementing function commands.	
Get the status byte.	getStatusByteDARWIN
Sends the status byte output command and receives the response.	
Send a trigger command (ESC T), and receive the response.	sendTriggerDARWIN
Used when implementing a new talker function.	
Set the communication timeout.	setTimeoutDARWIN

Note

Setting of the communication timeout is not recommended because unexpected disconnection may occur due to the conflict with the timeout time when data is retrieved.

Control Functions

Functionality	Command	Function
Switch the setting mode.	DS	transModeDARWIN
System reconfiguration	RS	initSystemDARWIN
RAM clear (Initialize the operation mode setup parameter.)	RC	
Alarm reset		AR
Date/Time setting	SD	setDateDARWIN
	SD	setDateNowDARWIN
Calculation start/stop	EX	computeDARWIN
Report start/stop	DR	reportingDARWIN
Establish setup mode	XE	establishDARWIN

Setup Functions

Function	Command	Function
Range Settings		
SKIP (not used)	SR	setSKIPDARWIN
DC voltage input	SR	setVOLTAGE
Thermocouple input	SR	setTCDARWIN
RTD input	SR	setRTDDARWIN
Contact input (DI)	SR	setDIDARWIN
Difference computation between channels	SR	setDELTADARWIN
Remote RJC	SR	setRRJCDARWIN
DC current	SR	setMADARWIN
Strain	SR	setSTRAIN
Pulse	SR	setPULSE
Power monitor	SR	setPOWER
Set the scale unit.	SN	setScalingUnitDARWIN
Set the alarm.	SA	setAlarmDARWIN

Data Retrieval Functions

Function	Command	Function
Get system configuration data.	TS, CF	getSystemConfigDARWIN
Declare the retrieval of the channel information data.	TS, LF	talkChInfoDARWIN
Get channel information data.		getChInfoDARWIN
Declare the retrieval of the measured data (ASCII code).	TS, FM	talkDataByASCIIDARWIN
Get the measured data (ASCII code).		getChDataByASCIIDARWIN
Declare the retrieval of the measured data (binary code).	TS, FM	talkDataByBinaryDARWIN
Get the measured data (binary code).		getChDataByBinaryDARWIN
Declare the retrieval of the setup data (operation mode).	TS, LF	talkOperationDataDARWIN
Get the setup data (operation mode).		getSetDataByLineDARWIN
Declare the retrieval of the setup data (setup mode).	TS, LF	talkSetupDataDARWIN
Get the setup data (setup mode).		getSetDataByLineDARWIN
Declare the retrieval of the setup data(A/D calibration mode).	TS, LF	talkCalibrationDataDARWIN
Get the setup data (A/D calibration mode)		getSetDataByLineDARWIN
Retrieve the report status	TS, RF	getReportStatusDARWIN

Utilities

Function	Function
Convert the measured value into double-precision floating point number.	toDoubleValueDARWIN
Convert the measured value into string.	toStringValueDARWIN
Alarm Get the alarm type string.	toAlarmNameDARWIN
Get the maximum length of the alarm string.	getAlarmNameDARWIN
Get the version number of this API.	getMaxLenAlarmNameDARWIN
Get the revision number of this API.	getVersionAPIDARWIN
Get the error message string.	getRevisionAPIDARWIN
Get the maximum length of the error message string.	getErrorMessageDARWIN
	getMaxLenErrorMessageDARWIN

Implementing Function Commands

Function commands can be implemented by using the DARWIN communication function commands. Below are the DARWIN communication function commands that can be used.

- All communication commands for the DA100 Data Acquisition Unit
- All communication commands for the DC100 Data Collector
- All communication commands for the DR130, DR231, DR232, DR241, and DR242 Hybrid Recorders.

8.2 Programming - DARWIN/Visual C -

Adding the Path to the Include File

Add the path of the include file (DAQDARWIN.h) to the project. The method of adding the include file varies depending on the environment used.

Declaration in the Source File

Write the declaration in the source file.

```
#include "DAQDARWIN.h"
```

Note

The include file of the common section (DAQHandler.h) is referenced from the include file described above. Thus, declaration for it is not necessary.

Load Library Statement

The statement below is added so that the executable module (.dll) of the API can link to the process.

The executable module (.dll) of the API is mapped within the address space (LoadLibrary). Next, the address of the export function in the executable module is retrieved (GetProcAddress).

The callback type of the function pointer is the function name with a prefix "DLL" added and converted to uppercase. It is defined in the include file of the API.

```
HMODULE pDll = LoadLibrary("DAQDARWIN");  
DLLOPENDARWIN openDARWIN = (DLLOPENMX)GetProcAddress(pDll,  
"openDARWIN");
```

Retrieval of the Measured Data

Program Example 1

This program retrieves measured data.

```

////////////////////////////////////
// DARWIN sample for measurement
#include <stdio.h>
#include "DAQDARWIN.h"
////////////////////////////////////
int main(int argc, char* argv[])
{
    int rc; //return code
    DAQDARWIN comm; //discriptor
    int flag;
    DarwinDateTime datetime;
    DarwinChInfo chinfo;
    DarwinDataInfo datainfo;
#ifdef WIN32
    HMODULE pDll; //DLL handle
    //callback
    DLOPENDARWIN openDARWIN;
    DLLCLOSEDARWIN closeDARWIN;
    DLLTALKDATABYBINARYDARWIN talkDataByBinaryDARWIN;
    DLLGETCHDATABYBINARYDARWIN getChDataByBinaryDARWIN;
    //load
    pDll = LoadLibrary("DAQDARWIN");
    //get address
    openDARWIN = (DLOPENDARWIN)GetProcAddress(pDll,
"openDARWIN");
    closeDARWIN = (DLLCLOSEDARWIN)GetProcAddress(pDll,
"closeDARWIN");
    talkDataByBinaryDARWIN =
(DLLTALKDATABYBINARYDARWIN)GetProcAddress(pDll,
"talkDataByBinaryDARWIN");
    getChDataByBinaryDARWIN =
(DLLGETCHDATABYBINARYDARWIN)GetProcAddress(pDll,
"getChDataByBinaryDARWIN");
#endif //WIN32
    //connect
    comm = openDARWIN("192.168.1.11", &rc);
    //get
    rc = talkDataByBinaryDARWIN(comm, 0, 1, 0, 2, &datetime);
    do { //measured data
        rc = getChDataByBinaryDARWIN(comm, &chinfo, &datainfo,
&flag);
    } while (! (flag & DAQDARWIN_FLAG_ENDDATA));
    //disconnect
    rc = closeDARWIN(comm);
#ifdef WIN32
    FreeLibrary(pDll);
#endif
    return rc;
}
////////////////////////////////////

```

Description

Overview

When retrieving data, the talker is executed first, and then data retrieval is executed in units of channels or lines. The end is determined by the flag.

Include File Statement

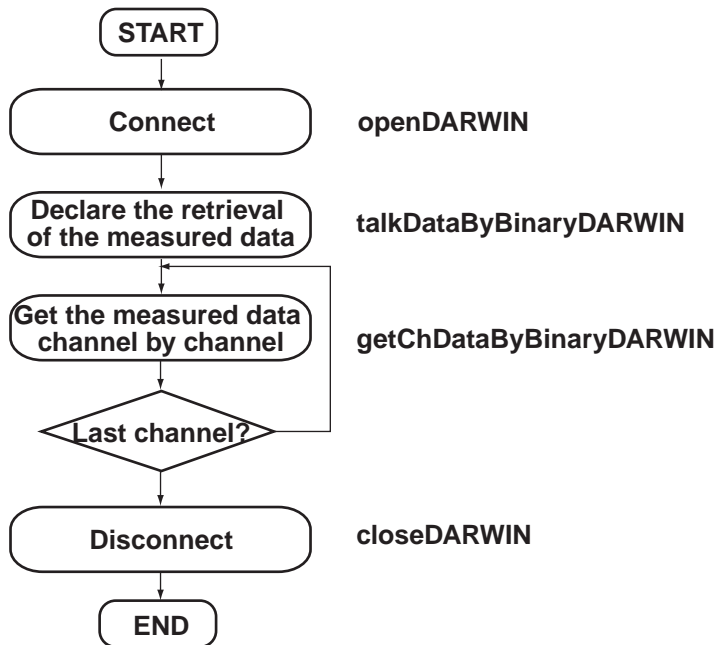
```
#include "DAQDARWIN.h"
```

Load Library Statement

The load library statement is from #ifdef WIN32 to #endif //WIN32. A callback type (such as DLLOPENDARWIN) is used.

Flow of the Process

The flow chart shown below omits the declaration section.



Communication Process

First, make a connection. After making the connection, the functions become available. As a termination procedure, disconnect the communication.

Communication Connection

```
openDARWIN("192.168.1.11", &rc)
```

The IP address of the DARWIN is specified.

The communication report specifies the communication constant DAQDARWIN report number.”

Talker

```
talkDataByBinaryDARWIN(comm, 0, 1, 0, 2, &datetime)
```

Sends the retrieval request of the measured data of channels 1 and 2 of subunit number 0 and retrieves the time information (declares the retrieval of the measured data).

Retrieval of the Measured Data

```
getChDataByBinaryDARWIN(comm, &chinfo, &datainfo, &flag)
```

Gets the measured data channel by channel. It is repeated up to the specified channel.

The end is determined by the flag status of “end data.”.

Comm. cut

```
closeDARWIN(comm)
```

Drops the connection.

Retrieval of Setup Data and Configuration

Program Example 2

This program executes the following two items. This program contains both items, but each item can be written and executed separately.

- Retrieval of setup data
- Setting a DC voltage range to the channel

```

////////////////////////////////////
// DARWIN sample for configuration
#include <stdio.h>
#include "DAQDARWIN.h"////////////////////////////////////
////////////////////////////////////
int main(int argc, char* argv[])
{
    int rc; //return code
    DAQDARWIN comm; //discriptor
    int flag;
    char line[BUFSIZ];
    int len;
#ifdef WIN32
    HMODULE pDll; //DLL handle
    //callback
    DLLOPENDARWIN openDARWIN;
    DLLCLOSEDARWIN closeDARWIN;
    DLLTALKOPERATIONDATADARWIN talkOperationDataDARWIN;
    DLLGETSETDATABYLINEDARWIN getSetDataByLineDARWIN;
    DLLSETVOLTGDARWIN setVOLTGDARWIN;
    //load
    pDll = LoadLibrary("DAQDARWIN");
    //get address
    openDARWIN = (DLLOPENDARWIN)GetProcAddress(pDll,
"openDARWIN");
    closeDARWIN = (DLLCLOSEDARWIN)GetProcAddress(pDll,
"closeDARWIN");
    talkOperationDataDARWIN =
(DLLTALKOPERATIONDATADARWIN)GetProcAddress(pDll,
"talkOperationDataDARWIN");
    getSetDataByLineDARWIN =
(DLLGETSETDATABYLINEDARWIN)GetProcAddress(pDll,
"getSetDataByLineDARWIN");
    setVOLTGDARWIN = (DLLSETVOLTGDARWIN)GetProcAddress(pDll,
"setVOLTGDARWIN");
#endif //WIN32
    //connect
    comm = openDARWIN("192.168.1.11", &rc);

```

```

//get
rc = talkOperationDataDARWIN(comm, 0, 1, 0, 2);
do {
    rc = getSetDataByLineDARWIN(comm, line, BUFSIZ, &len,
&flag);
} while (! (flag & DAQDARWIN_FLAG_ENDDATA));
//range
rc = setVOLTDARWIN(comm, DAQDARWIN_RANGE_VOLT_20MV, 0, 1, 2,
0, 0, 0, 0, 0);
//disconnect
rc = closeDARWIN(comm);
#ifdef WIN32
    FreeLibrary(pDll);
#endif
return rc;
}
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

```

Description

Load Library Statement

The load library state is from `#ifdef WIN32` to `#endif //WIN32`. A callback type (such as `DLLOPENDARWIN`) is used.

Talker

```
talkOperationDataDARWIN(comm, 0, 1, 0, 2)
```

Specifies the type of setup data to be retrieved (setup data of the operation mode) and the channel range (channels 1 and 2 of subunit number 0).

Retrieval of Setup Data

```
getSetDataByLineDARWIN(comm, line, BUFSIZ, &len, &flag)
```

Gets the output by the talker function line by line.

The end is determined by the flag status of "end data."

Setting a DC Voltage Range to the Channel

```
setVOLTDARWIN(comm, DAQDARWIN_RANGE_VOLT_20MV, 0, 1, 2, 0, 0,
0, 0, 0)
```

Sets the measurement range of channels 1 and 2 of subunit number 0 to 20 mV.

The scaling function is not used.

The constant 20 mV is used to specify the range type.

Implementing Function Commands

Program Example 3

This program switches the DARWIN to the operation mode. The program uses the DS command of the DARWIN communication function.

```

////////////////////////////////////
// DARWIN sample for command
#include <stdio.h>
#include "DAQDARWIN.h"
////////////////////////////////////
int main(int argc, char* argv[])
{
    int rc; //return code
    DAQDARWIN comm; //discriptor
    char line[BUFSIZ];
#ifdef WIN32
    HMODULE pDll; //DLL handle
    //callback
    DLLOPENDARWIN openDARWIN;
    DLLCLOSEDARWIN closeDARWIN;
    DLLRUNCOMMANDDARWIN runCommandDARWIN;
    //laod
    pDll = LoadLibrary("DAQDARWIN");
    //get address
    openDARWIN = (DLLOPENDARWIN)GetProcAddress(pDll,
"openDARWIN");
    closeDARWIN = (DLLCLOSEDARWIN)GetProcAddress(pDll,
"closeDARWIN");
    runCommandDARWIN = (DLLRUNCOMMANDDARWIN)GetProcAddress(pDll,
"runCommandDARWIN");
#endif //WIN32
    //connect
    comm = openDARWIN("192.168.1.11", &rc);
    //run
    sprintf(line, "DS%d" DAQDARWIN_MODE_OPE);
    rc = runCommandDARWIN(comm, line);
    //disconnect
    rc = closeDARWIN(comm);
#ifdef WIN32
    FreeLibrary(pDll);
#endif
    return rc;
}
////////////////////////////////////

```

Description

Creating the Message

```
sprintf(line, "DS%d" DAQDARWIN_MODE_OPE)
```

Stores the DS0 (switch to operation mode) command message of the DARWIN communication function in the line array.

The constant "operation mode" is used to specify operation mode.

Sending Messages

```
runCommandDARWIN(comm, line)
```

Sends the command message and receives the response. The number of bytes of the message is not specified (omitted). This member adds a terminator to the message and sends it.

Implementing the Talker Function

Program Example 4

This program retrieves the system configuration data. The program executes the TS and CF commands of the DARWIN communication function.

```

////////////////////////////////////
// DARWIN sample for talker
#include <stdio.h>
#include "DAQDARWIN.h"////////////////////////////////////
////////////////////////////////////
int main(int argc, char* argv[])
{
    int rc; //return code
    DAQDARWIN comm; //discriptor
    char line[BUFSIZ];
    int len;
#ifdef WIN32
    HMODULE pDll; //DLL handle
    //callback
    DLLOPENDARWIN openDARWIN;
    DLLCLOSEDARWIN closeDARWIN;
    DLLSENDLINEDARWIN sendLineDARWIN;
    DLLRECEIVELINEDARWIN receiveLineDARWIN;
    DLLSENDTRIGGERDARWIN sendTriggerDARWIN;
    DLLRUNCOMMANDDARWIN runCommandDARWIN;
    //laod
    pDll = LoadLibrary("DAQDARWIN");
    //get address
    openDARWIN = (DLLOPENDARWIN)GetProcAddress(pDll,
"openDARWIN");
    closeDARWIN = (DLLCLOSEDARWIN)GetProcAddress(pDll,
"closeDARWIN");
    sendLineDARWIN = (DLLSENDLINEDARWIN)GetProcAddress(pDll,
"sendLineDARWIN");
    receiveLineDARWIN =
(DLLRECEIVELINEDARWIN)GetProcAddress(pDll,
"receiveLineDARWIN");
    sendTriggerDARWIN =
(DLLSENDTRIGGERDARWIN)GetProcAddress(pDll,
"sendTriggerDARWIN");
    runCommandDARWIN = (DLLRUNCOMMANDDARWIN)GetProcAddress(pDll,
"runCommandDARWIN");
#endif //WIN32
    //connect
    comm = openDARWIN("192.168.1.11", &rc);
    //talker
    sprintf(line, "TS%d" DAQDARWIN_TALK_SYSINFODATA);
    rc = runCommandDARWIN(comm, line);
    rc = sendTriggerDARWIN(comm);
    rc = sendLineDARWIN(comm, "CF0");

    do {
        rc = receiveLineDARWIN(comm, line, BUFSIZ, &len);
    } while ((rc == 0) && (line[0] != 'E'));
}

```

```

    //disconnect
    rc = closeDARWIN(comm);
#ifdef WIN32
    FreeLibrary(pDll);
#endif
    return rc;
}
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

```

Description

Load Library Statement

The load library statement is from #ifdef WIN32 to #endif //WIN32. A callback type (such as DLLOPENDARWIN) is used.

Talker

```
sprintf(line, "TS%d" DAQDARWIN_TALK_SYSINFODATA)
```

Stores the TS5 (declares the retrieval of the system configuration data) command message of the DARWIN communication function to line.

The constant "system configuration data output" is used to specify the output of the system configuration data.

```
runCommandDARWIN(comm, line)
```

Sends the message and receives the response. This member adds a terminator to the message and sends it.

```
sendTriggerDARWIN(comm)
```

Sends a trigger (device trigger).

Designation of System Configuration Output Format

```
sendLineDARWIN(comm, "CF0"
```

Sends the CF0 communication function command (specify the module information that has been configured for the system). This member adds a terminator to the message and sends it.

Data Retrieval

```
receiveLineDARWIN(comm, line, BUFSIZ, &len)
```

Gets the system configuration data line by line. The program ends when an end mark (E) is returned.

Note

The receiveLine function simply receives the data. The user must write statements for determining the end of the data.

Error Processing

- Most functions return the result of the function process using an error number.
- The function `getErrorMessageDARWIN` can be used to get the error message string corresponding to the error number. A function for retrieving the maximum length of the error message string is also available.

9.1 Functions and Their Functionalities - DARWIN/ Visual Basic -

This section indicates the correspondence between the functionalities that the API supports and the Visual Basic functions.

Note

This API provides a portion of the functions common to the DARWIN series instruments. Model-specific functions, setup functions of the setup mode, and A/D calibration functions are not implemented. The functions can be added by using the commands of the DARWIN communication function.

The word “command” in the table signifies the command of the DARWIN communication function. For the details on the command, see the Communication Interface User’s Manual.

Communication Functions

Function	Function
Connect to DARWIN.	openDARWIN
Disconnect from DARWIN.	closeDARWIN
Send data line by line. Used when controlling the data transmission in a special way.	sendLineDARWIN
Receive data line by line. Used when controlling the data reception in a special way.	receiveLineDARWIN
Receives data in units of bytes. Used when controlling the data reception in a special way.	receiveByteDARWIN
Sends the command and receive the response. Used when implementing function commands.	runCommandDARWIN
Get the status byte. Sends the status byte output command and receives the response.	getStatusByteDARWIN
Send a trigger command (ESC T), and receive the response. Used when implementing a new talker function.	sendTriggerDARWIN
Set the communication timeout.	setTimeoutDARWIN

Note

Setting of the communication timeout is not recommended because unexpected disconnection may occur due to the conflict with the timeout time when data is retrieved.

Control Functions

Function	Command	Function
Switch the setting mode.	DS	transModeDARWIN
System reconfiguration	RS	initSystemDARWIN
RAM clear (Initialize the operation mode setup parameter.)	RC	
Alarm reset	AR	
Date/time setting	SD	setDateTimeDARWIN
	SD	setDateTimeNowDARWIN
Calculation start/stop	EX	computeDARWIN
Report start/stop	DR	reportingDARWIN
Finalize setup mode	XE	establishDARWIN

Setup Functions

Function	Command	Function
Range Settings SKIP (not used)	SR	setSKIPDARWIN
DC voltage input	SR	setVOLTDARWIN
Thermocouple input	SR	setTCDARWIN
RTD input	SR	setRTDDARWIN
Contact input (DI)	SR	setDIDARWIN
Difference computation between channels	SR	setDELTADARWIN
Remote RJC	SR	setRRJCDARWIN
DC current	SR	setMADARWIN
Strain	SR	setSTRAIN DARWIN
Pulse	SR	setPULSEDARWIN
Power monitor	SR	setPOWERDARWIN
Set the scale unit.	SN	setScalingUnitDARWIN
Set the alarm.	SA	setAlarmDARWIN

Data Retrieval Functions

Function	Command	Function
Get system configuration data.	TS, CF	getSystemConfigDARWIN
Declare the retrieval of the channel information data.	TS, LF	talkChInfoDARWIN
Get channel information data.		getChInfoDARWIN
Declare the retrieval of the measured data (ASCII code).	TS, FM	talkDataByASCIIDARWIN
Get the measured data (ASCII code).		getChDataByASCIIDARWIN
Declare the retrieval of the measured data (binary code).	TS, FM	talkDataByBinaryDARWIN
Get the measured data (binary code).		getChDataByBinaryDARWIN
Declare the retrieval of the setup data (operation mode).	TS, LF	talkOperationDataDARWIN
Get the setup data (operation mode).		getSetDataByLineDARWIN
Declare the retrieval of the setup data (setup mode).	TS, LF	talkSetupDataDARWIN
Get the setup data (setup mode).		getSetDataByLineDARWIN
Declare the retrieval of the setup data (A/D calibration mode).	TS, LF	talkCalibrationDataDARWIN
Get the setup data (A/D calibration mode)		getSetDataByLineDARWIN
Retrieve the report status	TS, RF	getReportStatusDARWIN

Utilities

Function	Function
Convert the measured value into double-precision floating point number.	toDoubleValueDARWIN
Convert the measured value into string.	toStringValueDARWIN
Alarm	Get the alarm type string.
	Get the maximum length of the alarm string.
	toAlarmNameDARWIN
	getMaxLenAlarmNameDARWIN
Get the version number of this API.	getVersionAPIDARWIN
Get the revision number of this API.	getRevisionAPIDARWIN
Get the error message string.	toErrorMessageDARWIN
Get the maximum length of the error message string.	getMaxLenErrorMessageDARWIN

Implementing Function Commands

Function commands can be implemented by using the DARWIN communication function commands. Below are the DARWIN communication function commands that can be used.

- All communication commands for the DA100 Data Acquisition Unit
- All communication commands for the DC100 Data Collector
- All communication commands for the DR130, DR231, DR232, DR241, and DR242 Hybrid Recorders.

9.2 Programming - DARWIN/Visual Basic -

Declaration of Types, Functions, and Constants

To use types, functions, and constants for Visual Basic, they must be declared in advance. The following methods of declaration statements are available.

Statement of All Declarations

Adding the standard module library file for Visual Basic (DAQDARWIN.bas) to the project is equivalent to declaring all types, functions, and constants.

Statement of Selective Declarations

The API Viewer that comes with Visual Studio can be used to copy the declaration statements of arbitrary types, functions, and constants. Load the text file for the API Viewer (DAQDARWIN.txt) on the API Viewer to use this function.

For a description of how to use the API Viewer, read the operation manual for Visual Studio.

Writing Declarations Directly

Below is an example of a declaration statement.

```
Public Declare Function openDARWIN Lib "DAQDARWIN" (ByVal  
strAddress As String, ByVal errorCode As Long) As Long
```

Retrieval of the Measured Data

Program Example 1

This program retrieves measured data.

```
Public Function Main()
Dim datetime As DarwinDateTime
Dim chinfo As DarwinChInfo
Dim datainfo As DarwinDataInfo
'connect
host = "192.168.1.11"
comm = openDARWIN(host, rc)
'get
rc = talkDataByBinaryDARWIN(comm, 0, 1, 0, 2, datetime)
Do
    rc = getChDataByBinaryDARWIN(comm, chinfo, datainfo, flag)
Loop While (flag And DAQDARWIN_FLAG_ENDDATA) = 0
'disconnect
rc = closeDARWIN(comm)
End Function
```

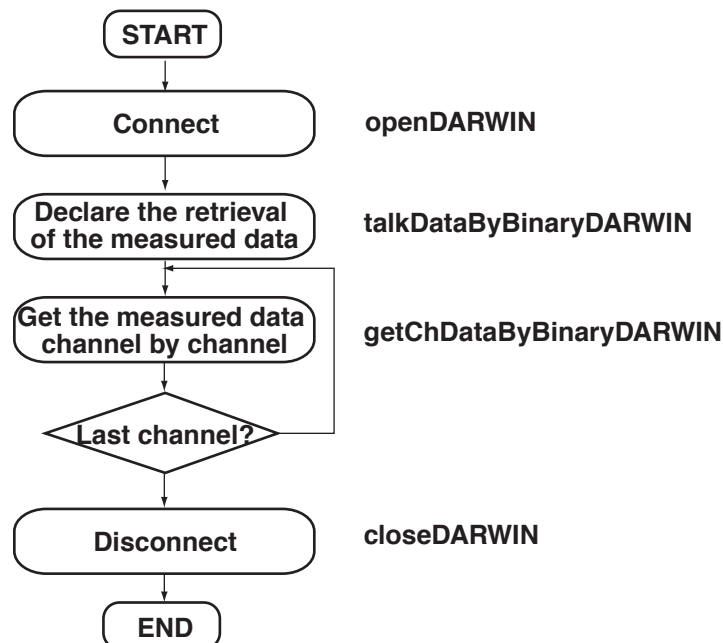
Description

Overview

When retrieving data, the talker is executed first, and then data retrieval is executed in units of channels or lines. The end is determined by the flag.

Flow of the Process

The flow chart shown below omits the declaration section.



Communication Process

First, make a connection. After making the connection, the functions become available. As a termination procedure, disconnect the communication.

Communication Connection

`openDARWIN(host, rc)`

The IP address of the DARWIN is specified. The communication report specifies the communication constant "DAQDARWIN report number."

Talker

`talkDataByBinaryDARWIN(comm, 0, 1, 0, 2, datetime)`

Sends the retrieval request of the measured data of channels 1 and 2 of subunit number 0 and retrieves the time information (declares the retrieval of the measured data).

Retrieval of the Measured Data

`getChDataByBinaryDARWIN(comm, chinfo, datainfo, flag)`

Gets the measured data channel by channel. It is repeated up to the specified channel.

The end is determined by the flag status of "end data."

Comm. cut

`closeDARWIN(comm)`

Drops the connection.

Retrieval of Setup Data and Configuration

Program Example 2

This program executes the following two items. This program contains both items, but each item can be written and executed separately.

- Retrieval of setup data
- Setting a DC voltage range to the channel

```
Public Function Main()
Dim line As String * 256
'connect
host = "192.168.1.11"
comm = openDARWIN(host, rc)
'get
rc = talkOperationDataDARWIN(comm, 0, 1, 0, 2)
Do
    rc = getSetDataByLineDARWIN(comm, line, 256, lenLine,
flag)
Loop While (flag And DAQDARWIN_FLAG_ENDDATA) = 0
'range
rc = setVOLTDARWIN(comm, DAQDARWIN_RANGE_VOLT_20MV, 0, 1, 2,
0, 0, 0, 0, 0)
'disconnect
rc = closeDARWIN(comm)
End Function
```

Description

Talker

```
talkOperationDataDARWIN(comm, 0, 1, 0, 2)
```

Specifies the type of setup data to be retrieved (setup data of the operation mode) and the channel range (channels 1 and 2 of subunit number 0).

Retrieval of Setup Data

```
getSetDataByLineDARWIN(comm, line, 256, lenLine, flag)
```

Gets the output by the talker function line by line in a 256-byte field.

The end is determined by the flag status of "end data."

Setting a DC voltage range to the channel

```
setVOLTDARWIN(comm, DAQDARWIN_RANGE_VOLT_20MV, 0, 1, 2, 0, 0,
0, 0, 0)
```

Sets the measurement range of channels 1 and 2 of subunit number 0 to 20 mV.

The scaling function is not used.

The constant 20 mV is used to specify the range type.

Implementing Function Commands

Program Example 3

This program switches DARWIN to the operation mode. The program uses the DS command of DARWIN communication function.

```
Public Function Main()  
    'connect  
    host = "192.168.1.11"  
    comm = openDARWIN(host, rc)  
    'run  
    Line = "DS0"rc = runCommandDARWIN(comm, Line)  
    'disconnect  
    rc = closeDARWIN(comm)  
End Function
```

Description

Sending Messages

runCommandDARWIN(comm, line)

Sends the command message and receives the response. This member adds a terminator to the message and sends it.

Implementing the Talker Function

Program Example 4

This program retrieves the system configuration data. The program executes the TS and CF commands of the DARWIN communication function.

```
Public Function Main()
Dim lenLine As Long
Dim line As String * 256
'connect
host = "192.168.1.11"
comm = openDARWIN(host, rc)
'talker
rc = runCommandDARWIN(comm, "TS5")
rc = sendTriggerDARWIN(comm)
rc = sendLineDARWIN(comm, "CF0")
Do
    rc = receiveLineDARWIN(comm, line, 256, lenLine)
Loop While ((rc = 0) And (Left(line, 1) <> "E"))
'disconnect
rc = closeDARWIN(comm)
End Function
```

Description

Talker

```
runCommandDARWIN(comm, "TS5")
```

Sends the TS5 (declares the retrieval of the system configuration data) command message of the DARWIN communication function and receives the response. This member adds a terminator to the message and sends it.

```
sendTriggerDARWIN(comm)
```

Sends a trigger (device trigger).

Designation of System Configuration Output Format

```
sendLineDARWIN(comm, "CF0")
```

Sends the CF0 communication function command (specify the module information that has been configured for the system). This member adds a terminator to the message and sends it.

Data Retrieval

```
receiveLineDARWIN(comm, line, 256, lenLine)
```

Stores the system configuration data line by line to a 256-byte field. The program ends when an end mark (E) is returned.

Note

The receiveLine function simply receives the data. The user must write statements for determining the end of the data.

Error Processing

- Most functions return the result of the function process using an error number.
- The function `toErrorMessageDARWIN` can be used to get the error message string corresponding to the error number. A function for retrieving the maximum length of the error message string is also available.

10.1 Details of Functions - DARWIN (Visual C/Visual Basic) -

This section describes the DARWIN functions that are used in C and Visual Basic. The functions are listed in alphabetical order by the function name.

For details on constants and types, see chapter 11.

For DARWIN terminology, see appendix 2.

Most functions return an error number as a return value. Error number 0 is returned if there is no error.

closeDARWIN

Syntax

```
int closeDARWIN(DAQDARWIN daqdarwin);
```

Declaration

```
Public Declare Function closeDARWIN Lib "DAQDARWIN" (ByVal  
daqdarwin As Long) As Long
```

Parameters

daqdarwin Specify the device descriptor.

Description

Disconnects the communication using the specified device descriptor.

- When the communication is disconnected, the value of the device descriptor is meaningless.
- After disconnection, do not use the value of the device descriptor.

Return value

Returns an error number.

Error:

Not descriptor No device descriptor.

Reference

CDAQDARWIN::close

computeDARWIN

Syntax

```
int computeDARWIN(DAQDARWIN daqdarwin, int iCompute);
```

Declaration

```
Public Declare Function computeDARWIN Lib "DAQDARWIN" (ByVal  
daqdarwin As Long, ByVal iCompute As Long) As Long
```

Parameters

daqdarwin	Specify the device descriptor.
iCompute	Specify the computation.

Description

Starts/stops computation.

- Valid with the computation function.
- This function executes the EX command of the DARWIN communication function.

Return value

Returns an error number.

Error:

Not descriptor	No device descriptor.
----------------	-----------------------

Reference

CDAQDARWIN::compute

establishDARWIN

Syntax

```
int establishDARWIN(DAQDARWIN daqdarwin, int iSetup);
```

Declaration

```
Public Declare Function establishDARWIN Lib "DAQDARWIN" (ByVal  
daqdarwin As Long, ByVal iSetup As Long) As Long
```

Parameters

daqdarwin	Specify the device descriptor.
iSetup	Specifies establishment of setup.

Description

Establishes setting contents for setup mode.

- It is only valid in setup mode.
- This function executes the EX command of the DARWIN communication function.

Return value

Returns an error number.

Error:

Not descriptor	No device descriptor.
----------------	-----------------------

Reference

CDAQDARWIN::establish

getAlarmNameDARWIN **[Visual C only]**

Syntax

```
const char * getAlarmNameDARWIN(int iAlarmType);
```

Parameters

iAlarmType Specify the alarm type.

Description

Gets the string corresponding to the specified alarm type.

- In Visual Basic, use the toAlarmNameDARWIN function.

Return value

Returns a pointer to the alarm type string.

Reference

CDAQDARWINDataInfo::getAlarmName

getChDataByASCIIDARWIN

Syntax

```
int getChDataByASCIIDARWIN(DAQDARWIN daqdarwin, DarwinChInfo *  
pDarwinChInfo, DarwinDataInfo * pDarwinDataInfo, int * pFlag);
```

Declaration

```
Public Declare Function getChDataByASCIIDARWIN Lib  
"DAQDARWIN"(ByVal daqdarwin As Long, ByVal pDarwinChInfo As  
DarwinChInfo, ByVal pDarwinDataInfo As DarwinDataInfo, ByVal  
pFlag As Long) As Long
```

Parameters

daqdarwin	Specify the device descriptor.
pDarwinChInfo	Specify the destination where the channel information data is to be returned.
pDarwinDataInfo	Specify the destination where the measured data is to be returned.
pFlag	Specify the destination where the flag is to be returned.

Description

Gets the output of each channel using the talker function declared by the talkDataByASCIIDARWIN function.

- Analyzes received information channel by channel and stores it in the structure.
- Stores channel information data and measured data if the return destination is specified.
- When the last set of data is retrieved, the flag status is set. The flag status is also set when the function ends in error.
- Do not perform communications using other functions until the data retrieval is completed. Other functions may not operate properly.

Return value

Returns an error number.

Error:

Not descriptor	No device descriptor.
----------------	-----------------------

Reference

```
CDAQDARWIN::getChDataByASCII  
CDAQDARWINChInfo::getDarwinChInfo  
CDAQDARWINDataInfo::getDarwinDataInfo
```

getChDataByBinaryDARWIN

Syntax

```
int getChDataByBinaryDARWIN(DAQDARWIN daqdarwin, DarwinChInfo
* pDarwinChInfo, DarwinDataInfo * pDarwinDataInfo, int *
pFlag);
```

Declaration

```
Public Declare Function getChDataByBinaryDARWIN Lib
"DAQDARWIN" (ByVal daqdarwin As Long, ByRef pDarwinChInfo As
DarwinChInfo, ByRef pDarwinDataInfo As DarwinDataInfo, ByRef
pFlag As Long) As Long
```

Parameters

daqdarwin	Specify the device descriptor.
pDarwinChInfo	Specify the destination where the channel information data is to be returned.
pDarwinDataInfo	Specify the destination where the measured data is to be returned.
pFlag	Specify the destination where the flag is to be returned.

Description

Gets the output of each channel using talkDataByBinaryDARWIN.

- Analyzes received information channel by channel and stores it in the structure.
- Stores channel information data and measured data if the return destination is specified.
- When the last set of data is retrieved, the flag status is set. The flag status is also set when the function ends in error.
- Do not perform communications using other functions until the data retrieval is completed. Other functions may not operate properly.

Return value

Returns an error number.

Error:

Not descriptor No device descriptor.

Reference

```
CDAQDARWIN::getChDataByBinary
CDAQDARWINChInfo::getDarwinChInfo
CDAQDARWINDataInfo::getDarwinDataInfo
```


getChInfoDARWIN

Syntax

```
int getChInfoDARWIN(DAQDARWIN daqdarwin, DarwinChInfo *  
pDarwinChInfo, int * pFlag);
```

Declaration

```
Public Declare Function getChInfoDARWIN Lib "DAQDARWIN" (ByVal  
daqdarwin As Long, ByRef pDarwinChInfo As DarwinChInfo, ByRef  
pFlag As Long) As Long
```

Parameters

daqdarwin	Specify the device descriptor.
pDarwinChInfo	Specify the destination where the channel information data is to be returned.
pFlag	Specify the destination where the flag is to be returned.

Description

Gets the channel information data output by channel using the talker function declared by the talkChInfoDARWIN function.

- Analyzes received information channel by channel and stores it in the structure.
- Stores channel information data if the return destination is specified.
- When the last set of data is retrieved, the flag status is set. The flag status is also set when the function ends in error.
- Do not perform communications using other functions until the data retrieval is completed. Other functions may not operate properly.

Return value

Returns an error number.

Error:

Not descriptor No device descriptor.

Reference

CDAQDARWIN::getChInfo

CDAQDARWINChInfo::getDarwinChInfo

getMessageDARWIN**[Visual C only]**

Syntax

```
const char * getMessageDARWIN(int errorCode);
```

Parameters

errorCode Specify the error number.

Description

Gets the error message string corresponding to the error number.

- In Visual Basic, use the function toErrorMessageDARWIN.

Return value

Returns the pointer to the error message string corresponding to the error number.

Reference

CDAQDARWIN::getMessage

getMaxLenAlarmNameDARWIN

Syntax

```
int getMaxLenAlarmNameDARWIN(void);
```

Declaration

```
Public Declare Function getMaxLenAlarmNameDARWIN Lib  
"DAQDARWIN"() As Long
```

Description

Gets the maximum length of the alarm type string.

- The return value does not include the terminator.

Return value

Returns the length of the string.

Reference

```
CDAQDARWINDataInfo::getMaxLenAlarmName
```

getMaxLenErrorMessageDARWIN

Syntax

```
int getMaxLenErrorMessageDARWIN(void);
```

Declaration

```
Public Declare Function getMaxLenErrorMessageDARWIN Lib  
"DAQDARWIN"() As Long
```

Description

Gets the maximum length of the error message string.

- The return value does not include the terminator.

Return value

Returns the length of the string.

Reference

CDAQDARWIN::getMaxLenErrorMessage

getReportStatusDARWIN

Syntax

```
int getReportStatusDARWIN(DAQDARWIN daqdarwin, int *  
pReportStatus);
```

Declaration

```
Public Declare Function getReportStatusDARWIN Lib  
"DAQDARWIN"(ByVal daqdarwin As Long, ByRef pReportStatus As  
Long) As Long
```

Parameters

daqdarwin Specify the device descriptor.
pReportStatus Specify the destination where the report status is to be returned.

Description

Gets the report status.

- Receives the report status output using the talker function.
- Stores the report status in the specified location if the return destination is specified.
- This function executes the TS and RF commands of the DARWIN communication function.

Return value

Returns an error number.

Error

Not descriptor No device descriptor.

Reference

CDAQDARWIN::getReportStatus

getRevisionAPIDARWIN

Syntax

```
const int getRevisionAPIDARWIN(void);
```

Declaration

```
Public Declare Function getRevisionAPIDARWIN Lib "DAQDARWIN"()  
As Long
```

Description

Gets the revision number of this API.

Return value

Returns the revision number.

Reference

CDAQDARWIN::getRevisionAPI

getSetDataByLineDARWIN

Syntax

```
int getSetDataByLineDARWIN(DAQDARWIN daqdarwin, char *  
strLine, int maxLine, int * lenLine, int * pFlag);
```

Declaration

```
Public Declare Function getSetDataByLineDARWIN Lib  
"DAQDARWIN"(ByVal daqdarwin As Long, ByVal strLine As String,  
ByVal maxLine As Long, ByRef lenLine As Long, ByRef flag As  
Long) As Long
```

Parameters

daqdarwin	Specify the device descriptor.
strLine	Specify the field where the string received by lines is to be stored.
maxLine	Specify the byte size of the field where the string received by lines is to be stored.
lenLine	Specify the destination where the byte size of the actual string received is returned.
pFlag	Specify the destination where the flag is to be returned.

Description

Gets the output from the talker function in units of lines after execution of the declaration for the retrieval of setup data.

- Stores the received string excluding line feeds.
- When the last set of data is retrieved, the flag status is set. The flag status is also set when the function ends in error.
- Do not perform communications using other functions until the data retrieval is completed. Other functions may not operate properly.

Return value

Returns an error number.

Error:

Not descriptor No device descriptor.

Reference

CDAQDARWIN::getSetDataByLine

getStatusByteDARWIN

Syntax

```
int getStatusByteDARWIN(DAQDARWIN daqdarwin, int *  
pStatusByte);
```

Declaration

```
Public Declare Function getStatusByteDARWIN Lib  
"DAQDARWIN" (ByVal daqdarwin As Long, ByRef pStatusByte As  
Long) As Long
```

Parameters

daqdarwin Specify the device descriptor.
pStatusByte Specify the destination where the status byte is to be returned.

Description

Sends the status byte output command (ESC S) and receives the status bytes.
Stores the status byte as an integer to the specified destination if the return destination is specified.

Return value

Returns an error number.
Error:
Not descriptor No device descriptor.

Reference

CDAQDARWIN::getStatusByte

getSystemConfigDARWIN

Syntax

```
int getSystemConfigDARWIN(DAQDARWIN daqdarwin, double *  
interval, DarwinSystemInfo * pDarwinSystemInfo);
```

Declaration

```
Public Declare Function getSystemConfigDARWIN Lib  
"DAQDARWIN"(ByVal daqdarwin As Long, ByRef interval As Double,  
ByRef pDarwinSystemInfo As DarwinSystemInfo) As Long
```

Parameters

daqdarwin	Specify the device descriptor.
interval	Specify the destination where the measurement interval is to be returned.
pDarwinSystemInfo	Specify the destination where the system configuration data is to be returned.

Description

Gets the system configuration data.

- Receives the system configuration data using the talker function.
- Stores the measurement interval and system configuration data if the return destination is specified.
- This function executes the TS and CF commands of the DARWIN communication function.

Return value

Returns an error number.

Error:

Not descriptor	No device descriptor.
----------------	-----------------------

Reference

```
CDAQDARWIN::getSystemConfig  
CDAQDARWINSysInfo::getDarwinSystemInfo  
CDAQDARWINSysInfo::getInterval
```

getVersionAPIDARWIN

Syntax

```
const int getVersionAPIDARWIN(void);
```

Declaration

```
Public Declare Function getVersionAPIDARWIN Lib "DAQDARWIN"()  
As Long
```

Description

Gets the version number of this API.

Return value

Returns the version number.

Reference

CDAQDARWIN::getVersionAPI

initSystemDARWIN

Syntax

```
int initSystemDARWIN(DAQDARWIN daqdarwin, int iCtrl);
```

Declaration

```
Public Declare Function initSystemDARWIN Lib "DAQDARWIN" (ByVal  
daqdarwin As Long, ByVal iCtrl As Long) As Long
```

Parameters

daqdarwin Specify the device descriptor.
iCtrl Specify the system control type.

Description

Executes the operation of the specified system control type.

This function executes the RS, RC, or AR command of the DARWIN communication function.

Return value

Returns an error number.

Error:

Not descriptor No device descriptor.

Reference

CDAQDARWIN::initSystem

openDARWIN

Syntax

```
DAQDARWIN openDARWIN(const char * strAddress, int *
  errorCode);
```

Declaration

```
Public Declare Function openDARWIN Lib "DAQDARWIN" (ByVal
  strAddress As String, ByRef errorCode As Long) As Long
```

Parameters

strAddress	Specify the IP address as a string.
errorCode	Specify the destination where the error number is to be returned.

Description

Connects to the device with the IP address specified by the parameters.

- Creates a device descriptor and returns the value as a return value.
- Stores the error number in the return specified destination if one is specified.
- The port number fixed, and set to the communication constant “communication port number.”
- If unsuccessful, returns NULL in Visual C or 0 in Visual Basic.

Return value

Returns the device descriptor.

Error

Creating descriptor is failure	Failed to create the device descriptor.
--------------------------------	---

Reference

CDAQDARWIN::open

receiveByteDARWIN

Syntax

```
int receiveByteDARWIN(DAQDARWIN daqdarwin, unsigned char *  
byteData, int maxData, int * lenData)
```

Declaration

```
Public Declare Function receiveByteDARWIN Lib  
"DAQDARWIN"(ByVal daqdarwin As Long, ByRef byteData() As Byte,  
ByVal maxData As Long, ByRef lenData As Long) As Long
```

Parameters

daqdarwin	Specify the device descriptor.
byteData	Specify the field where the received byte data is to be stored.
maxData	Specify the byte size of the received data.
lenData	Specify the destination where the byte size of the actual data received is returned.

Description

Stores the received data in the field specified by the parameter up to the specified byte size.

- Returns the byte size of the actual data received if the return destination is specified.
- If multiple bytes of data exist, repeat the function.
- Do not perform communications using other functions until the data retrieval is completed. Other functions may not operate properly.
- The user must carry out determination of the data end.
- Used for receiving binary output when implementing a model-specific talker function.

Return value

Returns an error number.

Error:

Not descriptor No device descriptor.

Reference

CDAQDARWIN::receiveByte

receiveLineDARWIN

Syntax

```
int receiveLineDARWIN(DAQDARWIN daqdarwin, char * strLine, int
maxLine, int * lenLine);
```

Declaration

```
Public Declare Function receiveLineDARWIN Lib
"DAQDARWIN" (ByVal daqdarwin As Long, ByVal strLine As String,
ByVal maxLine As Long, ByRef lenLine As Long) As Long
```

Parameters

daqdarwin	Specify the device descriptor.
strLine	Specify the field where the received string is to be stored.
maxLine	Specify the byte size of the field where the received string is to be stored.
lenLine	Specify the destination where the byte size of the actual string received is returned.

Description

Receives data in the field specified for storing received strings by the parameter, until a line feed is detected or up to the specified byte size.

- Stores the received string excluding line feeds in the storage field.
- Stores in the specified destination the byte size of the actual data received and stored if the return destination is specified.
- If multiple lines of data exist, repeat the function.
- Do not perform communications using other functions until the data retrieval is completed. Other functions may not operate properly.
- The user must carry out determination of the data end.

Return value

Returns an error number.

Error:

Not descriptor No device descriptor.

Reference

CDAQDARWIN::receiveLine

reportingDARWIN

Syntax

```
int reportingDARWIN(DAQDARWIN daqdarwin, int iReportRun);
```

Declaration

```
Public Declare Function reportingDARWIN Lib "DAQDARWIN" (ByVal  
daqdarwin As Long, ByVal iReportRun As Long) As Long
```

Parameters

daqdarwin Specify the device descriptor.
iReportRun Specify the report execution type.

Description

Starts/stops reporting.

- Valid with the report option.
- This function executes the DR command of the DARWIN communication function.

Return value

Returns an error number.

Error:

Not descriptor No device descriptor.

Reference

CDAQDARWIN::reporting

runCommandDARWIN

Syntax

```
int runCommandDARWIN(DAQDARWIN daqdarwin, const char *  
strCmd);
```

Declaration

```
Public Declare Function runCommandDARWIN Lib "DAQDARWIN" (ByVal  
daqdarwin As Long, ByVal strCmd As String) As Long
```

Parameters

daqdarwin	Specify the device descriptor.
strCmd	Specify the command message to be sent.

Description

Sends the specified command message and terminator and receives the response.

- This function adds a terminator to the command message at the time of transmission. Therefore, do not include the terminator in the command message.
- This function does not support simultaneous transmission of multiple commands or command messages that include the terminator.
- Like the data output request command of the talker function, does not support commands that do not send responses.

Return value

Returns an error number.

Error:

Not descriptor	No device descriptor.
----------------	-----------------------

Reference

CDAQDARWIN::runCommand

sendLineDARWIN

Syntax

```
int sendLineDARWIN(DAQDARWIN daqdarwin, const char * strLine);
```

Declaration

```
Public Declare Function sendLineDARWIN Lib "DAQDARWIN" (ByVal daqdarwin As Long, ByVal strLine As String) As Long
```

Parameters

daqdarwin Specify the device descriptor.
strLine Specify the string to be sent.

Description

Sends the string data specified by the parameter.

- When sending the command, the terminator is also part of the data.
- This function does not receive a response. Receive the returned data using another receive function.

Return value

Returns an error number.

Error:

Not descriptor No device descriptor.

Reference

CDAQDARWIN::sendLine

sendTriggerDARWIN

Syntax

```
int sendTriggerDARWIN(DAQDARWIN daqdarwin);
```

Declaration

```
Public Declare Function sendTriggerDARWIN Lib  
"DAQDARWIN" (ByVal daqdarwin As Long) As Long
```

Parameters

daqdarwin Specify the device descriptor.

Description

Sends a trigger command (ESC T), and receives the response.

Return value

Returns an error number.

Error:

Not descriptor No device descriptor.

Reference

CDAQDARWIN::sendTrigger

setAlarmDARWIN

Syntax

```
int setAlarmDARWIN(DAQDARWIN daqdarwin, int levelNo, int chType, int startChNo, int endChNo, int iAlarmType, int value, int relayType, int relayNo);
```

Declaration

```
Public Declare Function setAlarmDARWIN Lib "DAQDARWIN" (ByVal daqdarwin As Long, ByVal levelNo As Long, ByVal chType As Long, ByVal startChNo As Long, ByVal endChNo As Long, ByVal iAlarmType As Long, ByVal value As Long, ByVal relayType As Long, ByVal relayNo As Long) As Long
```

Parameters

daqdarwin	Specify the device descriptor.
levelNo	Specify the alarm level.
chType	Specify the channel type.
startChNo	Specify the start channel number.
endChNo	Specify the end channel number.
iAlarmType	Specify the alarm type.
value	Specify the data value of the alarm.
relayType	Specify the relay type.
relayNo	Specify the relay number.

Description

Sets the specified alarm (alarm level and alarm type) and alarm value to the channels in the specified channel range (specified by channel type, start channel number, and end channel number).

- With relay specification, when the relay number is less than or equal to 0, the relay is not specified (turned OFF).
- This function executes the SA command of the DARWIN communication function.

Return value

Returns an error number.

Error:

Not descriptor No device descriptor.

Reference

CDAQDARWIN::setAlarm

setDateTimeDARWIN

Syntax

```
int setDateTimeDARWIN(DAQDARWIN daqdarwin, DarwinDateTime *  
pDarwinDateTime);
```

Declaration

```
Public Declare Function setDateTimeDARWIN Lib  
"DAQDARWIN" (ByVal daqdarwin As Long, ByRef pDarwinDateTime As  
DarwinDateTime) As Long
```

Parameters

daqdarwin	Specify the device descriptor.
pDarwinDateTime	Specify the time information data.

Description

Sets the date and time on the device.

- In Visual C, if the parameter's time information data is set to NULL, the current date/time of the PC is used.
- This function executes the SD command of the DARWIN communication function.

Return value

Returns an error number.

Error:

Not descriptor	No device descriptor.
----------------	-----------------------

Reference

CDAQDARWIN::setDateTime

setDateTimeNowDARWIN

Syntax

```
int setDateTimeNowDARWIN(DAQDARWIN daqdarwin);
```

Declaration

```
Public Declare Function setDateTimeNowDARWIN Lib  
"DAQDARWIN" (ByVal daqdarwin As Long) As Long
```

Parameters

daqdarwin Specify the device descriptor.

Description

Sets the current date/time.

Return value

Returns an error number.

Reference

setDateTimeDARWIN

setDELTADARWIN

Syntax

```
int setDELTADARWIN(DAQDARWIN daqdarwin, int refChNo, int
chType, int startChNo, int endChNo, int spanMin, int spanMax);
```

Declaration

```
Public Declare Function setDELTADARWIN Lib "DAQDARWIN" (ByVal
daqdarwin As Long, ByVal refChNo As Long, ByVal chType As
Long, ByVal startChNo As Long, ByVal endChNo As Long, ByVal
spanMin As Long, ByVal spanMax As Long) As Long
```

Parameters

daqdarwin	Specify the device descriptor.
refChNo	Specify the channel number of the reference channel.
chType	Specify the channel type.
startChNo	Specify the start channel number.
endChNo	Specify the end channel number.
spanMin	Specify the left value of the span.
spanMax	Specify the right value of the span.

Description

Sets difference computation with respect to the specified reference channel to the channels in the specified channel range (specified by channel type, start channel number, and end channel number).

- With the span specification, if the left and right values are the same, it is considered omitted.
- This function executes the SR command of the DARWIN communication function.

Return value

Returns an error number.

Error:

Not descriptor No device descriptor.

Reference

CDAQDARWIN::setDELTA

setDIDARWIN

Syntax

```
int setDIDARWIN(DAQDARWIN daqdarwin, int iRangeDI, int chType,  
int startChNo, int endChNo, int spanMin, int spanMax, int  
scaleMin, int scaleMax, int scalePoint);
```

Declaration

```
Public Declare Function setDIDARWIN Lib "DAQDARWIN" (ByVal  
daqdarwin As Long, ByVal iRangeDI As Long, ByVal chType As  
Long, ByVal startChNo As Long, ByVal endChNo As Long, ByVal  
spanMin As Long, ByVal spanMax As Long, ByVal scaleMin As  
Long, ByVal scaleMax As Long, ByVal scalePoint As Long) As  
Long
```

Parameters

daqdarwin	Specify the device descriptor.
iRangeDI	Specify the contact range.
chType	Specify the channel type.
startChNo	Specify the start channel number.
endChNo	Specify the end channel number.
spanMin	Specify the left value of the span.
spanMax	Specify the right value of the span.
scaleMin	Specify the left value of the scale.
scaleMax	Specify the right value of the scale.
scalePoint	Specify the decimal point position for scaling.

Description

Sets the specified contact range to the channels in the specified channel range (specified by channel type, start channel number, and end channel number).

- With the span and scale specification, if the left and right values are the same, they are considered omitted.
- This function executes the SR command of the DARWIN communication function.

Return value

Returns an error number.

Error:

Not descriptor No device descriptor.

Reference

CDAQDARWIN::setDI

setMADARWIN

Syntax

```
int setMADARWIN(DAQDARWIN daqdarwin, int iRangeMA, int chType,
int startChNo, int endChNo, int spanMin, int spanMax, int
scaleMin, int scaleMax, int scalePoint);
```

Declaration

```
Public Declare Function setMADARWIN Lib "DAQDARWIN" (ByVal
daqdarwin As Long, ByVal iRangeMA As Long, ByVal chType As
Long, ByVal startChNo As Long, ByVal endChNo As Long, ByVal
spanMin As Long, ByVal spanMax As Long, ByVal scaleMin As
Long, ByVal scaleMax As Long, ByVal scalePoint As Long) As
Long
```

Parameters

daqdarwin	Specify the device descriptor.
iRangeMA	Specify the DC current range.
chType	Specify the channel type.
startChNo	Specify the start channel number.
endChNo	Specify the end channel number.
spanMin	Specify the left value of the span.
spanMax	Specify the right value of the span.
scaleMin	Specify the left value of the scale.
scaleMax	Specify the right value of the scale.
scalePoint	Specify the decimal point position for scaling.

Description

Sets the specified DC current range to the channels in the specified channel range (specified by channel type, start channel number, and end channel number).

- With the span and scale specification, if the left and right values are the same, they are considered omitted.
- This function executes the SR command of the DARWIN communication function.

Return value

Returns an error number.

Error:

Not descriptor No device descriptor.

Reference

CDAQDARWIN::setMA

setPOWERDARWIN

Syntax

```
int setPOWERDARWIN(DAQDARWIN daqdarwin, int iRangePOWER, int chType, int chNo, int iItem, int iWire, int spanMin, int spanMax, int scaleMin, int scaleMax, int scalePoint);
```

Declaration

```
Public Declare Function setPOWERDARWIN Lib "DAQDARWIN" (ByVal daqdarwin As Long, ByVal iRangePOWER As Long, ByVal chType As Long, ByVal chNo As Long, ByVal iItem As Long, ByVal iWire As Long, ByVal spanMin As Long, ByVal spanMax As Long, ByVal scaleMin As Long, ByVal scaleMax As Long, ByVal scalePoint As Long) As Long
```

Parameters

daqdarwin	Specify the device descriptor.
iRangePOWER	Specify the power monitor range.
chType	Specify the channel type.
startChNo	Specify the channel number.
iltem	Specify the power measurement parameter.
iWire	Specify the power connection method.
spanMin	Specify the left value of the span.
spanMax	Specify the right value of the span.
scaleMin	Specify the left value of the scale.
scaleMax	Specify the right value of the scale.
scalePoint	Specify the decimal point position for scaling.

Description

Sets the specified power range on the specified channel (specified under channel type and channel number).

- With the span and scale specification, if the left and right values are the same, they are considered omitted.
- This function executes the SR command of the DARWIN communication function.

Return value

Returns an error number.

Error:

Not descriptor No device descriptor.

Reference

CDAQDARWIN::setPOWER

setPULSEDARWIN

Syntax

```
int setPULSEDARWIN(DAQDARWIN daqdarwin, int iRangePULSE, int
chType, int startChNo, int endChNo, int spanMin, int spanMax,
int scaleMin, int scaleMax, int scalePoint, int bFilter);
```

Declaration

```
Public Declare Function setPULSEDARWIN Lib "DAQDARWIN" (ByVal
daqdarwin As Long, ByVal iRangePULSE As Long, ByVal chType As
Long, ByVal startChNo As Long, ByVal endChNo As Long, ByVal
spanMin As Long, ByVal spanMax As Long, ByVal scaleMin As
Long, ByVal scaleMax As Long, ByVal scalePoint As Long, ByVal
bFilter As Long) As Long
```

Parameters

daqdarwin	Specify the device descriptor.
iRangePULSE	Specify the pulse range.
chType	Specify the channel type.
startChNo	Specify the start channel number.
endChNo	Specify the end channel number.
spanMin	Specify the left value of the span.
spanMax	Specify the right value of the span.
scaleMin	Specify the left value of the scale.
scaleMax	Specify the right value of the scale.
scalePoint	Specify the decimal point position for scaling.
bFilter	Specify a filter using a boolean value.

Description

Sets the specified pulse range to the channels in the specified channel range (specified by channel type, start channel number, and end channel number).

- With the span and scale specification, if the left and right values are the same, they are considered omitted.
- This function executes the SR command of the DARWIN communication function.

Return value

Returns an error number.

Error:

Not descriptor No device descriptor.

Reference

CDAQDARWIN::setPULSE

setRRJCDARWIN

Syntax

```
int setRRJCDARWIN(DAQDARWIN daqdarwin, int refChNo, int  
chType, int startChNo, int endChNo, int spanMin, int spanMax);
```

Declaration

```
Public Declare Function setRRJCDARWIN Lib "DAQDARWIN" (ByVal  
daqdarwin As Long, ByVal refChNo As Long, ByVal chType As  
Long, ByVal startChNo As Long, ByVal endChNo As Long, ByVal  
spanMin As Long, ByVal spanMax As Long) As Long
```

Parameters

daqdarwin	Specify the device descriptor.
refChNo	Specify the channel number of the reference channel.
chType	Specify the channel type.
startChNo	Specify the start channel number.
endChNo	Specify the end channel number.
spanMin	Specify the left value of the span.
spanMax	Specify the right value of the span.

Description

Sets the remote RJC with respect to the specified reference channel to the channels in the specified channel range (specified by channel type, start channel number, and end channel number).

- With the span specification, if the left and right values are the same, it is considered omitted.
- This function executes the SR command of the DARWIN communication function.

Return value

Returns an error number.

Error:

Not descriptor No device descriptor.

Reference

CDAQDARWIN::setRRJC

setRTDDARWIN

Syntax

```
int setRTDDARWIN(DAQDARWIN daqdarwin, int iRangeRTD, int
chType, int startChNo, int endChNo, int spanMin, int spanMax,
int scaleMin, int scaleMax, int scalePoint);
```

Declaration

```
Public Declare Function setRTDDARWIN Lib "DAQDARWIN" (ByVal
daqdarwin As Long, ByVal iRangeRTD As Long, ByVal chType As
Long, ByVal startChNo As Long, ByVal endChNo As Long, ByVal
spanMin As Long, ByVal spanMax As Long, ByVal scaleMin As
Long, ByVal scaleMax As Long, ByVal scalePoint As Long) As
Long
```

Parameters

daqdarwin	Specify the device descriptor.
iRangeRTD	Specify the range type of the RTD input.
chType	Specify the channel type.
startChNo	Specify the start channel number.
endChNo	Specify the end channel number.
spanMin	Specify the left value of the span.
spanMax	Specify the right value of the span.
scaleMin	Specify the left value of the scale.
scaleMax	Specify the right value of the scale.
scalePoint	Specify the decimal point position for scaling.

Description

Sets the specified RTD range to the channels in the specified channel range (specified by channel type, start channel number, and end channel number).

- With the span and scale specification, if the left and right values are the same, they are considered omitted.
- This function executes the SR command of the DARWIN communication function.

Return value

Returns an error number.

Error:

Not descriptor No device descriptor.

Reference

CDAQDARWIN::setRTD

setScallingUnitDARWIN

Syntax

```
int setScallingUnitDARWIN(DAQDARWIN daqdarwin, const char *  
strUnit, int chType, int startChNo, int endChNo);
```

Declaration

```
Public Declare Function setScallingUnitDARWIN Lib  
"DAQDARWIN"(ByVal daqdarwin As Long, ByVal strUnit As String,  
ByVal chType As Long, ByVal startChNo As Long, ByVal endChNo  
As Long) As Long l11
```

Parameters

daqdarwin	Specify the device descriptor.
strUnit	Specify the unit name using a string.
chType	Specify the channel type.
startChNo	Specify the start channel number.
endChNo	Specify the end channel number.

Description

Sets the specified unit to the channels in the specified channel range (specified by channel type, start channel number, and end channel number).

This function executes the SN command of the DARWIN communication function.

Return value

Returns an error number.

Error:

Not descriptor No device descriptor.

Reference

CDAQDARWIN::setScallingUnit

setSKIPDARWIN

Syntax

```
int setSKIPDARWIN(DAQDARWIN daqdarwin, int chType, int startChNo, int endChNo);
```

Declaration

```
Public Declare Function setSKIPDARWIN Lib "DAQDARWIN" (ByVal daqdarwin As Long, ByVal chType As Long, ByVal startChNo As Long, ByVal endChNo As Long) As Long
```

Parameters

daqdarwin	Specify the device descriptor.
chType	Specify the channel type.
startChNo	Specify the start channel number.
endChNo	Specify the end channel number.

Description

Sets the channels in the specified channel range (specified by channel type, start channel number, and end channel number) to SKIP (not used).

This function executes the SR command of the DARWIN communication function.

Return value

Returns an error number.

Error:

Not descriptor No device descriptor.

Reference

CDAQDARWIN::setSKIP

setSTRAIN DARWIN

Syntax

```
int setSTRAIN DARWIN(DAQDARWIN daqdarwin, int iRangeSTRAIN, int chType, int startChNo, int endChNo, int spanMin, int spanMax, int scaleMin, int scaleMax, int scalePoint);
```

Declaration

```
Public Declare Function setSTRAIN DARWIN Lib "DAQDARWIN" (ByVal daqdarwin As Long, ByVal iRangeSTRAIN As Long, ByVal chType As Long, ByVal startChNo As Long, ByVal endChNo As Long, ByVal spanMin As Long, ByVal spanMax As Long, ByVal scaleMin As Long, ByVal scaleMax As Long, ByVal scalePoint As Long) As Long
```

Parameters

daqdarwin	Specify the device descriptor.
iRangeSTRAIN	Specify the strain input range.
chType	Specify the channel type.
startChNo	Specify the start channel number.
endChNo	Specify the end channel number.
spanMin	Specify the left value of the span.
spanMax	Specify the right value of the span.
scaleMin	Specify the left value of the scale.
scaleMax	Specify the right value of the scale.
scalePoint	Specify the decimal point position for scaling.

Description

Sets the specified strain range on the channels in the specified channel range (specified by channel type, start channel number, and end channel number).
With the span and scale specification, if the left and right values are the same, they are considered omitted.

This function executes the SR command of the DARWIN communication function.

Return value

Returns an error number.
Error:
Not descriptor No device descriptor.

Reference

CDAQDARWIN::setSTRAIN

setTCDARWIN

Syntax

```
int setTCDARWIN(DAQDARWIN daqdarwin, int iRangeTC, int chType,
int startChNo, int endChNo, int spanMin, int spanMax, int
scaleMin, int scaleMax, int scalePoint);
```

Declaration

```
Public Declare Function setTCDARWIN Lib "DAQDARWIN" (ByVal
daqdarwin As Long, ByVal iRangeTC As Long, ByVal chType As
Long, ByVal startChNo As Long, ByVal endChNo As Long, ByVal
spanMin As Long, ByVal spanMax As Long, ByVal scaleMin As
Long, ByVal scaleMax As Long, ByVal scalePoint As Long) As
Long
```

Parameters

daqdarwin	Specify the device descriptor.
iRangeTC	Specify the range type of the thermocouple input.
chType	Specify the channel type.
startChNo	Specify the start channel number.
endChNo	Specify the end channel number.
spanMin	Specify the left value of the span.
spanMax	Specify the right value of the span.
scaleMin	Specify the left value of the scale.
scaleMax	Specify the right value of the scale.
scalePoint	Specify the decimal point position for scaling.

Description

Sets the specified thermocouple range to the channels in the specified channel range (specified by channel type, start channel number, and end channel number).

- With the span and scale specification, if the left and right values are the same, they are considered omitted.
- This function executes the SR command of the DARWIN communication function.

Return value

Returns an error number.

Error:

Not descriptor No device descriptor.

Reference

CDAQDARWIN::setTC

setTimeoutDARWIN

Syntax

```
int setTimeoutDARWIN(DAQDARWIN daqdarwin, int seconds);
```

Declaration

```
Public Declare Function setTimeoutDARWIN Lib "DAQDARWIN" (ByVal daqdarwin As Long, ByVal seconds As Long) As Long
```

Parameters

daqdarwin Specify the device descriptor.
seconds Specify the communication timeout value in units of seconds.

Description

Sets a timeout for the communication with the device.

- If a negative value is specified, the timeout is discarded.
- Its use is not recommended.

Return value

Returns an error number.

Error:

Not descriptor No device descriptor.

Reference

CDAQDARWIN::setTimeout

setVOLTDARWIN

Syntax

```
int setVOLTDARWIN(DAQDARWIN daqdarwin, int iRangeVOLT, int
chType, int startChNo, int endChNo, int spanMin, int spanMax,
int scaleMin, int scaleMax, int scalePoint);
```

Declaration

```
Public Declare Function setVOLTDARWIN Lib "DAQDARWIN" (ByVal
daqdarwin As Long, ByVal iRangeVOLT As Long, ByVal chType As
Long, ByVal startChNo As Long, ByVal endChNo As Long, ByVal
spanMin As Long, ByVal spanMax As Long, ByVal scaleMin As
Long, ByVal scaleMax As Long, ByVal scalePoint As Long) As
Long
```

Parameters

daqdarwin	Specify the device descriptor.
iRangeVOLT	Specify the range type of the DC voltage input.
chType	Specify the channel type.
startChNo	Specify the start channel number.
endChNo	Specify the end channel number.
spanMin	Specify the left value of the span.
spanMax	Specify the right value of the span.
scaleMin	Specify the left value of the scale.
scaleMax	Specify the right value of the scale.
scalePoint	Specify the decimal point position for scaling.

Description

Sets the specified DC voltage range on the channels in the specified channel range (specified by channel type, start channel number, and end channel number).

- With the span and scale specification, if the left and right values are the same, they are considered omitted.
- This function executes the SR command of the DARWIN communication function.

Return value

Returns an error number.

Error:

Not descriptor No device descriptor.

Reference

CDAQDARWIN::setVOLT

talkCalibrationDataDARWIN

Syntax

```
int talkCalibrationDataDARWIN(DAQDARWIN daqdarwin, int startChType, int startChNo, int endChType, int endChNo);
```

Declaration

```
Public Declare Function talkCalibrationDataDARWIN Lib "DAQDARWIN" (ByVal daqdarwin As Long, ByVal startChType As Long, ByVal startChNo As Long, ByVal endChType As Long, ByVal endChNo As Long) As Long
```

Parameters

daqdarwin	Specify the device descriptor.
startChType	Specify the start channel type.
startChNo	Specify the start channel number.
endChType	Specify the end channel type.
endChNo	Specify the end channel number.

Description

Executes the declaration for retrieving setup data of A/D calibration mode from the start channel (start channel type and start channel number) to the end channel (end channel type and end channel number).

- The operation mode must be switched to A/D calibration mode in advance.
- This function executes the TS and LF commands of the DARWIN communication function.
- After executing this function, use the getSetDataByLineDARWIN function to retrieve the data by lines.

Return value

Returns an error number.

Error:

Not descriptor No device descriptor.

Reference

CDAQDARWIN::talkCalibrationData

talkChInfoDARWIN

Syntax

```
int talkChInfoDARWIN(DAQDARWIN daqdarwin, int startChType, int
startChNo, int endChType, int endChNo);
```

Declaration

```
Public Declare Function talkChInfoDARWIN Lib "DAQDARWIN" (ByVal
daqdarwin As Long, ByVal startChType As Long, ByVal startChNo
As Long, ByVal endChType As Long, ByVal endChNo As Long) As
Long
```

Parameters

daqdarwin	Specify the device descriptor.
startChType	Specify the start channel type.
startChNo	Specify the start channel number.
endChType	Specify the end channel type.
endChNo	Specify the end channel number.

Description

Executes the declaration for retrieving channel information data from the start channel (start channel type and start channel number) to the end channel (end channel type and end channel number).

- This function executes the TS and LF commands of the DARWIN communication function.
- After executing this function, use the getChInfoDARWIN function to retrieve the data for each channel.

Return value

Returns an error number.

Error:

Not descriptor No device descriptor.

Reference

CDAQDARWIN::talkChInfo

talkDataByASCIIDARWIN

Syntax

```
int talkDataByASCIIDARWIN(DAQDARWIN daqdarwin, int startChType, int startChNo, int endChType, int endChNo, DarwinDateTime * pDarwinDateTime);
```

Declaration

```
Public Declare Function talkDataByASCIIDARWIN Lib "DAQDARWIN" (ByVal daqdarwin As Long, ByVal startChType As Long, ByVal startChNo As Long, ByVal endChType As Long, ByVal endChNo As Long, ByRef pDarwinDateTime As DarwinDateTime) As Long
```

Parameters

daqdarwin	Specify the device descriptor.
startChType	Specify the start channel type.
startChNo	Specify the start channel number.
endChType	Specify the end channel type.
endChNo	Specify the end channel number.
pDarwinDateTime	Specify the destination where the time information data is to be returned.

Description

Executes the declaration for retrieving measured data in ASCII format from the start channel (start channel type and start channel number) to the end channel (end channel type and end channel number).

- Stores time information of the measured data if the return destination is specified.
- Specify measurement channels and computation channels separately.
- This function executes the TS and FM commands of the DARWIN communication function.
- After executing this function, use the getChDataByASCIIDARWIN function to retrieve the data for each channel.

Return value

Returns an error number.

Error:

Not descriptor	No device descriptor.
----------------	-----------------------

Reference

```
CDAQDARWIN::talkDataByASCII  
CDAQDARWINDateTime::getDarwinDateTime
```

talkDataByBinaryDARWIN

Syntax

```
int talkDataByBinaryDARWIN(DAQDARWIN daqdarwin, int
startChType, int startChNo, int endChType, int endChNo,
DarwinDateTime * pDarwinDateTime);
```

Declaration

```
Public Declare Function talkDataByBinaryDARWIN Lib
"DAQDARWIN" (ByVal daqdarwin As Long, ByVal startChType As
Long, ByVal startChNo As Long, ByVal endChType As Long, ByVal
endChNo As Long, ByRef pDarwinDateTime As DarwinDateTime) As
Long
```

Parameters

daqdarwin	Specify the device descriptor.
startChType	Specify the start channel type.
startChNo	Specify the start channel number.
endChType	Specify the end channel type.
endChNo	Specify the end channel number.
pDarwinDateTime	Specify the destination where the time information data is to be returned.

Description

Executes the declaration for retrieving measured data in binary format from the start channel (start channel type and start channel number) to the end channel (end channel type and end channel number).

- Stores time information data of the measured data if the return destination is specified.
- Specify measurement channels and computation channels separately.
- MSB is specified for the byte output order.
- This function executes the TS and FM commands of the DARWIN communication function.
- After executing this function, use the getChDataByBinaryDARWIN function to retrieve the data for each channel.

Return value

Returns an error number.

Error:

Not descriptor	No device descriptor.
----------------	-----------------------

Reference

```
CDAQDARWIN::talkDataByBinary
CDAQDARWINDateTime::getDarwinDateTime
```

talkOperationDataDARWIN

Syntax

```
int talkOperationDataDARWIN(DAQDARWIN daqdarwin, int  
startChType, int startChNo, int endChType, int endChNo);
```

Declaration

```
Public Declare Function talkOperationDataDARWIN Lib  
"DAQDARWIN"(ByVal daqdarwin As Long, ByVal startChType As  
Long, ByVal startChNo As Long, ByVal endChType As Long, ByVal  
endChNo As Long) As Long
```

Parameters

daqdarwin	Specify the device descriptor.
startChType	Specify the start channel type.
startChNo	Specify the start channel number.
endChType	Specify the end channel type.
endChNo	Specify the end channel number.

Description

Executes the declaration for retrieving setup data of the operation mode from the start channel (start channel type and start channel number) to the end channel (end channel type and end channel number).

- This function executes the TS and LF commands of the DARWIN communication function.
- After executing this function, use the getSetDataByLineDARWIN function to retrieve the data by lines.

Return value

Returns an error number.

Error:

Not descriptor No device descriptor.

Reference

CDAQDARWIN::talkOperationData

talkSetupDataDARWIN

Syntax

```
int talkSetupDataDARWIN(DAQDARWIN daqdarwin, int startChType,
int startChNo, int endChType, int endChNo);
```

Declaration

```
Public Declare Function talkSetupDataDARWIN Lib
"DAQDARWIN" (ByVal daqdarwin As Long, ByVal startChType As
Long, ByVal startChNo As Long, ByVal endChType As Long, ByVal
endChNo As Long) As Long
```

Parameters

daqdarwin	Specify the device descriptor.
startChType	Specify the start channel type.
startChNo	Specify the start channel number.
endChType	Specify the end channel type.
endChNo	Specify the end channel number.

Description

Executes the declaration for retrieving setup data of the setup mode from the start channel (start channel type and start channel number) to the end channel (end channel type and end channel number).

- This function executes the TS and LF commands of the DARWIN communication function.
- After executing this function, use the getSetDataByLineDARWIN function to retrieve the data by lines.

Return value

Returns an error number.

Error:

Not descriptor No device descriptor.

Reference

CDAQDARWIN::talkSetupData

toAlarmNameDARWIN

Syntax

```
int toAlarmNameDARWIN(int iAlarmType, char * strAlarm, int lenAlarm);
```

Declaration

```
Public Declare Function toAlarmNameDARWIN Lib "DAQDARWIN" (ByVal iAlarmType As Long, ByVal strAlarm As String, ByVal lenAlarm As Long) As Long
```

Parameters

iAlarmType	Specify the alarm type.
strAlarm	Specify the field where the string is to be stored.
lenAlarm	Specify the byte size of the field where the string is to be stored.

Description

Stores the string corresponding to the specified alarm type to the specified field.

- The string stored in the field includes the terminator (NULL).
- The return value is the length of the actual string. The return value does not include the terminator.

Return value

Returns the length of the string.

Reference

getAlarmNameDARWIN

toDoubleValueDARWIN

Syntax

```
double toDoubleValueDARWIN(int dataValue, int point);
```

Declaration

```
Public Declare Function toDoubleValueDARWIN Lib  
"DAQDARWIN" (ByVal dataValue As Long, ByVal point As Long) As  
Double
```

Parameters

dataValue	Specify the data value.
point	Specify the decimal point position.

Description

Generates the measured value from the specified data value and decimal point position.

Return value

Returns the measured value as a double-precision floating number.

Reference

CDAQDARWINDataInfo::toDoubleValue

toErrorMessageDARWIN

Syntax

```
int toErrorMessageDARWIN(int errorCode, char * errStr, int  
errLen);
```

Declaration

```
Public Declare Function toErrorMessageDARWIN Lib  
"DAQDARWIN"(ByVal errorCode As Long, ByVal errStr As String,  
ByVal errLen As Long) As Long
```

Parameters

errorCode	Specify the error number.
errStr	Specify the field where the string is to be stored.
errLen	Specify the byte size of the field where the string is to be stored.

Description

Stores the error message string corresponding to the error number to the specified field.

- The string stored to the field includes the terminator (NULL).
- The return value is the length of the actual string. The return value does not include the terminator.

Return value

Returns the length of the string.

Reference

getErrorMessageDARWIN

toStringValueDARWIN

Syntax

```
int toStringValueDARWIN(int dataValue, int point, char *  
strValue, int lenValue);
```

Declaration

```
Public Declare Function toStringValueDARWIN Lib  
"DAQDARWIN" (ByVal dataValue As Long, ByVal point As Long,  
ByVal strValue As String, ByVal lenValue As Long) As Long
```

Parameters

dataValue	Specify the data value.
point	Specify the decimal point position.
strValue	Specify the field where the string is to be stored.
lenValue	Specify the byte size of the field where the string is to be stored.

Description

Generates the measured value from the specified data value and decimal point position.

- Converts the generated measured value into a string and stores it in the specified field.
- The string stored to the field includes the terminator (NULL).
- The return value is the length of the actual string. The return value does not include the terminator.

Return value

Returns the length of the string.

Reference

CDAQDARWINDataInfo::toStringValue

transModeDARWIN

Syntax

```
int transModeDARWIN(DAQDARWIN daqdarwin, int iMode);
```

Declaration

```
Public Declare Function transModeDARWIN Lib "DAQDARWIN" (ByVal daqdarwin As Long, ByVal iMode As Long) As Long
```

Parameters

daqdarwin	Specify the device descriptor.
iMode	Specify the mode.

Description

Switches to the specified mode.

- This function executes the DS command of the DARWIN communication function.

Return value

Returns an error number.

Error:

Not descriptor	No device descriptor.
----------------	-----------------------

Reference

CDAQDARWIN::transMode

11.1 Overview of the DARWIN Constants

The types of constants provided are listed below. The constants are common to Visual C++, Visual C, and Visual Basic.

Type	Description	Page
Communication constants	Communication port number of the DARWIN	11-2
Enumeration constants	Number of subunits, etc.	11-2
Maximum values	Maximum length of the channel name string, etc.	11-2
String	Terminator string	11-2
Boolean value	Valid (ON) setting or Invalid (OFF) setting	11-2
Flag statuses	Identifies the last data set when data retrieved	11-3
Data status values	Status of the measured data	11-3
Alarm types	Upper-limit alarm, etc.	11-3
System control types	System control operation	11-4
Channel types/relay types	Channel or relay types	11-4
Operation modes	Operation, setup, and A/D calibration	11-4
Talker function types	Talker supporting output data	11-5
Status byte	Various statuses	11-5
Establish setup mode	Abort, establish	11-5
Unit number	Extension model, standalone model	11-5
Computation	Calculation start, stop, clear, etc.	11-6
Report execution type	Report start/stop	11-6
Report type	Hourly, daily, monthly, status	11-6
Report status	All invalid, newest, valid	11-6
DC voltage range types	20 mV, etc.	11-6
TC range types	Type R, etc.	11-7
RTD range types	Pt100: 1 mA, etc.	11-7
Contact input range types	Voltage level or contact input	11-7
Strain input ranges	2 k, 20 k, 200 k	11-8
Pulse ranges	GATE,RATE	11-8
Power monitor ranges	25 V 0.5 A, 25 V 5 A, 250 V 0.5 A, 250 V 5 A	11-8
DC current ranges	20 mA	11-8
Power connection methods	Single-phase two-wire, etc.	11-9
Power measurement parameters	Effective current, etc.	11-10

11.2 DARWIN Constants

This section describes the mnemonic and the meaning of the constants. For DARWIN terminology, see appendix 2.

Communication Constants

Mnemonic	Description
DAQDARWIN_COMMPORT	Communication port number of DARWIN.

Enumeration Constants

Sets the number of items such as the number of modules or units.

Mnemonic	Description
DAQDARWIN_NUMCHANNEL	The number of channels.
DAQDARWIN_NUMALARM	The number of alarms.
DAQDARWIN_NUMUNIT	The number of subunits.
DAQDARWIN_NUMSLOT	The number of slots per subunit.
DAQDARWIN_NUMTERM	The number of terminals per slot (module).

Maximum Values

Mnemonic	Description
DAQDARWIN_MAXCHNAMELEN	Maximum length of the channel name string.
DAQDARWIN_MAXCHRANGLLEN	Maximum length of the channel range name string.
DAQDARWIN_MAXUNITLEN	Maximum length of the unit name string.
DAQDARWIN_MAXMODULELEN	Maximum length of the module name string.
DAQDARWIN_MAXRELAYLEN	Maximum length of the relay name string. Same as the maximum length of the channel name string. Relay refers to the output relay of the alarm output module or the DI/DO module.
DAQDARWIN_MAXALARMLEN	Maximum length of the alarm type string.
DAQDARWIN_MAXDECIMALPOINT	Maximum value of the decimal point position.

The maximum length of the string does not include the terminator (NULL).

String

Mnemonic	Description
DAQDARWIN_TERMINATE	Terminator string.

Boolean Value (valid/invalid)

Mnemonic	Description
DAQDARWIN_VALID_OFF	Invalid (OFF) value.
DAQDARWIN_VALID_ON	Valid (ON) value.

Flag Status

Mnemonic	Description
DAQDARWIN_FLAG_OFF	All OFF.
DAQDARWIN_FLAG_ENDDATA	The data line retrieved using ASCII codes or in units of lines is at the last data set.

Can be synthesized using logical OR operators.

Data Status Values

Mnemonic	Description
DAQDARWIN_UNKNWON	The data status is not set.
DAQDARWIN_DATA_NORMAL	Normal.
DAQDARWIN_DATA_DIFFINPUT	Difference computation between channels being performed.
DAQDARWIN_DATA_PLUSOVER	Positive overrange.
DAQDARWIN_DATA_MINUSOVER	Negative overrange.
DAQDARWIN_DATA_SKIP	SKIP (not used).
DAQDARWIN_DATA_ILLEGAL	Illegal data status.
DAQDARWIN_DATA_ABNORMAL	Abnormal data status.
DAQDARWIN_DATA_NODATA	No data status.
DAQDARWIN_DATA_READER	Status when loading instantaneous value data from the communication port and communicating.

The status when using the communication port for loading instantaneous value data and communicating is the channel status in which channel information data is retrieved.

Alarm Type

◇ indicates a space.

Mnemonic	Description	String
DAQDARWIN_ALARM_NONE	No alarm (alarm OFF)	◇◇
DAQDARWIN_ALARM_UPPER	Upper limit alarm	H◇
DAQDARWIN_ALARM_LOWER	Lower limit alarm	L◇
DAQDARWIN_ALARM_UPDIFF	Difference upper limit alarm	dH
DAQDARWIN_ALARM_LOWDIFF	Difference lower limit alarm	dL
DAQDARWIN_ALARM_INCRATE	High limit on rate-of-change alarm	RH
DAQDARWIN_ALARM_DECRATE	Low limit on rate-of-change alarm	RL

System Control Types

Used when specifying the system control operation.

Mnemonic	Description
DAQDARWIN_SYSTEM_RECONSTRUCT	System reconfiguration
DAQDARWIN_SYSTEM_INITOPE	RAM clear (Initialize the operation mode setup parameter)
DAQDARWIN_SYSTEM_RESEALARM	Alarm reset

Channel/Relay types

Type values for channels, relays, communication input, and computation constants. Can be used to specify the channel or relay. Definitions consisting of a single character are available also to simplify the statements.

Mnemonic	Char	Description	Channel	Relay
DAQDARWIN_CHTYPE_MAINUNIT	I	Value representing the main unit expandable model	-	Yes
DAQDARWIN_CHTYPE_STANDALONE		Value representing the model of the standalone unit. Same as subunit number of 0.	Yes	Yes
DAQDARWIN_CHTYPE_MATHTYPE	A	Value representing the computation channel.	Yes	-
DAQDARWIN_CHTYPE_SWITCH	S	Value representing the internal switch.	-	Yes
DAQDARWIN_CHTYPE_COMMDATA	C	Value representing the communication input.	-	-
DAQDARWIN_CHTYPE_CONSTANT	K	Value representing the computation constant.	-	-
DAQDARWIN_CHTYPE_REPORT	R	Value representing the report.	-	-

Yes: Channel/Relay of the type exists.
 - : Channel/Relay of the type does not exist.

Note

The subunit number used to identify the subunit that is connected to the expandable model is also a type number. The subunit number is an integer value between 0 and 5. See appendix 2.

Operation Modes

Mnemonic	Description
DAQDARWIN_MODE_OPE	Operation mode
DAQDARWIN_MODE_SETUP	Setup mode
DAQDARWIN_MODE_CALIB	A/D calibration mode

Talker Function Types

Mnemonic	Description
DAQDARWIN_TALK_MEASUREDDATA	Outputs measured and computed data
DAQDARWIN_TALK_OPEDATA	Outputs the setup data of operation mode.
DAQDARWIN_TALK_CHINFODATA	Outputs the channel information data
DAQDARWIN_TALK_SYSINFODATA	Outputs the system configuration data
DAQDARWIN_TALK_CALIBDATA	Outputs the calibration data (setup data of calibration mode)
DAQDARWIN_TALK_SETUPDATA	Outputs the setup data of setup mode
DAQDARWIN_TALK_REPORTDATA	Outputs the report status

Status Byte Value

Can be synthesized using logical OR operators.

Mnemonic	Description
DAQDARWIN_STATUS_OFF	Value when all status bytes are invalid
DAQDARWIN_STATUS_ADCONV	A/D conversion complete
DAQDARWIN_STATUS_SYNTAX	Command syntax error
DAQDARWIN_STATUS_TIMER	Internal timer start/report creation
DAQDARWIN_STATUS_MEDIA	Access to medium (DC100)
DAQDARWIN_STATUS_RELEASE	Measurement dropout during computation
DAQDARWIN_STATUS_ALL	Mask value that enables all status bytes
DAQDARWIN_STATUS_SRQ	SRQ

For details on the meaning of the status byte value, see the communication interface user's manual for the DARWIN instrument.

Establish setup mode

Mnemonic	Description
DAQDARWIN_SETUP_ABORT	Abort
DAQDARWIN_SETUP_STORE	Establish

Unit Number

Mnemonic	Description
DAQDARWIN_UNITNO_MAINUNIT	Main unit of the expandable model
DAQDARWIN_UNITNO_STANDALONE	Standalone model unit

The subunit number is numerical. See Channel/Relay types.

Computation

Mnemonic	Description
DAQDARWIN_COMPUTE_START	Computation start
DAQDARWIN_COMPUTE_STOP	Computation stop
DAQDARWIN_COMPUTE_RESTART	After clearing computation data, restart
DAQDARWIN_COMPUTE_CLEAR	Clears computation data
DAQDARWIN_COMPUTE_RELEASE	Clears status display of measurement dropouts

Report execution type

Mnemonic	Description
DAQDARWIN_REPORT_RUN_START	Report start
DAQDARWIN_REPORT_RUN_STOP	Report stop

Report type

Mnemonic	Description
DAQDARWIN_REPORT_HOURLY	Hourly
DAQDARWIN_REPORT_DAILY	Daily
DAQDARWIN_REPORT_MONTHLY	Montly
DAQDARWIN_REPORT_STATUS	Status

Report status

Can be synthesized using logical OR operators.

Mnemonic	Description
DAQDARWIN_REPSTATUS_NONE	All invalid
DAQDARWIN_REPSTATUS_HOURLY_NEW	Newest hourly
DAQDARWIN_REPSTATUS_HOURLY_VALID	Valid hourly
DAQDARWIN_REPSTATUS_DAILY_NEW	Newest daily
DAQDARWIN_REPSTATUS_DAILY_VALID	Valid daily
DAQDARWIN_REPSTATUS_MONTHLY_NEW	Newest monthly
DAQDARWIN_REPSTATUS_MONTHLY_VALID	Valid monthly

DC Voltage Range Types

Mnemonic	Description	Setting range
DAQDARWIN_RANGE_VOLT_20MV	20 mV	-20.000-20.000 mV
DAQDARWIN_RANGE_VOLT_60MV	60 mV	-60.00-60.00 mV
DAQDARWIN_RANGE_VOLT_200MV	200 mV	-200.00-200.00 mV
DAQDARWIN_RANGE_VOLT_2V	2 V	-2.0000-2.0000 V
DAQDARWIN_RANGE_VOLT_6V	6 V	-6.000-6.000 V
DAQDARWIN_RANGE_VOLT_20V	20 V	-20.000-20.000 V
DAQDARWIN_RANGE_VOLT_50V	50 V	-50.00-50.00 V

TC Range

Mnemonic	Description	Setting range
DAQDARWIN_RANGE_TC_R	R	0.0 to 1760.0°C
DAQDARWIN_RANGE_TC_S	S	0.0 to 1760.0°C
DAQDARWIN_RANGE_TC_B	B	0.0 to 1820.0°C
DAQDARWIN_RANGE_TC_K	K	-200.0 to 1370.0°C
DAQDARWIN_RANGE_TC_E	E	-200.0 to 800.0°C
DAQDARWIN_RANGE_TC_J	J	-200.0 to 1100.0°C
DAQDARWIN_RANGE_TC_T	T	-200.0 to 400.0°C
DAQDARWIN_RANGE_TC_N	N	0.0 to 1300.0°C
DAQDARWIN_RANGE_TC_W	W	0.0 to 2315.0°C
DAQDARWIN_RANGE_TC_L	L	-200.0 to 900.0°C
DAQDARWIN_RANGE_TC_U	U	-200.0 to 400.0°C
DAQDARWIN_RANGE_TC_KP	KpAu7Fe	0.0 to 300.0 K

RTD Range

Mnemonic	Description	Setting range
DAQDARWIN_RANGE_RTD_1MAPT	Pt100:1mA	-200.0 to 600.0°C
DAQDARWIN_RANGE_RTD_2MAPT	Pt100:2mA	-200.0 to 250.0°C
DAQDARWIN_RANGE_RTD_1MAJPT	JPt100:1mA	-200.0 to 550.0°C
DAQDARWIN_RANGE_RTD_2MAJPT	JPt100:2mA	-200.0 to 250.0°C
DAQDARWIN_RANGE_RTD_2MAPT50	Pt50:2mA	-200.0 to 550.0°C
DAQDARWIN_RANGE_RTD_1MAPTH	Pt100:1mA-H	-140.00 to 150.00°C
DAQDARWIN_RANGE_RTD_2MAPTH	Pt100:2mA-H	-70.00 to 70.00°C
DAQDARWIN_RANGE_RTD_1MAJPTH	JPt100:1mA-H	-140.00 to 150.00°C
DAQDARWIN_RANGE_RTD_2MAJPTH	JPt100:2mA-H	-70.00 to 70.00°C
DAQDARWIN_RANGE_RTD_1MANIS	Ni100:1mA-S	-200.0 to 250.0°C
DAQDARWIN_RANGE_RTD_1MANID	Ni100:1mA-D	-60.0 to 180.0°C
DAQDARWIN_RANGE_RTD_1MANI120	Ni120:1mA	-70.0 to 200.0°C
DAQDARWIN_RANGE_RTD_CU10GE	Cu10:GE	-200.0 to 300.0°C
DAQDARWIN_RANGE_RTD_CU10LN	Cu10:L&N	-200.0 to 300.0°C
DAQDARWIN_RANGE_RTD_CU10WEED	Cu10:WEED	-200.0 to 300.0°C
DAQDARWIN_RANGE_RTD_CU10BAILEY	Cu10:BAILEY	-200.0 to 300.0°C
DAQDARWIN_RANGE_RTD_J263B	J263*B	-0.0 to 300.0 K

Contact Input (DI) Range

Mnemonic	Description	Setting range
DAQDARWIN_RANGE_DI_LEVEL	Voltage input	Less than 2.4 V, Greater than or equal to 2.4 V
DAQDARWIN_RANGE_DI_CONTACT	Contact input	0:open, 1:close

Strain input range

Mnemonic	Description Setting range
DAQDARWIN_RANGE_STRAIN_2K	2k -2000 to 2000 μ Strain (One-Gauge Method) -1000 to 1000 μ strain (two-gauge method), -500 to 500 μ strain (4-gauge method)
DAQDARWIN_RANGE_STRAIN_20K	20k -20000 to 20000 μ strain (1-gauge method) -10000 to 10000 μ strain (2-gauge method) -5000 to 5000 μ strain (4-gauge method)
DAQDARWIN_RANGE_STRAIN_200K	200k -200000 to 200000 μ strain (1-gauge method) -100000 to 100000 μ strain (2-gauge method) -50000 to 50000 μ strain (4-gauge method)

Pulse range

Mnemonic	Description	Setting range
DAQDARWIN_RANGE_PULSE_RATE	RATE	0 - 30000
DAQDARWIN_RANGE_PULSE_GATE	GATE	0 - 30000

Power monitor range

Mnemonic	Description	Setting range
DAQDARWIN_RANGE_POWER_25V05A	25 V 0.5 A	Voltage 25 V, current 0.5 A
DAQDARWIN_RANGE_POWER_25V5A	25 V 5 A	Voltage 25 V, current 5 A
DAQDARWIN_RANGE_POWER_250V05A	250 V 0.5 A	Voltage 250 V, current 0.5 A
DAQDARWIN_RANGE_POWER_250V5A	250 V 5 A	Voltage 250 V, current 5 A

DC current range

Mnemonic	Description	Setting range
DAQDARWIN_RANGE_MA_20MA	20 mA	-20.000-20.000mA

Power connection method

Mnemonic	Description
DAQDARWIN_WIRE_1PH2W	Single-phase two-wire
DAQDARWIN_WIRE_1PH3W	Single phase, 3-wire (for 3-wire only)
DAQDARWIN_WIRE_3PH3W2I	3-phase 3-wire (for 2 voltage 2 current 3-wire only)
DAQDARWIN_WIRE_3PH3W3I	3-phase 3-wire (for 3 voltage 3 current 3-wire only)
DAQDARWIN_WIRE_3PH4W	3-phase, 4-wire (for 3-wire only)

Power measurement parameters

Mnemonic	Description
DAQDARWIN_POWERITEM_I0	$(I1+I2+I3)/3$
DAQDARWIN_POWERITEM_I1	Effective current 1
DAQDARWIN_POWERITEM_I2	Effective current 2
DAQDARWIN_POWERITEM_I3	Effective current 3
DAQDARWIN_POWERITEM_I13	$(I1+I3)/2$
DAQDARWIN_POWERITEM_P0	$P1+P2+P3$
DAQDARWIN_POWERITEM_P1	Active power 1
DAQDARWIN_POWERITEM_P2	Active power 2
DAQDARWIN_POWERITEM_P3	Active power 3
DAQDARWIN_POWERITEM_P13	$P1+P3$
DAQDARWIN_POWERITEM_PF0	$P0/(P0^2+VAR0^2)^{1/2}=P0/VA0$
DAQDARWIN_POWERITEM_PF1	Power factor 1
DAQDARWIN_POWERITEM_PF2	Power factor 2
DAQDARWIN_POWERITEM_PF3	Power factor 3
DAQDARWIN_POWERITEM_PF13	$P13/(P13^2+VAR13^2)^{1/2}=P13/VA13$
DAQDARWIN_POWERITEM_PH0	$\tan^{-1}(VAR0/P0)$
DAQDARWIN_POWERITEM_PH1	Phase 1
DAQDARWIN_POWERITEM_PH2	Phase 2
DAQDARWIN_POWERITEM_PH3	Phase 3
DAQDARWIN_POWERITEM_PH13	$\tan^{-1}(VAR13/P13)$
DAQDARWIN_POWERITEM_V0	$(V1+V2+V3)/3$
DAQDARWIN_POWERITEM_V1	Effective power 1
DAQDARWIN_POWERITEM_V2	Effective power 2
DAQDARWIN_POWERITEM_V3	Effective power 3
DAQDARWIN_POWERITEM_V13	$(V1+V3)/2$
DAQDARWIN_POWERITEM_VA0	$VA1+VA2+VA3$
DAQDARWIN_POWERITEM_VA1	Apparent power 1
DAQDARWIN_POWERITEM_VA2	Apparent power 2
DAQDARWIN_POWERITEM_VA3	Apparent power 3
DAQDARWIN_POWERITEM_VA13	$VA1+VA3$
DAQDARWIN_POWERITEM_VAR0	$VAR1+VAR2+VAR3$
DAQDARWIN_POWERITEM_VAR1	Reactive power 1
DAQDARWIN_POWERITEM_VAR2	Reactive power 2
DAQDARWIN_POWERITEM_VAR3	Reactive power 3
DAQDARWIN_POWERITEM_VAR13	$VAR1+VAR3$
DAQDARWIN_POWERITEM_FREQ	Frequency

11.3 Overview of the DARWIN Types

The data types below are provided.

Type	Description	Page
DAQDARWIN	Device descriptor type.	11-13
DarwinDateTime	Time information structure.	11-13
DarwinChInfo	Channel information data structure.	11-14
DarwinDataInfo	Measured data structure.	11-14
DarwinModuleInfo	Module information structure.	11-14
DarwinUnitInfo	Unit information structure.	11-15
DarwinSystemInfo	System configuration data structure.	11-15

Type	Description
Callback type	Add prefix "DLL" to the function name and write in uppercase. Example: callback type of the openDARWIN function: DLLOPENDARWIN

The callback type is used to link the executable module (.dll) when using the Visual C.

11.4 DARWIN Types

Explanation on the Description

Visual C/Visual C++, and Visual Basic Types

Indicates the type in Visual C/Visual C++ and Visual Basic.

Types without signs in Visual C/Visual C++ become types with signs in Visual Basic.

Number of array elements for Visual C/Visual C++ types is omitted.

Retrieval

The markings below are used to indicate items that can be retrieved, those that the user can set, and so on.

Talker

Data retrieved using the talker function related to “retrieval of the measured data.”

R: Item that can be retrieved.

Channel Information

Data retrieved using the “retrieval of the channel information data” function.

R: Item that can be retrieved.

ASCII

Data retrieved using the “retrieval of measured data in ASCII code” function.

R: Item that can be retrieved.

Binary

Data retrieved using the “retrieval of measured data in binary code” function.

R: Item that can be retrieved.

Terminology

In the explanation of types, terminology representing DARWIN functions is used. For a description of the DARWIN terminology, see appendix 2.

DAQDARWIN

Type for storing the device descriptor.

Handled as Long type in Visual Basic and int in Visual C.

DarwinDateTime

DarwinDateTime structure

Visual C/C++ Type	Name	Description	Visual Basic Type
char	aYear	Last two digits of the year (0 to 99)	Byte
char	aMonth	Month (1 to 12) R	Byte
char	aDay	Day (1 to 31)	Byte
char	aHour	Hours (0 to 23)	Byte
char	aMinute	Minutes (0 to 59)	Byte
char	aSecond	Seconds (0 to 59)	Byte
short	aMilliSecond	Not used	Integer

Retrieval

Name	Description	Talker
aYear	Lower two digits of the year (0 to 99)	R
aMonth	Month (1 to 12)	R
aDay	Day (1 to 31)	R
aHour	Hours (0 to 23)	R
aMinute	Minutes (0 to 59)	R
aSecond	Seconds (0 to 59)	R

Time information data structure.

Visual C++: The wrapper class is CDAQDARWINDateTime.

Milliseconds are not used with the command support port.

DarwinChInfo

DarwinChInfo structure

Visual C/ Visual C++ Type	Name	Description	Visual Basic Type
int	aChNo	Channel number	Long
int	aPoint	Decimal point pos.	Long
int	aStatus	Channel status	Long
int	aChType	Channel type	Long
char []	aUnit	Unit name	String * DAQDARWIN_MAXUNITLEN
char	align	Not used.	(0 To 1) As Byte

Retrieval

Name	Description	ASCII	Binary	Channel Information
aChNo	Channel number	R	R	R
aPoint	Decimal Point Position	R	-	R
aStatus	Channel Status	-	-	R
aChType	Channel Type	R	R	R
aUnit	Unit Name	R	-	R

Channel information data structure.

Visual C++: The wrapper class is CDAQDARWINChInfo.

Note

For functions that retrieve measured data in binary code, the decimal point position and unit information is not retrieved. To convert the measured data into a value with an engineering unit, retrieve the decimal point position separately.

DarwinDataInfo

DarwinDataInfo structure

Visual C/ Visual C++ Type	Name	Description	Visual Basic Type
int	aValue	Data Value	Long
int	aStatus	Data Status	Long
int []	aAlarm	Array of alarms for the number of alarm levels	(1 To 4) As Long

Retrieval

Name	Description	ASCII	Binary
aValue	Data Value	R	R
aStatus	Data Status	R	R
aAlarm	Array of alarms for the number of alarm levels	R	R

Measured data structure.

The wrapper class is CDAQDARWINDataInfo.

DarwinModuleInfo

DarwinModuleInfo structure

Visual C/ Visual C++ Type	Name	Description	Visual Basic Type
int	aSlotNo	Slot number	Long
int	aInternalCode	Internal code	Long
char []	aName	Module name	String * DAQDARWIN_MAXMODULELEN
char	align	Not used.	(0 To 1) As Byte

Module information structure.

DarwinUnitInfo

DarwinUnitInfo structure

Visual C/ Visual C++ Type	Name	Description	Visual Basic Type
int	aExist	Valid/Invalid of Reports	Long
int	aUnitNo	Unit Number	Long
DarwinModuleInfo []	aModule	Array of module information of the specified number of slots	(0 To 5) As DarwinModuleInfo

Unit information structure.

DarwinSystemInfo

DarwinSystemInfo structure

Visual C/ Visual C++ Type	Name	Description	Visual Basic Type
DarwinUnitInfo	aMainUnit	Main unit's unit information	DarwinUnitInfo
DarwinUnitInfo []	aSubUnit	Array of unit information of of the specified number of subunits	(0 To 5) As DarwinUnitInfo

System configuration data structure.

Visual C++: The wrapper class is CDAQDARWINSysInfo.

12.1 MX100 Class

The Extended API consists of the MX100-dedicated classes below.

- CDAQConfig
 - CDAQMXItemConfig
 - CDAQHandler
 - CDAQMX
 - CDAQMX100
 - CDAQMXDataBuffer
 - CDAQMXList
 - CDAQMXAOPWMList
 - CDAQMXBalanceList
 - CDAQMXDOList
 - CDAQMXTransmitList
- : Class of API.
 • : Class added by the extended API

CDAQMX100 Class

A handler class for the main unit. Executes status transitions.

CDAQMXAOPWMList Class

Class for managing command AO/PWM channel data.

CDAQMXBalanceList Class

Class for managing initial balance data.

CDAQMXDataBuffer Class

Class for storing the measured data by channel.

CDAQMXDOList Class

Class for managing command DO channel data.

CDAQMXItemConfig Class

Class for the setup data handled by setting items.

CDAQMXList Class

Common class for managing user data in lists.

CDAQMXTransmitList Class

Class for managing transmission output data.

Notes

Memory usage is large since the various kinds of data are stored internally. Performance may suffer since the status is retrieved during status transitions. Also, if memory or disk space is insufficient, data may not be saved correctly.

Problems may occur when communications are performed for functions other than status transitions.

If access to the MX100 main unit is lost, communications are closed.

If you want to keep the connection open, run status data retrieval.

The following items are restricted, and not supported.

- Auto control of the FIFO cannot be used.
- User count cannot be used.
- Data numbers cannot be used.
- Status of the 7-segment LED status cannot be retrieved.
- Timeout value of the CF card cannot be set.
- Communication timeout cannot be set arbitrarily. It is set automatically to 180 seconds after communications are opened.

12.2 Correspondence between the Functions and Class/Member Functions - MX100 -

This section indicates the correspondence between the functions that the Extended API supports and the class member functions.

There are two types, status transition functions and retrieval functions.

Status transition functions control the MX100. When measured data is retrieved with the data retrieval function, the measured data advances by only one measurement interval's worth of data (the status of the Extended API changes).

The retrieval function returns the parameter value. When data values are retrieved, the data value of the current status held by the Extended API is returned (the status of the Extended API does not change).

Status Transition Functions

The FIFO column in the table indicates the FIFO operation when the function is executed while FIFO is running.

Stop: Stops the FIFO when the function is executed.

Continue: Continues the FIFO even when the function is executed.

Communication Functions

Function	FIFO	Class and Member Function
Connect to the MX100.	Continue	CDAQMX100:: open
Disconnect from the MX100.	Continue	CDAQMX100:: close

Starting/Stopping the FIFO

Function	FIFO	Class and Member Function
Start the FIFO	Continue	CDAQMX100:: measStart
Stop the FIFO	Stop	CDAQMX100:: measStop

Control Functions

Function		FIFO	Class and Member Function
Set date/time	Current time	Stop	CDAQMX100:: setDateTime
Set backup valid/invalid		Continue	CDAQMX100:: switchBackup
Format of the CF card		Stop	CDAQMX100:: formatCF
Unit	Reconfigure	Stop	CDAQMX100:: reconstruct
	Initialize	Stop	CDAQMX100:: initSetValue
	Reset alarm (alarm ACK)	Stop	CDAQMX100:: ackAlarm
7-segment LED display		Continue	CDAQMX100:: displaySegment
Initialize stored data	Specify channel	Continue	CDAQMX100:: initDataCh
	Specify FIFO	Continue	CDAQMX100:: initDataFIFO

At the end of communications, the control function updates the status.
 See the data manipulation functions for more about the data transmission and setup functions.

Setup Functions

Function		FIFO	Class and Member Function
Setup data	Collectively (send collectively)	All setup data	Stop CDAQMX100:: sendConfig
		Basic setup data	Stop CDAQMX100:: sendConfig
	Individually	System config data	Stop CDAQMX100:: sendConfig
		Channel setup data	Stop CDAQMX100:: sendConfig
		Initial balance data	Stop CDAQMX100:: sendConfig
		Output channel data	Stop CDAQMX100:: sendConfig
Initial balance data	Execute	Stop CDAQMX100:: initBalance	
	Reset	Stop CDAQMX100:: clearBalance	

The setup data setup functions send the data being stored.
 When setting arbitrary initial balance data, see the initial balance data sending function under data manipulation functions.

Setup Change Functions

The setup functions send the settings then update the status.

These settings are for individual channels. If the settings could not be entered, an error is usually returned.

Specification is possible with data values or measured values (Double).

Range Settings

Function	FIFO	Class and Member Function
Skip	Stop	CDAQMX100:: setRange
DC voltage input	Stop	CDAQMX100:: setRange
Thermocouple input	Stop	CDAQMX100:: setRange
RTD	Stop	CDAQMX100:: setRange
Digital input	Stop	CDAQMX100:: setRange
Resistance	Stop	CDAQMX100:: setRange
Strain	Stop	CDAQMX100:: setRange
AO	Stop	CDAQMX100:: setRange
PWM	Stop	CDAQMX100:: setRange
Difference computation between channels	Stop	CDAQMX100:: setChDELTA
Remote RJC	Stop	CDAQMX100:: setChRRJC

Channel Settings

Function	FIFO	Class and Member Function	
Unit name	Stop	CDAQMX100:: setChUnit	
Tag	Stop	CDAQMX100:: setChTag	
Comment	Stop	CDAQMX100:: setChComment	
AI/DI/AO/PWM	Span	Stop	CDAQMX100:: setSpan
AI/DI	Scale	Stop	CDAQMX100:: setScale
	Alarm	Stop	CDAQMX100:: setAlarm
	Hysteresis	Stop	CDAQMX100:: setHisteresys
AI	Filter coefficient	Stop	CDAQMX100:: setFilter
	Ref. junction compensation (RJC)	Stop	CDAQMX100:: setRJCType
	Burnout	Stop	CDAQMX100:: setBurnout
DO	De-energize	Stop	CDAQMX100:: setDeenergize
	Hold	Stop	CDAQMX100:: setHold
	Reference alarm	Stop	CDAQMX100:: setRefAlarm
Channel kind	DO type	Stop	CDAQMX100:: setChKind
	AO type	Stop	CDAQMX100:: setChKind
	PWM type	Stop	CDAQMX100:: setChKind

Module Settings

Function	FIFO	Class and Member Function
Interval type	Stop	CDAQMX100:: setInterval
A/D integration time type	Stop	CDAQMX100:: setIntegral

Unit Settings

Function	FIFO	Class and Member Function
Unit number	Stop	CDAQMX100:: setUnitNo
Temperature unit type	Stop	CDAQMX100:: setUnitTemp
CF write mode	Stop	CDAQMX100:: setCFWriteMode

Output Channel Data

Function	FIFO	Class and Member Function
Output type	Stop	CDAQMX100:: setOutputType
Selected value	Stop	CDAQMX100:: setChoice
Pulse interval integer multiple	Stop	CDAQMX100:: setPulseTime

Data Manipulation Functions

DO Data

Function		FIFO	Class and Member Function
Create		Continue	CDAQMXDOList:: create
Delete		Continue	CDAQMXDOList:: del
Partial chng	User spec	Continue	CDAQMXDOList:: change
	Copy	Continue	CDAQMXDOList:: copy
Transmission	Existing spec	Continue	CDAQMX100:: commandDO
	Change spec	Continue	CDAQMX100:: switchDO

AO/PWM Data

Function		FIFO	Class and Member Function
Create		Continue	CDAQMXAOPWMList:: create
Delete		Continue	CDAQMXAOPWMList:: del
Partial change	Output data value	Continue	CDAQMXAOPWMList:: change
	Actual output value	Continue	CDAQMX100:: changeAOPWMValue
	Copy	Continue	CDAQMXAOPWMList:: copy
Transmit		Continue	CDAQMX100:: commandAOPWM

Initial Balance Data

Function		FIFO	Class and Member Function
Create		Continue	CDAQMXBalanceList:: create
Delete		Continue	CDAQMXBalanceList:: del
Partial change	User spec	Continue	CDAQMXBalanceList:: change
	Copy	Continue	CDAQMXBalanceList:: copy
Transmit		Stop	CDAQMX100::reloadBalance

Transmission Output Data

Function		FIFO	Class and Member Function
Create		Continue	CDAQMXTransmitList:: create
Delete		Continue	CDAQMXTransmitList:: del
Partial change	User specification	Continue	CDAQMXTransmitList:: change
	Copy	Continue	CDAQMXTransmitList:: copy
Transmit	Existing specification	Continue	CDAQMX100:: commandTransmit
	Change specification	Continue	CDAQMX100:: switchTransmit

Manipulates each data by identifier.

Follows the CDAQMX100 class.

For manipulation other than transmission, status is not updated (no communication).

Retrieval Functions

Function		FIFO	Class and Member Function	
Status data		Continue	CDAQMX100:: updateStatus	
System configuration data		Continue	CDAQMX100:: updateSystem	
Setup data		Continue	CDAQMX100:: updateConfig	
Output data	DO Data	Continue	CDAQMX100:: updateDOData	
	AO/PWM data	Continue	CDAQMX100:: updateAOPWMData	
	Transmission Output Data			
Channel Information Data		Continue	CDAQMX100:: updateInfoCh	
Measured Data	Specify	FIFO val	Continue	CDAQMX100:: measDataCh
	channel	Inst val	Continue	CDAQMX100:: measInstCh
	Specify	FIFO val	Continue	CDAQMX100:: measDataFIFO
	FIFO	Inst val	Continue	CDAQMX100:: measInstFIFO
Initial balance data		Continue	CDAQMX100:: updateBalance	
Output channel data		Continue	CDAQMX100:: updateOutput	

Data retrieval is performed collectively and internally by this Extended API. Depending on the acquisition, the status may also be updated. Channel information data and setup data (including system configuration data, initial balance data, and output channel data) are stored internally, but the user can update stored data explicitly.

Setup Item Functions

Function		FIFO	Class and Member Function
Setup data	Receive collectively	Continue	CDAQMX100:: getItemAll
	Send collectively	Stop	CDAQMX100:: setItemAll
Setup items	Read	Continue	CDAQMXItemConfig:: readItem
	Write	Continue	CDAQMXItemConfig:: writeItem
	Initialize	Continue	CDAQMXItemConfig:: initialize

Loading, writing, and initializing of setting items is performed through access to the field where the item is stored, and validity checks are not performed on those fields. Also, status is not updated (no communication).

Retrieval Functions

Each data is stored in its class according to type. They are retrieved from the CDAQMX100 class.

Measured Data

Data Name		Class and Member Function
Data value		CDAQMXDataInfo:: getValue
Data status values		CDAQMXDataInfo:: getStatus
Alarm (presence/absence)		CDAQMXDataInfo:: isAlarm
Measured value	Double integer	CDAQMXDataInfo:: getDoubleValue
	String	CDAQMXDataInfo:: getStringValue
Time	No. of seconds	CDAQMXDateTime:: getTime
	Milliseconds	CDAQMXDateTime:: getMilliSecond
Valid data (presence/absence)		CDAQMXDataBuffer:: isCurrent

The data is retrieved from DAQMX100::getClassMXDataBuffer through CDAQMXDataBuffer::currentDataInfo and CDAQMXDataBuffer::currentDateTime.

Channel Information Data

Data Name		Class and Member Function
FIFO number		CDAQMXChInfo:: getFIFONo
Channel sequence number in the FIFO		CDAQMXChInfo:: getFIFOIndex
Display minimum value		CDAQMXChInfo:: getDisplayMin
Display maximum value		CDAQMXChInfo:: getDisplayMax
Measurable minimum value		CDAQMXChInfo:: getRealMin
Measurable maximum value		CDAQMXChInfo:: getRealMax

The data is retrieved from DAQMX100::getClassMXDataBuffer through CDAQMXDataBuffer::getClassMXChInfo.

Channel Setup Data

Data Name		Class and Member Function	
Channel status (valid/invalid)		CDAQMXChConfig:: isValid	
Decimal Point Position		CDAQMXChConfig:: getPoint	
Channel kind		CDAQMXChConfig:: getKind	
Range type		CDAQMXChConfig:: getRange	
Scale type		CDAQMXChConfig:: getScale	
Unit name		CDAQMXChConfig:: getUnit	
Tag		CDAQMXChConfig:: getTag	
Comment		CDAQMXChConfig:: getComment	
AI/DI/ Span	Min value	Data value	CDAQMXChConfig:: getSpanMin
		Meas values	CDAQMXItemConfig:: getDoubleSpanMin
	Max value	Data value	CDAQMXChConfig:: getSpanMax
		Meas value	CDAQMXItemConfig:: getDoubleSpanMax
AI/DIAO/PWM			
Scale	Min value	Data value	CDAQMXChConfig:: getScaleMin
		Meas value	CDAQMXItemConfig:: getDoubleScaleMin
	Max value	Data value	CDAQMXChConfig:: getScaleMax
		Meas values	CDAQMXItemConfig:: getDoubleScaleMax
Alarm type		CDAQMXChConfig:: getAlarmType	
Alarm value (ON)	Data Value	CDAQMXChConfig:: getAlarmValueON	
	Meas values	CDAQMXItemConfig:: getDoubleAlarmON	
Alarm value (OFF)	Data Value	CDAQMXChConfig:: getAlarmValueOFF	
	Meas value	CDAQMXItemConfig:: getDoubleAlarmOFF	
Hysteresis	Data Value	CDAQMXItemConfig:: getHisterisys	
	Meas value	CDAQMXItemConfig:: getDoubleHisterisys	
AI	Filter		CDAQMXChConfig:: getFilter
	RJC type		CDAQMXChConfig:: getRJCType
	RJC voltage		CDAQMXChConfig:: getRJCVolt
	Burnout		CDAQMXChConfig:: getBurnout
DO	De-energize		CDAQMXChConfig:: isDeenergize
	Hold		CDAQMXChConfig:: isHold
	Reference alarm		CDAQMXChConfig:: isRefAlarm
Channels under going difference between channels computation/RRJC/AO/PWM			
	Ref. channel	CDAQMXChConfig:: getRefChNo number	
Initial bal data	Boolean value		CDAQMXBalanceData:: getBalanceValid
	Initial balance value		CDAQMXBalanceData:: getBalanceValue
Output	Output type		CDAQMXOutputData:: getOutputType
channel data	Selection when idle		CDAQMXOutputData:: getIdleChoice
	Selection during error		CDAQMXOutputData:: getErrorChoice
	Value if the selected value is the "specified value."		
	Data value	CDAQMXOutputData:: getPresetValue	
	Meas value	CDAQMXItemConfig:: getDoublePresetValue	
	Pls intrvl integer mult.		CDAQMXOutputData:: getPulseTime

The data is retrieved from CDAQMX100::getClassMXItemConfig through CDAQMXItemConfig::getClassMXChConfig. The initial balance data is the same as the current data. It is retrieved from CDAQMX100::getClassMXItemConfig through CDAQMXItemConfig::getClassMXBalanceData.

The output channel data is retrieved from CDAQMX100::getClassMXItemConfig through CDAQMXItemConfig::getClassMXOutputData.

Network Information Data

Data Name	Class and Member Function
Host name	CDAQMXNetInfo:: getHost
IP address	CDAQMXNetInfo:: getAddress
Port number	CDAQMXNetInfo:: getPort
Subnet mask	CDAQMXNetInfo:: getSubMask
Gateway address	CDAQMXNetInfo:: getGateway

The data is retrieved from CDAQMX100::getClassMXItemConfig through CDAQMXItemConfig::getClassMXNetInfo.

System Configuration Data

Data Name	Class and Member Function	
Modules	Module type	CDAQMXSysInfo: getModuleType
	Number of channels	CDAQMXSysInfo: getChNum
	Interval type	CDAQMXSysInfo: getInterval
	AD integration time type	CDAQMXSysInfo: getIntegral
	Valid/Invalid value	CDAQMXSysInfo: isModuleValid
	Module type at startup	CDAQMXSysInfo: getStandbyType
	Actual module type	CDAQMXSysInfo: getRealType
	Terminal type	CDAQMXSysInfo: getTerminalType
	Version	CDAQMXSysInfo: getModuleVersion
	FIFO number	CDAQMXSysInfo: getFIFONo
	Serial number	CDAQMXSysInfo: getModuleSerial
	Unit	Unit type
Style		CDAQMXSysInfo: getStyle
Unit number		CDAQMXSysInfo: getUnitNo
Temperature unit type		CDAQMXSysInfo: getTempUnit
Power supply frequency		CDAQMXSysInfo: getFrequency
Part number		CDAQMXSysInfo: getPartNo
Option		CDAQMXSysInfo: getOption
Serial number		CDAQMXSysInfo: getUnitSerial
MAC address		CDAQMXSysInfo: getMAC
CF write mode	CDAQMXSysInfo: getCFWriteMode	

The data is retrieved from CDAQMX100::getClassMXItemConfig through CDAQMXItemConfig::getClassMXSysInfo.

Status Data

Data Name		Class and Member Function
Unit status value		CDAQMXStatus:: getUnitStatus
Valid number of FIFOs		CDAQMXStatus:: getFIFONum
Backup (presence or absence)		CDAQMXStatus:: isBackup
FIFO	FIFO status value	CDAQMXStatus:: getFIFOStatus
	Interval type	CDAQMXStatus:: getInterval
CF	CF status type	CDAQMXStatus:: getCFStatus
	Size	CDAQMXStatus:: getCFSize
	Remaining capacity	CDAQMXStatus:: getCFRemain
Status	No. of seconds	CDAQMXStatus:: getTime
return time	Milliseconds	CDAQMXStatus:: getMilliSecond

The data is retrieved from CDAQMX100::getClassMXItemConfig through CDAQMXItemConfig::getClassMXStatus.

Current Data

Data Name		Class and Member Function
DO data	Valid/Invalid Value	CDAQMXDOData:: getDOValid
	ON/OFF status	CDAQMXDOData:: getDOONOFF
AO/PWM data	Valid/Invalid Value	CDAQMXAOPWMData:: getAOPWMValid
	Output data value	CDAQMXAOPWMData:: getAOPWMValue
	Output value	CDAQMX100:: currentDoubleAOPWMValue
Initial balance data	Valid/Invalid Value	CDAQMXBalanceResult:: getBalanceValid
	Initial balance value	CDAQMXBalanceResult:: getBalanceValue
	Initial balance result	CDAQMXBalanceResult:: getResult
Trans. outpt data	Transmission status	CDAQMXTransmit:: getTransmit

This is the status of each data retrieved with the data retrieval functions.

The initial balance result of the initial balance data is the result executed by the setup function. DO data is retrieved from CDAQMX100::getClassMXDOList through CDAQMXDOList::getCurrent.

AO/PWM data is retrieved from CDAQMX100::getMXAOPWMList through CDAQMXAOPWMList::getCurrent.

Initial balance data is retrieved from CDAQMX100::getClassMXBalanceList through CDAQMXBalanceList::getCurrent.

Transmission output data is retrieved from CDAQMX100::getClassMXTransmitList through CDAQMXTransmitList::getCurrent.

Actually-output output statuses such as DO data and AO/PWM data can be retrieved as current data. However, immediately after sending data, the specified value is returned, and the actual output may occur at the next timing.

Held data are the values retrieved upon a status update. It is not data from the time the retrieval function was called.

User Data

Data Name		Class and Member Function
DO data	Valid/Invalid value	CDAQMXDOData:: getDOValid
	ON/OFF status	CDAQMXDOData:: getDOONOFF
AO/PWM data	Valid/Invalid value	CDAQMXAOPWMData:: getAOPWMValid
	Output data value	CDAQMXAOPWMData:: getAOPWMValue
	Output value	CDAQMX100:: userDoubleAOPWMValue
Initial balance data	Valid/Invalid value	CDAQMXBalanceData:: getBalanceValid
	Initial balance value	CDAQMXBalanceData:: getBalanceValue
Trans.outpt data	Transmission status	CDAQMXTransmit:: getTransmit

Gets the data values created by the user with data manipulation functions.

DO data is retrieved from CDAQMX100::getClassMXDOList through CDAQMXDOList::getClassMXDOData.

AO/PWM data is retrieved from CDAQMX100::getMXAOPWMList through CDAQMXAOPWMList::getClassMXAOPWM.

Initial balance data is retrieved from CDAQMX100::getClassMXBalanceList through CDAQMXBalanceList::getClassMXBalanceData.

Transmission output data is retrieved from CDAQMX100::getClassMXTransmitList through CDAQMXTransmitList::getClassMXTransmit.

Utilities

Function/Data Name		Class and Member Function
No. of remaining data	Retrieve by channel	CDAQMXDataBuffer:: getDataNum
	Retrieve by FIFO	CDAQMX100:: getDataNum
Error	Get MX-specific error	CDAQMX100:: getLastError
	Get the error message string.	CDAQMX100:: getErrorMessage
	Get the maximum length of the error message string.	CDAQMX100:: getMaxLenErrorMessage
	Get the number of the parameter on which an error was detected	CDAQMX100:: getItemError
Change from FIFO information to channel number	CDAQMX100:: toChNo	
Get the decimal point position by range type.	CDAQMXConfig:: getRangePoint	
Meas value	Change to double integer	CDAQMXDataInfo:: toDoubleValue
	Convert into string.	CDAQMXDataInfo:: toStringValue
Alarm	Get the alarm type string.	CDAQMXDataInfo:: getAlarmName
	Get the maximum length of the alarm string.	CDAQMXDataInfo:: getMaxLenAlarmName
Get the version number of this API.	CDAQMX100:: getVersionAPI	
Get the revision number of this API.	CDAQMX100:: getRevisionAPI	
Get a portion of the IP address.	CDAQMXNetInfo:: getPart	
AO/PWM	Convert the output values to output data values.	CDAQMXAOPWMData:: toAOPWMValue
	Convert the output data values to output values.	CDAQMXAOPWMData:: toRealValue
Setup items	Get setup itm strng from setup itm no.	CDAQMXItemConfig:: toItemName
	Gets the setup item number from the setup item string	CDAQMXItemConfig:: getItemNo
	Get the maximum length of the setup item string.	CDAQMXItemConfig:: getMaxLenItemName
Converts to style version.	CDAQMXSysInfo: toStyleVersion	

12.3 Programming - MX100/Visual C++ -

Adding the Path to the Include File

Add the path of the include file (DAQMX100.h) to the project. The method of adding the include file varies depending on the environment used.

Declaration in the Source File

Write the declaration in the source file.

```
#include "DAQMX100.h"
```

Note

The include file of the common section (DAQHandler.h) and the MX100 include file (DAQMX.h) is referenced from the include file described above. Thus, declaration for it is not necessary.

Library Designation

Adds libraries (DAQMX100.lib, DAQMX.lib, and DAQHandler.lib) to the project. The method of adding the include file varies depending on the environment used.

This enables the use of all classes. It also enables the use of all Visual C functions.

Data Retrieval

Program Example

```
////////////////////////////////////  
// MX100 sample for measurement  
#include <stdio.h>  
#include "DAQMX100.h"  
////////////////////////////////////  
int main(int argc, char* argv[])  
{  
    int rc; //return code  
    CDAQMX100 daqmx100; //class  
    int value;  
    //connect  
    rc = daqmx100.open("192.168.1.12");  
    //get  
    rc = daqmx100.measStart();  
    rc = daqmx100.measDataCh(1);  
    value = ((daqmx100.getClassMXDataBuffer(1))->  
    currentDataInfo()->getValue());  
    rc = daqmx100.measStop();  
    //disconnect  
    rc = daqmx100.close();  
    return rc;  
}  
////////////////////////////////////
```

Using Setup Items

Program Example

```

////////////////////////////////////
// MX100 sample for items
#include <stdio.h>
#include "DAQMX100.h"#include "DAQMXItems.h"
////////////////////////////////////
int main(int argc, char* argv[])
{
    int rc; //return code
    CDAQMX100 daqmx100; //class
    int i; //counter
    char strItem[BUFSIZ];
    int realLen;
    //connect
    rc = daqmx100.open("192.168.1.12");
    //get
    rc = daqmx100.getItemAll();
    //loop by items
    for (i = DAQMX_ITEM_ALL_START; i <= DAQMX_ITEM_ALL_END; i++)
    {
        //read
        realLen = (daqmx100.getClassMXItemConfig()).readItem(i,
strItem, BUFSIZ);
        //write
        rc = (daqmx100.getClassMXItemConfig()).writeItem(i,
strItem);
    }
    //set
    rc = daqmx100.setItemAll();
    //disconnect
    rc = daqmx100.close();
    return rc;
}
////////////////////////////////////

```

The program is an example of reading and writing all setup items.

Be sure to prepare string fields of sufficient size. By saving and loading groups of item numbers and item strings, you can backup the setup data.

For setup item numbers, see section 6.3.

12.4 Details of the MX100 Class

Classes are listed in alphabetical order by the class name.

CDAQMX100 Class

This class is a CDAQMX derived class.

This class communicates with the MX100, and stores the retrieved data.

It supports status transition functions. In general, after a function is executed, the status is updated and saved.

For setup functions, after each setting is executed the settings are received again and the status is updated. When collective transmission is carried out by a setup function or setting item function, the FIFO stops. The following data can be stored.

- Setup Data
- Channel Information Data
- Measured Data
- Time Information data

Data that is actively manipulated by the user such as command DO can be stored and managed.

By creating data to be sent in advance and assigning identifiers to it, the data can be easily transmitted. The following data can be stored.

- DO Data
- AO/PWM Data
- Transmission Output Data
- Initial Balance Data

Public Members

Construct/Destruct

CDAQMX100	Constructs an object.
~CDAQMX100	Destructs an object.

FIFO Functions

measStart	Starts data acquisition.
measStop	Stops data acquisition.

Control Functions

switchBackup	Switches backup.
reconstruct	Reconfigures the system.
initSetValue	Initializes the system.
ackAlarm	Resets alarms.
displaySegment	Displays the 7-segment LED.
sendConfig	Sends the setup data collectively.
initBalance	Executes initial balancing.
clearBalance	Initializes the initial balance value.

Setup Functions

setRange	Sets the range.
setChDELTA	Sets difference computation between channels.
setChRRJC	Sets remote RJC.
setChUnit	Sets the unit name.
setChTag	Sets the tag.
setChComment	Sets the comment.
setSpan	Sets the span.
setScale	Sets the scale.
setAlarm	Sets the alarm value.
setHisterisys	Sets the hysteresis.
setFilter	Set the filter time constant.
setRJCType	Sets the RJC type.
setBurnout	Sets the burnout type.
setDeenergize	Sets de-energize.
setHold	Sets hold.
setRefAlarm	Sets the reference alarm.
setChKind	Sets the channel type.
setInterval	Sets the interval type.
setIntegral	Sets the type of A/D integration time.
setUnitNo	Sets the unit number.
setUnitTemp	Sets the temperature unit type.
setCFWriteMode	Sets the CF write mode.
setOutputType	Sets the output type.
setChoice	Sets the selected value.
setPulseTime	Sets the integer multiple of the pulse interval.

Data Update Functions

updateStatus	Updates status data.
updateSystem	Updates the system configuration data.
updateConfig	Updates the setup data.
updateDOData	Updates the current DO data.
updateAOPWMData	Updates the current AO/PWM data.
updateBalance	Updates the current initial balance data.
updateOutput	Updates the current output channel data.
updateInfoCh	Updates the channel information data.

Data Retrieval Functions

measDataCh	Gets FIFO values by channel.
measDataFIFO	Gets FIFO values by FIFO.
measInstCh	Gets instantaneous values by channel.
measInstFIFO	Gets instantaneous values by FIFO.

Data Manipulation Functions

commandDO	Sends DO data.
switchDO	Switches DO data.
changeAOPWMValue	Changes AO/PWM data.
commandAOPWM	Sends AO/PWM data.
reloadBalance	Sends initial balance data.
commandTransmit	Sends transmission output data.
switchTransmit	Switches transmission output data.
currentDoubleAOPWMValue	Gets the current output data value.
userDoubleAOPWMValue	Gets the user created output data value.

Setup Item Functions

getItemAll	Gets the setup data collectively.
setItemAll	Sends the setup data collectively.

Member Data Manipulation

getClassMXItemConfig	Gets the setup data.
getClassMXDataBuffer	Gets the type information for each channel.
getClassMXDOList	Gets the DO data management information.
getClassMXAOPWMList	Gets the AO/PWM data management information.
getClassMXTransmitList	Gets the transmission output data management information.
getClassMXBalanceList	Gets initial balance data management information.

Utilities

initDataCh	Initializes the stored data of the specified channel.
initDataFIFO	Initializes the stored data of the specified FIFO.
getDataNum	Gets the remaining number of data.
toChNo	Gets the channel number.

• Overridden Members**Communication Functions**

open	Establishes connection.
------	-------------------------

Control Functions

setDateTime	Sets the time information.
formatCF	Formats the CF card.

Utilities

isObject	Checks an object.
----------	-------------------

• Inherited Members

See CDAQHandler.

closeget getErrorMessage getMaxLengthErrorMessage getRevisionAPI
getVersionAPI receiveLine sendLine setTimeout

See CDAQMX

autoFIFO getAOPWMData getBalance getChannel getChConfig
getChData getChDataNo getChInfo getConfig getData getDOData
getFIFOData getFIFODataNo getItemError getLastError
getMXConfig getOutput getStatusData getUserTime getTimeData
initSystem resetBalance runBalance setAOPWMData setBackup
setBalance setConfig setDOData setMXConfig setOutput
setSegment setTransmit setUserTime startFIFO stopFIFO
talkChData talkChInfo talkConfig

Protected Members**Data Members**

m_cMXItemConfig	Field for storing the setup data.
m_cMXDataBuffer	Field for storing various types of information on each measurement channel.
m_cMXDOList	Field for managing the DO data.
m_cMXAOPWMList	Field for managing the AO/PWM data.
m_cMXTransmitList	Field for managing the transmission output data.
m_cMXBalanceList	Field for managing the initial balance data.

Member Data Manipulation

measClear	Initializes the data member for retrieval of the measured data.
userClear	Initializes the data member of the management field.

Data Update Functions

updateAll	Updates all status and information data.
updateRenew	Updates the status.

Data Retrieval Functions

getDataCh	Gets the FIFO values by channel.
getDataFIFO	Gets FIFO values by FIFO.
getInstCh	Gets instantaneous values by channel.
getInstFIFO	Gets instantaneous values by FIFO.

Utilities

nextFIFO	Increments the current index number by FIFO.
getVersionMX100DLL	Gets the version number of the DLL.
getRevisionMX100DLL	Gets the revision number of the DLL.

• Inherited Members

See CDAQHandler.

m_comm m_nRemainSize receive receiveRemain send

See CDAQMX

m_nNo m_nLastError m_bAutoFIFO m_llUserTime m_nSessionNo
m_chFIFONo m_chFIFOIndex m_chDataType m_chDeciPos
m_lastFIFODataNo m_lastChDataNo m_startChNo m_endChNo
m_curChNo m_startFIFOIdx m_endFIFOIdx m_curFIFOIdx
m_startDataNo m_endDataNo m_curDataNo m_nFIFONo m_nDataNum
m_nChNum runCommand sendPacket receivePacket receiveBlock nop
registry getNo incCurDataNo incCurFIFOIdx getDataNo searchChNo
clearAttr clearData runPacket receiveBuffer m_nTimeNum
m_packetVer m_nItemError m_bTalkConfig m_bTalkChInfo
m_bTalkData getPacketVersion clearLastDataNoCh
clearLastDataNoFIFO getVersionDLL getRevisionDLL

Private Members

None.

Member Functions (Alphabetical Order)

CDAQMX100::ackAlarm

Syntax

```
int ackAlarm(void);
```

Description

Resets alarms.

Updates the status if successful.

Return value

Returns an error number.

Reference

```
initSystem
updateRenew
```

CDAQMX100::CDAQMX100

Syntax

```
CDAQMX100(void);
CDAQMX100(const char * strAddress, unsigned int uiPort =
DAQMX_COMMPORT, int * errCode = NULL);
virtual ~CDAQMX100(void);
```

Parameters

strAddress	Specify the IP address as a string.
uiPort	Specify the port number.
errCode	Specify the destination where the error number is to be returned.

Description

Constructs or destructs an object.

When constructing, the data member is initialized. The default value as a general rule is 0 (NULL). When the parameters are specified, a connection is established. If the return destination is specified, the error number during connection is returned.

When destructing, the data member field is released. The connection is dropped (close) when the communication descriptor exists. The error number is not returned.

Reference

```
measClear open userClear
CDAQMX::CDAQMX
```

CDAQMX100::changeAOPWMValue

Syntax

```
void changeAOPWMValue(int idAOPWM, int aopwmNo, int bValid,  
double realValue);
```

Parameters

idAOPWM	Specify the AO/PWM data identifier.
aopwmNo	Specify the AO/PWM data number.
bValid	Specify valid/invalid using a Boolean value.
realValue	Specify the actual output value.

Description

Changes the AO/PWM data of the specified AO/PWM data identifier.
Converts the specified actual output values to output data values and stores them.

Reference

```
getClassMXAOPWMList getClassMXItemConfig  
CDAQMXAOPWMData::toAOPWMValue  
CDAQMXAOPWMList::change  
CDAQMXItemConfig::getClassMXOutputData  
CDAQMXItemConfig::getRangePoint  
CDAQMXOutputData::getOutputType
```

CDAQMX100::clearBalance

Syntax

```
int clearBalance(void);
```

Description

Resets initial balancing.
Stores the result in the current data of the initial balance data management field of the data member.
Updates the status if successful.

Return value

Returns an error number.

Reference

```
getClassMXBalanceList resetBalance updateRenew  
CDAQMXBalanceList::getCurrent
```

CDAQMX100::commandAOPWM

Syntax

```
int commandAOPWM(int idAOPWM);
int commandAOPWM(CDAQMXAOPWMData & cMXAOPWMData);
```

Parameters

idAOPWM	Specify the AO/PWM data identifier.
cMXAOPWMData	Specify AO/PWM data.

Description

Sends the specified AO/PWM data.

Updates the status if successful.

Return value

Returns an error number.

Error:

Not Data	There is no data.
----------	-------------------

Reference

```
getClassMXAOPWMList setAOPWMData updateRenew
CDAQMXAOPWMList::getClassMXAOPWMData
```

CDAQMX100::commandDO

Syntax

```
int commandDO(int idDO);
int commandDO(CDAQMXDODData & cMXDODData);
```

Parameters

idDO	Specify the DO data identifier.
cMXDODData	Specify the DO data.

Description

Sends the specified DO data.

Updates the status if successful.

Return value

Returns an error number.

Error:

Not Data	There is no data.
----------	-------------------

Reference

```
getClassMXDOList setDODData updateRenew
CDAQMXDOList::getClassMXDODData
```

CDAQMX100::commandTransmit

Syntax

```
int commandTransmit(int idTrans);  
int commandTransmit(CDAQMXTransmit & cMXTransmit);
```

Parameters

idTrans Specify the transmission output data identifier.
cMXTransmit Specify the transmission output data.

Description

Sends the specified transmission output data.
Updates the status if successful.

Return value

Returns an error number.
Error:
Not Data There is no data.

Reference

```
getClassMXTransmitList setTransmit updateRenew  
CDAQMXTransmitList::getClassMXTransmit
```

CDAQMX100::currentDoubleAOPWMValue

Syntax

```
double currentDoubleAOPWMValue(int aopwmNo);
```

Parameters

aopwmNo Specify the AO/PWM data number.

Description

Gets the output data value of the specified AO/PWM data number from the current data of the AO/PWM data management field of the data member.
Returns 0.0 if it does not exist.

Return value

Returns the actual output value.

Reference

```
getClassMXAOPWMList getClassMXItemConfig getOutputRange  
CDAQMXAOPWMList::getCurrent  
CDAQMXAOPWMList::getAOPWMValue  
CDAQMXAOPWMList::toRealValue
```

CDAQMX100::displaySegment

Syntax

```
int displaySegment(int dispPattern0, int dispPattern1, int
dispType, int dispTime);
```

Parameters

dispPattern0	Specify the display pattern of segment number 0.
dispPattern1	Specify the display pattern of segment number 1.
dispType	Specify the display format.
dispTime	Specify the display time.

Description

Sets the display of the 7-segment LED.
Returns the display pattern prior to setting.
Updates the status if successful.

Return value

Returns an error number.

Reference

setSegment updateRenew

CDAQMX100::formatCF

Syntax

```
virtual int formatCF(void);
```

Description

Formats the CF card.
Updates the status if successful.

Return value

Returns an error number.

Reference

updateRenew
CDAQMX::formatCF

CDAQMX100::CDAQMXItemConfig

Syntax

```
CDAQMXItemConfig & getClassMXItemConfig(void);
```

Description

Gets the object of the setup data field from the data member.

Return value

Returns a reference to the object.

CDAQMX100::getClassMXAOPWMList

Syntax

```
CDAQMXAOPWMList & getClassMXAOPWMList(void);
```

Description

Gets the object of the AO/PWM data management field from the data member.

Return value

Returns a reference to the object.

CDAQMX100::getClassMXBalanceList

Syntax

```
CDAQMXBalanceList & getClassMXBalanceList(void);
```

Description

Gets the object of the initial balance data management field from the data member.

Return value

Returns a reference to the object.

CDAQMX100::getClassMXDataBuffer

Syntax

```
CDAQMXDataBuffer * getClassMXDataBuffer(int chNo);
```

Parameters

chNo Specify the channel number.

Description

Gets the field for storing measured data by channel from the data member by object.

Returns the object of the specified specified channel number.

Returns NULL if it does not exist.

Return value

Returns a pointer to the object.

CDAQMX100::getClassMXDOList

Syntax

```
CDAQMXDOList & getClassMXDOList(void);
```

Description

Gets the object of the DO data management field from the data member.

Return value

Returns a reference to the object.

CDAQMX100::getClassMXItemConfig

Syntax

```
CDAQMXItemConfig & getClassMXItemConfig(void);
```

Description

Gets the object of the setup data field from the data member.

Return value

Returns a reference to the object.

CDAQMX100::getClassMXTransmitList

Syntax

```
CDAQMXTransmitList & getClassMXTransmitList(void);
```

Description

Gets the object of the transmission output data management field from the data member.

Return value

Returns a reference to the object.

CDAQMX100::getDataCh

Syntax

```
int getDataCh(int chNo, int * bComm);
```

Parameters

chNo	Specify the channel number.
bComm	Specify the return destination where the communication will or will not occur.

Description

Gets the measured data of the specified channel number.

Increments the current index of various kinds of information for each channel of the data member.

When no stored data exists, communication is opened and new data is retrieved.

Gets FIFO values by channel, and stores them in the data member.

Returns the Boolean value indicating whether or not the communication was actually carried out if the return destination is specified.

Return value

Returns an error number.

Error:

Not Data	All kinds of information fields for each channel do not exist.
----------	--

Reference

```
getChData getChDataNo getClassMXDataBuffer getTimeData
talkChData
CDAQMXDataBuffer::create CDAQMXDataBuffer::next
CDAQMXDataBuffer::setDataInfo CDAQMXDataBuffer::setDateTime
CDAQMXStatus::isDataNo
```

CDAQMX100::getDataFIFO

Syntax

```
int getDataFIFO(int fifoNo, int * bComm);
```

Parameters

fifoNo Specify the FIFO number.

bComm Specify the return destination where the communication will or will not occur.

Description

Retrieves the measured data of the specified FIFO number.

Increments the current index of various kinds of information for each channel of the data member.

When no stored data exists, communication is opened and new data is retrieved.

Gets FIFO values by FIFO, and stores them in the data member.

Returns the Boolean value indicating whether or not the communication was actually carried out if the return destination is specified.

Return value

Returns an error number.

Error:

Not Data Various kinds of information fields for each channel do not exist.

Reference

```
getChData    getClassMXDataBuffer    getFIFODataNo    getTimeData  
nextFIFO    searchChNo    talkFIFOData  
CDAQMXDataBuffer::create  
CDAQMXDataBuffer::setDataInfo  
CDAQMXDataBuffer::setDateTime  
CDAQMXStatus::isDataNo
```

CDAQMX100::getDataNum

Syntax

```
int getDataNum(int fifoNo);
```

Parameters

fifoNo Specify the FIFO number.

Description

Gets all types of information fields for each channel of the data member in the remaining number of data of the specified FIFO number.

Returns the minimum value of the channels in the FIFO.

Returns 0 if it does not exist.

Return value

Returns the remaining number of data.

Reference

```
getClassMXDataBuffer  
CDAQMXDataBuffer::getDataNum
```

CDAQMX100::getInstCh

Syntax

```
int getInstCh(int chNo);
```

Parameters

chNo Specify the channel number.

Description

Receives the measured data of the specified channel number.

Gets instantaneous values by channel, and stores them in the data member.

Return value

Returns an error number.

Error:

Not Data Various kinds of information fields for each channel do not exist.

Reference

```
getChData getClassMXDataBuffer getTimeData talkChData
CDAQMXDataBuffer::create
CDAQMXDataBuffer::setDataInfo
CDAQMXDataBuffer::setDateTime
```

CDAQMX100::getInstFIFO

Syntax

```
int getInstFIFO(int fifoNo);
```

Parameters

fifoNo Specify the FIFO number.

Description

Receives the measured data of the specified FIFO number.

Gets instantaneous values by FIFO, and stores them in the data member.

Return value

Returns an error number.

Reference

```
getChData getClassMXDataBuffer getTimeData searchChNo
talkFIFOData
CDAQMXDataBuffer::create
CDAQMXDataBuffer::setDataInfo
CDAQMXDataBuffer::setDateTime
```

CDAQMX100::getItemAll

Syntax

```
int getItemAll(void);
```

Description

Receives setup data and stores it in the data member.
Updates the status and all information data if successful.

Return value

Returns an error number.

Reference

```
getClassMXItemConfig getConfig updateAll
```

CDAQMX100::getRevisionMX100DLL

Syntax

```
static const int getRevisionMX100DLL(void);
```

Description

Gets the revision number of this DLL.

Return value

Returns the revision number of this DLL.

CDAQMX100::getVersionMX100DLL

Syntax

```
static const int getVersionMX100DLL(void);
```

Description

Gets the version number of this DLL.

Return value

Returns the version number of this DLL.

CDAQMX100::initBalance

Syntax

```
int initBalance(void);
```

Description

Executes initial balancing.
Stores the result in the current data of the initial balance data management field of the data member.
Updates the status if successful.

Return value

Returns an error number.

Reference

```
getClassMXBalanceList runBalance updateRenew  
CDAQMXBalanceList::getCurrent
```

CDAQMX100::initDataCh

Syntax

```
void initDataCh(int chNo = DAQMX_CHNO_ALL);
```

Parameters

chNo Specify the channel number.

Description

Initializes the various information of the specified channel.

If the constant for “Specify All Channel numbers” is specified for the channel number, all channels are processed.

Updates the status if successful.

Reference

```
clearLastDataNoCh getClassMXDataBuffer updateRenew
CDAQMXDataBuffer::initialize
```

CDAQMX100::initDataFIFO

Syntax

```
void initDataFIFO(int fifoNo = DAQMX_FIFONO_ALL);
```

Parameters

fifoNo Specify the FIFO number.

Description

Initializes the various information of the channels in the specified FIFO.

If the constant for “Specify all FIFO numbers” is specified for the FIFO numbers, all FIFOs are processed.

Updates the status if successful.

Reference

```
clearLastDataNoFIFO getClassMXDataBuffer updateRenew
CDAQMXDataBuffer::initialize
```

CDAQMX100::initSetValue

Syntax

```
int initSetValue(void);
```

Description

Initializes the system.

Updates the status and all information data if successful.

Return value

Returns an error number.

Reference

```
initSystem updateAll
```

CDAQMX100::isObject

Syntax

```
virtual int isObject(const char * classname = "CDAQMX100");
```

Parameters

classname Specify the class name using a string.

Description

Checks whether the specified class name was inherited.

If parameters are omitted, checks whether it is this class.

Classes that inherit this class must be overridden in order to check their own classes.

Returns true (valid) if the class was inherited. Otherwise, returns false (invalid).

If different from this class, checks the parent class.

Return value

Returns a Boolean value.

Reference

CDAQMX::isObject

CDAQMX100::measClear

Syntax

```
void measClear(void);
```

Description

Initializes the data members below for retrieval of the measured data.

- Setup data field
- Various information fields per channel.

Reference

getClassMXItemConfig
CDAQMXDataBuffer::initialize
CDAQMXItemConfig::initialize

CDAQMX100::measDataCh

Syntax

```
int measDataCh(int chNo = DAQMX100_CHNO_ALL);
```

Parameters

chNo Specify the channel number.

Description

Increments the current measurement point of the measured data of the specified channel number just one point.

If the constant for “Specify All Channels” is specified for the channel numbers, all channels are processed.

Updates the status if communication executed.

Return value

Returns an error number.

Reference

getDataCh updateRenew

CDAQMX100::measDataFIFO

Syntax

```
int measDataFIFO(int fifoNo = DAQMX100_FIFONO_ALL);
```

Parameters

fifoNo Specify the FIFO number.

Description

Increments the current measurement point of the measured data of the channels of the specified FIFO number just one point.

If the constant for “Specify all FIFO numbers” is specified for the FIFO numbers, all FIFOs are processed.

Updates the status if communication executed.

Return value

Returns an error number.

Reference

getDataFIFO updateRenew

CDAQMX100::measInstCh

Syntax

```
int measInstCh(int chNo = DAQMX100_CHNO_ALL);
```

Parameters

chNo Specify the channel number.

Description

Receives the measured data of the specified channel number.
Gets instantaneous values by channel, and stores them in the data member.
If the constant for “Specify All Channels” is specified for the channel numbers, all channels are processed.
Updates the status if successful.

Return value

Returns an error number.

Reference

getInstCh updateRenew

CDAQMX100::measInstFIFO

Syntax

```
int measInstFIFO(int fifoNo = DAQMX100_FIFONO_ALL);
```

Parameters

fifoNo Specify the FIFO number.

Description

Receives the measured data of the specified FIFO number.
Gets instantaneous values by FIFO, and stores them in the data member.
If the constant for “Specify all FIFO numbers” is specified for the FIFO numbers, all FIFOs are processed.
Updates the status if successful.

Return value

Returns an error number.

Reference

getInstFIFO updateRenew

CDAQMX100::measStart

Syntax

```
int measStart(void);
```

Description

Starts data acquisition.

Starts the FIFO.

Updates the status if successful.

Return value

Returns an error number.

Reference

startFIFO updateRenew

CDAQMX100::measStop

Syntax

```
int measStop(void);
```

Description

Stops data acquisition.

Stops the FIFO.

Updates the status if successful.

Return value

Returns an error number.

Reference

stopFIFO updateRenew

CDAQMX100::nextFIFO

Syntax

```
int nextFIFO(int fifoNo);
```

Parameters

fifoNo Specify the FIFO number.

Description

Increments current index number of the channels of the specified FIFO number.

Returns the increment result as a Boolean value. If a channel is incremented and no data is present, Invalid is returned.

Return value

Returns a Boolean value.

Reference

getClassMXDataBuffer
CDAQMXDataBuffer::next

CDAQMX100::open

Syntax

```
virtual int open(const char * strAddress, unsigned int uiPort  
= DAQMX_COMMPORT);
```

Parameters

strAddress	Specify the IP address as a string.
uiPort	Specify the port number.

Description

Connects to the device with the IP address and port number specified by the parameters.

The port number can be omitted. If omitted, it is set to the communication constant, "MX100 communication port number."

When initializing the data member for retrieval of the measured data and connection is successful, those items are retrieved and stored.

Sets the communication timeout to 3 minutes (the time after which the measuring instrument automatically closes the connection).

Return value

Returns an error number.

Reference

```
close measClear setTimeOut updateAll  
CDAQMX::open
```

CDAQMX100::reconstruct

Syntax

```
int reconstruct(void);
```

Description

Reconfigures the system.

Updates the status and all information data if successful.

Return value

Returns an error number.

Reference

```
initSystem updateAll
```

CDAQMX100::reloadBalance

Syntax

```
int reloadBalance(int idBalance);
int reloadBalance(CDAQMXBalanceData & cMXBalanceData);
```

Parameters

idBalance	Specify the initial balance data identifier.
cMXBalanceData	Specify initial balance data.

Description

Sends specified initial balance data.
Updates the status if successful.

Return value

Returns an error number.
Error:
Not Data There is no data.

Reference

```
getClassMXBalanceList setBalance updateRenew
CDAQMXBalanceList::getClassMXBalanceData
```

CDAQMX100::sendConfig

Syntax

```
int sendConfig(void);
```

Description

Sends the setup data field of the data member.

Return value

Returns an error number.

Reference

```
setItemAll
```

CDAQMX100::setAlarm

Syntax

```
int setAlarm(int chNo, int levelNo, int iAlarmType, double valueON, double valueOFF);  
int setAlarm(int chNo, int levelNo, int iAlarmType = DAQMX_ALARM_NONE, int valueON = 0, int valueOFF = 0);
```

Parameters

chNo	Specify the channel number.
levelNo	Specify the alarm level.
iAlarmType	Specify the alarm type.
valueON	Specify the threshold level (ON value) for alarm activation.
valueOFF	Specify the threshold level (OFF value) for alarm termination.

Description

Sets the alarm to the alarm level of the specified channel.
Changes the setup data fields of the data member and sends them collectively.
If the constant for “Specify All Channels” is specified for the channel numbers, all channels are processed.
If the alarm level is set to the constant for “Specify all alarm level numbers,” all alarm levels are processed.

Return value

Returns an error number.

Reference

```
getClassMXItemConfig setItemAll  
CDAQMXChConfig::getPoint  
CDAQMXChConfig::setAlarmValue  
CDAQMXItemConfig::getClassMXChConfig
```

CDAQMX100::setBurnout

Syntax

```
int setBurnout(int chNo, int iBurnout);
```

Parameters

chNo	Specify the channel number.
iBurnout	Specify the burnout type.

Description

Sets the burnout type to the channels of the specified channel numbers.
Changes the setup data fields of the data member and sends them collectively.
If the constant for “Specify All Channels” is specified for the channel numbers, all channels are processed.

Return value

Returns an error number.

Reference

```
getClassMXItemConfig setItemAll  
CDAQMXChConfig::setBurnout  
CDAQMXItemConfig::getClassMXChConfig
```

CDAQMX100::setCFWriteMode

Syntax

```
int setCFWriteMode(int iCFWriteMode);
```

Parameters

iCFWriteMode Specify the CF write mode.

Description

Sets the CF write mode.

Changes the setup data fields of the data member and sends them collectively.

Return value

Returns an error number.

Reference

```
getClassMXItemConfig setItemAll
CDAQMXItemConfig::getClassMXSysInfo
CDAQMXSysInfo::setCFWriteMode
```

CDAQMX100::setChComment

Syntax

```
int setChComment(int chNo, const char * strComment);
```

Parameters

chNo Specify the channel number.

strComment Specify the comment.

Description

Sets the comment on the channel of the specified channel number.

Changes the setup data fields of the data member and sends them collectively.

If the constant for "Specify All Channels" is specified for the channel numbers, all channels are processed.

Return value

Returns an error number.

Reference

```
getClassMXItemConfig setItemAll
CDAQMXChConfig::setComment
CDAQMXItemConfig::getClassMXChConfig
```

CDAQMX100::setChDELTA

Syntax

```
int setChDELTA(int chNo, int refChNo, int iRange =  
DAQMX_RANGE_REFERENCE);
```

Parameters

chNo	Specify the channel number.
refChNo	Specify the channel number of the reference channel.
iRange	Specify the range type for the input of the target channel.

Description

Sets the difference computation for the specified reference channels.

Channel settings outside the range take default values.

Changes the setup data fields of the data member and sends them collectively.

If the constant for “Specify All Channels” is specified for the channel numbers, all channels are processed.

If the range type is set to “Reference Channel” the range of the channel number that references the target channel’s measurement range is applied.

Return value

Returns an error number.

Reference

```
getClassMXItemConfig setItemAll  
CDAQMXItemConfig::getClassMXSysInfo  
CDAQMXItemConfig::setDELTA  
CDAQMXSysInfo::getModuleType
```

CDAQMX100::setChKind

Syntax

```
int setChKind(int chNo, int iKind, int refChNo =
DAQMX_REFCHNO_NONE);
```

Parameters

chNo	Specify the channel number.
iKind	Specify the channel type.
refChNo	Specify the reference channel number.

Description

Sets the channel type on the channel of the specified channel number.

The channel settings are set to the default values.

Changes the setup data fields of the data member and sends them collectively.

If the constant for “Specify All Channels” is specified for the channel numbers, all channels are processed.

The reference channel number is valid if the channel type is AI (difference between channels), DI (difference between channels), AI (remote RJC), AO (transmission output), or PWM (transmission output).

Return value

Returns an error number.

Reference

```
getClassMXItemConfig setItemAll
CDAQMXItemConfig::setAOType
CDAQMXItemConfig::setDELTA
CDAQMXItemConfig::setDI
CDAQMXItemConfig::setDOType
CDAQMXItemConfig::setPWMTType
CDAQMXItemConfig::setRRJC
CDAQMXItemConfig::setSKIP
CDAQMXItemConfig::setVOLT
```

CDAQMX100::setChoice

Syntax

```
int setChoice(int outputNo, int idleChoice, int errorChoice,  
int presetValue);  
int setChoice(int outputNo, int idleChoice, int errorChoice,  
double presetValue);
```

Parameters

outputNo	Specify the output channel data number.
idleChoice	Specify the selected value when idling.
errorChoice	Specify the selected value when an error occurs.
presetValue	Specify the value if the selected value is the “specified value.”

Description

Sets the selection on the output channel data of the specified output channel data number.

Changes the setup data fields of the data member and sends them collectively.

If the constant for “Specify All output data numbers” is specified for the output data number, all channels are processed.

Return value

Returns an error number.

Reference

```
getClassMXItemConfig setItemAll  
CDAQMXItemConfig::getClassMXOutputData  
CDAQMXItemConfig::getRangePoint  
CDAQMXOutputData::setChoice
```

CDAQMX100::setChRRJC

Syntax

```
int setChRRJC(int chNo, int refChNo);
```

Parameters

chNo	Specify the channel number.
refChNo	Specify the reference channel number.

Description

Sets the remote RJC for the specified reference channel.

Channel settings outside the range take default values.

Changes the setup data fields of the data member and sends them collectively.

If the constant for “Specify All Channels” is specified for the channel numbers, all channels are processed.

Return value

Returns an error number.

Reference

```
getClassMXItemConfig setItemAll  
CDAQMXItemConfig::setRRJC
```

CDAQMX100::setChTag

Syntax

```
int setChTag(int chNo, const char * strTag);
```

Parameters

chNo	Specify the channel number.
strTag	Specify the tag.

Description

Sets the tag on the channel of the specified channel number.

Changes the setup data fields of the data member and sends them collectively.

If the constant for “Specify All Channels” is specified for the channel numbers, all channels are processed.

Return value

Returns an error number.

Reference

```
getClassMXItemConfig setItemAll
CDAQMXChConfig::setTag
CDAQMXItemConfig::getClassMXChConfig
```

CDAQMX100::setChUnit

Syntax

```
int setChUnit(int chNo, const char * strUnit);
```

Parameters

chNo	Specify the channel number.
strUnit	Specify the unit name.

Description

Sets the unit name on the channel of the specified channel number.

Changes the setup data fields of the data member and sends them collectively.

If the constant for “Specify All Channels” is specified for the channel numbers, all channels are processed.

Return value

Returns an error number.

Reference

```
getClassMXItemConfig setItemAll
CDAQMXChConfig::setUnit
CDAQMXItemConfig::getClassMXChConfig
```


CDAQMX100::setDateTime

Syntax

```
virtual int setDateTime(CDAQMXDateTime * pcMXDateTime = NULL);
```

Parameters

pcMXDateTime Specify the time information data.

Description

Sets time information data on the device.

Updates the status if successful.

Return value

Returns an error number.

Reference

updateRenew
CDAQMX::setDateTime

CDAQMX100::setDeenergize

Syntax

```
int setDeenergize(int doNo, int bDeenergize);
```

Parameters

doNo Specify the data number.

bDeenergize Specify de-energize using a Boolean value.

Description

Sets de-energize on the specified DO data number of the specified channel.

Changes the setup data fields of the data member and sends them collectively.

If the constant for "Specify All Channels" is specified for the channel numbers, all channels are processed.

Return value

Returns an error number.

Reference

getClassMXItemConfig setItemAll
CDAQMXChConfig::setDeenergize
CDAQMXItemConfig::getClassMXChConfig

CDAQMX100::setFilter

Syntax

```
int setFilter(int chNo, int iFilter);
```

Parameters

chNo	Specify the channel number.
iFilter	Specify the filter coefficient.

Description

Sets the filter coefficient on the channel of the specified channel number.

Changes the setup data fields of the data member and sends them collectively.

If the constant for “Specify All Channels” is specified for the channel numbers, all channels are processed.

Return value

Returns an error number.

Reference

```
getClassMXItemConfig setItemAll
CDAQMXChConfig::setFilter
CDAQMXItemConfig::getClassMXChConfig
```

CDAQMX100::setHisterisys

Syntax

```
int setHisterisys(int chNo, int levelNo, double histerisys);
int setHisterisys(int chNo, int levelNo, int histerisys = 0);
```

Parameters

chNo	Specify the channel number.
levelNo	Specify the alarm level.
histerisys	Specify the hysteresis.

Description

Sets the hysteresis for the alarm level of the specified channel number.

Changes the setup data fields of the data member and sends them collectively.

If the constant for “Specify All Channels” is specified for the channel numbers, all channels are processed.

If the alarm level is set to the constant for “Specify all alarm level numbers,” all alarm levels are processed.

Return value

Returns an error number.

Reference

```
getClassMXItemConfig setItemAll
CDAQMXChConfig::getAlarmType
CDAQMXChConfig::getAlarmValueON
CDAQMXChConfig::getPoint
CDAQMXChConfig::setAlarm
CDAQMXItemConfig::getClassMXChConfig
```

CDAQMX100::setHold

Syntax

```
int setHold(int doNo, int bHold);
```

Parameters

doNo	Specify the data number.
bHold	Specify hold using a Boolean value.

Description

Sets hold on the specified DO data number of the specified channel.
Changes the setup data fields of the data member and sends them collectively.
If the constant for “Specify All Channels” is specified for the channel numbers, all channels are processed.

Return value

Returns an error number.

Reference

```
getClassMXItemConfig setItemAll  
CDAQMXChConfig::setHold  
CDAQMXItemConfig::getClassMXChConfig
```

CDAQMX100::setIntegral

Syntax

```
int setIntegral(int moduleNo, int iHz);
```

Parameters

moduleNo	Specify the module number.
iHz	Specify the type of A/D integration time.

Description

Sets the A/D integration time type on the module of the specified module number.
Changes the setup data fields of the data member and sends them collectively.
If the constant for “Specify all module numbers” is specified for the module numbers, all modules are processed.

Return value

Returns an error number.

Reference

```
getClassMXItemConfig setItemAll  
CDAQMXItemConfig::getClassMXSysInfo  
CDAQMXItemConfig::setInterval  
CDAQMXSysInfo::getInterval
```

CDAQMX100::setInterval

Syntax

```
int setInterval(int moduleNo, int iInterval);
```

Parameters

moduleNo	Specify the module number.
iInterval	Specify the interval type.

Description

Sets the interval type on the module of the specified module number.

Changes the setup data fields of the data member and sends them collectively.

If the constant for “Specify all module numbers” is specified for the module numbers, all modules are processed.

Return value

Returns an error number.

Reference

```
getClassMXItemConfig setItemAll
CDAQMXItemConfig::getClassMXSysInfo
CDAQMXItemConfig::setInterval
CDAQMXSysInfo::getIntegral
```

CDAQMX100::setItemAll

Syntax

```
int setItemAll(void);
```

Description

Sends the setup data field of the data member.

Updates the status and all information data if successful.

Return value

Returns an error number.

Reference

```
getClassMXItemConfig setConfig updateAll
```

CDAQMX100::setOutputType

Syntax

```
int setOutputType(int outputNo, int iOutput);
```

Parameters

outputNo	Specify the output channel data number.
iOutput	Specify the output type.

Description

Sets the output type on the output channel data of the specified output channel data number.

The settings other than the output type are set to the default values.

Changes the setup data fields of the data member and sends them collectively.

If the constant for “Specify All output data numbers” is specified for the output data number, all channels are processed.

Return value

Returns an error number.

Reference

```
getClassMXItemConfig setItemAll  
CDAQMXItemConfig::setAO  
CDAQMXItemConfig::setPWM
```

CDAQMX100::setPulseTime

Syntax

```
int setPulseTime(int outputNo, int pulseTime);
```

Parameters

outputNo	Specify the output channel data number.
pulserTime	Specify the integer multiple of the pulse interval.

Description

Sets the pulse integer multiple on the output channel data of the specified output channel data number.

Changes the setup data fields of the data member and sends them collectively.

If the constant for “Specify All output data numbers” is specified for the output data number, all channels are processed.

Return value

Returns an error number.

Reference

```
getClassMXItemConfig setItemAll  
CDAQMXItemConfig::getClassMXOutputData  
CDAQMXOutputData::setPulseTime
```

CDAQMX100::setRange

Syntax

```
int setRange(int chNo, int iRange);
```

Parameters

chNo	Specify the channel number.
iRange	Specify the range type.

Description

Sets the range.

For the contact range and SKIP range types, see the definitions for the new constants.

Channel settings outside the range take default values.

Changes the setup data fields of the data member and sends them collectively.

If the constant for “Specify All Channels” is specified for the channel numbers, all channels are processed.

Return value

Returns an error number.

Reference

```
getClassMXItemConfig setItemAll
CDAQMXItemConfig::setAO
CDAQMXItemConfig::setDI
CDAQMXItemConfig::setPWM
CDAQMXItemConfig::setRES
CDAQMXItemConfig::setRTD
CDAQMXItemConfig::setSKIP
CDAQMXItemConfig::setSTRAIN
CDAQMXItemConfig::setTC
CDAQMXItemConfig::setVOLT
```

CDAQMX100::setRefAlarm

Syntax

```
int setRefAlarm(int doNo, int refChNo, int levelNo, int  
bValid);
```

Parameters

doNo	Specify the data number.
refChNo	Specify the reference channel number.
levelNo	Specify the alarm level.
bValid	Specify the Boolean value.

Description

Sets the reference alarm on the specified DO data number of the specified channel. The reference alarm is specified by reference channel number and alarm level. If the constant for “Specify All Channels” is specified for the channel numbers, all channels are processed. If the alarm level is set to the constant for “Specify all alarm level numbers,” all alarm levels are processed.

Return value

Returns an error number.

Reference

```
getClassMXItemConfig setItemAll  
CDAQMXChConfig::setRefAlarm  
CDAQMXItemConfig::getClassMXChConfig
```

CDAQMX100::setRJCType

Syntax

```
int setRJCType(int chNo, int iRJCType, int volt = 0);
```

Parameters

chNo	Specify the channel number.
iRJCType	Specify the RJC type.
volt	Specify the RJC voltage.

Description

Sets the RJC type on the channel of the specified channel number. Changes the setup data fields of the data member and sends them collectively. If the constant for “Specify All Channels” is specified for the channel numbers, all channels are processed.

Return value

Returns an error number.

Reference

```
getClassMXItemConfig setItemAll  
CDAQMXChConfig::setRJCType  
CDAQMXItemConfig::getClassMXChConfig
```

CDAQMX100::setScale

Syntax

```
int setScale(int chNo, double scaleMin, double scaleMax, int
scalePoint);
int setScale(int chNo, int scaleMin = 0, int scaleMax = 0, int
scalePoint = 0);
```

Parameters

chNo	Specify the channel number.
scaleMin	Specify the scale minimum.
scaleMax	Specify the scale maximum.
scalePoint	Specify the decimal point position.

Description

Sets the scale on the channel of the specified channel number.
Changes the setup data fields of the data member and sends them collectively.
If the constant for “Specify All Channels” is specified for the channel numbers, all channels are processed.

Return value

Returns an error number.

Reference

```
getClassMXItemConfig setItemAll
CDAQMXItemConfig::setScale
```

CDAQMX100::setSpan

Syntax

```
int setSpan(int chNo, double spanMin, double spanMax);
int setSpan(int chNo, int spanMin = 0, int spanMax = 0);
```

Parameters

chNo	Specify the channel number.
spanMin	Specify the span minimum.
spanMax	Specify the span maximum.

Description

Sets the span on the channel of the specified channel number.
Changes the setup data fields of the data member and sends them collectively.
If the constant for “Specify All Channels” is specified for the channel numbers, all channels are processed.

Return value

Returns an error number.

Reference

```
getClassMXItemConfig setItemAll
CDAQMXChConfig::setSpan
CDAQMXItemConfig::getClassMXChConfig
CDAQMXItemConfig::getClassMXSysInfo
CDAQMXItemConfig::getSpanPoint
CDAQMXSysInfo::getTempUnit
```


CDAQMX100::setUnitNo

Syntax

```
int setUnitNo(int unitNo);
```

Parameters

unitNo Specify the unit number.

Description

Sets the unit number.

Changes the setup data fields of the data member and sends them collectively.

Return value

Returns an error number.

Reference

```
getClassMXItemConfig setItemAll  
CDAQMXItemConfig::getClassMXSysInfo  
CDAQMXSysInfo::setUnitNo
```

CDAQMX100::setUnitTemp

Syntax

```
int setUnitTemp(int iTempUnit);
```

Parameters

iTempUnit Specify the temperature unit type.

Description

Sets the temperature unit type.

Changes the setting values of the channels affected.

Changes the setup data fields of the data member and sends them collectively.

Return value

Returns an error number.

Reference

```
getClassMXItemConfig setItemAll  
CDAQMXItemConfig::setTempUnit
```

CDAQMX100::switchBackup

Syntax

```
int switchBackup(int bBackup);
```

Parameters

bBackup Specify backup using a Boolean value.

Description

Switches backup.

Updates the status if successful.

Return value

Returns an error number.

Reference

setBackup updateRenew

CDAQMX100::switchDO

Syntax

```
int switchDO(int idDO, int bONOFF);
```

Parameters

idDO Specify the DO data identifier.

bONOFF Specify ON/OFF using a Boolean value.

Description

Sends the specified DO data.

Changes the valid channels of the DO data to the specified ON/OFF value and sends them.

Updates the status if successful.

Return value

Returns an error number.

Error:

Not Data There is no data.

Reference

commandDO getClassMXDOList
CDAQMXDOData::setDOONOFF
CDAQMXDOList::getClassMXDOData

CDAQMX100::switchTransmit

Syntax

```
int switchTransmit(int idTrans, int iTransmit);
```

Parameters

idTrans	Specify the transmission output data identifier.
iTransmit	Specify the transmission status.

Description

Sends the specified transmission output data.

Changes all channels of the transmission output data to the specified transmission status and sends them.

Updates the status if successful.

Return value

Returns an error number.

Error:

Not Data	There is no data.
----------	-------------------

Reference

```
commandTransmit getClassMXTransmitList  
CDAQMXTransmit::setTransmit  
CDAQMXTransmitList::getClassMXTransmit
```

CDAQMX100::toChNo

Syntax

```
int toChNo(int fifoNo, int fifoIndex);
```

Parameters

fifoNo	Specify the FIFO number.
fifoIndex	Returns the channel sequence number in the FIFO.

Description

Gets the channel number from the specified information.

Returns 0 if it does not exist.

Return value

Returns the channel number.

Reference

```
searchChNo
```

CDAQMX100::updateAll

Syntax

```
int updateAll(void);
```

Description

Updates all status and information data.

Gets the following information and stores it in the data member.

- Setup data
- Channel information data
- Status of randomly changing instrument

Return value

Returns an error number.

Reference

updateConfig updateInfoCh updateRenew

CDAQMX100::updateAOPWMDData

Syntax

```
int updateAOPWMDData(void);
```

Description

Gets the AO/PWM data and transmission output data and stores it in the data member.

Stores it in the current data fields of the AO/PWM data management field and the transmission output management field.

If the communication packet version is not supported by the instrument, the process concludes normally.

Return value

Returns an error number.

Reference

```
getAOPWMDData getClassMXAOPWMList getClassMXTransmitList
getPacketVersion
CDAQMXAOPWMList::getCurrent
CDAQMXTransmitList::getCurrent
```

CDAQMX100::updateBalance

Syntax

```
int updateBalance(void);
```

Description

Gets initial balance data and stores it in the data member.

Stores it in the current data of the initial balance data management field.

Also copies to the initial balance data of the setup data field.

If the communication packet version is not supported by the instrument, the process concludes normally.

Return value

Returns an error number.

Reference

```
getBalance getClassMXBalanceList getClassMXItemConfig  
getPacketVersion  
CDAQMXBalanceList::getCurrent  
CDAQMXItemConfig::getClassMXBalanceData
```

CDAQMX100::updateConfig

Syntax

```
int updateConfig(void);
```

Description

Receives setup data and stores it in the data member.

Copies initial balance data to the current data of the initial balance data management field.

Return value

Returns an error number.

Reference

```
getClassMXBalanceList getClassMXItemConfig getConfig  
CDAQMXBalanceList::getCurrent  
CDAQMXItemConfig::getClassMXBalanceData
```

CDAQMX100::updateDOData

Syntax

```
int updateDOData(void);
```

Description

Gets DO data and stores it in the data member.

Stores it in the current data of the DO data management field.

Return value

Returns an error number.

Reference

```
getClassMXDOList getDOData  
CDAQMXDOList::getCurrent
```

CDAQMX100::updateInfoCh

Syntax

```
int updateInfoCh(int chNo = DAQMX_CHNO_ALL);
```

Parameters

chNo Specify the channel number.

Description

Gets the channel information data of the specified channel number and stores it in the data member.

If the constant for “Specify All Channels” is specified for the channel numbers, all channels are processed.

Return value

Returns an error number.

Reference

```
getChInfo getClassMXDataBuffer talkChInfo
CDAQMXDataBuffer::setChInfo
```

CDAQMX100::updateOutput

Syntax

```
int updateOutput(void);
```

Description

Gets the output channel data and stores it in the data member.

Return value

Returns an error number.

Reference

```
getClassMXItemConfig getOutput
CDAQMXItemConfig::getClassMXOutputData
```

CDAQMX100::updateRenew

Syntax

```
int updateRenew(void);
```

Description

Updates the status.

Gets the following status of the randomly changing instrument and stores it in the data member.

- Status data
- Current DO data
- Current AO/PWM data and transmission output data.

Return value

Returns an error number.

Reference

```
updateAOPWMData updateDOData updateStatus
```

CDAQMX100::updateStatus

Syntax

```
int updateStatus(void);
```

Description

Gets status data and stores it in the data member.

Return value

Returns an error number.

Reference

```
getClassMXItemConfig getStatusData  
CDAQMXItemConfig::getClassMXStatus
```

CDAQMX100::updateSystem

Syntax

```
int updateSystem(void);
```

Description

Gets system configuration data and stores it in the data member.

Return value

Returns an error number.

Reference

```
getClassMXItemConfig getSystemConfig  
CDAQMXItemConfig::getClassMXSysInfo
```

CDAQMX100::userClear

Syntax

```
void userClear(void);
```

Description

Initializes the current data from the following data of the management field.

- DO data management field
- AO/PWM data management field
- Transmission output data management field
- Initial balance data management field

Reference

```
getClassMXAOPWMList getClassMXBalanceList getClassMXDOList  
getClassMXTransmitList  
CDAQMXAOPWMList::initCurrent  
CDAQMXBalanceList::initCurrent  
CDAQMXDOList::initCurrent  
CDAQMXTransmitList::initCurrent
```

CDAQMX100::userDoubleAOPWMValue

Syntax

```
double userDoubleAOPWMValue(int idAOPWM, int aopwmNo);
```

Parameters

idAOPWM	Specify the AO/PWM data identifier.
aopwmNo	Specify the AO/PWM data number.

Description

Gets, as actual output values, the output data values of the specified AO/PWM data number from the AO/PWM data of the specified AO/PWM data identifier of the AO/PWM data management field of the data member.

Returns 0.0 if it does not exist.

Return value

Returns the actual output value.

Reference

```
getClassMXAOPWMList getClassMXItemConfig getOutputRange  
CDAQMXAOPWMList::getClassMXAOPWMData  
CDAQMXAOPWMData::getAOPWMValue  
CDAQMXAOPWMData::toRealValue
```


CDAQMXAOPWMList Class

CDAQMXList
 CDAQMXAOPWMList

This class is for managing AO/PWM data from the output of command AO/PWM. You can create the AO/PWM data to output in advance and store it. The data is labeled with indentifiers.

Public Members

Construct/Destruct

CDAQMXAOPWMList	Constructs an object.
~CDAQMXAOPWMList	Destructs an object.

Member Data Manipulation

add	Adds AO/PWM data.
change	Changes AO/PWM data.
copyData	Copies AO/PWM data.
getClassMXAOPWMLData	Gets AO/PWM data.
initCurrent	Initializes the current AO/PWM data.
getCurrent	Gets the current AO/PWM data.

- **Overridden Members**

Member Data Manipulation

create	Creates AO/PWM data.
copy	Copies AO/PWM data.

- **Inherited Members**

See CDAQMXList.
 del
 getMaxNo
 getNum
 initialize
 isData

Protected Members

Data Members

m_cCurrent	Field for storing the current AO/PWM data.
------------	--

Inherited Members

See CDAQMXList.
 m_list
 m_num
 addData
 DelData
 getData

Private Members

None.

Member Functions (Alphabetical Order)

CDAQMXAOPWMList::add

Syntax

```
int add(CDAQMXAOPWMLData * pcMXAOPWMLData);
```

Parameters

pcMXAOPWMLData Specify the data using a pointer.

Description

Adds the specified data to the list and generates data identifiers.
Returns a negative value if not added.

Return value

Returns the data identifier.

Reference

addData

CDAQMXAOPWMList::CDAQMXAOPWMList

Syntax

```
CDAQMXAOPWMList(void);  
virtual ~CDAQMXAOPWMList(void);
```

Description

Constructs or destructs an object.
When constructing, the data member is initialized.
When destructing, the data in the list is deleted.

Reference

initCurrent
CDAQMXList::CDAQMXList

CDAQMXAOPWMList::change

Syntax

```
void change(int idAOPWM, int aopwmNo, int bValid, int  
iAOPWMValue = 0);
```

Parameters

idAOPWM	Specify the AO/PWM data identifier.
aopwmNo	Specify the AO/PWM data number.
bValid	Specify valid/invalid using a Boolean value.
iAOPWMValue	Specify the output data value.

Description

Changes the AO/PWM data of the specified AO/PWM data identifier.

If the data identifier is set to the constant for “Specify all data identifiers,” all data in the list is processed.

If AO/PWM data is set to the constant for “Specify all AO/PWM data numbers,” all numbers within the data are processed.

Reference

```
getClassMXAOPWMData  
CDAQMXAOPWMData::setAOPWM
```

CDAQMXAOPWMList::copy

Syntax

```
virtual void copy(int idxNo, int idxSrc);
```

Parameters

idxNo	Specify the data identifier of the copy destination.
idxSrc	Specify the data identifier of the copy source.

Description

Copies the contents of the data indicated by the data identifier from the copy source to the copy destination.

Reference

```
copyData  
getClassMXAOPWMData
```

CDAQMXAOPWMList::copyData

Syntax

```
void copyData(int idAOPWM, CDAQMXAOPWMData * pcMXAOPWMData);
```

Parameters

idAOPWM Specify the AO/PWM data identifier.
pcMXAOPWMData Specify the data using a pointer.

Description

Copies the data specified by the pointer to the data of the specified data identifier. If the data identifier is set to the constant for “Specify all data identifiers,” all data in the list is processed.

Reference

getClassMXAOPWMData

CDAQMXAOPWMList::create

Syntax

```
virtual int create(void);
```

Description

Creates data and adds it to the list.

Return value

Returns the data identifier.

Reference

add
CDAQMXAOPWMData::CDAQMXAOPWMData

CDAQMXAOPWMList::getClassMXAOPWMData

Syntax

```
CDAQMXAOPWMData * getClassMXAOPWMData(int idAOPWM);
```

Parameters

idAOPWM Specify the AO/PWM data identifier.

Description

Gets the data of the specified data identifier.

If the data identifier is set to the constant for “specify current data,” gets the current data field from the data member.

Returns NULL if it does not exist.

Return value

Returns a pointer to the data.

Reference

getCurrent getData

CDAQMXAOPWMList::getCurrent

Syntax

```
CDAQMXAOPWMList & getCurrent(void);
```

Description

Gets the current data field from the data member.

Return value

Returns a reference to the data.

CDAQMXAOPWMList::initCurrent

Syntax

```
void initCurrent(void);
```

Description

Initializes the current data field of the data member.

Reference

```
getCurrent  
CDAQMXAOPWMList::initialize
```

CDAQMXBalanceList Class

CDAQMXList
CDAQMXBalanceList

This class manages initial balance data.

You can create the initial balance data to send in advance and store it. The data is identified by an identifier.

You can also store the results of the current data.

Public Members

Construct/Destruct

CDAQMXBalanceList	Constructs an object.
~CDAQMXBalanceList	Destructs an object.

Member Data Manipulation

add	Adds initial balance data.
change	Changes initial balance data.
copyData	Copies initial balance data.
getClassMXBalanceData	Gets initial balance data.
initCurrent	Initializes the current initial balance data.
getCurrent	Gets the current initial balance data.

- **Overridden Members**

Member Data Manipulation

create	Creates initial balance data.
copy	Copies initial balance data.

- **Inherited Members**

See CDAQMXList.
del
getMaxNo
getNum
initialize
isData

Protected Members

Data Members

m_cCurrent	Field for storing the current initial balance data.
------------	---

Inherited Members

See CDAQMXList.
m_list
m_num
addData
delData
getData

Private Members

None.

Member Functions (Alphabetical Order)

CDAQMXBalanceList::add

Syntax

```
int add(CDAQMXBalanceData * pcMXBalanceData);
```

Parameters

pcMXBalanceData Specify the data using a pointer.

Description

Adds the specified data to the list and generates data identifiers.
Returns a negative value if not added.

Return value

Returns the data identifier.

Reference

addData

CDAQMXBalanceList::CDAQMXBalanceList

Syntax

```
CDAQMXBalanceList(void);  
virtual ~CDAQMXBalanceList(void);
```

Description

Constructs or destructs an object.
When constructing, the data member is initialized.
When destructing, the data in the list is deleted.

Reference

```
initCurrent  
CDAQMXList::CDAQMXList
```

CDAQMXBalanceList::change

Syntax

```
void change(int idBalance, int balanceNo, int bValid, int
iValue = 0);
```

Parameters

idBalance	Specify the initial balance data identifier.
balanceNo	Specify the initial balance data number.
bValid	Specify valid/invalid using a Boolean value.
iValue	Specify the initial balance value.

Description

Changes the initial balance data of the specified initial balance data identifier.

If the data identifier is set to the constant for “Specify all data identifiers,” all data in the list is processed.

If the initial balance data number is set to the constant for “Specify all initial balance numbers,” all items within the data are changed.

Reference

```
getClassMXBalanceData
CDAQMXBalanceData::setBalance
```

CDAQMXBalanceList::copy

Syntax

```
virtual void copy(int idxNo, int idxSrc);
```

Parameters

idxNo	Specify the data identifier of the copy destination.
idxSrc	Specify the data identifier of the copy source.

Description

Copies the contents of the data indicated by the data identifier from the copy source to the copy destination.

Reference

```
copyData
getClassMXBalanceData
```

CDAQMXBalanceList::copyData

Syntax

```
void copyData(int idBalance, CDAQMXBalanceData *  
pcMXBalanceData);
```

Parameters

idBalance Specify the initial balance data identifier.
pcMXBalanceData Specify the data using a pointer.

Description

Copies the data specified by the pointer to the data of the specified data identifier.
If the data identifier is set to the constant for “Specify all data identifiers,” all data in
the list is processed.

Reference

getClassMXBalanceData

CDAQMXBalanceList::create

Syntax

```
virtual int create(void);
```

Description

Creates data and adds it to the list.

Return value

Returns the data identifier.

Reference

add
CDAQMXBalanceData::CDAQMXBalanceData

CDAQMXBalanceList::getClassMXBalanceData

Syntax

```
CDAQMXBalanceData * getClassMXBalanceData(int idBalance);
```

Parameters

idBalance Specify the initial balance data identifier.

Description

Gets the data of the specified data identifier.
If the data identifier is set to the constant for “specify current data,” gets the current
data field from the data member.
Returns NULL if it does not exist.

Return value

Returns a pointer to the data.

Reference

getCurrent
getData

CDAQMXBalanceList::getCurrent

Syntax

```
CDAQMXBalanceResult & getCurrent(void);
```

Description

Gets the current data field from the data member.

Return value

Returns a reference to the data.

CDAQMXBalanceList::initCurrent

Syntax

```
void initCurrent(void);
```

Description

Initializes the current data field of the data member.

Reference

```
getCurrent  
CDAQMXBalanceResult::initialize
```

CDAQMXDataBuffer Class

This class stores each type of information for each channel of the MX100 in a group. You can store multiple data with the FIFO. The data is managed in a list. It is specified with a storage location index number. You can indicate the current location using the current index number.

The following data can be stored.

- Channel Information Data
- Measured Data
- Time information data

Public Members

Construct/Destruct

CDAQMXDataBuffer	Constructs an object.
~CDAQMXDataBuffer	Destructs an object.

Member Data Manipulation

initialize	Initializes the data member.
getClassMXChInfo	Gets the channel information data.
create	Creates the storage field.
setChInfo	Stores the channel information data.
setDataInfo	Stores the measured data.
setDateTime	Stores the time information data.
currentDataInfo	Gets the current measured data.
currentDateTime	Gets the current time information data.

Utilities

next	Increments the current index number.
getDataNum	Gets the remaining number of data.
isCurrent	Checks whether the current data exists.

Data Members

m_cMXChInfo	Field for storing the channel information data.
m_pDataBuf list.	Field for the pointer to the top of the measured data
m_pTimeBuf information data list.	Field for the pointer to the top of the time
m_cur	Field for storing the current index number.
m_max	Field for storing the number of valid elements.
m_num	Field for storing the number of list elements.

Member Data Manipulation

getDataInfo	Gets the measured data.
getDateTime	Gets the time information data.

Private Members

None.

Member Functions (Alphabetical Order)

CDAQMXDataBuffer::CDAQMXDataBuffer

Syntax

```
CDAQMXDataBuffer(void);
CDAQMXDataBuffer(CDAQMXChInfo & cMXChInfo);
virtual ~CDAQMXDataBuffer(void);
```

Parameters

cMXChInfo Specify the channel information data.

Description

Constructs or destructs an object.

When constructing, the data member is initialized. If a channel information data is specified in the parameter, it is copied to the data member.

When destructing, each type of information in the data member is destructed.

Reference

```
getClassMXChInfo initialize setChInfo
CDAQMXChInfo::initialize
```

CDAQMXDataBuffer::create

Syntax

```
int create(int num);
```

Parameters

num Specify the number of data.

Description

Creates a list with the specified number of data.

When complete, the current index number is set to at the top of the list.

If a list has already been created, the old list is appended with the specified number of data. The specified number of data becomes the number of valid elements.

There is no data in the elements of a list that is created or appended to.

If the list could not be configured, it is initialized.

Return value

Returns an error number.

Error:

Not Data There is no data. Specified value is negative.

Exception An exception occurred. The field could not be created.

Reference

```
initialize
```

CDAQMXDataBuffer::currentDataInfo

Syntax

```
CDAQMXDataInfo * currentDataInfo(void);
```

Description

Retrieves the measured data indicated by the current index number.
Returns NULL if it does not exist.

Return value

Returns a pointer to the object.

CDAQMXDataBuffer::currentDateTime

Syntax

```
CDAQMXDateTime * currentDateTime(void);
```

Description

Retrieves the time information indicated by the current index number.
Returns NULL if it does not exist.

Return value

Returns a pointer to the object.

CDAQMXDataBuffer::getClassMXChInfo

Syntax

```
CDAQMXChInfo & getClassMXChInfo(void);
```

Description

Gets the field for storing the channel information data from the data member.

Return value

Returns the reference to the object.

CDAQMXDataBuffer::getDataInfo

Syntax

```
CDAQMXDataInfo * getDataInfo(int index);
```

Parameters

index Specify the storage location index number.

Description

Gets the measured data of the specified location from the data member.
Returns NULL if it does not exist.

Return value

Returns a pointer to the measured data.

CDAQMXDataBuffer::getDataNum

Syntax

```
int getDataNum(void);
```

Description

Gets the number of remaining data from the difference between the number of valid elements and the current index number.

Return value

Returns the remaining number of data.

CDAQMXDataBuffer::getDateTime

Syntax

```
CDAQMXDateTime * getDateTime(int index);
```

Parameters

index Specify the storage location index number.
 cMXDateTime Specify the time information data.

Description

Gets the time information data of the specified location from the data member.
 Returns NULL if it does not exist.

Return value

Returns a pointer to the time information data.

CDAQMXDataBuffer::initialize

Syntax

```
void initialize(void);
```

Description

Initializes the data member.
 Destructs the storage field for the measured data and time information data.
 The field for storing the channel information data is not initialized.

CDAQMXDataBuffer::isCurrent

Syntax

```
int isCurrent(void);
```

Description

Checks whether the data in the current location (of the current index number) is valid.

Return value

Returns a Boolean value.

Reference

currentDataInfo

CDAQMXDataBuffer::next

Syntax

```
int next(void);
```

Description

Increments the current index number.

If the valid number of elements is exceeded, the storage location is considered to be nonexistent, and a negative number is set.

Return value

Returns the current index number.

CDAQMXDataBuffer::setChInfo

Syntax

```
void setChInfo(CDAQMXChInfo & cMXChInfo);
```

Parameters

cMXChInfo Specify the channel information data.

Description

Copies the specified channel information data to the data member.

CDAQMXDataBuffer::setDataInfo

Syntax

```
int setDataInfo(int index, CDAQMXDataInfo & cMXDataInfo);
```

Parameters

index Specify the storage location index number.

cMXDataInfo Specify the measured data.

Description

Copies the specified measured data to the specified location field of the data member.

If no data exists in the element of the specified location in the list, data is created.

References to the channel information data are not copied, and become the channel information data field of the data member.

Return value

Returns an error number.

Error:

Not Data There is no data. The storage location index number is out of range.

Exception An exception occurred. The field could not be configured.

Reference

```
getClassMXChInfo  
CDAQMXDataInfo::CDAQMXDataInfo  
CDAQMXDataInfo::setClassMXChInfo
```

CDAQMXDataBuffer::setDateTime

Syntax

```
int setDateTime(int index, CDAQMXDateTime & cMXDateTime);
```

Parameters

index Specify the storage location index number.
cMXDateTime Specify the time information data.

Description

Copies the specified time information data to the specified location of the data member.

If no data exists in the element of the specified location in the list, data is created.

Return value

Returns an error number.

Error:

Not Data There is no data. The storage location index number is out of range.

Exception An exception occurred. The field could not be configured.

Reference

CDAQMXDateTime::CDAQMXDateTime

CDAQMXDOList Class

CDAQMXList
CDAQMXDOList

This class is for managing DO data from the output of command DO.
You can create the DO data to send in advance and store it. The data is identified by an identifier.

Public Members

Construct/Destruct

CDAQMXDOList	Constructs an object.
~CDAQMXDOList	Destructs an object.

Member Data Manipulation

add	Adds the DO data.
change	Changes the DO data.
copyData	Copies the DO data.
getClassMXDOData	Gets the DO data.
initCurrent	Initializes the DO data.
getCurrent	Gets the current DO data.

Overridden Members

Member Data Manipulation

create	Creates the DO data.
copy	Copies the DO data.

Inherited Members

See CDAQMXList.
del
getMaxNo
getNum
initialize
isData

Protected Members

Data Members

m_list	Pointer to the top of the list.
m_num	Field for storing the number of elements in the list.

Member Data Manipulation

addData	Adds data to the list.
delData	Deletes data from the list.
getData	Gets data from the list.

Private Members

None.

Member Functions (Alphabetical Order)

CDAQMXDOList::add

Syntax

```
int add(CDAQMXDOData * pcMXDOData);
```

Parameters

pcMXDOData Specify the data using a pointer.

Description

Adds the specified data to the list and generates data identifiers.

Returns a negative value if not added.

Reference

addData

CDAQMXDOList::CDAQMXDOList

Syntax

```
CDAQMXDOList(void);  
virtual ~CDAQMXDOList(void);
```

Description

Constructs or destructs an object.

When constructing, the data member is initialized.

When destructing, the data in the list is deleted.

Reference

```
initCurrent  
CDAQMXList::CDAQMXList
```

CDAQMXDOList::change

Syntax

```
void change(int idDO, int doNo, int bValid, int bONOFF =  
DAQMX_VALID_OFF);
```

Parameters

idDO	Specify the DO data identifier.
doNo	Specify the data number.
bValid	Specify valid/invalid using a Boolean value.
bONOFF	Specify ON/OFF using a Boolean value.

Description

Changes the DO data of the specified DO data identifier.

If the data identifier is set to the constant for “Specify all data identifiers,” all data in the list is processed.

If the DO data number is set to the constant for “Specify all DO numbers,” all items within the data are processed.

Reference

getClassMXDOData
CDAQMXDOData::setDO

CDAQMXList::copy

Syntax

```
virtual void copy(int idxNo, int idxSrc);
```

Parameters

idxNo	Specify the data identifier of the copy destination.
idxSrc	Specify the data identifier of the copy source.

Description

Copies the contents of the data indicated by the data identifier from the copy source to the copy destination.

Does nothing if not overridden.

CDAQMXDOList::copyData

Syntax

```
void copyData(int idDO, CDAQMXDOData * pcMXDOData);
```

Parameters

idDO Specify the DO data identifier.
pcMXDOData Specify the data using a pointer.

Description

Copies the data specified by the pointer to the data of the specified data identifier. If the data identifier is set to the constant for “Specify all data identifiers,” all data in the list is processed.

Reference

getClassMXDOData

CDAQMXDOList::create

Syntax

```
virtual int create(void);
```

Description

Creates data and adds it to the list.

Return value

Returns the data identifier.

Reference

add
CDAQMXDOData::CDAQMXDOData

CDAQMXDOList::getClassMXDOData

Syntax

```
CDAQMXDOData * getClassMXDOData(int idDO);
```

Parameters

idDO Specify the DO data identifier.

Description

Gets the data of the specified data identifier. If the data identifier is set to the constant for “specify current data,” gets the current data field from the data member. Returns NULL if it does not exist.

Return value

Returns a pointer to the data.

Reference

getCurrent getData

CDAQMXDOList::getCurrent

Syntax

```
CDAQMXDOData & getCurrent(void);
```

Description

Gets the current data field from the data member.

Return value

Returns a reference to the data.

CDAQMXDOList::initCurrent

Syntax

```
void initCurrent(void);
```

Description

Initializes the current data field of the data member.

Reference

```
getCurrent  
CDAQMXDOData::initialize
```

CDAQMXItemConfig Class

CDAQCongig
CDAQMXItemConfig

This is a setup data class providing functions that access the setting contents using setup items.

Access using setup items involves the loading from or writing to the fields of the content. In principle, all contents are processed even they are unused, indefinite, or illegal. The contents are not checked for validity.

The class supports functions that indicate content fields using strings. If a type is defined with a constant, it is expressed with the string that indicates the type. If multiple notations are used such as with the CF status type and reference alarm, they are delimited with commas.

See the constants for the item name strings.

If it does not exist in the return value of the member function, it is usually the case that the field does not exist.

Functions are supported that retrieve the string as a floating point value from the item name and decimal point position.

Public Members

Construct/Destruct

CDAQMXItemConfig	Constructs an object.
~CDAQMXItemConfig	Destructs an object.

Setup Item Manipulation

readItem	Loads setup items.
writeItem	Writes setup items.

Member Data Manipulation

getHisterisys	Gets the hysteresis.
getDoubleHisterisys	Gets the hysteresis as a floating point number.
getDoubleAlarmON	Gets the alarm ON value as a floating point number.
getDoubleAlarmOFF	Gets the alarm OFF value as a floating point number.
getDoubleSpanMin	Gets the span minimum as a floating point number.
getDoubleSpanMax	Gets the span maximum as a floating point number.
getDoubleScaleMin	Gets the scale minimum as a floating point number.
getDoubleScaleMax	Gets the scale maximum as a floating point number.
getDoublePresetValue	Gets the value if the selected value is the "specified value" as a floating point number.

Utilities

toltemName	Gets the setup item string.
getItemNo	Gets the setting item number.
getMaxLenItemName	Get the maximum length of the setup item string.

Overridden Members

Utilities

isObject	Checks an object.
----------	-------------------

Inherited Members

See CDAQMXConfig.

getChName getClassMXBalanceData getClassMXChConfig
getClassMXChConfigData getClassMXNetInfo getClassMXOutputData
getClassMXStatus getClassMXSysInfo getItemError
getMXConfigData getRangePoint getSpanPoint
initialize initMXConfigData isCorrect isObject reconstruct
setAO setAOType setChKind setDELTA setDI setDOType setInterval
setMXConfigData setPWM setPWMType setRES setRRJC setRTD
setScaling setSKIP setSTRAIN setTC setTempUnit setVOLT

Protected Members

Inherited Members

See CDAQMXConfig.

m_cMXSysInfo m_cMXStatus m_cMXNetInfo m_cMXChConfigData

Private Members

None.

Member Functions (Alphabetical Order)

CDAQMXItemConfig::CDAQMXItemConfig

Syntax

```
CDAQMXItemConfig(MXConfigData * pMXConfigData = NULL);
virtual ~CDAQMXItemConfig(void);
```

Parameters

pMXConfigData Specify the setup data.

Description

Constructs or destructs an object.

When constructing, the data member is set to the specified value. If not specified, the data member is initialized.

Reference

CDAQMXConfig::CDAQMXConfig

CDAQMXItemConfig::getDoubleAlarmOFF

Syntax

```
double getDoubleAlarmOFF(int chNo, int levelNo);
```

Parameters

chNo Specify the channel number.

levelNo Specify the alarm level.

Description

Gets the specified channel number and threshold level for alarm termination (OFF value) as a floating point number.

Returns 0.0 if it does not exist.

Return value

Returns the threshold level (OFF value) for alarm termination as a floating point number.

Reference

```
getClassMXChConfig
CDAQMXChConfig::getAlarmValueOFF
CDAQMXChConfig::getPoint
CDAQMXDataInfo::toDoubleValue
```


CDAQMXItemConfig::getDoubleAlarmON

Syntax

```
double getDoubleAlarmON(int chNo, int levelNo);
```

Parameters

chNo	Specify the channel number.
levelNo	Specify the alarm level.

Description

Gets the specified channel number and threshold level for alarm generation (ON value) as a floating point number.

Returns 0.0 if it does not exist.

Return value

Returns the threshold level (ON value) for alarm generation as a floating point number.

Reference

```
getClassMXChConfig  
CDAQMXChConfig::getAlarmValueON  
CDAQMXChConfig::getPoint  
CDAQMXDataInfo::toDoubleValue
```

CDAQMXItemConfig::getDoubleHisterisys

Syntax

```
double getDoubleHisterisys(int chNo, int levelNo);
```

Parameters

chNo	Specify the channel number.
levelNo	Specify the alarm level.

Description

Gets the specified channel number and hysteresis for the alarm level as a floating point number.

Returns 0.0 if it does not exist.

Return value

Returns the hysteresis as a floating point number.

Reference

```
getClassMXChConfig  
getHisterisys  
CDAQMXChConfig::getPoint  
CDAQMXDataInfo::toDoubleValue
```

CDAQMXItemConfig::getDoublePresetValue

Syntax

```
double getDoublePresetValue(int outputNo);
```

Parameters

outputNo Specify the output channel data number.

Description

Gets the value if the selected value of the specified output channel data number is the “specified value,” as a floating point number.

Returns 0.0 if it does not exist.

Return value

Returns the value if the selected value is the “specified value” as a floating point number.

Reference

```
getClassMXOutputData  
getSpanPoint  
CDAQMXOutputData::getPresetValue  
CDAQMXDataInfo::toDoubleValue
```

CDAQMXItemConfig::getDoubleScaleMax

Syntax

```
double getDoubleScaleMax(int chNo);
```

Parameters

chNo Specify the channel number.

Description

Gets the maximum scale value for the specified channel number as a floating point number.

Returns 0.0 if it does not exist.

Return value

Gets the scale maximum value as a floating point number.

Reference

```
getClassMXChConfig  
CDAQMXChConfig::getPoint  
CDAQMXChConfig::getScaleMax  
CDAQMXDataInfo::toDoubleValue
```

CDAQMXItemConfig::getDoubleScaleMin

Syntax

```
double getDoubleScaleMin(int chNo);
```

Parameters

chNo Specify the channel number.

Description

Gets the minimum scale value for the specified channel number as a floating point number.

Returns 0.0 if it does not exist.

Return value

Gets the scale minimum value as a floating point number.

Reference

```
getClassMXChConfig  
CDAQMXChConfig::getPoint  
CDAQMXChConfig::getScaleMin  
CDAQMXDataInfo::toDoubleValue
```

CDAQMXItemConfig::getDoubleSpanMax

Syntax

```
double getDoubleSpanMax(int chNo);
```

Parameters

chNo Specify the channel number.

Description

Gets the maximum span value for the specified channel number as a floating point number.

Returns 0.0 if it does not exist.

Return value

Gets the span maximum value as a floating point number.

Reference

```
getClassMXChConfig  
getSpanPoint  
CDAQMXChConfig::getSpanMax  
CDAQMXDataInfo::toDoubleValue
```

CDAQMXItemConfig::getDoubleSpanMin

Syntax

```
double getDoubleSpanMin(int chNo);
```

Parameters

chNo Specify the channel number.

Description

Gets the minimum span value for the specified channel number as a floating point number.

Returns 0.0 if it does not exist.

Return value

Gets the span minimum value as a floating point number.

Reference

```
getClassMXChConfig  
getSpanPoint  
CDAQMXChConfig::getSpanMin  
CDAQMXDataInfo::toDoubleValue
```

CDAQMXItemConfig::getHisterisys

Syntax

```
int getHisterisys(int chNo, int levelNo);
```

Parameters

chNo Specify the channel number.

levelNo Specify the alarm level.

Description

Gets the hysteresis of the specified channel number and alarm level.

Returns 0 if it does not exist.

Return value

Returns the hysteresis.

Reference

```
getClassMXChConfig  
CDAQMXChConfig::getAlarmValueOFF  
CDAQMXChConfig::getAlarmValueON
```

CDAQMXItemConfig::getMaxLenItemName

Syntax

```
static int getMaxLenItemName(void);
```

Description

Gets the maximum length of string of the item name of the setup item.

The return value does not include the terminator.

Return value

Returns the length of the string.

CDAQMXConfig::isObject

Syntax

```
virtual int isObject(const char * classname =  
"CDAQMXItemConfig");
```

Parameters

classname Specify the class name using a string.

Description

Checks whether the specified class name was inherited.

If parameters are omitted, checks whether it is this class.

Classes that inherit this class must be overridden in order to check their own classes.

Returns true (valid) if the class was inherited. Otherwise, returns false (invalid).

If different from this class, checks the parent class.

Return value

Returns a Boolean value.

Reference

CDAQMXConfig::isObject

CDAQMXItemConfig::readItem

Syntax

```
int readItem(int itemNo, char * strItem, int lenItem);
```

Parameters

itemNo	Specify the setup item.
strItem	Specify the field where the string is to be stored.
lenItem	Specify the byte size of the field where the string is to be stored.

Description

Gets the contents of the specified setup item as a string.

Stores the string in the specified storage destination.

The strings that can be stored are, in general, ASCII strings.

Returns 0 if it does not exist.

Return value

Returns the length of the actual string.

Reference

```
getClassMXBalanceData getClassMXChConfig getClassMXNetInfo
getClassMXOutputData getClassMXStatus getClassMXSysInfo
```

CDAQMXItemConfig::toItemName

Syntax

```
static int toItemName(int itemNo, char * strItem, int
lenItem);
```

Parameters

itemNo	Specify the setup item.
strItem	Specify the field where the string is to be stored.
lenItem	Specify the byte size of the field where the string is to be stored.

Description

Gets the setup item name of the specified setup item as a string.

Stores the string in the specified storage destination.

The string stored to the field includes the terminator. The return value does not include the terminator.

Stores an empty string and returns 0 if it does not exist.

The strings that can be stored are, in general, ASCII strings.

Return value

Returns the length of the string.

CDAQMXItemConfig::toItemName

Syntax

```
static int toItemName(int itemNo, char * strItem, int lenItem);
```

Parameters

itemNo	Specify the setup item.
strItem	Specify the field where the string is to be stored.
lenItem	Specify the byte size of the field where the string is to be stored.

Description

Gets the setup item name of the specified setup item as a string.
Stores the string in the specified storage destination.
The string stored to the field includes the terminator. The return value does not include the terminator.
Stores an empty string and returns 0 if it does not exist.
The strings that can be stored are, in general, ASCII strings.

Return value

Returns the length of the string.

CDAQMXItemConfig::toItemNo

Syntax

```
static int toItemNo(const char * strItem);
```

Parameters

strItem	Specify the item name of the setup item using a string.
---------	---

Description

Gets the setup item from the string.
If it does not exist, "Unknown" is returned.
The specified string is, in general, an ASCII string.

Return value

Returns the setup item.

CDAQMXItemConfig::writeItem

Syntax

```
int writeItem(int itemNo, const char * strItem);
```

Parameters

itemNo	Specify the setup item.
strItem	Specify the contents using a string.

Description

Sets the specified contents to the specified setup item.
The format of the string should match the output of setup item loading function.
The specified string is, in general, an ASCII string.

Return value

Returns an error number.

Error:

Not Support	Setup item or contents not supported.
-------------	---------------------------------------

Reference

`getClassMXBalanceData` `getClassMXChConfig` `getClassMXNetInfo`
`getClassMXOutputData` `getClassMXStatus` `getClassMXSysInfo`

CDAQMXList Class

CDAQMXList

CDAQMXList

This class manages data created by the user.

The registered data is identified by a data identifier. The data identifier corresponds to the index number of the list.

This class does not define data to be created. It overrides data member manipulation function members.

Public Members

Construct/Destruct

CDAQMXList Constructs an object.

~CDAQMXList Destructs an object.

Member Data Manipulation

initialize Initializes the data member.

create Creates the data.

del Deletes the data.

copy Copies the data.

Utilities

getNum Gets the specified number of data.

getMaxNo Gets the maximum value of the data identifier.

isData Checks for the existence of data.

Protected Members

Data Members

m_list Pointer to the top of the list.

m_num Field for storing the number of elements in the list.

Member Data Manipulation

addData Adds data to the list.

delData Deletes the data from the list.

getData Gets data from the list.

Private Members

None.

Member Functions (Alphabetical Order)

CDAQMXList::addData

Syntax

```
int addData(void * pData);
```

Parameters

`pData` Specify the data using a pointer.

Description

Adds the specified data to the list and generates data identifiers.

Returns a negative value if not added.

Return value

Returns the data identifier.

CDAQMXList::CDAQMXList

Syntax

```
CDAQMXList(void);
virtual ~CDAQMXList(void);
```

Description

Constructs or destructs an object.

When constructing, the data member is initialized.

When destructing, the data in the list is deleted.

Reference

`initialize`

CDAQMXList::copy

Syntax

```
virtual void copy(int idxNo, int idxSrc);
```

Parameters

`idxNo` Specify the data identifier of the copy destination.

`idxSrc` Specify the data identifier of the copy source.

Description

Copies the contents of the data indicated by the data identifier from the copy source to the copy destination.

Does nothing if not overridden.

CDAQMXList::create

Syntax

```
virtual int create(void);
```

Description

Creates data and adds it to the list.

Data not created if not overridden.

Return value

Returns the data identifier.

Reference

addData

CDAQMXList::del

Syntax

```
virtual void del(int idxNo);
```

Parameters

idxNo Specify the data identifier.

Description

Deletes the data of the specified data identifier from the list and destructs.

If set to the constant for “Specify all data identifiers,” all data in the list is processed.

Reference

delData

CDAQMXList::delData

Syntax

```
void delData(int idxNo);
```

Parameters

idxNo Specify the data identifier.

Description

Deletes the data of the specified data identifier from the list and destructs.

If set to the constant for “Specify all data identifiers,” all data in the list is processed.

Reference

getData

CDAQMXList::getData

Syntax

```
void * getData(int idxNo);
```

Parameters

idxNo Specify the data identifier.

Description

Gets the data of the specified data identifier.
Returns NULL if it does not exist.

Return value

Returns a pointer to the data.

CDAQMXList::getMaxNo

Syntax

```
int getMaxNo(void);
```

Description

Gets the maximum value of the data identifier of the data that exists in the list.

Return value

Returns the data identifier.

CDAQMXList::getNum

Syntax

```
int getNum(void);
```

Description

Gets the specified number of data.
Gets the number of data that exists in the list.

Return value

Returns the number of data.

CDAQMXList::initialize

Syntax

```
virtual void initialize(void);
```

Description

Initializes the data member.
Destructs all data in the list.

Reference

delData

CDAQMXList::isData

Syntax

```
int isData(int idxNo);
```

Parameters

idxNo Specify the data identifier.

Description

Checks whether the data of the specified data identifier exists in the list.
If it exists, Valid is returned. Otherwise, returns Invalid.

Return value

Returns a Boolean value.

CDAQMXTransmitList Class

CDAQMXList
CDAQMXTransmitList

This class is for managing transmission output data specified under transmission output.

You can create the transmission output data to send in advance and store it. The data is identified by an identifier.

Public Members

Construct/Destruct

CDAQMXTransmitList Constructs an object.

~CDAQMXTransmitList Destructs an object.

Member Data Manipulation

add Adds transmission output data.

change Changes transmission output data.

copyData Copies transmission output data.

getClassMXTransmit Gets transmission output data.

initCurrent Initializes the current transmission output data.

getCurrent Gets the current transmission output data.

• Overridden Members

Member Data Manipulation

create Creates transmission output data.

copy Copies transmission output data.

• Inherited Members

See CDAQMXList.

del

getMaxNo

getNum

initialize

isData

Protected Members

Data Members

m_cCurrent Field for storing the current transmission output data.

Inherited Members

See CDAQMXList.

m_list

m_num

addData

delData

getData

Private Members

None.

Member Functions (Alphabetical Order)

CDAQMXTransmitList::add

Syntax

```
int add(CDAQMXTransmit * pcMXTransmit);
```

Parameters

pcMXTransmit Specify the data using a pointer.

Description

Adds the specified data to the list and generates data identifiers.

Returns a negative value if not added.

Return value

Returns the data identifier.

Reference

addData

CDAQMXTransmitList::CDAQMXTransmitList

Syntax

```
CDAQMXTransmitList(void);  
virtual ~CDAQMXTransmitList(void);
```

Description

Constructs or destructs an object.

When constructing, the data member is initialized.

When destructing, the data in the list is deleted.

Reference

initCurrent
CDAQMXList::CDAQMXList

CDAQMXTransmitList::change

Syntax

```
void change(int idTransmit, int aopwmNo, int iTransmit);
```

Parameters

idTransmit	Specify the transmission output data identifier.
aopwmNo	Specify the AO/PWM data number.
iTransmit	Specify the transmission status.

Description

Changes the transmission output data of the specified transmission output data identifier.

If the data identifier is set to the constant for “Specify all data identifiers,” all data in the list is processed.

If the transmission output data number is set to the constant for “Specify all transmission output data numbers,” all items within the data are processed.

Reference

```
getClassMXTransmit
CDAQMXTransmit::setTransmit
```

CDAQMXTransmitList::copy

Syntax

```
virtual void copy(int idxNo, int idxSrc);
```

Parameters

idxNo	Specify the data identifier of the copy destination.
idxSrc	Specify the data identifier of the copy source.

Description

Copies the contents of the data indicated by the data identifier from the copy source to the copy destination.

Reference

```
copyData
getClassMXTransmit
```

CDAQMXTransmitList::copyData

Syntax

```
void copyData(int idTransmit, CDAQMXTransmit * pcMXTransmit);
```

Parameters

idTransmit	Specify the transmission output data identifier.
pcMXTransmit	Specify the data using a pointer.

Description

Copies the data specified by the pointer to the data of the specified data identifier. If the data identifier is set to the constant for “Specify all data identifiers,” all data in the list is processed.

Reference

getClassMXTransmit

CDAQMXTransmitList::create

Syntax

```
virtual int create(void);
```

Description

Creates data and adds it to the list.

Return value

Returns the data identifier.

Reference

add
CDAQMXTransmit::CDAQMXTransmit

CDAQMXTransmitList::getClassMXTransmit

Syntax

```
CDAQMXTransmit * getClassMXTransmit(int idTransmit);
```

Parameters

idTransmit	Specify the transmission output data identifier.
------------	--

Description

Gets the data of the specified data identifier. If the data identifier is set to the constant for “specify current data,” gets the current data field from the data member. Returns NULL if it does not exist.

Return value

Returns a pointer to the data.

Reference

getCurrent
getData

CDAQMXTransmitList::getCurrent

Syntax

```
CDAQMXTransmit & getCurrent(void);
```

Description

Gets the current data field from the data member.

Return value

Returns a reference to the data.

CDAQMXTransmitList::initCurrent

Syntax

```
void initCurrent(void);
```

Description

Initializes the current data field of the data member.

Reference

```
getCurrent  
CDAQMXTransmit::initialize
```

13.1 Functions and Their Functionalities - MX100/ Visual C -

This section indicates the correspondence between the functionalities that the extension API supports and the Visual C functions.

There are two types, status transition functions and retrieval functions.

Status retrieval functions control the MX100. When measured data is retrieved with the data transition function, the measured data advances by only one interval's worth of data (the status of the extension API changes).

The retrieval function returns the parameter value. When data is retrieved, the data value of the current status is returned (the status of the extension API does not change).

Status Transition Functions

The FIFO column in the table indicates the FIFO operation when the function is executed while FIFO is running.

Stop: Stops the FIFO when the function is executed.

Continue: Continues the FIFO even when the function is executed.

Communication Functions

Function	FIFO	Function
Connect to the MX100	Continue	openMX100
Disconnect from the MX100	Continue	closeMX100

Starting/Stopping the FIFO

Function	FIFO	Function
Start the FIFO	Continue	measStartMX100
Stop the FIFO	Continue	measStopMX100

Control Functions

Function		FIFO	Class and Member Function
Set date/time	Current time	Stop	setDateTimeNowMX100
Set backup	valid/invalid	Continue	switchBackupMX100
Format of the CF card		Stop	formatCFMX100
Unit	Reconfigure	Stop	reconstructMX100
	Initialize	Stop	initSetValueMX100
	Reset alarm (alarm ACK)	Stop	ackAlarmMX100
	7-segment LED display	Continue	displaySegmentMX100
Initialize stored data	Specify channel	Continue	initDataChMX100
	Specify FIFO	Continue	initDataFIFOMX100

At the end of communications, the control function updates the status. See the data manipulation functions for more about the data transmission and setup functions. Time cannot be set arbitrarily.

Setup Functions

Function		FIFO	Function
Set all setup data (send collectively)	All setup data	Stop	sendConfigMX100
	Basic setup data	Stop	sendConfigMX100
Set setup data individually	System configuration data	Stop	sendConfigMX100
	Channel setup data	Stop	sendConfigMX100
	Initial balance data	Stop	sendConfigMX100
	Output channel data	Stop	sendConfigMX100
Initial balance data	Execute	Stop	initBalanceMX100
	Reset	Stop	clearBalanceMX100

The setup data setup functions send the stored data. When setting arbitrary initial balance data, see the initial balance data sending function under data manipulation functions.

Setup Change Functions

Range Settings

Function	FIFO	Function
Skip	Stop	setRangeMX100
DC voltage input	Stop	setRangeMX 100
Thermocouple input	Stop	setRangeMX100
RTD	Stop	setRangeMX100
Digital input	Stop	setRangeMX100
Resistance	Stop	setRangeMX100
Strain	Stop	setRangeMX100
AO	Stop	setRangeMX100
PWM	Stop	setRangeMX100
Difference computation between channels	Stop	setChDELTAMX100
Remote RJC	Stop	setChRRJCMX100

Channel Settings

Function	FIFO	Function		
Unit name	Stop	setChUnitMX100		
Tag	Stop	setChTagMX100		
Comment	Stop	setChCommentMX100		
AI/DI/AO/PWM Span	Stop	setSpanMX100 setDoubleSpanMX100		
AI/DI	Scale	Stop	setScaleMX100 setDoubleScaleMX100	
		Alarm	Stop	setAlarmMX100 setDoubleAlarmMX100 setAlarmValueMX100 setDoubleAlarmValueMX100
			Hysteresis	Stop
AI	Filter coefficient	Stop	setFilterMX100	
	Ref. junction compensation (RJC)	Stop	setRJCTypeMX100	
	Burnout	Stop	setBurnoutMX100	
DO	De-energize	Stop	setDeenergizeMX100	
	Hold	Stop	setHoldMX100	
	Reference alarm	Stop	setRefAlarmMX100	
Channel kind	DO type	Stop	setChKindMX100	
	AO type	Stop	setChKindMX100	
	PWM type	Stop	setChKindMX100	

Module Settings

Function	FIFO	Function
Interval type	Stop	setIntervalMX100
A/D integral time type	Stop	setIntegralMX100

Unit Settings

Function	FIFO	Function
Unit number	Stop	setUnitNoMX100
Temperature unit type	Stop	setUnitTempMX100
CF write mode	Stop	setCFWriteModeMX100

Output Channel Data

Function	FIFO	Function
Output type	Stop	setOutputTypeMX100
Selected value	Stop	setChoiceMX100
		setDoubleChoiceMX100
Pulse interval integer multiple	Stop	setPulseTimeMX100

The setup functions send the settings then update the status.

These settings are for individual channels. If the settings could not be entered, an error is usually returned.

Specification is possible with data values or measured values (Double).

Data Manipulation Functions

DO Data

Function		FIFO	Function
Create		Continue	createDOMX100
Delete		Continue	deleteDOMX100
Partial	User specification	Continue	changeDOMX100
chng	Copy	Continue	copyDOMX100
Transmit	Existing specification	Continue	commandDOMX100
	Change specification	Continue	switchDOMX100

AO/PWM data

Function		FIFO	Function
Create		Continue	createAOPWMMX100
Delete		Continue	deleteAOPWMMX100
Partial	Output data value	Continue	changeAOPWMMX100
chng	Actual output value	Continue	hangeAOPWMValueMX100
	Copy	Continue	copyAOPWMMX100
Transmit		Continue	commandAOPWMMX100

Initial Balance Data

Function		FIFO	Function
Create		Continue	createBalanceMX100
Delete		Continue	deleteBalanceMX100
Partial	User specification	Continue	changeBalanceMX100
chng	Copy	Continue	copyBalanceMX100
Transmit		Stop	commandBalanceMX100

Transmission Output Data

Function		FIFO	Function
Create		Continue	createTransmitMX100
Delete		Continue	deleteTransmitMX100
Partial change	User specification	Continue	changeTransmitMX100
	Copy	Continue	copyTransmitMX100
Transmit	Existing specification	Continue	commandTransmitMX100
	Change specification	Continue	switchTransmitMX100

Manipulates each data by identifier.

For manipulation other than transmission, status is not updated (no communication).

Retrieval Functions

Function		FIFO	Function	
Status data		Continue	updateStatusMX100	
System configuration data		Continue	updateSystemMX100	
Setup data		Continue	updateConfigMX100	
Output data	DO data	Continue	updateDODataMX100	
	AO/PWM data	Continue	dateAOPWMDataMX100	
	Transmission output data			
Channel information data		Continue	updateInfoChMX100	
Measured data	Specify ch.	FIFO value	Continue	measDataChMX100
		Inst value	Continue	measInstChMX100
	Specify FIFO	FIFO value	Continue	measDataFIFOMX100
		Inst value	Continue	measInstFIFOMX100
Initial balance data		Continue	updateBalanceMX100	
Output channel data		Continue	updateOutputMX100	

Data retrieval is performed collectively and internally by this API. Depending on the acquisition, the status may also be updated. Channel information data and setup data (including system configuration data, initial balance data, and output channel data) are stored internally, but the user can update stored data explicitly.

Setup Items

Function		FIFO	Function
Setup data	Receive collectively	Continue	getItemAllMX100
	Send collectively	Stop	setItemAllMX100
Setup items	Read	Continue	readItemMX100
	Write	Continue	writeItemMX100
	Initialize	Continue	initItemMX100

Loading, writing, and initializing of setting items is performed through access to the field where the item is stored, and validity checks are not performed on those fields. Also, status is not updated (no communication).

Retrieval Functions

Measured Data

Data Name		Function
Data value		dataValueMX100
Data status Values		dataStatusMX100
Alarm (presence/absence)		dataAlarmMX100
Measured values	Double integer	dataDoubleValueMX100
	String	dataStringValueMX100
Time	No. of seconds	dataTimeMX100
	Milliseconds	dataMilliSecMX100
	Year	dataYearMX100
	Month	dataMonthMX100
	Day	dataDayMX100
	Hour	dataHourMX100
	Minute	dataMinuteMX100
	Second	dataSecondMX100
Valid data (presence/absence)		dataValidMX100

Channel Information Data

Data Name	Function
FIFO number	channelFIFONoMX100
Channel sequence number in the FIFO	channelFIFOIndexMX100
Display minimum value	channelDisplayMinMX100
Display maximum value	channelDisplayMaxMX100
Measurable minimum value	channelRealMinMX100
Measurable maximum value	channelRealMaxMX100

Channel Setup Data

Data Name				Function
Channel status (valid/invalid)				channelValidMX100
Decimal Point Position				channelPointMX100
Channel kind				channelKindMX100
Range type				channelRangeMX100
Scale type				channelScaleTypeMX100
Unit name				toChannelUnitMX100 getChannelUnitMX100
Tag				toChannelTagMX100 getChannelTagMX100
Comment				toChannelCommentMX100 getChannelCommentMX100
AI/DI/AO/ PWM	Span	Min val	Data Value	channelSpanMinMX100
			Meas val	channelDoubleSpanMinMX100
	Max val	Data Value	channelSpanMaxMX100	
		Meas value	channelDoubleSpanMaxMX100	
AI/DI	Scale	Min val	Data Value	channelScaleMinMX100
			Meas value	channelDoubleScaleMinMX100
		Max value	Data Value	channelScaleMaxMX100
			Meas value	channelDoubleScaleMaxMX100
	Alarm type			alarmTypeMX100
	Alarm value (ON)	Data Value	alarmValueONMX100	
		Meas value	alarmDoubleValueONMX100	
	Alarm value (OFF)	Data Value	alarmValueOFFMX100	
Meas value		alarmDoubleValueOFFMX100		
Hysteresis	Data Value	alarmHisterisysMX100		
	Meas value	alarmDoubleHisterisysMX100		
AI	Filter coefficient			channelFilterMX100
	RJC type			channelRJCTypeMX100
	RJC voltage			channelRJCVoltMX100
	Burnout			channelBurnoutMX100
DO	De-energize			channelDeenergizeMX100
	Hold			channelHoldMX100
	Reference alarm			channelRefAlarmMX100
Channels undergoing difference between channels computation/Remote RJC/AO/PWM				
	channelRefChNoMX100		Reference channel number	
Initial balance data	Boolean value			channelBalanceValidMX100
	Initial balance value			channelBalanceValueMX100
Output channel data	Output type			channelOutputTypeMX100
	Value selected when idle			channelIdleChoiceMX100
	Value selected during error			channelErrorChoiceMX100
	Value if the selected value is the "specified value."			
		Data value		channelPresetValueMX100
		Meas value		channelDoublePresetValueMX100
	Pulse interval integer multiple			channelPulseTimeMX100

Network Information Data

Data Name	Function
Host name	toNetHostMX100 getNetHostMX100
IP address	netAddressMX100
Port number	netPortMX100
Subnet mask	netSubmaskMX100
Gateway address	netGatewayMX100

System Configuration Data

Data Name	Function	
Module	Module type	moduleTypeMX100
	Number of channels	moduleChNumMX100
	Interval type	moduleIntervalMX100
	AD integral time type	moduleIntegralMX100
	Valid/Invalid value	moduleValidMX100
	Module type at startup	moduleStandbyTypeMX100
	Actual module type	moduleRealTypeMX100
	Terminal type	moduleTerminalMX100
	Version	moduleVersionMX100
	FIFO Number	moduleFIFONoMX100
	Serial number	toModuleSerialMX100 getModuleSerialMX100
Unit	Unit type	unitTypeMX100
	Style	unitStyleMX100
	Unit number	unitNoMX100
	Temperature unit type	unitTempMX100
	Power supply frequency	unitFrequencyMX100
	Part number	toUnitPartNoMX100 getUnitPartNoMX100
	Option	unitOptionMX100
	Serial number	toUnitSerialMX100 getUnitSreialMX100
	MAC address	unitMACMX100
	CF write mode	unitCFWriteModeMX100

Status Data

Data Name		Function
Unit status value		statusUnitMX100
Valid number of FIFOs		statusFIFONumMX100
Backup (presence or absence)		statusBackupMX100
FIFO	FIFO status value	statusFIFOMX100
	Interval type	statusFIFOIntervalMX100
CF	CF status type	statusCFMX100
	Size	statusCFSizeMX100
	Remaining capacity	statusCFRemainMX100
Status return time	No. of seconds	statusTimeMX100
	Milliseconds	statusMilliSecMX100
	Year	statusYearMX100
	Month	statusMonthMX100
	Day	statusDayMX100
	Hour	statusHourMX100
	Minute	statusMinuteMX100
	Second	statusSecondMX100

Current Data

Data Name		Function
DO Data	Valid/Invalid value	currentDOValidMX100
	ON/OFF status	currentDOValueMX100
AO/PWM data	Valid/Invalid value	currentAOPWMValidMX100
	Output data value	currentAOPWMValueMX100
	Output value	currentDoubleAOPWMValueMX100
Initial balance data	Valid/Invalid value	currentBalanceValidMX100
	Initial balance value	currentBalanceValueMX100
	Initial balance result	currentBalanceResultMX100
Trans output data	Transmission status	currentTransmitMX100

This is the status of each data retrieved with the data retrieval functions.

The initial balance result of the initial balance data is the result executed by the setup function.

Actually-output output statuses such as DO data and AO/PWM data can be retrieved as current data. However, immediately after sending data, the specified value is returned, and the actual output may occur at the next timing.

Held data are the values retrieved upon a status update. It is not data from the time the retrieval function was called.

User Data

Data Name		Function
DO data	Valid/Invalid Value	userDOValidMX100
	ON/OFF status	userDOValueMX100
AO/PWM data	Valid/Invalid value	userAOPWMValidMX100
	Output data value	userAOPWMValueMX100
	Output value	userDoubleAOPWMValueMX100
Initial balance data	Valid/Invalid value	userBalanceValidMX100
	Initial balance value	userBalanceValueMX100
Trans output data	Transmission status	userTransmitMX100

Gets the data values created by the user with data manipulation functions.

Utilities

Function/Data Name		Function
No. of remaining data	Retrieve by channel	dataNumChMX100
	Retrieve by FIFO	dataNumFIFOMX100
Error	Get MX-specific error	lastErrorMX100
	Get the error message string	toErrorMessageMX100
		getErrorMessageMX100
	Get max length of the error message string	errorMaxLengthMX100
Get no. of parameter on which error detected	itemErrorMX100	
Change from FIFO information to channel number		channelNumberMX100
Get the decimal point position by range type		rangePointMX100
Meas value	Change to double integer	toDoubleValueMX100
	Convert into string.	toStringValueMX100
Alarm	Get the alarm type string.	toAlarmNameMX100
		getAlarmNameMX100
	Get the maximum length of the alarm string	alarmMaxLengthMX100
Get the version number of this API		versionAPIMX100
Get the revision number of this API		revisionAPIMX100
Get a portion of the IP address		addressPartMX100
AO/PWM	Convert the output values to output data values	toAOPWMValueMX100
	Convert the output data values to output values	toRealValueMX100
Setup items	Gets the setup item string from the setup item number	toItemNameMX100
	Gets the setup item number from the setup item string	toItemNoMX100
	Get max. length of the setup item string	itemMaxLengthMX100
	Convert to style version.	toStyleVersionMX100

13.2 Programming - MX100/Visual C -

Adding the Path to the Include File

Add the path of the include file (DAQMX100.h) to the project. The method of adding the include file varies depending on the environment used.

Declaration in the Source File

Write the declaration in the source file.

```
#include "DAQMX100.h"
```

Note

The include file of the common section (DAQHandler.h) and the MX100 include file (DAQMX.h) is referenced from the include file described above. Thus, declaration for it is not necessary.

Load Library Statement

The statement below is added so that the executable module (.dll) of the API can link to the process.

The executable module (.dll) of the API is mapped within the address space (LoadLibrary). Next, the address of the export function in the executable module is retrieved (GetProcAddress).

The callback type of the function pointer is the function name with a prefix "DLL" added and converted to uppercase. It is defined in the include file of the API.

```
HMODULE pDll = LoadLibrary("DAQMX100");  
DLLOPENMX100 openMX100 = (DLLOPENMX100)GetProcAddress(pDll,  
"openMX100");
```

Retrieval of the Measured Data

Program Example

```

////////////////////////////////////
// MX100 sample for measurement
#include <stdio.h>
#include "DAQMX100.h"
////////////////////////////////////
int main(int argc, char* argv[])
{
    int rc; //return code
    DAQMX100 comm; //discriptor
    int value;
#ifdef WIN32
    HMODULE pDll; //DLL handle
    //callback
    DLLOPENMX100 openMX100;
    DLLCLOSEMX100 closeMX100;
    DLLMEASSTARTMX100 measStartMX100;
    DLLMEASSTOPMX100 measStopMX100;
    DLLMEASDATAACHMX100 measDataChMX100;
    DLLDATAVALUEMX100 dataValueMX100;
    //laod
    pDll = LoadLibrary("DAQMX100");
    //get address
    openMX100 = (DLLOPENMX100)GetProcAddress(pDll, "openMX100");
    closeMX100 = (DLLCLOSEMX100)GetProcAddress(pDll,
"closeMX100");
    measStartMX100 = (DLLMEASSTARTMX100)GetProcAddress(pDll,
"measStartMX100");
    measStopMX100 = (DLLMEASSTOPMX100)GetProcAddress(pDll,
"measStopMX100");
    measDataChMX100 = (DLLMEASDATAACHMX100)GetProcAddress(pDll,
"measDataChMX100");
    dataValueMX100 = (DLLDATAVALUEMX100)GetProcAddress(pDll,
"dataValueMX100");
#endif //WIN32
    //connect
    comm = openMX100("192.168.1.12", &rc);
    //get
    rc = measStartMX100(comm);
    rc = measDataChMX100(comm, 1);
    value = dataValueMX100(comm, 1);
    rc = measStopMX100(comm);
    //disconnect
    rc = closeMX100(comm);
#ifdef WIN32
    FreeLibrary(pDll);
#endif
    return rc;
}
////////////////////////////////////

```

Using Setup Items

Program Example

```

////////////////////////////////////
// MX100 sample for items
#include <stdio.h>
#include "DAQMX100.h"
#include "DAQMXItems.h"
////////////////////////////////////
int main(int argc, char* argv[])
{
    int rc; //return code
    DAQMX100 comm; //discriptor
    int i; //counter
    char strItem[BUFSIZ];
    int realLen;
#ifdef WIN32
    HMODULE pDll; //DLL handle
    //callback
    DLLOPENMX100 openMX100;
    DLLCLOSEMX100 closeMX100;
    DLLGETITEMALLMX100 getItemAllMX100;
    DLLSETITEMALLMX100 setItemAllMX100;
    DLLREADITEMMX100 readItemMX100;
    DLLWRITEITEMMX100 writeItemMX100;
    //load
    pDll = LoadLibrary("DAQMX100");
    //get address
    openMX100 = (DLLOPENMX100)GetProcAddress(pDll, "openMX100");
    closeMX100 = (DLLCLOSEMX100)GetProcAddress(pDll,
"closeMX100");
    getItemAllMX100 = (DLLGETITEMALLMX100)GetProcAddress(pDll,
"getItemAllMX100");
    setItemAllMX100 = (DLLSETITEMALLMX100)GetProcAddress(pDll,
"setItemAllMX100");
    readItemMX100 = (DLLREADITEMMX100)GetProcAddress(pDll,
"readItemMX100");
    writeItemMX100 = (DLLWRITEITEMMX100)GetProcAddress(pDll,
"writeItemMX100");
#endif //WIN32
}

```



```

//connect
comm = openMX100("192.168.1.12", &rc);
//get
rc = getItemAllMX100(comm);
//loop by items
for (i = DAQMX_ITEM_ALL_START; i <= DAQMX_ITEM_ALL_END; i++)
{
    //read
    rc = readItemMX100(comm, i, strItem, BUFSIZ, &realLen);
    //write
    rc = writeItemMX100(comm, i, strItem);
}
//set
rc = setItemAllMX100(comm);
//disconnect
rc = closeMX100(comm);
#ifdef WIN32
    FreeLibrary(pDll);
#endif
return rc;
}
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

```

The program is an example of reading and writing all setup items.

Be sure to prepare string fields of sufficient size.

By saving and loading groups of item numbers and item strings, you can backup the setup data.

For setup item numbers, see section 6.3.

14.1 Functions and Their Functionalities - MX100/ Visual Basic -

This section indicates the correspondence between the functionalities that the extension API supports and the Visual Basic functions.

There are two types, status transition functions and retrieval functions.

Status transition functions control the MX100. When using status transition functions, if measured data is retrieved with a data retrieval function, the measured data increments by only 1 (the status of the extension API changes).

The retrieval function returns the parameter value. When data is retrieved, the data value of the current status is returned (the status of the extension API does not change).

Status Transition Functions

The FIFO column in the table indicates the FIFO operation when the function is executed while FIFO is running.

Stop: Stops the FIFO when the function is executed.

Continue: Continues the FIFO even when the function is executed.

Communication Functions

Function	FIFO	Function
Connect to the MX100	Continue	openMX100
Disconnect from the MX100	Continue	closeMX100

Starting/Stopping the FIFO

Function	FIFO	Function
Start the FIFO	Continue	measStartMX100
Stop the FIFO	Stop	measStopMX100

Control Functions

Function		FIFO	Function
Set date/time	Current time	Stop	setDateTimeNowMX100
Set backup	valid/invalid	Continue	switchBackupMX100
Format of the CF card.		Stop	formatCFMX100
Unit	Reconfigure	Stop	reconstructMX100
	Initialize	Stop	initSetValueMX100
	Reset alarm (alarm ACK)	Stop	ackAlarmMX100
7-segment LED display		Continue	displaySegmentMX100
Initialize stored data	Specify channel	Continue	initDataChMX100
	Specify FIFO	Continue	initDataFIFOMX100

At the end of communications, the control function updates the status.

See the data manipulation functions for more about the data transmission and setup functions. Time cannot be set arbitrarily.

Setup Functions

Function		FIFO	Function
Set setup data (send collectively)	All setup data	Stop	sendConfigMX100
	Basic setup data	Stop	sendConfigMX100
Set setup data individually	System configuration data	Stop	sendConfigMX100
	Channel setup data	Stop	sendConfigMX100
	Initial balance data	Stop	sendConfigMX100
	Output channel data	Stop	sendConfigMX100
Initial balance data	Execute	Stop	initBalanceMX100
	Reset	Stop	clearBalanceMX100

The setup data setting function sends the data being stored.

When setting arbitrary initial balance data, see the initial balance data sending function under data manipulation functions.

Setup Change Functions

The setup functions send the settings then update the status. These settings are for individual channels. If the settings could not be entered, an error is usually returned. Specification is possible with data values or measured values (Double).

Range Settings

Function	FIFO	Function
Skip	Stop	setRangeMX
DC voltage input	Stop	setRangeMX
Thermocouple input	Stop	setRangeMX
RTD	Stop	setRangeMX
Digital input	Stop	setRangeMX
Resistance	Stop	setRangeMX
Strain	Stop	setRangeMX
AO	Stop	setRangeMX
PWM	Stop	setRangeMX
Difference computation between channels	Stop	setChDELTAMX100
Remote RRJC	Stop	setChRRJCMX100

Channel Settings

Function	FIFO	Function
Unit name	Stop	setChUnitMX100
Tag	Stop	setChTagMX100
Comment	Stop	setChCommentMX100
AI/DI/AO/PWM	Span	setSpanMX100 setDoubleSpanMX100
AI/DI	Scale	setScaleMX100 setDoubleScaleMX100
	Alarm	setAlarmMX100 setDoubleAlarmMX100 setAlarmValueMX100 setDoubleAlarmValueMX100
	Hysteresis	setHisterisysMX100 setDoubleHisterisysMX100
AI	Filter coefficient	setFilterMX100
	Ref. junction compensation (RJC)	setRJCTypeMX100
	Burnout types	setBurnoutMX100
DO	De-energize	setDeenergizeMX100
	Hold	setHoldMX100
	Reference Alarm	setRefAlarmMX100
Channel kind	DO type	setChKindMX100
	AO type	setChKindMX100
	PWM type	setChKindMX100

Module Settings

Function	FIFO	Function
Interval type	Stop	setIntervalMX100
A/D integral time type	Stop	setIntegralMX100

Unit Settings

Function	FIFO	Function
Unit number	Stop	setUnitNoMX100
Temperature unit type	Stop	setUnitTempMX100
CF write mode	Stop	setCFWriteModeMX100

Output Channel Data

Function	FIFO	Function
Output type	Stop	setOutputTypeMX100
Selected value	Stop	setChoiceMX100 setDoubleChoiceMX100
Pulse interval integer multiple	Stop	setPulseTimeMX100

Data Manipulation Functions

DO Data

Function	FIFO	Function	
Create	Continue	createDOMX100	
Delete	Continue	deleteDOMX100	
Partial chng	User specification	Continue	changeDOMX100
	Copy	Continue	copyDOMX100
Transmit	Existing specification	Continue	commandDOMX100
	Change specification	Continue	switchDOMX100

AO/PWM Data

Function	FIFO	Function	
Create	Continue	createAOPWMMX100	
Delete	Continue	deleteAOPWMMX100	
Partial chng	Output data value	Continue	changeAOPWMMX100
	Actual output value	Continue	hangeAOPWMValueMX100
	Copy	Continue	copyAOPWMMX100
Transmit		Continue	commandAOPWMMX100

Initial Balance Data

Function	FIFO	Function	
Create	Continue	createBalanceMX100	
Delete	Continue	deleteBalanceMX100	
Partial chng	User specification	Continue	changeBalanceMX100
	Copy	Continue	copyBalanceMX100
Transmit		Stop	commandBalanceMX100

Transmission Output Data

Function	FIFO	Function	
Create	Continue	createTransmitMX100	
Delete	Continue	deleteTransmitMX100	
Partial chng	User specification	Continue	changeTransmitMX100
	Copy	Continue	copyTransmitMX100
Transmit	Existing specification	Continue	commandTransmitMX100
	Change specification	Continue	switchTransmitMX100

Manipulates each data by identifier.

For manipulation other than transmission, status is not updated (no communication).

Retrieval Functions

Function		FIFO	Function	
Status data		Continue	updateStatusMX100	
System configuration data		Continue	updateSystemMX100	
Setup data		Continue	updateConfigMX100	
Output data	DO Data	Continue	updateDODataMX100	
	AO/PWM data	Continue	updateAOPWMDataMX100	
	Transmission output data			
Channel Information Data		Continue	updateInfoChMX100	
Measured Data	Specify ch	FIFO val	Continue	measDataChMX100
		Inst val	Continue	measInstChMX100
	Specify FIFO	FIFO val	Continue	measDataFIFOMX100
		Inst val	Continue	measInstFIFOMX100
Initial balance data		Continue	updateBalanceMX100	
Output channel data		Continue	updateOutputMX100	

Data retrieval is performed collectively and internally by this API. Depending on the acquisition, the status may also be updated. Channel information data and setup data (including system configuration data, initial balance data, and output channel data) are stored internally, but the user can update stored data explicitly.

Setup Items

Function		FIFO	Function
Setup data	Receive collectively	Continue	getItemAllMX100
	Send collectively	Stop	setItemAllMX100
Setup items	Read	Continue	readItemMX100
	Write	Continue	writeItemMX100
	Initialize	Continue	initItemMX100

Loading, writing, and initializing of setting items is performed through access to the field where the item is stored, and validity checks are not performed on those fields. Also, status is not updated (no communication).

Retrieval Functions

Measured Data

Data Name		Function
Data value		dataValueMX100
Data status values		dataStatusMX100
Alarm (presence/absence)		dataAlarmMX100
Measured values	Double integer	dataDoubleValueMX100
	String	dataStringValueMX100
Time	No. of seconds	dataTimeMX100
	Milliseconds	dataMilliSecMX100
	Year	dataYearMX100
	Month	dataMonthMX100
	Day	dataDayMX100
	Hour	dataHourMX100
	Minute	dataMinuteMX100
	Second	dataSecondMX100
Valid data (presence/absence)		dataValidMX100

Channel Information Data

Data Name	Function
FIFO number	channelFIFONoMX100
Channel sequence number in the FIFO	channelFIFOIndexMX100
Display minimum value	channelDisplayMinMX100
Display maximum value	channelDisplayMaxMX100
Measurable minimum value	channelRealMinMX100
Measurable maximum value	channelRealMaxMX100

Channel Setup Data

Data Name				Function	
Channel status (valid/invalid)				channelValidMX100	
Decimal point position				channelPointMX100	
Channel kind				channelKindMX100	
Range type				channelRangeMX100	
Scale type				channelScaleTypeMX100	
Unit name				toChannelUnitMX100 getChannelUnitMX100	
Tag				toChannelTagMX100 getChannelTagMX100	
Comment				toChannelCommentMX100 getChannelCommentMX100	
AI/DI/AO/ PWM	Span	Min value	Data Value	channelSpanMinMX100	
			Meas value	channelDoubleSpanMinMX100	
	Max value	Data Value	channelSpanMaxMX100		
		Meas value	channelDoubleSpanMaxMX100		
AI/DI	Scale	Min value	Data Value	channelScaleMinMX100	
			Meas value	channelDoubleScaleMinMX100	
	Max value	Data Value	channelScaleMaxMX100		
		Meas value	channelDoubleScaleMaxMX100		
Alarm type				alarmTypeMX100	
Alarm value (ON)			Data Value	alarmValueONMX100	
			Meas value	alarmDoubleValueONMX100	
Alarm value (OFF)			Data Value	alarmValueOFFMX100	
			Meas value	alarmDoubleValueOFFMX100	
Hysteresis			Data Value	alarmHisterisysMX100	
			Meas value	alarmDoubleHisterisysMX100	
AI	Filter			channelFilterMX100	
	RJC type			channelRJCTypeMX100	
	RJC voltage			channelRJCVoltMX100	
	Burnout			channelBurnoutMX100	
DO	De-energize			channelDeenergizeMX100	
	Hold			channelHoldMX100	
	Reference alarm			channelRefAlarmMX100	
Channels undergoing difference between channels computation/Remote RJC/AO/PWM					
			Reference channel number	channelRefChNoMX100	
Initial balance data	Boolean value			channelBalanceValidMX100	
	Initial balance value			channelBalanceValueMX100	
Output channel data	Output type			channelOutputTypeMX100	
	Value selected when idle			channelIdleChoiceMX100	
	Value selected during error			channelErrorChoiceMX100	
	Value if the selected value is the "specified value."			Data Value	channelPresetValueMX100
				Meas value	channelDoublePresetValueMX100
Pulse interval integer multiple				channelPulseTimeMX100	

Network Information Data

Data Name	Function
Host name	toNetHostMX100
IP address	netAddressMX100
Port number	netPortMX100
Subnet mask	netSubmaskMX100
Gateway address	netGatewayMX100

System Configuration Data

Data Name	Function		
Module	Module type	moduleTypeMX100	
	Number of channels	moduleChNumMX100	
	Interval type	moduleIntervalMX100	
	AD integral time type	moduleIntegralMX100	
	Valid/Invalid value	moduleValidMX100	
	Module type at startup	moduleStandbyTypeMX100	
	Actual module type	moduleRealTypeMX100	
	Terminal type	moduleTerminalMX100	
	Version	moduleVersionMX100	
	FIFO number	moduleFIFONoMX100	
	Serial number	toModuleSerialMX100	
	Unit	Unit type	unitTypeMX100
		Style	unitStyleMX100
Unit number		unitNoMX100	
Temperature unit type		unitTempMX100	
Power supply frequency		unitFrequencyMX100	
Part number		toUnitPartNoMX100	
Options		unitOptionMX100	
Serial number		toUnitSerialMX100	
MAC address		unitMACMX100	
CF write mode	unitCFWriteModeMX100		

Status Data

Data Name		Function
Unit status value		statusUnitMX100
Valid number of FIFOs		statusFIFONumMX100
Backup (presence or absence)		statusBackupMX100
FIFO	FIFO status value	statusFIFOMX100
	Interval type	statusFIFOIntervalMX100
CF	CF status type	statusCFMX100
	Size	statusCFSizeMX100
	Remaining capacity	statusCFRemainMX100
Status return time	No. of seconds	statusTimeMX100
	Milliseconds	statusMilliSecMX100
	Year	statusYearMX100
	Month	statusMonthMX100
	Day	statusDayMX100
	Hour	statusHourMX100
	Minute	statusMinuteMX100
	Second	statusSecondMX100

Current Data

Data Name		Function
DO data	Valid/Invalid value	currentDOValidMX100
	ON/OFF status	currentDOValueMX100
AO/PWM data	Valid/Invalid value	currentAOPWMValidMX100
	Output data value	currentAOPWMValueMX100
	Output value	currentDoubleAOPWMValueMX100
Initial balance data	Valid/Invalid value	currentBalanceValidMX100
	Initial balance value	currentBalanceValueMX100
	Initial balance result	currentBalanceResultMX100
Trans output data	Transmission status	currentTransmitMX100

This is the status of each data retrieved with the data retrieval functions.

The initial balance result of the initial balance data is the result executed by the setup function.

Actually output output statuses such as DO data and AO/PWM data can be retrieved as current data. However, immediately after sending data, the specified value is returned, and the actual output may occur at the next timing.

Held data are the values retrieved upon a status update. It is not data from the time the retrieval function was called.

User Data

Data Name		Function
DO data	Valid/Invalid value	userDOValidMX100
	ON/OFF status	userDOValueMX100
AO/PWM data	Valid/Invalid value	userAOPWMValidMX100
	Output data value	userAOPWMValueMX100
	Output value	userDoubleAOPWMValueMX100
Initial balance data	Valid/Invalid value	userBalanceValidMX100
	Initial balance value	userBalanceValueMX100
Trans output data	Transmission status	userTransmitMX100

Gets the data values created by the user with data manipulation functions.

Utilities

Function/Data Name		Function
No. of remaining data	Retrieve by channel	dataNumChMX100
	Retrieve by FIFO	dataNumFIFOMX100
Error	Get MX-specific error	lastErrorMX100
	Get the error message string.	toErrorMessageMX100
	Get max length of the error message string	errorMaxLengthMX100
	Get no. of parameter on which err detected	itemErrorMX100
Change from FIFO information to channel number		channelNumberMX100
Get the decimal point position by range type		rangePointMX100
Meas val	Change to double integer	toDoubleValueMX100
	Convert into string.	toStringValueMX100
Alarm	Get the alarm type string	toAlarmNameMX100
	Get maximum length of the alarm string	alarmMaxLengthMX100
Get the version number of this API		versionAPIMX100
Get the revision number of this API		revisionAPIMX100
Get a portion of the IP address		addressPartMX100
AO/PWM	Convert output val to output data val	toAOPWMValueMX100
	Convert output data val to output val	toRealValueMX100
Setup items	Get setup item string from setup item no	toltemNameMX100
	Get setup item no from setup item string	toltemNoMX100
	Get max length of the setup item string	itemMaxLengthMX100
Convert to style version		toStyleVersionMX100

14.2 Programming - MX100/Visual Basic -

Declaration of Types, Functions, and Constants

To use types, functions, and constants for Visual Basic, they must be declared in advance. The following methods of declaration statements are available.

Statement of All Declarations

Adding the standard module file for Visual Basic (DAQMX100.bas) to the project is equivalent to declaring all types, functions, and constants.

Statement of Selective Declarations

The API Viewer that comes with Visual Studio can be used to copy the declaration statements of arbitrary types, functions, and constants. Load the text file for the API Viewer (DAQMX100.txt) on the API Viewer to use this function.

For a description of how to use the API Viewer, read the operation manual for Visual Studio.

Writing Declarations Directly

Below is an example of a declaration statement.

```
Public Declare Function openMX100 Lib "DAQMX100" (ByVal  
strAddress As String, ByRef errorCode As Long) As Long
```

Retrieval of the Measured Data

Program Example

```
Attribute VB_Name = "Module1"
Public Function Main()
    'connect
    comm = openMX100("192.168.1.12", rc)
    'get
    rc = measStartMX100(comm)
    rc = measDataChMX100(comm, 1)
    value = dataValueMX100(comm, 1)
    rc = measStopMX100(comm)
    'disconnect
    rc = closeMX100(comm)
End Function
```

Using Setup Items

Program Example

```
Attribute VB_Name = "Module1"
Public Function Main()
    Dim comm As Long           '/descriptor
    Dim rc As Long            '/return (error) code
    Dim i As Long             '/counter
    Dim strItem As String * 512 '/string buffer
    Dim lenItem As Long       '/size of buffer
    Dim realLen As Long       '/real size of string by
function returned
    '/set size
    lenItem = 512
    '/open
    comm = openMX100("192.168.1.12", rc)
    '/get
    rc = getItemAllMX100(comm)
    '/loop by items
    For i = DAQMX_ITEM_ALL_START To DAQMX_ITEM_ALL_END
        '/read
        rc = readItemMX100(comm, i, strItem, lenItem, realLen)
        '/write
        rc = writeItemMX100(comm, i, strItem)
    Next i
    '/set
    rc = setItemAllMX100(comm)
    '/close
    rc = closeMX100(comm)
End Function
```

The program is an example of reading and writing all setup items. Be sure to prepare string fields of sufficient size. By saving and loading groups of item numbers and item strings, you can backup the setup data.

For setup item numbers, see section 6.3.

15.1 Functions and Their Functionalities - MX100/ Visual Basic.NET -

This section indicates the correspondence between the functionalities that the extension API supports and the Visual C functions.

There are two types, status transition functions and retrieval functions.

Status retrieval functions control the MX100. When measured data is retrieved with the data transition function, the measured data advances by only one interval's worth of data (the status of the extension API changes).

The retrieval function returns the parameter value. When data is retrieved, the data value of the current status is returned (the status of the extension API does not change).

Status Transition Functions

The FIFO column in the table indicates the FIFO operation when the function is executed while FIFO is running.

Stop: Stops the FIFO when the function is executed.

Continue: Continues the FIFO even when the function is executed.

Communication Functions

Function	FIFO	Function
Connect to the MX100	Continue	openMX100
Disconnect from the MX100	Continue	closeMX100

Starting/Stopping the FIFO

Function	FIFO	Function
Start the FIFO	Continue	measStartMX100
Stop the FIFO	Stop	measStopMX100

Control Functions

Function		FIFO	Class and Member Function
Set date/time	Current time	Stop	setDateTimeNowMX100
Set backup	valid/invalid	Continue	switchBackupMX100
Format of the CF card		Stop	formatCFMX100
Unit	Reconfigure	Stop	reconstructMX100
	Initialize	Stop	initSetValueMX100
	Reset alarm (alarm ACK)	Stop	ackAlarmMX100
7-segment LED display		Continue	displaySegmentMX100
Initialize stored data	Specify channel	Continue	initDataChMX100
	Specify FIFO	Continue	initDataFIFOMX100

At the end of communications, the control function updates the status. See the data manipulation functions for more about the data transmission and setup functions. Time cannot be set arbitrarily.

Setup Functions

Function		FIFO	Function
Set setup data (send collectively)	All setup data	Stop	sendConfigMX100
	Basic setup data	Stop	sendConfigMX100
Set setup data individually	Sys configuration data	Stop	sendConfigMX100
	Channel setup data	Stop	sendConfigMX100
	Initial balance data	Stop	sendConfigMX100
	Output channel data	Stop	sendConfigMX100
Initial balance data	Execute	Stop	initBalanceMX100
	Reset	Stop	clearBalanceMX100

The setup data setup functions send the stored data. When setting arbitrary initial balance data, see the initial balance data sending function under data manipulation functions.

Setup Change Functions

The setup functions send the settings then update the status.

These settings are for individual channels. If the settings could not be entered, an error is usually returned.

Specification is possible with data values or measured values (Double).

Range Settings

Function	FIFO	Function
Skip	Stop	setRangeMX
DC voltage input	Stop	setRangeMX
Thermocouple input	Stop	setRangeMX
RTD	Stop	setRangeMX
Digital input	Stop	setRangeMX
Resistance	Stop	setRangeMX
Strain	Stop	setRangeMX
AO	Stop	setRangeMX
PWM	Stop	setRangeMX
Difference computation between channels	Stop	setChDELTAMX100
Remote RJC	Stop	setChRRJCMX100

Channel Settings

Function	FIFO	Function
Unit name	Stop	setChUnitMX100
Tag	Stop	setChTagMX100
Comment	Stop	setChCommentMX100
AI/DI/AO/ PWM	Span	setSpanMX100
		setDoubleSpanMX100
AI/DI	Scale	setScaleMX100
		setDoubleScaleMX100
	Alarm	setAlarmMX100
		setDoubleAlarmMX100
		setAlarmValueMX100
		setDoubleAlarmValueMX100
	Hysteresis	setHisterisysMX100
		setDoubleHisterisysMX100
AI	Filter coefficient	setFilterMX100
	Ref. junction compensation (RJC)	setRJCTypeMX100
	Burnout	setBurnoutMX100
DO	De-energize	setDeenergizeMX100
	Hold	setHoldMX100
	Reference alarm	setRefAlarmMX100
Channel	DO type	setChKindMX100
kind	AO type	setChKindMX100
	PWM type	setChKindMX100

Module Settings

Function	FIFO	Function
Interval type	Stop	setIntervalMX100
A/D integral time type	Stop	setIntegralMX100

Unit Settings

Function	FIFO	Function
Unit number	Stop	setUnitNoMX100
Temperature unit type	Stop	setUnitTempMX100
CF write mode	Stop	setCFWriteModeMX100

Output Channel Data

Function	FIFO	Function
Output type	Stop	setOutputTypeMX100
Selected value	Stop	setChoiceMX100 setDoubleChoiceMX100
Pulse interval integer multiple	Stop	setPulseTimeMX100

Data Manipulation Functions

DO Data

Function		FIFO	Function
Create		Continue	createDOMX100
Delete		Continue	deleteDOMX100
Partial chng	User specification	Continue	changeDOMX100
	Copy	Continue	copyDOMX100
Transmit	Existing specification	Continue	commandDOMX100
	Change specification	Continue	switchDOMX100

AO/PWM Data

Function		FIFO	Function
Create		Continue	createAOPWMMX100
Delete		Continue	deleteAOPWMMX100
Partial chng	Output data value	Continue	changeAOPWMMX100
	Actual output value	Continue	changeAOPWMValueMX100
	Copy	Continue	copyAOPWMMX100
Transmit		Continue	commandAOPWMMX100

Initial Balance Data

Function		FIFO	Function
Create		Continue	createBalanceMX100
Delete		Continue	deleteBalanceMX100
Partial chng	User specification	Continue	changeBalanceMX100
	Copy	Continue	copyBalanceMX100
Transmit		Stop	commandBalanceMX100

Transmission Output Data

Function		FIFO	Function
Create		Continue	createTransmitMX100
Delete		Continue	deleteTransmitMX100
Partial chng	User specification	Continue	changeTransmitMX100
	Copy	Continue	copyTransmitMX100
Transmit	Existing specification	Continue	commandTransmitMX100
	Change specification	Continue	switchTransmitMX100

Manipulates each data by identifier.

For manipulation other than transmission, status is not updated (no communication).

Retrieval Functions

Function		FIFO	Function	
Status data		Continue	updateStatusMX100	
System configuration data		Continue	updateSystemMX100	
Setup data		Continue	updateConfigMX100	
Output data	DO Data	Continue	updateDODataMX100	
	AO/PWM data	Continue	updateAOPWMDataMX100	
	Transmission output data			
Channel information data		Continue	updateInfoChMX100	
Measured data	Specify ch	FIFO value	Continue	measDataChMX100
		Inst value	Continue	measInstChMX100
	Specify FIFO	FIFO value	Continue	measDataFIFOMX100
		Inst value	Continue	measInstFIFOMX100
Initial balance data		Continue	updateBalanceMX100	
Output channel data		Continue	updateOutputMX100	

Data retrieval is performed collectively and internally by this API. Depending on the acquisition, the status may also be updated. Channel information data and setup data (including system configuration data, initial balance data, and output channel data) are stored internally, but the user can update stored data explicitly.

Setup Items

Function		FIFO	Function
Setup data	Receive collectively	Continue	getItemAllMX100
	Send collectively	Stop	setItemAllMX100
Setup items	Read	Continue	readItemMX100
	Write	Continue	writeItemMX100
	Initialize	Continue	initItemMX100

Loading, writing, and initializing of setting items is performed through access to the field where the item is stored, and validity checks are not performed on those fields. Also, status is not updated (no communication).

Retrieval Functions

Measured Data

Data Name		Function
Data value		dataValueMX100
Data status values		dataStatusMX100
Alarm (presence/absence)		dataAlarmMX100
Measured val	Double integer	dataDoubleValueMX100
	String	dataStringValueMX100
Time	No. of seconds	dataTimeMX100
	Milliseconds	dataMilliSecMX100
	Year	dataYearMX100
	Month	dataMonthMX100
	Day	dataDayMX100
	Hour	dataHourMX100
	Minute	dataMinuteMX100
	Second	dataSecondMX100
Valid data (presence/absence)		dataValidMX100

Channel Information Data

Data Name	Function
FIFO number	channelFIFONoMX100
Channel sequence number in the FIFO	channelFIFOIndexMX100
Display minimum value	channelDisplayMinMX100
Display maximum value	channelDisplayMaxMX100
Measurable minimum value	channelRealMinMX100
Measurable maximum value	channelRealMaxMX100

Channel Setup Data

Data Name				Function
Channel status (valid/invalid)				channelValidMX100
Decimal point position				channelPointMX100
Channel kind				channelKindMX100
Range type				channelRangeMX100
Scale type				channelScaleTypeMX100
Unit name				toChannelUnitMX100 getChannelUnitMX100
Tag				toChannelTagMX100 getChannelTagMX100
Comment				toChannelCommentMX100 getChannelCommentMX100
AI/DI/AO/ PWM	Span	Min value	Data Value	channelSpanMinMX100
			Meas value	channelDoubleSpanMinMX100
	Max value	Data Value	channelSpanMaxMX100	
		Meas value	channelDoubleSpanMaxMX100	
AI/DI	Scale	Min value	Data value	channelScaleMinMX100
			Meas value	channelDoubleScaleMinMX100
	Max value	Data value	channelScaleMaxMX100	
		Meas value	channelDoubleScaleMaxMX100	
	Alarm type			alarmTypeMX100
	Alarm value (ON)	Data value		alarmValueONMX100
Meas value		alarmDoubleValueONMX100		
Alarm value (OFF)	Data value		alarmValueOFFMX100	
	Meas value		alarmDoubleValueOFFMX100	
Hysteresis	Data value		alarmHisterisysMX100	
	Meas value		alarmDoubleHisterisysMX100	
AI	Filter			channelFilterMX100
	RJC type			channelRJCTypeMX100
	RJC voltage			channelRJCVoltMX100
	Burnout			channelBurnoutMX100
DO	De-energize			channelDeenergizeMX100
	Hold			channelHoldMX100
	Reference alarm			channelRefAlarmMX100
Channels undergoing difference between channels computation/Remote RJC/AO/PWM				
			Ref. channel number	channelRefChNoMX100
Initial balance data	Valid/Invalid value			channelBalanceValidMX100
	Initial balance value			channelBalanceValueMX100
Output channel data	Output type			channelOutputTypeMX100
	Value selected when idling			channelIdleChoiceMX100
	Value selected during error			channelErrorChoiceMX100
	Value if the selected value is the "specified value."			
			Data value	channelPresetValueMX100
		Meas value	channelDoublePresetValueMX100	
Pulse interval integer multiple			channelPulseTimeMX100	

Network Information Data

Data Name	Function
Host name	toNetHostMX100
IP address	netAddressMX100
Port number	netPortMX100
Subnet mask	netSubmaskMX100
Gateway address	netGatewayMX100

System Configuration Data

Data Name	Function	
Module	Module type	moduleTypeMX100
	Number of channels	moduleChNumMX100
	Interval type	moduleIntervalMX100
	AD integral time type	moduleIntegralMX100
	Valid/Invalid value	moduleValidMX100
	Module type at startup	moduleStandbyTypeMX100
	Actual module type	moduleRealTypeMX100
	Terminal type	moduleTerminalMX100
	Version	moduleVersionMX100
	FIFO number	moduleFIFONoMX100
	Serial number	toModuleSerialMX100
	Unit	Unit type
Style		unitStyleMX100
Unit number		unitNoMX100
Temperature unit type		unitTempMX100
Power supply frequency		unitFrequencyMX100
Part number		toUnitPartNoMX100
Options		unitOptionMX100
Serial number		toUnitSerialMX100
MAC address		unitMACMX100
CF write mode	unitCFWriteModeMX100	

Status Data

Data Name		Function
Unit status value		statusUnitMX100
Valid number of FIFOs		statusFIFONumMX100
Backup (presence or absence)		statusBackupMX100
FIFO	FIFO status value	statusFIFOMX100
	Interval type	statusFIFOIntervalMX100
CF	CF status type	statusCFMX100
	Size	statusCFSizeMX100
	Remaining capacity	statusCFRemainMX100
Status return time	No. of seconds	statusTimeMX100
	Milliseconds	statusMilliSecMX100
	Year	statusYearMX100
	Month	statusMonthMX100
	Day	statusDayMX100
	Hour	statusHourMX100
	Minute	statusMinuteMX100
	Second	statusSecondMX100

Current Data

Data Name		Function
DO Data	Valid/Invalid value	currentDOValidMX100
	ON/OFF status	currentDOValueMX100
AO/PWM data	Valid/Invalid value	currentAOPWMValidMX100
	Output data value	currentAOPWMValueMX100
	Output value	currentDoubleAOPWMValueMX100
Initial balance data	Valid/Invalid value	currentBalanceValidMX100
	Initial balance value	currentBalanceValueMX100
	Initial balance result	currentBalanceResultMX100
Trans output data	Transmission status	currentTransmitMX100

This is the status of each data retrieved with the data retrieval functions.

The initial balance result of the initial balance data is the result executed by the setup function.

Actually output output statuses such as DO data and AO/PWM data can be retrieved as current data. However, immediately after sending data, the specified value is returned, and the actual output may occur at the next timing.

Held data are the values retrieved upon a status update. It is not data from the time the retrieval function was called.

User Data

Data Name		Function
DO data	Valid/Invalid value	userDOValidMX100
	ON/OFF status	userDOValueMX100
AO/PWM data	Valid/Invalid Value	userAOPWMValidMX100
	Output data value	userAOPWMValueMX100
	Output value	userDoubleAOPWMValueMX100
Initial balance data	Valid/Invalid value	userBalanceValidMX100
	Initial balance value	userBalanceValueMX100
Transmission output data	Transmission status	userTransmitMX100

Gets the data values created by the user with data manipulation functions.

Utilities

Data Name		Function
No. of remaining data	Retrieve by channel	dataNumChMX100
	Retrieve by FIFO	dataNumFIFOMX100
Error	Get MX-specific error	lastErrorMX100
	Get the error message string	toErrorMessageMX100
	Get max length of the error message string	errorMaxLengthMX100
	Get no of parameter on which err detected	itemErrorMX100
Change from FIFO information to channel number		channelNumberMX100
Get the decimal point position by range type		rangePointMX100
Meas value	Change to double integer	toDoubleValueMX100
	Convert into string	toStringValueMX100
Alarm	Get the alarm type string	toAlarmNameMX100
	Get the maximum length of the alarm string	alarmMaxLengthMX100
Get the version number of this API		versionAPIMX100
Get the revision number of this API		revisionAPIMX100
Get a portion of the IP address		addressPartMX100
AO/PWM	Convert output val to output data val	toAOPWMValueMX100
	Convert output data val to output val	toRealValueMX100
Setup items	Get setup item string from setup item no	toItemNameMX100
	Get setup item no from setup item string	toItemNoMX100
	Get max length of setup item string	itemMaxLengthMX100
Convert to style version		toStyleVersionMX100

15.2 Programming - MX100/Visual Basic.NET -

Declaration of Functions and Constants

To use functions, and constants for Visual Basic.NET, they must be declared in advance. The following methods of declaration statements are available.

Declaration Statements

Adding the module file for Visual Basic.NET (DAQMX100.vb) to the project is equivalent to declaring all functions and constants.

Retrieval of Measured Data

Program Example 1

```
Module Module1
    Public Function Meas()
        Dim comm As Integer
        Dim rc As Integer
        Dim value As Integer
        'connect
        comm = openMX100("192.168.1.12" rc)
        'get
        rc = measStartMX100(comm)
        rc = measDataChMX100(comm, 1)
        value = dataValueMX100(comm, 1)
        rc = measStopMX100(comm)
        'disconnect
        rc = closeMX100(comm)
    End Function
End Module
```

Using Setup Items

Program Example 2

```
Module Module1
    Public Function Item()
        Dim rc As Integer
        Dim comm As Integer
        Dim i As Integer
        Dim strItem As String
        Dim lenItem As Integer
        Dim realLen As Integer
        lenItem = 512
        strItem = Space(lenItem)
        'connect
        comm = openMX100("192.168.1.12", rc)
        'get
        rc = getItemAllMX100(comm)
        'loop by item
        For i = DAQMX_ITEM_ALL_START To DAQMX_ITEM_ALL_END
            'read
            rc = readItemMX100(comm, i, strItem, lenItem,
realLen)
                'write
                rc = writeItemMX100(comm, i, strItem)
            Next i
            'set
            rc = setItemAllMX100(comm)
            'disconnect
            rc = closeMX100(comm)
        End Function
    End Module
```

The program is an example of reading and writing all setup items.

Be sure to prepare string fields of sufficient size.

By saving and loading pairs of item numbers and item strings, you can backup the setup data.

For setup item numbers, see section 6.3.

16.1 Functions and Their Functionalities - MX100/C# -

This section indicates the correspondence between the functionalities that the extension API supports and the Visual C functions.

There are two types, status transition functions and retrieval functions.

Status retrieval functions control the MX100. When measured data is retrieved with the data transition function, the measured data advances by only one interval's worth of data (the status of the extension API changes).

The retrieval function returns the parameter value. When data is retrieved, the data value of the current status is returned (the status of the extension API does not change).

Status Transition Functions

The FIFO column in the table indicates the FIFO operation when the function is executed while FIFO is running.

Stop: Stops the FIFO when the function is executed.

Continue: Continues the FIFO even when the function is executed.

Communication Functions

Function	FIFO	Function
Connect to the MX100	Continue	DAQMX100:: open
Disconnect from the MX100	Continue	DAQMX100:: close

Starting/Stopping the FIFO

Function	FIFO	Function
Start FIFO	Continue	DAQMX100. measStartMX100
Stop FIFO	Stop	DAQMX100. measStopMX100

Control Functions

Function		FIFO	Function
Set date/time	Current time	Stop	DAQMX100. setDateTimeNowMX100
Set backup	valid/invalid	Continue	DAQMX100. switchBackupMX100
Format of the CF card		Stop	DAQMX100. formatCFMX100
Unit	Reconfigure	Stop	DAQMX100. reconstructMX100
	Initialize	Stop	DAQMX100. initSetValueMX100
	Reset alarm (alarm ACK)	Stop	DAQMX100. ackAlarmMX100
7-segment LED display		Continue	DAQMX100. displaySegmentMX100
Initialize stored data	Specify channel	Continue	DAQMX100. initDataChMX100
	Specify FIFO	Continue	DAQMX100. initDataFIFOMX100

At the end of communications, the control function updates the status.
See the data manipulation functions for more about the data transmission and setup functions.

Setup Functions

Function		FIFO	Function
Set setup data (send collectively)	All setup data	Stop	DAQMX100. sendConfigMX100
	Basic setup data	Stop	DAQMX100. sendConfigMX100
Set setup data individually	Sys configuration data	Stop	DAQMX100. sendConfigMX100
	Channel setup data	Stop	DAQMX100. sendConfigMX100
	Initial balance data	Stop	DAQMX100. sendConfigMX100
	Output channel data	Stop	DAQMX100. sendConfigMX100
Initial balance data	Execute	Stop	DAQMX100. initBalanceMX100
	Reset	Stop	DAQMX100. clearBalanceMX100

The setup data setup functions send the stroed data.
When setting arbitrary initial balance data, see the initial balance data sending function under data manipulation functions.

Setup Change Functions

The setup functions send the settings then update the status.

These settings are for individual channels. If the settings could not be entered, an error is usually returned.

Specification is possible with data values or measured values (Double).

Range Settings

Function	FIFO	Function
Skip	Stop	DAQMX100. setRangMX100
DC voltage input	Stop	DAQMX100. setRangMX100
Thermocouple input	Stop	DAQMX100. setRangMX100
RTD	Stop	DAQMX100. setRangMX100
Digital input	Stop	DAQMX100. setRangMX100
Resistance	Stop	DAQMX100. setRangMX100
Strain	Stop	DAQMX100. setRangMX100
AO	Stop	DAQMX100. setRangMX100
PWM	Stop	DAQMX100. setRangMX100
Difference computation between channels	Stop	DAQMX100. setChDELTAMX100
Remote RRJC	Stop	DAQMX100. setChRRJCMX100

Channel Settings

Function	FIFO	Function
Unit name	Stop	DAQMX100. setChUnitMX100
Tag	Stop	DAQMX100. setChTagMX100
Comment	Stop	DAQMX100. setChCommentMX100
AI/DI/AO/ PWM	Span	DAQMX100. setSpanMX100
		DAQMX100. setDoubleSpanMX100
AI/DI	Scale	DAQMX100. setScaleMX100
		DAQMX100. setDoubleScaleMX100
	Alarm	DAQMX100. setAlarmMX100
		DAQMX100. setDoubleAlarmMX100
		DAQMX100. setAlarmValueMX100
		DAQMX100. setDoubleAlarmValueMX100
	Hysteresis	DAQMX100. setHisterisysMX100
		DAQMX100. setDoubleHisterisysMX100
AI	Filter coefficient	DAQMX100. setFilterMX100
	Ref. junction compensation (RJC)	DAQMX100. setRJCTypeMX100
	Burnout	DAQMX100. setBurnoutMX100
DO	De-energize	DAQMX100. setDeenergizeMX100
	Hold	DAQMX100. setHoldMX100
	Reference alarm	DAQMX100. setRefAlarmMX100
Channel kind	DO type	DAQMX100. setChKindMX100
	AO type	DAQMX100. setChKindMX100
	PWM type	DAQMX100. setChKindMX100

Module Settings

Function	FIFO	Function
Interval type	Stop	DAQMX100. setIntervalMX100
A/D integral time type	Stop	DAQMX100. setIntegralMX100

Unit Settings

Function	FIFO	Function
Unit number	Stop	DAQMX100. setUnitNoMX100
Temperature unit type	Stop	DAQMX100. setUnitTempMX100
CF write mode	Stop	DAQMX100. setCFWriteModeMX100

Output Channel Data

Function	FIFO	Class and Member Function
Output type	Stop	DAQMX100. setOutputTypeMX100
Selected value	Stop	DAQMX100. setChoiceMX100 DAQMX100. setDoubleChoiceMX100
Pulse interval integer multiple	Stop	DAQMX100. setPulseTimeMX100

Data Manipulation Functions

DO Data

Function	FIFO	Function	
Create	Continue	DAQMX100. createDOMX100	
Delete	Continue	DAQMX100. deleteDOMX100	
Partial chng	User specification	Continue	DAQMX100. changeDOMX100
	Copy	Continue	DAQMX100. copyDOMX100
Transmit	Existing specification	Continue	DAQMX100. commandDOMX100
	Change specification	Continue	DAQMX100. switchDOMX100

AO/PWM Data

Function	FIFO	Function	
Create	Continue	DAQMX100. createAOPWMMX100	
Delete	Continue	DAQMX100. deleteAOPWMMX100	
Partial chng	Output data value	Continue	DAQMX100. changeAOPWMMX100
	Actual output value	Continue	DAQMX100. changeAOPWMValueMX100
	Copy	Continue	DAQMX100. copyAOPWMMX100
Transmit		Continue	DAQMX100. commandAOPWMMX100

Initial Balance Data

Function	FIFO	Function	
Create	Continue	DAQMX100. createBalanceMX100	
Delete	Continue	DAQMX100. deleteBalanceMX100	
Partial chng	User specification	Continue	DAQMX100. changeBalanceMX100
	Copy	Continue	DAQMX100. copyBalanceMX100
Transmit		Stop	DAQMX100. commandBalanceMX100

Transmission Output Data

Function	FIFO	Function	
Create	Continue	DAQMX100. createTransmitMX100	
Delete	Continue	DAQMX100. deleteTransmitMX100	
Partial chng	User specification	Continue	DAQMX100. changeTransmitMX100
	Copy	Continue	DAQMX100. copyTransmitMX100
Transmit	Existing specification	Continue	DAQMX100. commandTransmitMX100
	Change specification	Continue	DAQMX100. switchTransmitMX100

Manipulates each data by identifier.

For manipulation other than transmission, status is not updated (no communication).

Retrieval Functions

Function		FIFO	Function
Status		Continue	DAQMX100. updateStatusMX100
System configuration data		Continue	DAQMX100. updateSystemMX100
Setup data		Continue	DAQMX100. updateConfigMX100
Output data	DO data	Continue	DAQMX100. updateDODataMX100
	AO/PWM data	Continue	DAQMX100. updateAOPWMDataMX100
	Trans output data		
Channel information data		Continue	DAQMX100. updateInfoChMX100
Measured data (Specify channel)	FIFO value	Continue	DAQMX100. measDataChMX100
	Instantaneous value	Continue	DAQMX100. measInstChMX100
Measured data (specify FIFO)	FIFO value	Continue	DAQMX100. measDataFIFOMX100
	Instantaneous value	Continue	DAQMX100. measInstFIFOMX100
Initial balance data		Continue	DAQMX100. updateBalanceMX100
Output channel data		Continue	DAQMX100. updateOutputMX100

Data retrieval is performed collectively and internally by this API.

Depending on the acquisition, the status may also be updated.

Channel information data and setup data (including system configuration data, initial balance data, and output channel data) are stored internally, but the user can update stored data explicitly.

Setup Items

Function		FIFO	Function
Set setup data	Receive collectively	Continue	DAQMX100. getItemAllMX100
	Send collectively	Stop	DAQMX100. setItemAllMX100
Setup items	Read	Continue	DAQMX100. readItemMX100
	Write	Continue	DAQMX100. writeItemMX100
	Initialize	Continue	DAQMX100. initItemMX100

Loading, writing, and initializing of setting items is performed through access to the field where the item is stored, and validity checks are not performed on those fields. Also, status is not updated (no communication).

Retrieval Functions

Measured Data

Data Name		Function
Data value		DAQMX100. dataValueMX100
Data status values		DAQMX100. dataStatusMX100
Alarm (presence/absence)		DAQMX100. dataAlarmMX100
Meas values	Double integer	DAQMX100. dataDoubleValueMX100
	String	DAQMX100. dataStringValueMX100
Time	No. of seconds	DAQMX100. dataTimeMX100
	Milliseconds	DAQMX100. dataMilliSecMX100
	Year	DAQMX100. dataYearMX100
	Month	DAQMX100. dataMonthMX100
	Day	DAQMX100. dataDayMX100
	Hour	DAQMX100. dataHourMX100
	Minute	DAQMX100. dataMinuteMX100
	Second	DAQMX100. dataSecondMX100
Valid data (presence/absence)		DAQMX100. dataValidMX100

Channel Information Data

Data Name	Function
FIFO number	DAQMX100. channelFIFONoMX100
Channel sequence number in the FIFO	DAQMX100. channelFIFOIndexMX100
Display minimum value	DAQMX100. channelDisplayMinMX100
Display maximum value	DAQMX100. channelDisplayMaxMX100
Measurable minimum value	DAQMX100. channelRealMinMX100
Measurable maximum value	DAQMX100. channelRealMaxMX100

Channel Setup Data

Data Name		Function	
Channel status (valid/invalid)		DAQMX100. channelValidMX100	
Decimal point position		DAQMX100. channelPointMX100	
Channel kind		DAQMX100. channelKindMX100	
Range type		DAQMX100. channelRangeMX100	
Scale type		DAQMX100. channelScaleTypeMX100	
Unit name		DAQMX100. toChannelUnitMX100	
Tag		DAQMX100. toChannelTagMX100	
Comment		DAQMX100. toChannelCommentMX100	
AI/DI/AO/ PWM	Span Min value	Data val	DAQMX100. channelSpanMinMX100
		Measval	DAQMX100. channelDoubleSpanMinMX100
	Max value	Data val	DAQMX100. channelSpanMaxMX100
		Meas val	DAQMX100. channelDoubleSpanMaxMX100
AI/DI	Scale Min value	Data val	DAQMX100. channelScaleMinMX100
		Meas val	DAQMX100. channelDoubleScaleMinMX100
	Max val	Data val	DAQMX100. channelScaleMaxMX100
		Meas val	DAQMX100. channelDoubleScaleMaxMX100
Alarm type		DAQMX100. alarmTypeMX100	
Alarm value (ON)	Data val	DAQMX100. alarmValueONMX100	
	Meas val	DAQMX100. alarmDoubleValueONMX100	
Alarm value (OFF)	Data val	DAQMX100. alarmValueOFFMX100	
	Meas val	DAQMX100. alarmDoubleValueOFFMX100	
Hysteresis	Data val	DAQMX100. alarmHisterisysMX100	
	Meas val	DAQMX100. alarmDoubleHisterisysMX100	
AI	Filter		DAQMX100. channelFilterMX100
	RJC type		DAQMX100. channelRJCTypeMX100
	RJC voltage		DAQMX100. channelRJCVoltMX100
	Burnout		DAQMX100. channelBurnoutMX100
DO	De-energize		DAQMX100. channelDeenergizeMX100
	Hold		DAQMX100. channelHoldMX100
	Reference alarm		DAQMX100. channelRefAlarmMX100
Channels undergoing difference between channels computation/Remote RJC/AO/PWM			
Reference channel number		DAQMX100. channelRefChNoMX100	
Initial balance data	Valid/Invalid value		DAQMX100. channelBalanceValidMX100
	Initial balance value		DAQMX100. channelBalanceValueMX100
Output channel data	Output type		DAQMX100. channelOutputTypeMX100
	Selection when idle		DAQMX100. channelIdleChoiceMX100
	Selection during err		DAQMX100. channelErrorChoiceMX100
	Value if the selected value is the "specified value."		
		Data val	DAQMX100. channelPresetValueMX100
		Meas val	DAQMX100. channelDoublePresetValueMX100
Pulse interval integer multiple		DAQMX100. channelPulseTimeMX100	

Network Information Data

Data Name	Function
Host name	DAQMX100. toNetHostMX100
IP address	DAQMX100. netAddressMX100
Port number	DAQMX100. netPortMX100
Subnet mask	DAQMX100. netSubmaskMX100
Gateway address	DAQMX100. netGatewayMX100

System Configuration Data

Data Name	Function	
Module	Module type	DAQMX100. moduleTypeMX100
	Number of channels	DAQMX100. moduleChNumMX100
	Interval type	DAQMX100. moduleIntervalMX100
	AD integral time type	DAQMX100. moduleIntegralMX100
	Valid/Invalid value	DAQMX100. moduleValidMX100
	Module type at startup	DAQMX100. moduleStandbyTypeMX100
	Actual module type	DAQMX100. moduleRealTypeMX100
	Terminal type	DAQMX100. moduleTerminalMX100
	Version	DAQMX100. moduleVersionMX100
	FIFO number	DAQMX100. moduleFIFONoMX100
	Serial number	DAQMX100. toModuleSerialMX100
Unit	Unit type	DAQMX100. unitTypeMX100
	Style	DAQMX100. unitStyleMX100
	Unit number	DAQMX100. unitNoMX100
	Temperature unit type	DAQMX100. unitTempMX100
	Power supply frequency	DAQMX100. unitFrequencyMX100
	Part number	DAQMX100. toUnitPartNoMX100
	Options	DAQMX100. unitOptionMX100
	Serial number	DAQMX100. toUnitSerialMX100
	MAC address	DAQMX100. unitMACMX100
CF write mode	DAQMX100. unitCFWriteModeMX100	

Status Data

Data Name		Function
Unit status value		DAQMX100. statusUnitMX100
Valid number of FIFOs		DAQMX100. statusFIFONumMX100
Backup (presence or absence)		DAQMX100. statusBackupMX100
FIFO	FIFO status value	DAQMX100. statusFIFOMX100
	Interval type	DAQMX100. statusFIFOIntervalMX100
CF	CF status type	DAQMX100. statusCFMX100
	Size	DAQMX100. statusCFSizeMX100
	Remaining capacity	DAQMX100. statusCFRemainMX100
Status return time	No. of seconds	DAQMX100. statusTimeMX100
	Milliseconds	DAQMX100. statusMilliSecMX100
	Year	DAQMX100. statusYearMX100
	Month	DAQMX100. statusMonthMX100
	Day	DAQMX100. statusDayMX100
	Hour	DAQMX100. statusHourMX100
	Minute	DAQMX100. statusMinuteMX100
	Second	DAQMX100. statusSecondMX100

Current Data

Data Name		Function
DO data	Valid/Invalid value	DAQMX100. currentDOValidMX100
	ON/OFF status	DAQMX100. currentDOValueMX100
AO/PWM data	Valid/Invalid value	DAQMX100. currentAOPWMValidMX100
	Output data value	DAQMX100. currentAOPWMValueMX100
	Output value	DAQMX100. currentDoubleAOPWMValueMX100
Initial balance data	Valid/Invalid value	DAQMX100. currentBalanceValidMX100
	Initial balance value	DAQMX100. currentBalanceValueMX100
	Initial balance result	DAQMX100. currentBalanceResultMX100
Trans output data	Transmission status	DAQMX100. currentTransmitMX100

This is the status of each data retrieved with the data retrieval functions.

The initial balance result of the initial balance data is the result executed by the setup function.

Actually output output statuses such as DO data and AO/PWM data can be retrieved as current data. However, immediately after sending data, the specified value is returned, and the actual output may occur at the next timing.

Held data are the values retrieved upon a status update. It is not data from the time the retrieval function was called.

User Data

Data Name		Function
DO data	Valid/Invalid value	DAQMX100. userDOValidMX100
	ON/OFF status	DAQMX100. userDOValueMX100
AO/PWM data	Valid/Invalid value	DAQMX100. userAOPWMValidMX100
	Output data value	DAQMX100. userAOPWMValueMX100
	Output value	DAQMX100. userDoubleAOPWMValueMX100
Initial balance data	Valid/Invalid value	DAQMX100. userBalanceValidMX100
	Initial balance value	DAQMX100. userBalanceValueMX100
Transmission output data	Transmission status	DAQMX100. userTransmitMX100

Gets the data values created by the user with data manipulation functions.

Utilities

Function/Data Name		Function
Remaining data number	Retrieve by channel	DAQMX100. dataNumChMX100
	Retrieve by FIFO	DAQMX100. dataNumFIFOMX100
Error	Get MX-specific error	DAQMX100. lastErrorMX100
	Get the error message string	DAQMX100. toErrorMessageMX100
	Get the error message string maximum length	DAQMX100. errorMaxLengthMX100
	Get no. of setup items on which errors were detected	DAQMX100. itemErrorMX100
Change from FIFO information to channel number		DAQMX100. channelNumberMX100
Get the decimal point position by range type		DAQMX100. rangePointMX100
Meas val	Change to double integer	DAQMX100. toDoubleValueMX100
	Convert into string	DAQMX100. toStringValueMX100
Alarm	Get the alarm type string	DAQMX100. toAlarmNameMX100
	Get the max length of the alarm string	DAQMX100. alarmMaxLengthMX100
Get the version number of this API		DAQMX100. versionAPIMX100
Get the revision number of this API		DAQMX100. revisionAPIMX100
Get a portion of the IP address		DAQMX100. addressPartMX100
AO/PWM	Convert output val to output data values	DAQMX100. toAOPWMValueMX100
	Convert output data val to output values	DAQMX100. toRealValueMX100
Setup items	Get setup item number from the setup item string	DAQMX100. toItemNameMX100
	Get the setup item string from the setup item number	DAQMX100. toItemNoMX100
	Get the maximum length of the setup item string	DAQMX100. itemMaxLengthMX100
Convert to style version		DAQMX100.toStyleVersionMX100

16.2 Programming - MX100/C# -

Declaration of Functions and Constants

To use functions and constants for C#, they must be declared in advance. The following methods of declaration statements are available.

Declaration Statements

Adding the module file for C# (DAQMX100.cs) to the project is equivalent to declaring all functions and constants.

Retrieval of Setup Data

Program Example 1

```
using System;
using System.Text;
using System.Runtime.InteropServices;

namespace MeasCS
{
    class Class1
    {
        [STAThread]
        static void Main(string[] args)
        {
            int rc;
            Encoding enc = Encoding.GetEncoding ("ascii");
            String address = "192.168.1.12";
            //connect
            int comm =
            DAQMX100.openMX100(enc.GetBytes(address), out rc);
            //get
            rc = DAQMX100.measStartMX100(comm);
            rc = DAQMX100.measDataChMX100(comm, 1);
            int val = DAQMX100.dataValueMX100(comm, 1);
            rc = DAQMX100.measStopMX100(comm);
            //disconnect
            rc = DAQMX100.closeMX100(comm);
        }
    }
}
```

Using Setup Items

Program Example 2

```

using System;
using System.Text;
using System.Runtime.InteropServices;

namespace ItemCS
{
    class Class1
    {
        [STAThread]
        static void Main(string[] args)
        {
            int rc;
            int lenItem = 512;
            byte[] strItem = new byte[lenItem];
            int realLen;
            Encoding enc = Encoding.GetEncoding ("ascii");
            String address = "192.168.1.12";
            //connect
            int comm =
DAQMX100.openMX100(enc.GetBytes(address), out rc);
            //get
            rc = DAQMX100.getItemAllMX100(comm);
            //loop by items
            for (int i = DAQMXItems.DAQMX_ITEM_ALL_START; i
<= DAQMXItems.DAQMX_ITEM_ALL_END; i++)
            {
                //read
                rc = DAQMX100.readItemMX100(comm, i,
strItem, lenItem, out realLen);
                //write
                rc = DAQMX100.writeItemMX100(comm, i,
strItem);
            }
            //set
            rc = DAQMX100.setItemAllMX100(comm);
            //disconnect
            rc = DAQMX100.closeMX100(comm);
        }
    }
}

```

The program is an example of reading and writing all setup items.

You can use setup items to process setup data collectively.

Be sure to prepare string fields of sufficient size.

By saving and loading pairs of item numbers and item strings, you can backup the setup data.

For setup item numbers, see section 6.3. .

17.1 Details of Function - MX00 (Visual C/Visual Basic/Visual Basic.NET/C#) - Status Transition Functions

This section describes the MX100 functions that are used in C and Visual Basic. The functions are listed in alphabetical order by the function name.

For details on constants and types, see chapter 18.

For information about the MX100, see appendix 1.

Most functions return an error number as a return value. Error number 0 is returned if there is no error.

In C#, the class (DAQMX100) member contains the declarations.

ackAlarmMX100

Syntax

```
int ackAlarmMX100(DAQMX100 daqmx100);
```

Declaration

Visual Basic

```
Public Declare Function ackAlarmMX100 Lib "DAQMX100" (ByVal daqmx100 As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function ackAlarmMX100 Lib "DAQMX100" (ByVal daqmx100 As Integer) As Integer
```

C#

```
[DllImport("DAQMX100.dll" CharSet=CharSet.Auto, EntryPoint="ackAlarmMX100")]  
public static extern int ackAlarmMX100(int daqmx100);
```

Parameters

daqmx100 Specify the device descriptor.

Description

Resets alarms.

Return value

Returns an error number.

Error:

Not descriptor No device descriptor.

Reference

CDAQMX100::ackAlarm

changeAOPWMMX100

Syntax

```
int changeAOPWMMX100(DAQMX100 daqmx100, int idAOPWM, int
aopwmNo, int bValid, int iAOPWMValue);
```

Declaration**Visual Basic**

```
Public Declare Function changeAOPWMMX100 Lib "DAQMX100" (ByVal
daqmx100 As Long, ByVal idAOPWM As Long, ByVal aopwmNo As
Long, ByVal bValid As Long, ByVal iAOPWMValue As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function changeAOPWMMX100 Lib
"DAQMX100" (ByVal daqmx100 As Integer, ByVal idAOPWM As
Integer, ByVal aopwmNo As Integer, ByVal bValid As Integer,
ByVal iAOPWMValue As Integer) As Integer
```

C#

```
[DllImport("DAQMX100.dll" CharSet=CharSet.Auto,
EntryPoint="changeAOPWMMX100")]
public static extern int changeAOPWMMX100(int daqmx100, int
idAOPWM, int aopwmNo, int bValid, int iAOPWMValue);
```

Parameters

daqmx100	Specify the device descriptor.
idAOPWM	Specify the AO/PWM data identifier.
aopwmNo	Specify the AO/PWM data number.
bValid	Specify valid/invalid using a Boolean value.
iAOPWMValue	Specify the output data value.

Description

Changes the AO/PWM data of the specified AO/PWM data identifier.

- The output data specifies the changed value of the actually output value.

Return value

Returns an error number.

Error:

Not descriptor	No device descriptor.
----------------	-----------------------

Reference

```
CDAQMX100::getClassMXAOPWMList
CDAQMXAOPWMList::change
```

changeAOPWMValueMX100

Syntax

```
int changeAOPWMValueMX100(DAQMX100 daqmx100, int idAOPWM, int
aopwmNo, int bValid, double realValue);
```

Declaration**Visual Basic**

```
Public Declare Function changeAOPWMValueMX100 Lib
"DAQMX100"(ByVal daqmx100 As Long, ByVal idAOPWM As Double,
ByVal aopwmNo As Long, ByVal bValid As Long, ByVal realValue
As Double) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function changeAOPWMValueMX100 Lib
"DAQMX100"(ByVal daqmx100 As Integer, ByVal idAOPWM As
Integer, ByVal aopwmNo As Integer, ByVal bValid As Integer,
ByVal realValue As Double) As Integer
```

C#

```
[DllImport("DAQMX100.dll" CharSet=CharSet.Auto,
EntryPoint="changeAOPWMValueMX100")]
public static extern int changeAOPWMValueMX100(int daqmx100,
int idAOPWM, int aopwmNo, int bValid, Double realValue);
```

Parameters

daqmx100	Specify the device descriptor.
idAOPWM	Specify the AO/PWM data identifier.
aopwmNo	Specify the AO/PWM data number.
bValid	Specify valid/invalid using a Boolean value.
realValue	Specify the actual output value.

Description

Changes the AO/PWM data of the specified AO/PWM data identifier.

- If the differences from the user specified output value are excluded, it is the same as the changeAOPWMMX100 function.
- The user-specified output value specifies a floating point value including the decimal point position.

Return value

Returns an error number.

Error:

Not descriptor	No device descriptor.
----------------	-----------------------

Reference

CDAQMX100::changeAOPWMValue

changeBalanceMX100

Syntax

```
int changeBalanceMX100(DAQMX100 daqmx100, int idBalance, int balanceNo, int bValid, int iValue);
```

Declaration**Visual Basic**

```
Public Declare Function changeBalanceMX100 Lib "DAQMX100" (ByVal daqmx100 As Long, ByVal idBalance As Long, ByVal balanceNo As Long, ByVal bValid As Long, ByVal iValue As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function changeBalanceMX100 Lib "DAQMX100" (ByVal daqmx100 As Integer, ByVal idBalance As Integer, ByVal balanceNo As Integer, ByVal bValid As Integer, ByVal iValue As Integer) As Integer
```

C#

```
[DllImport("DAQMX100.dll" CharSet=CharSet.Auto, EntryPoint="changeBalanceMX100")]
public static extern int changeBalanceMX100(int daqmx100, int idBalance, int balanceNo, int bValid, int iValue);
```

Parameters

daqmx100	Specify the device descriptor.
idBalance	Specify the initial balance data identifier.
balanceNo	Specify the initial balance data number.
bValid	Specify valid/invalid using a Boolean value.
iValue	Specify the initial balance value.

Description

Changes the initial balance data of the specified initial balance data identifier.

Return value

Returns an error number.

Error:

Not descriptor	No device descriptor.
----------------	-----------------------

Reference

```
CDAQMX100::getClassMXBalanceList
CDAQMXBalanceList::change
```

changeDOMX100

Syntax

```
int changeDOMX100(DAQMX100 daqmx100, int idDO, int doNo, int bValid, int bONOFF);
```

Declaration

Visual Basic

```
Public Declare Function changeDOMX100 Lib "DAQMX100" (ByVal daqmx100 As Long, ByVal idDO As Long, ByVal doNo As Long, ByVal bValid As Long, ByVal bONOFF As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function changeDOMX100 Lib "DAQMX100" (ByVal daqmx100 As Integer, ByVal idDO As Integer, ByVal doNo As Integer, ByVal bValid As Integer, ByVal bONOFF As Integer) As Integer
```

C#

```
[DllImport("DAQMX100.dll" CharSet=CharSet.Auto, EntryPoint="changeDOMX100")]  
public static extern int changeDOMX100(int daqmx100, int idDO, int doNo, int bValid, int bONOFF);
```

Parameters

daqmx100	Specify the device descriptor.
idDO	Specify the DO data identifier.
doNo	Specify the data number.
bValid	Specify valid/invalid using a Boolean value.
bONOFF	Specify ON/OFF using a Boolean value.

Description

Changes the DO data of the specified DO data identifier.

Return value

Returns an error number.

Error:

Not descriptor	No device descriptor.
----------------	-----------------------

Reference

CDAQMX100::getClassMXDOList
CDAQMXDOList::change

changeTransmitMX100

Syntax

```
int changeTransmitMX100(DAQMX100 daqmx100, int idTrans, int
aopwmNo, int iTransmit);
```

Declaration**Visual Basic**

```
Public Declare Function changeTransmitMX100 Lib
"DAQMX100"(ByVal daqmx100 As Long, ByVal idTrans As Long,
ByVal aopwmNo As Long, ByVal iTransmit As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function changeTransmitMX100 Lib
"DAQMX100"(ByVal daqmx100 As Integer, ByVal idTrans As
Integer, ByVal aopwmNo As Integer, ByVal iTransmit As Integer)
As Integer
```

C#

```
[DllImport("DAQMX100.dll" CharSet=CharSet.Auto,
EntryPoint="changeTransmitMX100")]
public static extern int changeTransmitMX100(int daqmx100, int
idTrans, int aopwmNo, int iTransmit);
```

Parameters

daqmx100	Specify the device descriptor.
idTrans	Specify the transmission output data identifier.
aopwmNo	Specify the AO/PWM data number.
iTransmit	Specify the transmission status.

Description

Changes the transmission output data of the specified transmission output data identifier.

Return value

Returns an error number.

Error:

Not descriptor	No device descriptor.
----------------	-----------------------

Reference

```
CDAQMX100::getClassMXTransmitList
CDAQMXTransmitList::change
```

clearBalanceMX100

Syntax

```
int clearBalanceMX100(DAQMX100 daqmx100);
```

Declaration

Visual Basic

```
Public Declare Function clearBalanceMX100 Lib "DAQMX100" (ByVal daqmx100 As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function clearBalanceMX100 Lib "DAQMX100" (ByVal daqmx100 As Integer) As Integer
```

C#

```
[DllImport("DAQMX100.dll" CharSet=CharSet.Auto, EntryPoint="clearBalanceMX100")]  
public static extern int clearBalanceMX100(int daqmx100);
```

Parameters

daqmx100 Specify the device descriptor.

Description

Initializes the initial balance value.

Return value

Returns an error number.

Error:

Not descriptor No device descriptor.

Reference

CDAQMX100::clearBalance

closeMX100

Syntax

```
int closeMX100(DAQMX100 daqmx100);
```

Declaration

Visual Basic

```
Public Declare Function closeMX100 Lib "DAQMX100" (ByVal daqmx100 As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function closeMX100 Lib "DAQMX100" (ByVal daqmx100 As Integer) As Integer
```

C#

```
[DllImport("DAQMX100.dll" CharSet=CharSet.Auto, EntryPoint="closeMX100")]  
public static extern int closeMX100(int daqmx100);
```

Parameters

daqmx100 Specify the device descriptor.

Description

Disconnects the communication using the specified device descriptor.

- When the communication is disconnected, the value of the device descriptor is meaningless.
- After disconnection, do not use the value of the device descriptor.

Return value

Returns an error number.

Error:

Not descriptor No device descriptor.

Reference

CDAQMX100::close

commandAOPWMMX100

Syntax

```
int commandAOPWMMX100(DAQMX100 daqmx100, int idAOPWM);
```

Declaration

Visual Basic

```
Public Declare Function commandAOPWMMX100 Lib "DAQMX100" (ByVal daqmx100 As Long, ByVal idAOPWM As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function commandAOPWMMX100 Lib "DAQMX100" (ByVal daqmx100 As Integer, ByVal idAOPWM As Integer) As Integer
```

C#

```
[DllImport("DAQMX100.dll" CharSet=CharSet.Auto, EntryPoint="commandAOPWMMX100")]  
public static extern int commandAOPWMMX100(int daqmx100, int idAOPWM);
```

Parameters

daqmx100	Specify the device descriptor.
idAOPWM	Specify the AO/PWM data identifier.

Description

Sends the AO/PWM data of the specified AO/PWM data identifier.

Return value

Returns an error number.

Error:

Not descriptor	No device descriptor.
----------------	-----------------------

Reference

CDAQMX100::commandAOPWM

commandBalanceMX100

Syntax

```
int commandBalanceMX100(DAQMX100 daqmx100, int idBalance);
```

Declaration**Visual Basic**

```
Public Declare Function commandBalanceMX100 Lib
"DAQMX100"(ByVal daqmx100 As Long, ByVal idBalance As Long) As
Long
```

Visual Basic.NET

```
Public Declare Ansi Function commandBalanceMX100 Lib
"DAQMX100"(ByVal daqmx100 As Integer, ByVal idBalance As
Integer) As Integer
```

C#

```
[DllImport("DAQMX100.dll" CharSet=CharSet.Auto,
EntryPoint="commandBalanceMX100")]
public static extern int commandBalanceMX100(int daqmx100, int
idBalance);
```

Parameters

daqmx100	Specify the device descriptor.
idBalance	Specify the initial balance data identifier.

Description

Sends the initial balance data of the specified initial balance data identifier.

Return value

Returns an error number.

Error:

Not descriptor	No device descriptor.
----------------	-----------------------

Reference

CDAQMX100::reloadBalance

commandDOMX100

Syntax

```
int commandDOMX100(DAQMX100 daqmx100, int idDO);
```

Declaration

Visual Basic

```
Public Declare Function commandDOMX100 Lib "DAQMX100" (ByVal daqmx100 As Long, ByVal idDO As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function commandDOMX100 Lib "DAQMX100" (ByVal daqmx100 As Integer, ByVal idDO As Integer) As Integer
```

C#

```
[DllImport("DAQMX100.dll" CharSet=CharSet.Auto, EntryPoint="commandDOMX100")]  
public static extern int commandDOMX100(int daqmx100, int idDO);
```

Parameters

daqmx100	Specify the device descriptor.
idDO	Specify the DO data identifier.

Description

Sends the DO data of the specified DO data identifier.

Return value

Returns an error number.

Error:

Not descriptor	No device descriptor.
----------------	-----------------------

Reference

CDAQMX100::commandDO

commandTransmitMX100

Syntax

```
int commandTransmitMX100(DAQMX100 daqmx100, int idTrans);
```

Declaration**Visual Basic**

```
Public Declare Function commandTransmitMX100 Lib
"DAQMX100"(ByVal daqmx100 As Long, ByVal idTrans As Long) As
Long
```

Visual Basic.NET

```
Public Declare Ansi Function commandTransmitMX100 Lib
"DAQMX100"(ByVal daqmx100 As Integer, ByVal idTrans As
Integer) As Integer
```

C#

```
[DllImport("DAQMX100.dll" CharSet=CharSet.Auto,
EntryPoint="commandTransmitMX100")]
public static extern int commandTransmitMX100(int daqmx100,
int idTrans);
```

Parameters

daqmx100	Specify the device descriptor.
idTrans	Specify the transmission output data identifier.

Description

Sends the transmission output data of the specified transmission output data identifier.

Return value

Returns an error number.

Error:

Not descriptor	No device descriptor.
----------------	-----------------------

Reference

CDAQMX100::commandTransmit

copyAOPWMMX100

Syntax

```
int copyAOPWMMX100(DAQMX100 daqmx100, int idAOPWM, int idAOPWMSrc);
```

Declaration

Visual Basic

```
Public Declare Function copyAOPWMMX100 Lib "DAQMX100" (ByVal daqmx100 As Long, ByVal idAOPWM As Long, ByVal idAOPWMSrc As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function copyAOPWMMX100 Lib "DAQMX100" (ByVal daqmx100 As Integer, ByVal idAOPWM As Integer, ByVal idAOPWMSrc As Integer) As Integer
```

C#

```
[DllImport("DAQMX100.dll" CharSet=CharSet.Auto, EntryPoint="copyAOPWMMX100")]  
public static extern int copyAOPWMMX100(int daqmx100, int idAOPWM, int idAOPWMSrc);
```

Parameters

daqmx100	Specify the device descriptor.
idAOPWM	Specify the copy destination for the AO/PWM data identifier.
idAOPWMSrc	Specify the copy source for the AO/PWM data identifier.

Description

Copies the AO/PWM data from the copy source of the specified AO/PWM data identifier to the copy destination.

- If the copy source is set to the constant for “specify current data,” gets the current data field from the data member.

Return value

Returns an error number.

Error:

Not descriptor	No device descriptor.
----------------	-----------------------

Reference

CDAQMX100::getClassMXAOPWMList
CDAQMXAOPWMList::copy

copyBalanceMX100

Syntax

```
int copyBalanceMX100(DAQMX100 daqmx100, int idBalance, int
idBalanceSrc);
```

Declaration**Visual Basic**

```
Public Declare Function copyBalanceMX100 Lib "DAQMX100" (ByVal
daqmx100 As Long, ByVal idBalance As Long, ByVal idBalanceSrc
As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function copyBalanceMX100 Lib
"DAQMX100" (ByVal daqmx100 As Integer, ByVal idBalance As
Integer, ByVal idBalanceSrc As Integer) As Integer
```

C#

```
[DllImport("DAQMX100.dll" CharSet=CharSet.Auto,
EntryPoint="copyBalanceMX100")]
public static extern int copyBalanceMX100(int daqmx100, int
idBalance, int idBalanceSrc);
```

Parameters

daqmx100	Specify the device descriptor.
idBalance	Specify the initial balance data identifier of the copy destination.
idBalanceSrc	Specify the initial balance data identifier of the copy source.

Description

Copies the initial balance data from the copy source of the specified initial balance data identifier to the copy destination.

- If the copy source is set to the constant for "specify current data," gets the current data field from the data member.

Return value

Returns an error number.

Error:

Not descriptor No device descriptor.

Reference

```
CDAQMX100::getClassMXBalanceList
CDAQMXBalanceList::copy
```

copyDOMX100

Syntax

```
int copyDOMX100(DAQMX100 daqmx100, int idDO, int idDOSrc);
```

Declaration

Visual Basic

```
Public Declare Function copyDOMX100 Lib "DAQMX100" (ByVal daqmx100 As Long, ByVal idDO As Long, ByVal idDOSrc As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function copyDOMX100 Lib "DAQMX100" (ByVal daqmx100 As Integer, ByVal idDO As Integer, ByVal idDOSrc As Integer) As Integer
```

C#

```
[DllImport("DAQMX100.dll" CharSet=CharSet.Auto, EntryPoint="copyDOMX100")]  
public static extern int copyDOMX100(int daqmx100, int idDO, int idDOSrc);
```

Parameters

daqmx100	Specify the device descriptor.
idDO	Specify the DO data identifier of the copy destination.
idDOSrc	Specify the DO data identifier of the copy source.

Description

Copies the DO data from the copy source of the specified DO data identifier to the copy destination.

- If the copy source is set to the constant for "specify current data," gets the current data field from the data member.

Return value

Returns an error number.

Error:

Not descriptor	No device descriptor.
----------------	-----------------------

Reference

```
CDAQMX100::getClassMXDOList  
CDAQMXDOList::copy
```

copyTransmitMX100

Syntax

```
int copyTransmitMX100(DAQMX100 daqmx100, int idTrans, int idTransSrc);
```

Declaration**Visual Basic**

```
Public Declare Function copyTransmitMX100 Lib "DAQMX100" (ByVal daqmx100 As Long, ByVal idTrans As Long, ByVal idTransSrc As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function copyTransmitMX100 Lib "DAQMX100" (ByVal daqmx100 As Integer, ByVal idTrans As Integer, ByVal idTransSrc As Integer) As Integer
```

C#

```
[DllImport("DAQMX100.dll" CharSet=CharSet.Auto, EntryPoint="copyTransmitMX100")]
public static extern int copyTransmitMX100(int daqmx100, int idTrans, int idTransSrc);
```

Parameters

daqmx100	Specify the device descriptor.
idTrans	Specify the transmission output data identifier of the copy destination.
idTransSrc	Specify the transmission output data identifier of the copy source.

Description

Copies the transmission output data from the copy source of the specified transmission output data identifier to the copy destination.

- If the copy source is set to the constant for "specify current data," gets the current data field from the data member.

Return value

Returns an error number.

Error:

Not descriptor	No device descriptor.
----------------	-----------------------

Reference

```
CDAQMX100::getClassMXTransmitList
CDAQMXTransmitList::copy
```

createAOPWMMX100

Syntax

```
int createAOPWMMX100(DAQMX100 daqmx100, int * errCode);
```

Declaration

Visual Basic

```
Public Declare Function createAOPWMMX100 Lib "DAQMX100" (ByVal daqmx100 As Long, ByRef errCode As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function createAOPWMMX100 Lib "DAQMX100" (ByVal daqmx100 As Integer, ByRef errCode As Integer) As Integer
```

C#

```
[DllImport("DAQMX100.dll" CharSet=CharSet.Auto, EntryPoint="createAOPWMMX100")]  
public static extern int createAOPWMMX100(int daqmx100, out int errCode);
```

Parameters

daqmx100	Specify the device descriptor.
errorCode	Specify the destination where the error number is to be returned.

Description

Creates new AO/PWM data.

- Returns the AOPWM data identifier as a return value.
- Returns a negative number if unsuccessful.
- Stores the error number if the return destination is specified.

Return value

Returns the data identifier.

Error:

Not descriptor	No device descriptor.
Not data	Data creation failed.

Reference

```
CDAQMX100::getClassMXAOPWMList  
CDAQMXAOPWMList::create
```

createBalanceMX100

Syntax

```
int createBalanceMX100(DAQMX100 daqmx100, int * errCode);
```

Declaration

Visual Basic

```
Public Declare Function createBalanceMX100 Lib
"DAQMX100"(ByVal daqmx100 As Long, ByRef errCode As Long) As
Long
```

Visual Basic.NET

```
Public Declare Ansi Function createBalanceMX100 Lib
"DAQMX100"(ByVal daqmx100 As Integer, ByRef errCode As
Integer) As Integer
```

C#

```
[DllImport("DAQMX100.dll" CharSet=CharSet.Auto,
EntryPoint="createBalanceMX100")]
public static extern int createBalanceMX100(int daqmx100, out
int errCode);
```

Parameters

daqmx100	Specify the device descriptor.
errorCode	Specify the destination where the error number is to be returned.

Description

Creates new initial balance data.

- Returns the initial balance data identifier as a return value.
- Returns a negative number if unsuccessful.
- Stores the error number if the return destination is specified.

Return value

Returns the data identifier.

Error:

Not descriptor	No device descriptor.
Not data	Data creation failed.

Reference

```
CDAQMX100::getClassMXBalanceList
CDAQMXBalanceList::create
```

createDOMX100

Syntax

```
int createDOMX100(DAQMX100 daqmx100, int * errorCode);
```

Declaration

Visual Basic

```
Public Declare Function createDOMX100 Lib "DAQMX100" (ByVal daqmx100 As Long, ByRef errorCode As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function createDOMX100 Lib "DAQMX100" (ByVal daqmx100 As Integer, ByRef errorCode As Integer) As Integer
```

C#

```
[DllImport("DAQMX100.dll" CharSet=CharSet.Auto, EntryPoint="createDOMX100")]  
public static extern int createDOMX100(int daqmx100, out int errorCode);
```

Parameters

daqmx100	Specify the device descriptor.
errorCode	Specify the destination where the error number is to be returned.

Description

Creates new DO data.

- Returns the DO data identifier as a return value.
- Returns a negative number if unsuccessful.
- Stores the error number if the return destination is specified.

Return value

Returns the data identifier.

Error:

Not descriptor	No device descriptor.
----------------	-----------------------

Not data	Data creation failed.
----------	-----------------------

Reference

CDAQMX100::getClassMXDOList

CDAQMXDOList::create

createTransmitMX100

Syntax

```
int createTransmitMX100(DAQMX100 daqmx100, int * errCode);
```

Declaration**Visual Basic**

```
Public Declare Function createTransmitMX100 Lib
"DAQMX100"(ByVal daqmx100 As Long, ByRef errCode As Long) As
Long
```

Visual Basic.NET

```
Public Declare Ansi Function createTransmitMX100 Lib
"DAQMX100"(ByVal daqmx100 As Integer, ByRef errCode As
Integer) As Integer
```

C#

```
[DllImport("DAQMX100.dll" CharSet=CharSet.Auto,
EntryPoint="createTransmitMX100")]
public static extern int createTransmitMX100(int daqmx100, out
int errCode);
```

Parameters

daqmx100	Specify the device descriptor.
errorCode	Specify the destination where the error number is to be returned.

Description

Creates new transmission output data.

- Returns the transmission output data identifier as a return value.
- Returns a negative number if unsuccessful.
- Stores the error number if the return destination is specified.

Return value

Returns the data identifier.

Error:

Not descriptor	No device descriptor.
Not data	Data creation failed.

Reference

```
CDAQMX100::getClassMXTransmitList
CDAQMXTransmitList::create
```

deleteAOPWMMX100

Syntax

```
int deleteAOPWMMX100(DAQMX100 daqmx100, int idaOPWM);
```

Declaration

Visual Basic

```
Public Declare Function deleteAOPWMMX100 Lib "DAQMX100" (ByVal daqmx100 As Long, ByVal idaOPWM As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function deleteAOPWMMX100 Lib "DAQMX100" (ByVal daqmx100 As Integer, ByVal idaOPWM As Integer) As Integer
```

C#

```
[DllImport("DAQMX100.dll" CharSet=CharSet.Auto, EntryPoint="deleteAOPWMMX100")]  
public static extern int deleteAOPWMMX100(int daqmx100, int idaOPWM);
```

Parameters

daqmx100	Specify the device descriptor.
idaOPWM	Specify the AO/PWM data identifier.

Description

Delete the AO/PWM data of the specified AO/PWM data identifier.

- If the data identifier is set to the constant for "Specify all data identifiers," the entire list is initialized.

Return value

Returns an error number.

Error:

Not descriptor	No device descriptor.
----------------	-----------------------

Reference

```
CDAQMX100::getClassMXAOPWMList  
CDAQMXAOPWMList::del  
CDAQMXAOPWMList::initialize
```

deleteBalanceMX100

Syntax

```
int deleteBalanceMX100(DAQMX100 daqmx100, int idBalance);
```

Declaration**Visual Basic**

```
Public Declare Function deleteBalanceMX100 Lib
"DAQMX100"(ByVal daqmx100 As Long, ByVal idBalance As Long) As
Long
```

Visual Basic.NET

```
Public Declare Ansi Function deleteBalanceMX100 Lib
"DAQMX100"(ByVal daqmx100 As Integer, ByVal idBalance As
Integer) As Integer
```

C#

```
[DllImport("DAQMX100.dll" CharSet=CharSet.Auto,
EntryPoint="deleteBalanceMX100")]
public static extern int deleteBalanceMX100(int daqmx100, int
idBalance);
```

Parameters

daqmx100	Specify the device descriptor.
idBalance	Specify the initial balance data identifier.

Description

Deletes the initial balance data of the specified initial balance data identifier.

- If the data identifier is set to the constant for "Specify all data identifiers," the entire list is initialized.

Return value

Returns an error number.

Error:

Not descriptor	No device descriptor.
----------------	-----------------------

Reference

```
CDAQMX100::getClassMXBalanceList
CDAQMXBalanceList::del
CDAQMXBalanceList::initialize
```

deleteDOMX100

Syntax

```
int deleteDOMX100(DAQMX100 daqmx100, int idDO);
```

Declaration

Visual Basic

```
Public Declare Function deleteDOMX100 Lib "DAQMX100" (ByVal daqmx100 As Long, ByVal idDO As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function deleteDOMX100 Lib "DAQMX100" (ByVal daqmx100 As Integer, ByVal idDO As Integer) As Integer
```

C#

```
[DllImport("DAQMX100.dll" CharSet=CharSet.Auto, EntryPoint="deleteDOMX100")]  
public static extern int deleteDOMX100(int daqmx100, int idDO);
```

Parameters

daqmx100	Specify the device descriptor.
idDO	Specify the DO data identifier.

Description

Deletes the DO data of the specified DO data identifier.

- If the data identifier is set to the constant for “Specify all data identifiers,” the entire list is initialized.

Return value

Returns an error number.

Error:

Not descriptor	No device descriptor.
----------------	-----------------------

Reference

```
CDAQMX100::getClassMXDOList  
CDAQMXDOList::del  
CDAQMXDOList::initialize
```

deleteTransmitMX100

Syntax

```
int deleteTransmitMX100(DAQMX100 daqmx100, int idTrans);
```

Declaration

Visual Basic

```
Public Declare Function deleteTransmitMX100 Lib
"DAQMX100"(ByVal daqmx100 As Long, ByVal idTrans As Long) As
Long
```

Visual Basic.NET

```
Public Declare Ansi Function deleteTransmitMX100 Lib
"DAQMX100"(ByVal daqmx100 As Integer, ByVal idTrans As
Integer) As Integer
```

C#

```
[DllImport("DAQMX100.dll" CharSet=CharSet.Auto,
EntryPoint="deleteTransmitMX100")]
public static extern int deleteTransmitMX100(int daqmx100, int
idTrans);
```

Parameters

daqmx100	Specify the device descriptor.
idTrans	Specify the transmission output data identifier.

Description

Deletes the transmission output data of the specified transmission output data identifier.

- If the data identifier is set to the constant for "Specify all data identifiers," the entire list is initialized.

Return value

Returns an error number.

Error:

Not descriptor	No device descriptor.
----------------	-----------------------

Reference

```
CDAQMX100::getClassMXTransmitList
CDAQMXTransmitList::del
CDAQMXTransmitList::initialize
```

displaySegmentMX100

Syntax

```
int displaySegmentMX100(DAQMX100 daqmx100, int dispPattern0,  
int dispPattern1, int dispType, int dispTime);
```

Declaration

Visual Basic

```
Public Declare Function displaySegmentMX100 Lib  
"DAQMX100"(ByVal daqmx100 As Long, dispPattern0 As Long,  
dispPattern1 As Long, dispType As Long, dispTime As Long) As  
Long
```

Visual Basic.NET

```
Public Declare Ansi Function displaySegmentMX100 Lib  
"DAQMX100"(ByVal daqmx100 As Integer, ByVal dispPattern0 As  
Integer, ByVal dispPattern1 As Integer, ByVal dispType As  
Integer, ByVal dispTime As Integer) As Integer
```

C#

```
[DllImport("DAQMX100.dll" CharSet=CharSet.Auto,  
EntryPoint="displaySegmentMX100")]  
public static extern int displaySegmentMX100(int daqmx100, int  
dispPattern0, int dispPattern1, int dispType, int dispTime);
```

Parameters

daqmx100	Specify the device descriptor.
dispPattern0	Specify the display pattern of segment number 0.
dispPattern1	Specify the display pattern of segment number 1.
dispType	Specify the display format.
dispTime	Specify the display time.

Description

Sets the display of the 7-segment LED.

- Returns the display pattern prior to changes.

Return value

Returns an error number.

Error:

Not descriptor No device descriptor.

Reference

CDAQMX100::displaySegmentMX100

formatCFMX100

Syntax

```
int formatCFMX100(DAQMX100 daqmx100);
```

Declaration

Visual Basic

```
Public Declare Function formatCFMX100 Lib "DAQMX100" (ByVal daqmx100 As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function formatCFMX100 Lib "DAQMX100" (ByVal daqmx100 As Integer) As Integer
```

C#

```
[DllImport("DAQMX100.dll", CharSet=CharSet.Auto, EntryPoint="formatCFMX100")]  
public static extern int formatCFMX100(int daqmx100);
```

Parameters

daqmx100 Specify the device descriptor.

Description

Formats the CF (Compact Flash) card.

Return value

Returns an error number.

Error:

Not descriptor No device descriptor.

Reference

CDAQMX100::formatCF

getItemAllMX100

Syntax

```
int getItemAllMX100(DAQMX100 daqmx100);
```

Declaration

Visual Basic

```
Public Declare Function getItemAllMX100 Lib "DAQMX100" (ByVal daqmx100 As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function getItemAllMX100 Lib "DAQMX100" (ByVal daqmx100 As Integer) As Integer
```

C#

```
[DllImport("DAQMX100.dll" CharSet=CharSet.Auto, EntryPoint="getItemAllMX100")]  
public static extern int getItemAllMX100(int daqmx100);
```

Parameters

daqmx100 Specify the device descriptor.

Description

Gets the setup data collectively.

Return value

Returns an error number.

Error:

Not descriptor No device descriptor.

Reference

CDAQMX100::getItemAll

initBalanceMX100

Syntax

```
int initBalanceMX100(DAQMX100 daqmx100);
```

Declaration

Visual Basic

```
Public Declare Function initBalanceMX100 Lib "DAQMX100" (ByVal daqmx100 As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function initBalanceMX100 Lib "DAQMX100" (ByVal daqmx100 As Integer) As Integer
```

C#

```
[DllImport("DAQMX100.dll" CharSet=CharSet.Auto, EntryPoint="initBalanceMX100")]  
public static extern int initBalanceMX100(int daqmx100);
```

Parameters

daqmx100 Specify the device descriptor.

Description

Executes initial balancing.

Return value

Returns an error number.

Error:

Not descriptor No device descriptor.

Reference

CDAQMX100::initBalance

initDataChMX100

Syntax

```
int initDataChMX100(DAQMX100 daqmx100, int chNo);
```

Declaration

Visual Basic

```
Public Declare Function initDataChMX100 Lib "DAQMX100" (ByVal daqmx100 As Long, ByVal chNo As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function initDataChMX100 Lib "DAQMX100" (ByVal daqmx100 As Integer, ByVal chNo As Integer) As Integer
```

C#

```
[DllImport("DAQMX100.dll" CharSet=CharSet.Auto, EntryPoint="initDataChMX100")]  
public static extern int initDataChMX100(int daqmx100, int chNo);
```

Parameters

daqmx100	Specify the device descriptor.
chNo	Specify the channel number.

Description

Initializes the measured data of the specified channel number.

- Retrieval of measured data starts from the top of the FIFO.
- If the constant for "Specify all channel numbers" is specified for the channel numbers, all channels are processed.

Return value

Returns an error number.

Error:

Not descriptor	No device descriptor.
----------------	-----------------------

Reference

CDAQMX100::initDataCh

initDataFIFOMX100

Syntax

```
int initDataFIFOMX100(DAQMX100 daqmx100, int fifoNo);
```

Declaration

Visual Basic

```
Public Declare Function initDataFIFOMX100 Lib "DAQMX100" (ByVal daqmx100 As Long, ByVal fifoNo As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function initDataFIFOMX100 Lib "DAQMX100" (ByVal daqmx100 As Integer, ByVal fifoNo As Integer) As Integer
```

C#

```
[DllImport("DAQMX100.dll" CharSet=CharSet.Auto, EntryPoint="initDataFIFOMX100")]
public static extern int initDataFIFOMX100(int daqmx100, int fifoNo);
```

Parameters

daqmx100	Specify the device descriptor.
fifoNo	Specify the FIFO number.

Description

Initializes the measured data of the specified FIFO number.

- Retrieval of measured data starts from the top of the FIFO.
- If the constant for "Specify all FIFO numbers" is specified for the FIFO numbers, all FIFOs are processed.

Return value

Returns an error number.

Error:

Not descriptor	No device descriptor.
----------------	-----------------------

Reference

CDAQMX100::initDataFIFO

initItemMX100

Syntax

```
int initItemMX100(DAQMX100 daqmx100);
```

Declaration

Visual Basic

```
Public Declare Function initItemMX100 Lib "DAQMX100" (ByVal daqmx100 As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function initItemMX100 Lib "DAQMX100" (ByVal daqmx100 As Integer) As Integer
```

C#

```
[DllImport("DAQMX100.dll" CharSet=CharSet.Auto, EntryPoint="initItemMX100")]  
public static extern int initItemMX100(int daqmx100);
```

Parameters

daqmx100 Specify the device descriptor.

Description

Initializes the stored setup data.

- Overwrites the stored field. Validity checks are not performed.
- The results of each retrieval function are not necessarily correct.

Return value

Returns an error number.

Error:

Not descriptor No device descriptor.

Reference

CDAQMX100::getClassMXItemConfig

CDAQMXItemConfig::initialize

initSetValueMX100

Syntax

```
int initSetValueMX100(DAQMX100 daqmx100);
```

Declaration

Visual Basic

```
Public Declare Function initSetValueMX100 Lib "DAQMX100" (ByVal daqmx100 As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function initSetValueMX100 Lib "DAQMX100" (ByVal daqmx100 As Integer) As Integer
```

C#

```
[DllImport("DAQMX100.dll" CharSet=CharSet.Auto, EntryPoint="initSetValueMX100")]  
public static extern int initSetValueMX100(int daqmx100);
```

Parameters

daqmx100 Specify the device descriptor.

Description

Initializes settings.

Return value

Returns an error number.

Error:

Not descriptor No device descriptor.

Reference

CDAQMX100::initSetValue

measDataChMX100

Syntax

```
int measDataChMX100(DAQMX100 daqmx100, int chNo);
```

Declaration

Visual Basic

```
Public Declare Function measDataChMX100 Lib "DAQMX100" (ByVal daqmx100 As Long, ByVal chNo As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function measDataChMX100 Lib "DAQMX100" (ByVal daqmx100 As Integer, ByVal chNo As Integer) As Integer
```

C#

```
[DllImport("DAQMX100.dll" CharSet=CharSet.Auto, EntryPoint="measDataChMX100")]  
public static extern int measDataChMX100(int daqmx100, int chNo);
```

Parameters

daqmx100	Specify the device descriptor.
chNo	Specify the channel number.

Description

Gets the measured data of the specified channel number.

- If the constant for "Specify all channel numbers" is specified for the channel numbers, all channels are processed.
- Advances one measurement point only.
- When combining FIFO specification and instantaneous value specification, the data order changes.
- First, the amount of retrievable data is retrieved and stored, and the first data is set as the current status data . Then, each time the function is called, the measurement point of the stored data advances by one, and the next data is set as the current status. When the last stored data is reached, the next retrievable amount of data is retrieved again.
- When retrieving data by communications, other status changes are performed.

Return value

Returns an error number.

Error:

Not descriptor	No device descriptor.
----------------	-----------------------

Reference

CDAQMX100::measDataCh

measDataFIFOMX100

Syntax

```
int measDataFIFOMX100(DAQMX100 daqmx100, int fifoNo);
```

Declaration**Visual Basic**

```
Public Declare Function measDataFIFOMX100 Lib "DAQMX100" (ByVal daqmx100 As Long, ByVal fifoNo As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function measDataFIFOMX100 Lib "DAQMX100" (ByVal daqmx100 As Integer, ByVal fifoNo As Integer) As Integer
```

C#

```
[DllImport("DAQMX100.dll" CharSet=CharSet.Auto, EntryPoint="measDataFIFOMX100")]
public static extern int measDataFIFOMX100(int daqmx100, int fifoNo);
```

Parameters

daqmx100	Specify the device descriptor.
fifoNo	Specify the FIFO number.

Description

Retrieves the measured data of the specified FIFO number.

- If the constant for “Specify all FIFO numbers” is specified for the FIFO numbers, all FIFOs are processed.
- Advances one measurement point only.
- The channels in the FIFO get the same measurement point data.
- When combining channel specification and instantaneous value specification, the data order changes.
- First, the amount of retrievable data is retrieved and stored, and the first data is set as the current status data . Then, each time the function is called, the measurement point of the stored data advances by one, and the next data is set as the current status. When the last stored data is reached, the next retrievable amount of data is retrieved again.
- When retrieving data by communications, other status changes are performed.

Return value

Returns an error number.

Error:

Not descriptor	No device descriptor.
----------------	-----------------------

Reference

CDAQMX100::measDataFIFO

measInstChMX100

Syntax

```
int measInstChMX100(DAQMX100 daqmx100, int chNo);
```

Declaration

Visual Basic

```
Public Declare Function measInstChMX100 Lib "DAQMX100" (ByVal daqmx100 As Long, ByVal chNo As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function measInstChMX100 Lib "DAQMX100" (ByVal daqmx100 As Integer, ByVal chNo As Integer) As Integer
```

C#

```
[DllImport("DAQMX100.dll" CharSet=CharSet.Auto, EntryPoint="measInstChMX100")]  
public static extern int measInstChMX100(int daqmx100, int chNo);
```

Parameters

daqmx100	Specify the device descriptor.
chNo	Specify the channel number.

Description

Retrieve the instantaneous value of the specified channel number.

- If the constant for "Specify all channel numbers" is specified for the channel numbers, all channels are processed.
- Performs other status changes.

Return value

Returns an error number.

Error:

Not descriptor	No device descriptor.
----------------	-----------------------

Reference

CDAQMX100::measInstCh

measInstFIFOMX100

Syntax

```
int measInstFIFOMX100(DAQMX100 daqmx100, int fifoNo);
```

Declaration

Visual Basic

```
Public Declare Function measInstFIFOMX100 Lib "DAQMX100" (ByVal daqmx100 As Long, ByVal fifoNo As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function measInstFIFOMX100 Lib "DAQMX100" (ByVal daqmx100 As Integer, ByVal fifoNo As Integer) As Integer
```

C#

```
[DllImport("DAQMX100.dll" CharSet=CharSet.Auto, EntryPoint="measInstFIFOMX100")]
public static extern int measInstFIFOMX100(int daqmx100, int chNo);
```

Parameters

daqmx100	Specify the device descriptor.
fifoNo	Specify the FIFO number.

Description

Retrieve the instantaneous value of the specified FIFO number.

- If the constant for "Specify all FIFO numbers" is specified for the FIFO numbers, all FIFOs are processed.
- Performs other status changes.

Return value

Returns an error number.

Error:

Not descriptor	No device descriptor.
----------------	-----------------------

Reference

CDAQMX100::measInstFIFO

measStartMX100

Syntax

```
int measStartMX100(DAQMX100 daqmx100);
```

Declaration

Visual Basic

```
Public Declare Function measStartMX100 Lib "DAQMX100" (ByVal daqmx100 As Long) As Long
```

Visual Basic.NET

```
Public Declare Auto Function measStartMX100 Lib "DAQMX100" (ByVal daqmx100 As Integer) As Integer
```

C#

```
[DllImport("DAQMX100.dll" CharSet=CharSet.Auto, EntryPoint="measStartMX100")]  
public static extern int measStartMX100(int daqmx100);
```

Parameters

daqmx100 Specify the device descriptor.

Description

Starts data acquisition.

- Starts the FIFO.
- If the FIFO is already started, it is continued.

Return value

Returns an error number.

Error:

Not descriptor No device descriptor.

Reference

CDAQMX100::measStart

measStopMX100

Syntax

```
int measStopMX100(DAQMX100 daqmx100);
```

Declaration

Visual Basic

```
Public Declare Function measStopMX100 Lib "DAQMX100" (ByVal daqmx100 As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function measStopMX100 Lib "DAQMX100" (ByVal daqmx100 As Integer) As Integer
```

C#

```
[DllImport("DAQMX100.dll", CharSet=CharSet.Auto, EntryPoint="measStopMX100")]  
public static extern int measStopMX100(int daqmx100);
```

Parameters

daqmx100 Specify the device descriptor.

Description

Stops data acquisition.

- Stops the FIFO and discards the acquired data.

Return value

Returns an error number.

Error:

Not descriptor No device descriptor.

Reference

CDAQMX100::measStop

openMX100

Syntax

```
DAQMX100 openMX100(const char * strAddress, int * errorCode);
```

Declaration

Visual Basic

```
Public Declare Function openMX100 Lib "DAQMX100" (ByVal  
strAddress As String, ByRef errorCode As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function openMX100 Lib "DAQMX100" (ByVal  
strAddress As String, ByRef errorCode As Integer) As Integer
```

C#

```
[DllImport("DAQMX100.dll" CharSet=CharSet.Auto,  
EntryPoint="openMX100")]  
public static extern int openMX100(byte[] strAddress, out int  
errorCode);
```

Parameters

strAddress	Specify the IP address as a string.
errorCode	Specify the destination where the error number is to be returned.

Description

Connects to the device with the address specified by the parameters.

- Creates a device descriptor and returns the value as a return value.
- Stores the error number if the return destination is specified.
- Initializes the stored data. Retrieves information about the status of the instrument such as setup data, channel information data, and stores the information.
- The specified string is, in general, an ASCII string.
- If unsuccessful, returns NULL in Visual C or 0 in Visual Basic, Visual Basic.NET/C#.

Return value

Returns the device descriptor.

Error:

Creating descriptor is failure	Failed to create the device descriptor.
--------------------------------	---

Reference

CDAQMX100::open

readItemMX100

Syntax

```
int readItemMX100(DAQMX100 daqmx100, int itemNo, char *
strItem, int lenItem, int * realLen);
```

Declaration**Visual Basic**

```
Public Declare Function readItemMX100 Lib "DAQMX100" (ByVal
daqmx100 As Long, ByVal itemNo As Long, ByVal strItem As
String, ByVal lenItem As Long, realLen As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function readItemMX100 Lib
"DAQMX100" (ByVal daqmx100 As Integer, ByVal itemNo As Integer,
ByVal strItem As String, ByVal lenItem As Integer, ByRef
realLen As Integer) As Integer
```

C#

```
[DllImport("DAQMX100.dll" CharSet=CharSet.Auto,
EntryPoint="readItemMX100")]
public static extern int readItemMX100(int daqmx100, int
itemNo, byte[] strItem, int lenItem, out int realLen);
```

Parameters

daqmx100	Specify the device descriptor.
itemNo	Specify the setup item number.
strItem	Specify the field where the string is to be stored.
lenItem	Specify the byte size of the field where the string is to be stored.
realLen	Specify the return destination for the length of the actual string.

Description

Stores the contents of the specified setup item as a string in the specified field.

- The string stored to the field includes the terminator (NULL).
- If the return destination is specified, returns the length of the actual string. The terminator is not included.
- The strings that can be stored are, in general, ASCII strings.

Return value

Returns an error number.

Error:

Not descriptor	No device descriptor.
Not Support	Unsupported setup item.
Not Data	The string storage field is insufficient.

Reference

```
CDAQMX100::getClassMXItemConfig
CDAQMXItemConfig::readItem
```

reconstructMX100

Syntax

```
int reconstructMX100(DAQMX100 daqmx100);
```

Declaration

Visual Basic

```
Public Declare Function reconstructMX100 Lib "DAQMX100" (ByVal daqmx100 As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function reconstructMX100 Lib "DAQMX100" (ByVal daqmx100 As Integer) As Integer
```

C#

```
[DllImport("DAQMX100.dll" CharSet=CharSet.Auto, EntryPoint="reconstructMX100")]  
public static extern int reconstructMX100(int daqmx100);
```

Parameters

daqmx100 Specify the device descriptor.

Description

Reconfigures the system.

Return value

Returns an error number.

Error:

Not descriptor No device descriptor.

Reference

CDAQMX100::reconstruct

sendConfigMX100

Syntax

```
int sendConfigMX100(DAQMX100 daqmx100);
```

Declaration

Visual Basic

```
Public Declare Function sendConfigMX100 Lib "DAQMX100" (ByVal daqmx100 As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function sendConfigMX100 Lib "DAQMX100" (ByVal daqmx100 As Integer) As Integer
```

C#

```
[DllImport("DAQMX100.dll", CharSet=CharSet.Auto, EntryPoint="sendConfigMX100")]  
public static extern int sendConfigMX100(int daqmx100);
```

Parameters

daqmx100 Specify the device descriptor.

Description

Sends the stored setup data.

Return value

Returns an error number.

Error:

Not descriptor No device descriptor.

Reference

CDAQMX100::sendConfig

setAlarmMX100

Syntax

```
int setAlarmMX100(DAQMX100 daqmx100, int chNo, int levelNo,
int iAlarmType, int value);
```

Declaration

Visual Basic

```
Public Declare Function setAlarmMX100 Lib "DAQMX100" (ByVal
daqmx100 As Long, ByVal chNo As Long, ByVal levelNo As Long,
ByVal iAlarmType As Long, ByVal value As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function setAlarmMX100 Lib
"DAQMX100" (ByVal daqmx100 As Integer, ByVal chNo As Integer,
ByVal levelNo As Integer, ByVal iAlarmType As Integer, ByVal
value As Integer) As Integer
```

C#

```
[DllImport("DAQMX100.dll" CharSet=CharSet.Auto,
EntryPoint="setAlarmMX100")]
public static extern int setAlarmMX100(int daqmx100, int chNo,
int levelNo, int iAlarmType, int value);
```

Parameters

daqmx100	Specify the device descriptor.
chNo	Specify the channel number.
levelNo	Specify the alarm level.
iAlarmType	Specify the alarm type.
value	Specify the alarm value.

Description

Sets the alarm to the alarm level of the specified channel.

- Set according to the specified range type.
- The hysteresis is 0.
- The alarm value is specified with an integer excluding the decimal point.

Return value

Returns an error number.

Error:

Not descriptor	No device descriptor.
----------------	-----------------------

Reference

CDAQMX100::setAlarm

setAlarmValueMX100

Syntax

```
int setAlarmValueMX100(DAQMX100 daqmx100, int chNo, int levelNo, int iAlarmType, int valueON, int valueOFF);
```

Declaration

Visual Basic

```
Public Declare Function setAlarmValueMX100 Lib "DAQMX100" (ByVal daqmx100 As Long, ByVal chNo As Long, ByVal levelNo As Long, ByVal iAlarmType As Long, ByVal valueON As Long, ByVal valueOFF As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function setAlarmValueMX100 Lib "DAQMX100" (ByVal daqmx100 As Integer, ByVal chNo As Integer, ByVal levelNo As Integer, ByVal iAlarmType As Integer, ByVal valueON As Integer, ByVal valueOFF As Integer) As Integer
```

C#

```
[DllImport("DAQMX100.dll" CharSet=CharSet.Auto, EntryPoint="setAlarmValueMX100")]
public static extern int setAlarmValueMX100(int daqmx100, int chNo, int levelNo, int iAlarmType, int valueON, int valueOFF);
```

Parameters

daqmx100	Specify the device descriptor.
chNo	Specify the channel number.
levelNo	Specify the alarm level.
iAlarmType	Specify the alarm type.
valueON	Specify the threshold level (ON value) for alarm activation.
valueOFF	Specify the threshold level (OFF value) for alarm termination.

Description

Sets the alarm to the alarm level of the specified channel.

- Set according to the specified range type.
- Hysteresis is specified by the threshold of alarm activation and release.
- The alarm value is specified with an integer excluding the decimal point.

Return value

Returns an error number.

Error:

Not descriptor No device descriptor.

Reference

CDAQMX100::setAlarm

setBurnoutMX100

Syntax

```
int setBurnoutMX100(DAQMX100 daqmx100, int chNo, int  
iBurnout);
```

Declaration

Visual Basic

```
Public Declare Function setBurnoutMX100 Lib "DAQMX100" (ByVal  
daqmx100 As Long, ByVal chNo As Long, ByVal iBurnout As Long)  
As Long
```

Visual Basic.NET

```
Public Declare Ansi Function setBurnoutMX100 Lib  
"DAQMX100" (ByVal daqmx100 As Integer, ByVal chNo As Integer,  
ByVal iBurnout As Integer) As Integer
```

C#

```
[DllImport("DAQMX100.dll" CharSet=CharSet.Auto,  
EntryPoint="setBurnoutMX100")]  
public static extern int setBurnoutMX100(int daqmx100, int  
chNo, int iBurnout);
```

Parameters

daqmx100	Specify the device descriptor.
chNo	Specify the channel number.
iBurnout	Specify the burnout type.

Description

Sets the burnout type to the specified channel numbers.

Return value

Returns an error number.

Error:

Not descriptor	No device descriptor.
----------------	-----------------------

Reference

CDAQMX100::setBurnout

setCFWriteModeMX100

Syntax

```
int setCFWriteModeMX100(DAQMX100 daqmx100, int iCFWriteMode);
```

Declaration**Visual Basic**

```
Public Declare Function setCFWriteModeMX100 Lib
"DAQMX100"(ByVal daqmx100 As Long, ByVal iCFWriteMode As Long)
As Long
```

Visual Basic.NET

```
Public Declare Ansi Function setCFWriteModeMX100 Lib
"DAQMX100"(ByVal daqmx100 As Integer, ByVal iCFWriteMode As
Integer) As Integer
```

C#

```
[DllImport("DAQMX100.dll" CharSet=CharSet.Auto,
EntryPoint="setCFWriteModeMX100")]
public static extern int setCFWriteModeMX100(int daqmx100, int
iCFWriteMode);
```

Parameters

daqmx100	Specify the device descriptor.
iCFWriteMode	Specify the CF write mode.

Description

Sets the CF write mode.

Return value

Returns an error number.

Error:

Not descriptor	No device descriptor.
----------------	-----------------------

Reference

CDAQMX100::setCFWriteMode

setChCommentMX100

Syntax

```
int setChCommentMX100(DAQMX100 daqmx100, int chNo, const char  
* strComment);
```

Declaration

Visual Basic

```
Public Declare Function setChCommentMX100 Lib "DAQMX100" (ByVal  
daqmx100 As Long, ByVal chNo As Long, ByVal strComment As  
String) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function setChCommentMX100 Lib  
"DAQMX100" (ByVal daqmx100 As Integer, ByVal chNo As Integer,  
ByVal strComment As String) As Integer
```

C#

```
[DllImport("DAQMX100.dll" CharSet=CharSet.Auto,  
EntryPoint="setChCommentMX100")]  
public static extern int setChCommentMX100(int daqmx100, int  
chNo, byte[] strComment);
```

Parameters

daqmx100	Specify the device descriptor.
chNo	Specify the channel number.
strTag	Specify the comment.

Description

Sets the comment to the specified channel number.

- The specified string is, in general, an ASCII string.

Return value

Returns an error number.

Error:

Not descriptor	No device descriptor.
----------------	-----------------------

Reference

CDAQMX100::setChComment

setChDELTAMX100

Syntax

```
int setChDELTAMX100(DAQMX100 daqmx100, int chNo, int refChNo,
int iRange);
```

Declaration**Visual Basic**

```
Public Declare Function setChDELTAMX100 Lib "DAQMX100" (ByVal
daqmx100 As Long, ByVal chNo As Long, ByVal refChNo As Long,
ByVal iRange As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function setChDELTAMX100 Lib
"DAQMX100" (ByVal daqmx100 As Integer, ByVal chNo As Integer,
ByVal refChNo As Integer, ByVal iRange As Integer) As Integer
```

C#

```
[DllImport("DAQMX100.dll" CharSet=CharSet.Auto,
EntryPoint="setChDELTAMX100")]
public static extern int setChDELTAMX100(int daqmx100, int
chNo, int refChNo, int iRange);
```

Parameters

daqmx100	Specify the device descriptor.
chNo	Specify the channel number.
refChNo	Specify the reference channel using a channel number.
iRange	Specify the range type for the input of the target channel.

Description

Sets the specified channel number to difference computation.

- If the range type is set to the reference range, the measurement range of the target channel is set to the same range as the reference channel.
- The span, scale, and alarm are set to the default values.

Return value

Returns an error number.

Error:

Not descriptor	No device descriptor.
----------------	-----------------------

Reference

CDAQMX100::setChDELTA

setChKindMX100

Syntax

```
int setChKindMX100(DAQMX100 daqmx100, int chNo, int iKind, int refChNo);
```

Declaration

Visual Basic

```
Public Declare Function setChKindMX100 Lib "DAQMX100" (ByVal daqmx100 As Long, ByVal chNo As Long, ByVal iKind As Long, ByVal refChNo As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function setChKindMX100 Lib "DAQMX100" (ByVal daqmx100 As Integer, ByVal chNo As Integer, ByVal iKind As Integer, ByVal refChNo As Integer) As Integer
```

C#

```
[DllImport("DAQMX100.dll" CharSet=CharSet.Auto, EntryPoint="setChKindMX100")]  
public static extern int setChKindMX100(int daqmx100, int chNo, int iKind, int refChNo);
```

Parameters

daqmx100	Specify the device descriptor.
chNo	Specify the channel number.
iKind	Specify the channel type.
refChNo	Specify the reference channel using a channel number.

Description

Sets the channel type to the specified channel numbers.

- The range, span, scale, alarm and other setup items are set to the default values.
- The reference channel specification is valid when the channel type is difference between channels, remote RJC, AO, or PWM.

Return value

Returns an error number.

Error:

Not descriptor	No device descriptor.
----------------	-----------------------

Reference

CDAQMX100::setChKind

setChoiceMX100

Syntax

```
int setChoiceMX100(DAQMX100 daqmx100, int outputNo, int
idleChoice, int errorChoice, int presetValue);
```

Declaration**Visual Basic**

```
Public Declare Function setChoiceMX100 Lib "DAQMX100" (ByVal
daqmx100 As Long, ByVal outputNo As Long, ByVal idleChoice As
Long, ByVal errorChoice As Long, ByVal presetValue As Long) As
Long
```

Visual Basic.NET

```
Public Declare Ansi Function setChoiceMX100 Lib
"DAQMX100" (ByVal daqmx100 As Integer, ByVal outputNo As
Integer, ByVal idleChoice As Integer, ByVal errorChoice As
Integer, ByVal presetValue As Integer) As Integer
```

C#

```
[DllImport("DAQMX100.dll" CharSet=CharSet.Auto,
EntryPoint="setChoiceMX100")]
public static extern int setChoiceMX100(int daqmx100, int
outputNo, int idleChoice, int errorChoice, int presetValue);
```

Parameters

daqmx100	Specify the device descriptor.
outputNo	Specify the output channel (AO/PWM data number).
idleChoice	Specify the selected value when idle.
errorChoice	Specify the selected value when an error occurs.
presetValue	Specify the output value if the selected value is the "specified value."

Description

Sets the output during idling and when errors occur for the specified output channel.

- The user-specified output value specifies an integer excluding the decimal point position.

Return value

Returns an error number.

Error:

Not descriptor	No device descriptor.
----------------	-----------------------

Reference

CDAQMX100::setChoice

setChRRJCMX100

Syntax

```
int setChRRJCMX100(DAQMX100 daqmx100, int chNo, int refChNo);
```

Declaration

Visual Basic

```
Public Declare Function setChRRJCMX100 Lib "DAQMX100" (ByVal daqmx100 As Long, ByVal chNo As Long, ByVal refChNo As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function setChRRJCMX100 Lib "DAQMX100" (ByVal daqmx100 As Integer, ByVal chNo As Integer, ByVal refChNo As Integer) As Integer
```

C#

```
[DllImport("DAQMX100.dll" CharSet=CharSet.Auto, EntryPoint="setChRRJCMX100")]  
public static extern int setChRRJCMX100(int daqmx100, int chNo, int refChNo);
```

Parameters

daqmx100	Specify the device descriptor.
chNo	Specify the channel number.
refChNo	Specify the reference channel using a channel number.

Description

Sets the remote RJC on the specified channel number.

- The span and alarm are set to the default values.

Return value

Returns an error number.

Error:

Not descriptor	No device descriptor.
----------------	-----------------------

Reference

CDAQMX100::setChRRJC

setChTagMX100

Syntax

```
int setChTagMX100(DAQMX100 daqmx100, int chNo, const char *
strTag);
```

Declaration**Visual Basic**

```
Public Declare Function setChTagMX100 Lib "DAQMX100" (ByVal
daqmx100 As Long, ByVal chNo As Long, ByVal strTag As String)
As Long
```

Visual Basic.NET

```
Public Declare Ansi Function setChTagMX100 Lib
"DAQMX100" (ByVal daqmx100 As Integer, ByVal chNo As Integer,
ByVal strTag As String) As Integer
```

C#

```
[DllImport("DAQMX100.dll" CharSet=CharSet.Auto,
EntryPoint="setChTagMX100")]
public static extern int setChTagMX100(int daqmx100, int chNo,
int byte[] strTag);
```

Parameters

daqmx100	Specify the device descriptor.
chNo	Specify the channel number.
strTag	Specify the tag.

Description

Sets the tag to the specified channel number.

- The specified string is, in general, an ASCII string.

Return value

Returns an error number.

Error:

Not descriptor	No device descriptor.
----------------	-----------------------

Reference

CDAQMX100::setChTag

setChUnitMX100

Syntax

```
int setChUnitMX100(DAQMX100 daqmx100, int chNo, const char * strUnit);
```

Declaration

Visual Basic

```
Public Declare Function setChUnitMX100 Lib "DAQMX100" (ByVal daqmx100 As Long, ByVal chNo As Long, ByVal strUnit As String) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function setChUnitMX100 Lib "DAQMX100" (ByVal daqmx100 As Integer, ByVal chNo As Integer, ByVal strUnit As String) As Integer
```

C#

```
[DllImport("DAQMX100.dll" CharSet=CharSet.Auto, EntryPoint="setChUnitMX100")]  
public static extern int setChUnitMX100(int daqmx100, int chNo, byte[] strUnit);
```

Parameters

daqmx100	Specify the device descriptor.
chNo	Specify the channel number.
strUnit	Specify the unit name.

Description

Sets the unit name to the specified channel number.

- The specified string is, in general, an ASCII string.

Return value

Returns an error number.

Error:

Not descriptor	No device descriptor.
----------------	-----------------------

Reference

CDAQMX100::setChUnit

setDateTimeNowMX100

Syntax

```
int setDateTimeNowMX100(DAQMX100 daqmx100);
```

Declaration

Visual Basic

```
Public Declare Function setDateTimeNowMX100 Lib
"DAQMX100"(ByVal daqmx100 As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function setDateTimeNowMX100 Lib
"DAQMX100"(ByVal daqmx100 As Integer) As Integer
```

C#

```
[DllImport("DAQMX100.dll" CharSet=CharSet.Auto,
EntryPoint="setDateTimeNowMX100")]
public static extern int setDateTimeNowMX100(int daqmx100);
```

Parameters

daqmx100 Specify the device descriptor.

Description

Sets the current date/time of the PC.

- Milliseconds are discarded.
- The response to this function may take one second or longer.

Return value

Returns an error number.

Error:

Not descriptor No device descriptor.

Reference

CDAQMX100::setDateTime

setDeenergizeMX100

Syntax

```
int setDeenergizeMX100(DAQMX100 daqmx100, int doNo, int bDeenergize);
```

Declaration

Visual Basic

```
Public Declare Function setDeenergizeMX100 Lib "DAQMX100" (ByVal daqmx100 As Long, ByVal doNo As Long, ByVal bDeenergize As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function setDeenergizeMX100 Lib "DAQMX100" (ByVal daqmx100 As Integer, ByVal doNo As Integer, ByVal bDeenergize As Integer) As Integer
```

C#

```
[DllImport("DAQMX100.dll" CharSet=CharSet.Auto, EntryPoint="setDeenergizeMX100")]  
public static extern int setDeenergizeMX100(int daqmx100, int doNo, int bDeenergize);
```

Parameters

daqmx100	Specify the device descriptor.
doNo	Specify the data number.
bDeenergize	Specify de-energize using a valid/invalid value.

Description

Sets de-energize on the specified DO data number of the specified DO channel.

Return value

Returns an error number.

Error:

Not descriptor	No device descriptor.
----------------	-----------------------

Reference

CDAQMX100::setDeenergize

setDoubleAlarmMX100

Syntax

```
int setDoubleAlarmMX100(DAQMX100 daqmx100, int chNo, int levelNo, int iAlarmType, double value);
```

Declaration**Visual Basic**

```
Public Declare Function setDoubleAlarmMX100 Lib "DAQMX100" (ByVal daqmx100 As Long, ByVal chNo As Long, ByVal levelNo As Long, ByVal iAlarmType As Long, ByVal value As Double) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function setDoubleAlarmMX100 Lib "DAQMX100" (ByVal daqmx100 As Integer, ByVal chNo As Integer, ByVal levelNo As Integer, ByVal iAlarmType As Integer, ByVal value As Double) As Integer
```

C#

```
[DllImport("DAQMX100.dll" CharSet=CharSet.Auto, EntryPoint="setDoubleAlarmMX100")]
public static extern int setDoubleAlarmMX100(int daqmx100, int chNo, int levelNo, int iAlarmType, double value);
```

Parameters

daqmx100	Specify the device descriptor.
chNo	Specify the channel number.
levelNo	Specify the alarm level.
iAlarmType	Specify the alarm type.
value	Specify the alarm value.

Description

Sets the alarm to the alarm level of the specified channel.

- If the differences from the specified alarm value are excluded, it is the same as the setAlarmMX100 function.
- The alarm value is specified with a floating point number including the decimal point position.

Return value

Returns an error number.

Error:

Not descriptor No device descriptor.

Reference

CDAQMX100::setAlarm

setDoubleAlarmValueMX100

Syntax

```
int setDoubleAlarmValueMX100(DAQMX100 daqmx100, int chNo, int levelNo, int iAlarmType, double valueON, double valueOFF)
```

Declaration

Visual Basic

```
Public Declare Function setDoubleAlarmValueMX100 Lib "DAQMX100" (ByVal daqmx100 As Long, ByVal chNo As Long, ByVal levelNo As Long, ByVal iAlarmType As Long, ByVal valueON As Double, ByVal valueOFF As Double) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function setDoubleAlarmValueMX100 Lib "DAQMX100" (ByVal daqmx100 As Integer, ByVal chNo As Integer, ByVal levelNo As Integer, ByVal iAlarmType As Integer, ByVal valueON As Double, ByVal valueOFF As Double) As Integer
```

C#

```
[DllImport("DAQMX100.dll" CharSet=CharSet.Auto, EntryPoint="setDoubleAlarmValueMX100")]  
public static extern int setDoubleAlarmValueMX100(int daqmx100, int chNo, int levelNo, int iAlarmType, double valueON, double valueOFF);
```

Parameters

daqmx100	Specify the device descriptor.
chNo	Specify the channel number.
levelNo	Specify the alarm level.
iAlarmType	Specify the alarm type.
valueON	Specify the threshold level (ON value) for alarm activation.
valueOFF	Specify the threshold level (OFF value) for alarm termination.

Description

Sets the alarm to the alarm level of the specified channel.

- If the differences from the specified threshold value are excluded, it is the same as the setAlarmValueMX100 function.
- The threshold value is specified with a floating point number including the decimal point position.

Return value

Returns an error number.

Error:

Not descriptor	No device descriptor.
----------------	-----------------------

Reference

CDAQMX100::setAlarm

setDoubleChoiceMX100

Syntax

```
int setDoubleChoiceMX100(DAQMX100 daqmx100, int outputNo, int
idleChoice, int errorChoice, double presetValue);
```

Declaration**Visual Basic**

```
Public Declare Function setDoubleChoiceMX100 Lib
"DAQMX100"(ByVal daqmx100 As Long, ByVal outputNo As Long,
ByVal idleChoice As Long, ByVal errorChoice As Long, ByVal
presetValue As Double) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function setDoubleChoiceMX100 Lib
"DAQMX100"(ByVal daqmx100 As Integer, ByVal outputNo As
Integer, ByVal idleChoice As Integer, ByVal errorChoice As
Integer, ByVal presetValue As Double) As Integer
```

C#

```
[DllImport("DAQMX100.dll" CharSet=CharSet.Auto,
EntryPoint="setDoubleChoiceMX100")]
public static extern int setDoubleChoiceMX100(int daqmx100,
int outputNo, int idleChoice, int errorChoice, double
presetValue);
```

Parameters

daqmx100	Specify the device descriptor.
outputNo	Specify the output channel (AO/PWM data number).
idleChoice	Specify the selected value when idling.
errorChoice	Specify the selected value when an error occurs.
presetValue	Specify the output value if the selected value is the "specified value."

Description

Sets the output during idling and when errors occur for the specified output channel.

- If the differences from the user specified output value are excluded, it is the same as the setChoiceMX100 function.
- The user-specified output value specifies a floating point value including the decimal point position.

Return value

Returns an error number.

Error:

Not descriptor	No device descriptor.
----------------	-----------------------

Reference

CDAQMX100::setChoice

setDoubleHisterisysMX100

Syntax

```
int setDoubleHisterisysMX100(DAQMX100 daqmx100, int chNo, int levelNo, double histerisys);
```

Declaration

Visual Basic

```
Public Declare Function setDoubleHisterisysMX100 Lib "DAQMX100" (ByVal daqmx100 As Long, ByVal chNo As Long, ByVal levelNo As Long, ByVal histerisys As Double) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function setDoubleHisterisysMX100 Lib "DAQMX100" (ByVal daqmx100 As Integer, ByVal chNo As Integer, ByVal levelNo As Integer, ByVal histerisys As Double) As Integer
```

C#

```
[DllImport("DAQMX100.dll" CharSet=CharSet.Auto, EntryPoint="setDoubleHisterisysMX100")]  
public static extern int setDoubleHisterisysMX100(int daqmx100, int chNo, int levelNo, double histerisys);
```

Parameters

daqmx100	Specify the device descriptor.
chNo	Specify the channel number.
levelNo	Specify the alarm level.
histerisys	Specify the hysteresis.

Description

Sets the hysteresis for the alarm level of the specified channel number.

- If the differences from the specified hysteresis value are excluded, it is the same as the setHisterisysMX100 function.
- The hysteresis value is specified with a floating point number including the decimal point position.

Return value

Returns an error number.

Error:

Not descriptor	No device descriptor.
----------------	-----------------------

Reference

CDAQMX100::setHisterisys

setDoubleScaleMX100

Syntax

```
int setDoubleScaleMX100(DAQMX100 daqmx100, int chNo, double
scaleMin, double scaleMax, int scalePoint);
```

Declaration**Visual Basic**

```
Public Declare Function setDoubleScaleMX100 Lib
"DAQMX100"(ByVal daqmx100 As Long, ByVal chNo As Long, ByVal
scaleMin As Double, ByVal scaleMax As Double, ByVal scalePoint
As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function setDoubleScaleMX100 Lib
"DAQMX100"(ByVal daqmx100 As Integer, ByVal chNo As Integer,
ByVal scaleMin As Double, ByVal scaleMax As Double, ByVal
scalePoint As Integer) As Integer
```

C#

```
[DllImport("DAQMX100.dll" CharSet=CharSet.Auto,
EntryPoint="setDoubleScaleMX100")]
public static extern int setDoubleScaleMX100(int daqmx100, int
chNo, double scaleMin, double scaleMax, int scalePoint);
```

Parameters

daqmx100	Specify the device descriptor.
chNo	Specify the channel number.
scaleMin	Specify the scale minimum.
scaleMax	Specify the scale maximum.
scalePoint	Specify the decimal point position for scaling.

Description

Sets the scale to the specified channel number.

- If the differences from the specified scale value are excluded, it is the same as the setScaleMX100 function.
- The scale value is specified with a floating point number including the decimal point position.

Return value

Returns an error number.

Error:

Not descriptor No device descriptor.

Reference

CDAQMX100::setScale

setDoubleSpanMX100

Syntax

```
int setDoubleSpanMX100(DAQMX100 daqmx100, int chNo, double spanMin, double spanMax);
```

Declaration

Visual Basic

```
Public Declare Function setDoubleSpanMX100 Lib "DAQMX100" (ByVal daqmx100 As Long, ByVal chNo As Long, ByVal spanMin As Double, ByVal spanMax As Double) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function setDoubleSpanMX100 Lib "DAQMX100" (ByVal daqmx100 As Integer, ByVal chNo As Integer, ByVal spanMin As Double, ByVal spanMax As Double) As Integer
```

C#

```
[DllImport("DAQMX100.dll" CharSet=CharSet.Auto, EntryPoint="setDoubleSpanMX100")] public static extern int setDoubleSpanMX100(int daqmx100, int chNo, double spanMin, double spanMax);
```

Parameters

daqmx100	Specify the device descriptor.
chNo	Specify the channel number.
spanMin	Specify the span minimum.
spanMax	Specify the span maximum.

Description

Sets the span to the specified channel number.

- If the differences from the specified span value are excluded, it is the same as the setSpanMX100 function.
- The span value is specified with a floating point number including the decimal point position.

Return value

Returns an error number.

Error:

Not descriptor	No device descriptor.
----------------	-----------------------

Reference

CDAQMX100::setSpan

setFilterMX100

Syntax

```
int setFilterMX100(DAQMX100 daqmx100, int chNo, int iFilter);
```

Declaration

Visual Basic

```
Public Declare Function setFilterMX100 Lib "DAQMX100" (ByVal daqmx100 As Long, ByVal chNo As Long, ByVal iFilter As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function setFilterMX100 Lib "DAQMX100" (ByVal daqmx100 As Integer, ByVal chNo As Integer, ByVal iFilter As Integer) As Integer
```

C#

```
[DllImport("DAQMX100.dll" CharSet=CharSet.Auto, EntryPoint="setFilterMX100")]
public static extern int setFilterMX100(int daqmx100, int chNo, int iFilter);
```

Parameters

daqmx100	Specify the device descriptor.
chNo	Specify the channel number.
iFilter	Specify the filter coefficient.

Description

Sets the filter coefficient to the specified channel number.

Return value

Returns an error number.

Error:

Not descriptor	No device descriptor.
----------------	-----------------------

Reference

CDAQMX100::setFilter

setHisterisysMX100

Syntax

```
int setHisterisysMX100(DAQMx100 daqmx100, int chNo, int levelNo, int histerisys);
```

Declaration

Visual Basic

```
Public Declare Function setHisterisysMX100 Lib "DAQMx100" (ByVal daqmx100 As Long, ByVal chNo As Long, ByVal levelNo As Long, ByVal histerisys As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function setHisterisysMX100 Lib "DAQMx100" (ByVal daqmx100 As Integer, ByVal chNo As Integer, ByVal levelNo As Integer, ByVal histerisys As Integer) As Integer
```

C#

```
[DllImport("DAQMx100.dll" CharSet=CharSet.Auto, EntryPoint="setHisterisysMX100")] public static extern int setHisterisysMX100(int daqmx100, int chNo, int levelNo, int histerisys);
```

Parameters

daqmx100	Specify the device descriptor.
chNo	Specify the channel number.
levelNo	Specify the alarm level.
histerisys	Specify the hysteresis.

Description

Sets the hysteresis for the alarm level of the specified channel number.

- The hysteresis is specified with an integer excluding the decimal point.

Return value

Returns an error number.

Error:

Not descriptor	No device descriptor.
----------------	-----------------------

Reference

CDAQMx100::setHisterisys

setHoldMX100

Syntax

```
int setHoldMX100(DAQMX100 daqmx100, int doNo, int bHold);
```

Declaration

Visual Basic

```
Public Declare Function setHoldMX100 Lib "DAQMX100" (ByVal daqmx100 As Long, ByVal doNo As Long, ByVal bHold As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function setHoldMX100 Lib "DAQMX100" (ByVal daqmx100 As Integer, ByVal doNo As Integer, ByVal bHold As Integer) As Integer
```

C#

```
[DllImport("DAQMX100.dll" CharSet=CharSet.Auto, EntryPoint="setHoldMX100")]
public static extern int setHoldMX100(int daqmx100, int doNo, int bHold);
```

Parameters

daqmx100	Specify the device descriptor.
doNo	Specify the data number.
bHold	Specify hold using a valid/invalid value.

Description

Sets hold on the specified DO data number of the specified DO channel.

Return value

Returns an error number.

Error:

Not descriptor	No device descriptor.
----------------	-----------------------

Reference

CDAQMX100::setHold

setIntegralMX100

Syntax

```
int setIntegralMX100(DAQMX100 daqmx100, int moduleNo, int iIntegral);
```

Declaration

Visual Basic

```
Public Declare Function setIntegralMX100 Lib "DAQMX100" (ByVal daqmx100 As Long, ByVal moduleNo As Long, ByVal iIntegral As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function setIntegralMX100 Lib "DAQMX100" (ByVal daqmx100 As Integer, ByVal moduleNo As Integer, ByVal iIntegral As Integer) As Integer
```

C#

```
[DllImport("DAQMX100.dll" CharSet=CharSet.Auto, EntryPoint="setIntegralMX100")]  
public static extern int setIntegralMX100(int daqmx100, int moduleNo, int iIntegral);
```

Parameters

daqmx100	Specify the device descriptor.
moduleNo	Specify the module number.
iIntegral	Specify the type of A/D integral time.

Description

Sets the A/D integral time type on the module of the specified module number.

Return value

Returns an error number.

Error:

Not descriptor	No device descriptor.
----------------	-----------------------

Reference

CDAQMX100::setIntegral

setIntervalMX100

Syntax

```
int setIntervalMX100(DAQMX100 daqmx100, int moduleNo, int
iInterval);
```

Declaration**Visual Basic**

```
Public Declare Function setIntervalMX100 Lib "DAQMX100" (ByVal
daqmx100 As Long, ByVal moduleNo As Long, ByVal iInterval As
Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function setIntervalMX100 Lib
"DAQMX100" (ByVal daqmx100 As Integer, ByVal moduleNo As
Integer, ByVal iInterval As Integer) As Integer
```

C#

```
[DllImport("DAQMX100.dll" CharSet=CharSet.Auto,
EntryPoint="setIntervalMX100")]
public static extern int setIntervalMX100(int daqmx100, int
moduleNo, int iInterval);
```

Parameters

daqmx100	Specify the device descriptor.
moduleNo	Specify the module number.
iInterval	Specify the interval type.

Description

Sets the interval type on the module of the specified module number.

Return value

Returns an error number.

Error:

Not descriptor	No device descriptor.
----------------	-----------------------

Reference

CDAQMX100::setInterval

setItemAllMX100

Syntax

```
int setItemAllMX100(DAQMX100 daqmx100);
```

Declaration

Visual Basic

```
Public Declare Function setItemAllMX100 Lib "DAQMX100" (ByVal daqmx100 As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function setItemAllMX100 Lib "DAQMX100" (ByVal daqmx100 As Integer) As Integer
```

C#

```
[DllImport("DAQMX100.dll" CharSet=CharSet.Auto, EntryPoint="setItemAllMX100")]  
public static extern int setItemAllMX100(int daqmx100);
```

Parameters

daqmx100 Specify the device descriptor.

Description

Sends the setup data collectively.

Return value

Returns an error number.

Error:

Not descriptor No device descriptor.

Reference

CDAQMX100::setItemAll

setOutputTypeMX100

Syntax

```
int setOutputTypeMX100(DAQMX100 daqmx100, int outputNo, int
iOutput);
```

Declaration**Visual Basic**

```
Public Declare Function setOutputTypeMX100 Lib
"DAQMX100"(ByVal daqmx100 As Long, ByVal outputNo As Long,
ByVal iOutput As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function setOutputTypeMX100 Lib
"DAQMX100"(ByVal daqmx100 As Integer, ByVal outputNo As
Integer, ByVal iOutput As Integer) As Integer
```

C#

```
[DllImport("DAQMX100.dll" CharSet=CharSet.Auto,
EntryPoint="setOutputTypeMX100")]
public static extern int setOutputTypeMX100(int daqmx100, int
outputNo, int iOutput);
```

Parameters

daqmx100	Specify the device descriptor.
outputNo	Specify the output channel (AO/PWM data number).
iOutput	Specify the output type.

Description

Sets the output type on the specified channel number.

- The specified channel name setting item is initialized to the default value.
- Data created with data manipulation is not changed.

Return value

Returns an error number.

Error:

Not descriptor	No device descriptor.
----------------	-----------------------

Reference

CDAQMX100::setOutputType

setPulseTimeMX100

Syntax

```
int setPulseTimeMX100(DAQMX100 daqmx100, int outputNo, int pulseTime);
```

Declaration

Visual Basic

```
Public Declare Function setPulseTimeMX100 Lib "DAQMX100" (ByVal daqmx100 As Long, ByVal outputNo As Long, ByVal pulseTime As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function setPulseTimeMX100 Lib "DAQMX100" (ByVal daqmx100 As Integer, ByVal outputNo As Integer, ByVal pulseTime As Integer) As Integer
```

C#

```
[DllImport("DAQMX100.dll" CharSet=CharSet.Auto, EntryPoint="setPulseTimeMX100")]  
public static extern int setPulseTimeMX100(int daqmx100, int outputNo, int pulseTime);
```

Parameters

daqmx100	Specify the device descriptor.
outputNo	Specify the output channel (PWM data number).
pulseTime	Specify the integer multiple of the pulse interval.

Description

Sets the integer multiple of the pulse interval on the specified channel number.

Return value

Returns an error number.

Error:

Not descriptor	No device descriptor.
----------------	-----------------------

Reference

CDAQMX100::setPulseTime

setRangeMX100

Syntax

```
int setRangeMX100(DAQMX100 daqmx100, int chNo, int iRange);
```

Declaration**Visual Basic**

```
Public Declare Function setRangeMX100 Lib "DAQMX100" (ByVal daqmx100 As Long, ByVal chNo As Long, ByVal iRange As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function setRangeMX100 Lib "DAQMX100" (ByVal daqmx100 As Integer, ByVal chNo As Integer, ByVal iRange As Integer) As Integer
```

C#

```
[DllImport("DAQMX100.dll" CharSet=CharSet.Auto, EntryPoint="setRangeMX100")]
public static extern int setRangeMX100(int daqmx100, int chNo, int iRange);
```

Parameters

daqmx100	Specify the device descriptor.
chNo	Specify the channel number.
iRange	Specify the range type.

Description

Sets the range on the specified channel number.

- The specified channel name setting item is initialized to the default value.
- Data created with data manipulation is not changed.

Return value

Returns an error number.

Error:

Not descriptor	No device descriptor.
----------------	-----------------------

Reference

CDAQMX100::setRange

setRefAlarmMX100

Syntax

```
int setRefAlarmMX100(DAQMX100 daqmx100, int doNo, int refChNo,
int levelNo, int bValid);
```

Declaration

Visual Basic

```
Public Declare Function setRefAlarmMX100 Lib "DAQMX100" (ByVal
daqmx100 As Long, ByVal doNo As Long, ByVal refChNo As Long,
ByVal levelNo As Long, ByVal bValid As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function setRefAlarmMX100 Lib
"DAQMX100" (ByVal daqmx100 As Integer, ByVal doNo As Integer,
ByVal refChNo As Integer, ByVal levelNo As Integer, ByVal
bValid As Integer) As Integer
```

C#

```
[DllImport("DAQMX100.dll" CharSet=CharSet.Auto,
EntryPoint="setRefAlarmMX100")]
public static extern int setRefAlarmMX100(int daqmx100, int
doNo, int refChNo, int levelNo, int bValid);
```

Parameters

daqmx100	Specify the device descriptor.
doNo	Specify the data number.
refChNo	Specify the reference channel using a channel number.
levelNo	Specify the alarm level.
bValid	Specify the Boolean value.

Description

Sets the reference alarm on the specified DO data number of the specified DO channel.

- The reference alarm is specified by channel number and alarm level.

Return value

Returns an error number.

Error:

Not descriptor	No device descriptor.
----------------	-----------------------

Reference

CDAQMX100::setRefAlarm

setRJCTypeMX100

Syntax

```
int setRJCTypeMX100(DAQMX100 daqmx100, int chNo, int iRJCType,
int volt);
```

Declaration**Visual Basic**

```
Public Declare Function setRJCTypeMX100 Lib "DAQMX100" (ByVal
daqmx100 As Long, ByVal chNo As Long, ByVal iRJCType As Long,
ByVal volt As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function setRJCTypeMX100 Lib
"DAQMX100" (ByVal daqmx100 As Integer, ByVal chNo As Integer,
ByVal iRJCType As Integer, ByVal volt As Integer) As Integer
```

C#

```
[DllImport("DAQMX100.dll" CharSet=CharSet.Auto,
EntryPoint="setRJCTypeMX100")]
public static extern int setRJCTypeMX100(int daqmx100, int
chNo, int iRJCType, int volt);
```

Parameters

daqmx100	Specify the device descriptor.
chNo	Specify the channel number.
iRJCType	Specify the RJC type.
volt	Specify the RJC voltage.

Description

Sets the RJC association on the specified channel number.

Return value

Returns an error number.

Error:

Not descriptor	No device descriptor.
----------------	-----------------------

Reference

CDAQMX100::setRJCType

setScaleMX100

Syntax

```
int setScaleMX100(DAQMX100 daqmx100, int chNo, int scaleMin,
int scaleMax, int scalePoint);
```

Declaration

Visual Basic

```
Public Declare Function setScaleMX100 Lib "DAQMX100" (ByVal
daqmx100 As Long, ByVal chNo As Long, ByVal scaleMin As Long,
ByVal scaleMax As Long, ByVal scalePoint As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function setScaleMX100 Lib
"DAQMX100" (ByVal daqmx100 As Integer, ByVal chNo As Integer,
ByVal scaleMin As Integer, ByVal scaleMax As Integer, ByVal
scalePoint As Integer) As Integer
```

C#

```
[DllImport("DAQMX100.dll" CharSet=CharSet.Auto,
EntryPoint="setScaleMX100")]
public static extern int setScaleMX100(int daqmx100, int chNo,
int scaleMin, int scaleMax, int scalePoint);
```

Parameters

daqmx100	Specify the device descriptor.
chNo	Specify the channel number.
scaleMin	Specify the scale minimum.
scaleMax	Specify the scale maximum.
scalePoint	Specify the decimal point position for scaling.

Description

Sets the scale to the specified channel number.

- Set according to the specified range type.
- If the scale value is outside the range, it is rounded to a possible value.
- If the minimum and maximum values are equal, the scale is "No scale."
- The scale value is specified with an integer excluding the decimal point.

Return value

Returns an error number.

Error:

Not descriptor	No device descriptor.
----------------	-----------------------

Reference

CDAQMX100::setScale

setSpanMX100

Syntax

```
int setSpanMX100(DAQMX100 daqmx100, int chNo, int spanMin, int spanMax);
```

Declaration**Visual Basic**

```
Public Declare Function setSpanMX100 Lib "DAQMX100" (ByVal daqmx100 As Long, ByVal chNo As Long, ByVal spanMin As Long, ByVal spanMax As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function setSpanMX100 Lib "DAQMX100" (ByVal daqmx100 As Integer, ByVal chNo As Integer, ByVal spanMin As Integer, ByVal spanMax As Integer) As Integer
```

C#

```
[DllImport("DAQMX100.dll" CharSet=CharSet.Auto, EntryPoint="setSpanMX100")]
public static extern int setSpanMX100(int daqmx100, int chNo, int spanMin, int spanMax);
```

Parameters

daqmx100	Specify the device descriptor.
chNo	Specify the channel number.
spanMin	Specify the span minimum.
spanMax	Specify the span maximum.

Description

Sets the span to the specified channel number.

- Set according to the specified range type.
- If the span value is outside the range, it is rounded to a possible value.
- If the minimum and maximum values are equal, the default value is set.
- The span value is specified with an integer excluding the decimal point.

Return value

Returns an error number.

Error:

Not descriptor	No device descriptor.
----------------	-----------------------

Reference

CDAQMX100::setSpan

setUnitNoMX100

Syntax

```
int setUnitNoMX100(DAQMX100 daqmx100, int unitNo);
```

Declaration

Visual Basic

```
Public Declare Function setUnitNoMX100 Lib "DAQMX100" (ByVal daqmx100 As Long, ByVal unitNo As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function setUnitNoMX100 Lib "DAQMX100" (ByVal daqmx100 As Integer, ByVal unitNo As Integer) As Integer
```

C#

```
[DllImport("DAQMX100.dll" CharSet=CharSet.Auto, EntryPoint="setUnitNoMX100")]  
public static extern int setUnitNoMX100(int daqmx100, int unitNo);
```

Parameters

daqmx100	Specify the device descriptor.
unitNo	Specify the unit number.

Description

Sets the unit number.

Return value

Returns an error number.

Error:

Not descriptor	No device descriptor.
----------------	-----------------------

Reference

CDAQMX100::setUnitNo

setUnitTempMX100

Syntax

```
int setUnitTempMX100(DAQMX100 daqmx100, int iTempUnit);
```

Declaration**Visual Basic**

```
Public Declare Function setUnitTempMX100 Lib "DAQMX100" (ByVal daqmx100 As Long, ByVal iTempUnit As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function setUnitTempMX100 Lib "DAQMX100" (ByVal daqmx100 As Integer, ByVal iTempUnit As Integer) As Integer
```

C#

```
[DllImport("DAQMX100.dll" CharSet=CharSet.Auto, EntryPoint="setUnitTempMX100")]
public static extern int setUnitTempMX100(int daqmx100, int iTempUnit);
```

Parameters

daqmx100	Specify the device descriptor.
iTempUnit	Specify the temperature unit type.

Description

Sets the temperature unit type.

Return value

Returns an error number.

Not descriptor	No device descriptor.
----------------	-----------------------

Reference

CDAQMX100::setUnitTemp

switchBackupMX100

Syntax

```
int switchBackupMX100(DAQMX100 daqmx100, int bBackup);
```

Declaration

Visual Basic

```
Public Declare Function switchBackupMX100 Lib "DAQMX100" (ByVal daqmx100 As Long, ByVal bBackup As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function switchBackupMX100 Lib "DAQMX100" (ByVal daqmx100 As Integer, ByVal bBackup As Integer) As Integer
```

C#

```
[DllImport("DAQMX100.dll" CharSet=CharSet.Auto, EntryPoint="switchBackupMX100")]  
public static extern int switchBackupMX100(int daqmx100, int bBackup);
```

Parameters

daqmx100	Specify the device descriptor.
bBackup	Specify the Boolean value.

Description

Sets backup.

Return value

Returns an error number.

Error:

Not descriptor	No device descriptor.
----------------	-----------------------

Reference

CDAQMX100::switchBackup

switchDOMX100

Syntax

```
int switchDOMX100(DAQMX100 daqmx100, int idDO, int bONOFF);
```

Declaration**Visual Basic**

```
Public Declare Function switchDOMX100 Lib "DAQMX100" (ByVal daqmx100 As Long, ByVal idDO As Long, ByVal bONOFF As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function switchDOMX100 Lib "DAQMX100" (ByVal daqmx100 As Integer, ByVal idDO As Integer, ByVal bONOFF As Integer) As Integer
```

C#

```
[DllImport("DAQMX100.dll" CharSet=CharSet.Auto, EntryPoint="switchDOMX100")]
public static extern int switchDOMX100(int daqmx100, int idDO, int bONOFF);
```

Parameters

daqmx100	Specify the device descriptor.
idDO	Specify the DO data identifier.
bONOFF	Specify ON/OFF using a Boolean value.

Description

Sends the DO data of the specified DO data identifier.

- Changes the valid channels of the DO data to the specified ON/OFF value and sends it.

Return value

Returns an error number.

Error:

Not descriptor	No device descriptor.
----------------	-----------------------

Reference

CDAQMX100::switchDO

switchTransmitMX100

Syntax

```
int switchTransmitMX100(DAQMX100 daqmx100, int idTrans, int  
iTransmit);
```

Declaration

Visual Basic

```
Public Declare Function switchTransmitMX100 Lib  
"DAQMX100"(ByVal daqmx100 As Long, ByVal idTrans As Long,  
ByVal iTransmit As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function switchTransmitMX100 Lib  
"DAQMX100"(ByVal daqmx100 As Integer, ByVal idTrans As  
Integer, ByVal iTransmit As Integer) As Integer
```

C#

```
[DllImport("DAQMX100.dll" CharSet=CharSet.Auto,  
EntryPoint="switchTransmitMX100")]  
public static extern int switchTransmitMX100(int daqmx100, int  
idTrans, int iTransmit);
```

Parameters

daqmx100	Specify the device descriptor.
idTrans	Specify the transmission output data identifier.
iTransmit	Specify the transmission status.

Description

Sends the transmission output data of the specified transmission output data identifier.

- Changes all channels of the transmission output data to the specified transmission status and sends it.

Return value

Returns an error number.

Error:

Not descriptor	No device descriptor.
----------------	-----------------------

Reference

CDAQMX100::switchTransmit

updateAOPWMDDataMX100

Syntax

```
int updateAOPWMDDataMX100(DAQMX100 daqmx100);
```

Declaration

Visual Basic

```
Public Declare Function updateAOPWMDDataMX100 Lib  
"DAQMX100"(ByVal daqmx100 As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function updateAOPWMDDataMX100 Lib  
"DAQMX100"(ByVal daqmx100 As Integer) As Integer
```

C#

```
[DllImport("DAQMX100.dll" CharSet=CharSet.Auto,  
EntryPoint="updateAOPWMDDataMX100")]  
public static extern int updateAOPWMDDataMX100(int daqmx100);
```

Parameters

daqmx100 Specify the device descriptor.

Description

Updates the stored AO/PWM data and transmission output data.

Return value

Returns an error number.

Error:

Not descriptor No device descriptor.

Reference

CDAQMX100::updateAOPWMDData

updateBalanceMX100

Syntax

```
int updateBalanceMX100(DAQMX100 daqmx100);
```

Declaration

Visual Basic

```
Public Declare Function updateBalanceMX100 Lib  
"DAQMX100"(ByVal daqmx100 As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function updateBalanceMX100 Lib  
"DAQMX100"(ByVal daqmx100 As Integer) As Integer
```

C#

```
[DllImport("DAQMX100.dll" CharSet=CharSet.Auto,  
EntryPoint="updateBalanceMX100")]  
public static extern int updateBalanceMX100(int daqmx100);
```

Parameters

daqmx100 Specify the device descriptor.

Description

Updates the stored initial balance data.

Return value

Returns an error number.

Error:

Not descriptor No device descriptor.

Reference

CDAQMX100::updateBalance

updateConfigMX100

Syntax

```
int updateConfigMX100(DAQMX100 daqmx100);
```

Declaration

Visual Basic

```
Public Declare Function updateConfigMX100 Lib "DAQMX100" (ByVal daqmx100 As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function updateConfigMX100 Lib "DAQMX100" (ByVal daqmx100 As Integer) As Integer
```

C#

```
[DllImport("DAQMX100.dll", CharSet=CharSet.Auto, EntryPoint="updateConfigMX100")]  
public static extern int updateConfigMX100(int daqmx100);
```

Parameters

daqmx100 Specify the device descriptor.

Description

Updates the stored setup data.

Return value

Returns an error number.

Error:

Not descriptor No device descriptor.

Reference

CDAQMX100::updateConfig

updateDODataMX100

Syntax

```
int updateDODataMX100(DAQMX100 daqmx100);
```

Declaration

Visual Basic

```
Public Declare Function updateDODataMX100 Lib "DAQMX100" (ByVal daqmx100 As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function updateDODataMX100 Lib "DAQMX100" (ByVal daqmx100 As Integer) As Integer
```

C#

```
[DllImport("DAQMX100.dll" CharSet=CharSet.Auto, EntryPoint="updateDODataMX100")]  
public static extern int updateDODataMX100(int daqmx100);
```

Parameters

daqmx100 Specify the device descriptor.

Description

Updates the stored DO data.

Return value

Returns an error number.

Error:

Not descriptor No device descriptor.

Reference

CDAQMX100::updateDOData

updateInfoChMX100

Syntax

```
int updateInfoChMX100(DAQMX100 daqmx100, int chNo);
```

Declaration

Visual Basic

```
Public Declare Function updateInfoChMX100 Lib "DAQMX100" (ByVal daqmx100 As Long, ByVal chNo As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function updateInfoChMX100 Lib "DAQMX100" (ByVal daqmx100 As Integer, ByVal chNo As Integer) As Integer
```

C#

```
[DllImport("DAQMX100.dll" CharSet=CharSet.Auto, EntryPoint="updateInfoChMX100")]
public static extern int updateInfoChMX100(int daqmx100, int chNo);
```

Parameters

daqmx100	Specify the device descriptor.
chNo	Specify the channel number.

Description

Changes the channel information data of the specified channel number.

Returns an error number.

Error:

Not descriptor	No device descriptor.
----------------	-----------------------

Reference

CDAQMX100::updateInfoCh

updateOutputMX100

Syntax

```
int updateOutputMX100(DAQMX100 daqmx100);
```

Declaration

Visual Basic

```
Public Declare Function updateOutputMX100 Lib "DAQMX100" (ByVal daqmx100 As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function updateOutputMX100 Lib "DAQMX100" (ByVal daqmx100 As Integer) As Integer
```

C#

```
[DllImport("DAQMX100.dll" CharSet=CharSet.Auto, EntryPoint="updateOutputMX100")]  
public static extern int updateOutputMX100(int daqmx100);
```

Parameters

daqmx100 Specify the device descriptor.

Description

Updates the stored output channel data.

Return value

Returns an error number.

Error:

Not descriptor No device descriptor.

Reference

CDAQMX100::updateOutput

updateStatusMX100

Syntax

```
int updateStatusMX100(DAQMX100 daqmx100);
```

Declaration

Visual Basic

```
Public Declare Function updateStatusMX100 Lib "DAQMX100" (ByVal daqmx100 As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function updateStatusMX100 Lib "DAQMX100" (ByVal daqmx100 As Integer) As Integer
```

C#

```
[DllImport("DAQMX100.dll" CharSet=CharSet.Auto, EntryPoint="updateStatusMX100")]  
public static extern int updateStatusMX100(int daqmx100);
```

Parameters

daqmx100 Specify the device descriptor.

Description

Updates the stored status data.

Return value

Returns an error number.

Error:

Not descriptor No device descriptor.

Reference

CDAQMX100::updateStatus

updateSystemMX100

Syntax

```
int updateSystemMX100(DAQMX100 daqmx100);
```

Declaration

Visual Basic

```
Public Declare Function updateSystemMX100 Lib "DAQMX100" (ByVal daqmx100 As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function updateSystemMX100 Lib "DAQMX100" (ByVal daqmx100 As Integer) As Integer
```

C#

```
[DllImport("DAQMX100.dll" CharSet=CharSet.Auto, EntryPoint="updateSystemMX100")]  
public static extern int updateSystemMX100(int daqmx100);
```

Parameters

daqmx100 Specify the device descriptor.

Description

Updates the stored system configuration data.

Return value

Returns an error number.

Error:

Not descriptor No device descriptor.

Reference

CDAQMX100::updateSystem

writeltemMX100

Syntax

```
int writeItemMX100(DAQMX100 daqmx100, int itemNo, const char *
strItem);
```

Declaration**Visual Basic**

```
Public Declare Function writeItemMX100 Lib "DAQMX100" (ByVal
daqmx100 As Long, ByVal itemNo As Long, ByVal strItem As
String, ByVal) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function writeItemMX100 Lib
"DAQMX100" (ByVal daqmx100 As Integer, ByVal itemNo As Integer,
ByVal strItem As String, ByVal) As Integer
```

C#

```
[DllImport("DAQMX100.dll" CharSet=CharSet.Auto,
EntryPoint="writeItemMX100")]
public static extern int writeItemMX100(int daqmx100, int
itemNo, byte[] strItem);
```

Parameters

daqmx100	Specify the device descriptor.
itemNo	Specify the setup item number.
strItem	Specify the item contents using strings.

Description

Writes the specified string contents to the specified setting item.

- Overwrites the stored field. Validity checks are not performed.
- The results of each retrieval function are not necessarily correct.
- The specified string is, in general, an ASCII string.

Return value

Returns an error number.

Error:

Not descriptor	No device descriptor.
----------------	-----------------------

Reference

```
CDAQMX100::getClassMXItemConfig
CDAQMXItemConfig::writeItem
```

17.2 Details of Function - MX00 (Visual C/Visual Basic/ Visual Basic.NET/C#) - Retrieval Functions

This section describes the MX100 functions that are used in C and Visual Basic. The functions are listed in alphabetical order by the function name.

For details on constants and types, see chapter 18.

For MX100 terminology, see appendix 1.

Most functions return an error number as a return value. Error number 0 is returned if there is no error.

See page v “Conventions Used in This Manual” for more information about return values.

“The value does not exist” means the specification of the function is different, the range is different, the field does not exist, or retrieval failed.

In C#, the class (DAQMX100) member contains the declarations.

addressPartMX100

Syntax

```
int addressPartMX100(unsigned int address, int index);
```

Declaration

Visual Basic

```
Public Declare Function addressPartMX100 Lib "DAQMX100" (ByVal  
address As Long, ByVal index As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function addressPartMX100 Lib  
"DAQMX100" (ByVal address As Integer, ByVal index As Integer)  
As Integer
```

C#

```
[DllImport("DAQMX100.dll" CharSet=CharSet.Auto,  
EntryPoint="addressPartMX100")]  
public static extern int addressPartMX100(int address, int  
index);
```

Parameters

address	Specify the IP address.
index	Specify the position of the part.

Description

Gets the byte value of the portion of the specified IP address.

- Returns the byte value of the specified part position.
- The part position is specified with the index value (starting from 0) in units of bytes. The range is from 0 to 3.
- Returns 0 if it does not exist.

Return value

Returns the byte value.

Reference

CDAQMXNetInfo::getPart

alarmDoubleHisterisysMX100

Syntax

```
double alarmDoubleHisterisysMX100(DAQMX100 daqmx100, int chNo,
int levelNo);
```

Declaration

Visual Basic

```
Public Declare Function alarmDoubleHisterisysMX100 Lib
"DAQMX100"(ByVal daqmx100 As Long, ByVal chNo As Long, ByVal
levelNo As Long) As Double
```

Visual Basic.NET

```
Public Declare Ansi Function alarmDoubleHisterisysMX100 Lib
"DAQMX100"(ByVal daqmx100 As Integer, ByVal chNo As Integer,
ByVal levelNo As Integer) As Double
```

C#

```
[DllImport("DAQMX100.dll" CharSet=CharSet.Auto,
EntryPoint="alarmDoubleHisterisysMX100")]
public static extern double alarmDoubleHisterisysMX100(int
daqmx100, int chNo, int levelNo);
```

Parameters

daqmx100	Specify the device descriptor.
chNo	Specify the channel number.
levelNo	Specify the alarm level.

Description

Gets the specified channel number and alarm level hysteresis from the stored current channel setting data.

- The return value is a floating point number including the decimal point position.
- Returns 0.0 if it does not exist.

Return value

Returns the hysteresis.

Reference

```
CDAQMX100::getClassMXItemConfig
CDAQMXItemConfig::getDoubleHisterisys
```

alarmDoubleValueOFFMX100

Syntax

```
double alarmDoubleValueOFFMX100(DAQMX100 daqmx100, int chNo,
int levelNo);
```

Declaration

Visual Basic

```
Public Declare Function alarmDoubleValueOFFMX100 Lib
"DAQMX100"(ByVal daqmx100 As Long, ByVal chNo As Long, ByVal
levelNo As Long) As Double
```

Visual Basic.NET

```
Public Declare Ansi Function alarmDoubleValueOFFMX100 Lib
"DAQMX100"(ByVal daqmx100 As Integer, ByVal chNo As Integer,
ByVal levelNo As Integer) As Double
```

C#

```
[DllImport("DAQMX100.dll" CharSet=CharSet.Auto,
EntryPoint="alarmDoubleValueOFFMX100")]
public static extern double alarmDoubleValueOFFMX100(int
daqmx100, int chNo, int levelNo);
```

Parameters

daqmx100	Specify the device descriptor.
chNo	Specify the channel number.
levelNo	Specify the alarm level.

Description

Gets the specified channel number and alarm level alarm value (OFF) from the stored current channel setting data.

- The return value is a floating point number including the decimal point position.
- Returns 0.0 if it does not exist.

Return value

Returns the alarm value (OFF value).

Reference

```
CDAQMX100::getClassMXItemConfig
CDAQMXItemConfig::getDoubleAlarmOFF
```

alarmDoubleValueONMX100

Syntax

```
double alarmDoubleValueONMX100(DAQMX100DAQMX100 daqmx100, int  
chNo, int levelNo);
```

Declaration

Visual Basic

```
Public Declare Function alarmDoubleValueONMX100 Lib  
"DAQMX100"(ByVal daqmx100 As Long, ByVal chNo As Long, ByVal  
levelNo As Long) As Double
```

Visual Basic.NET

```
Public Declare Ansi Function alarmDoubleValueONMX100 Lib  
"DAQMX100"(ByVal daqmx100 As Integer, ByVal chNo As Integer,  
ByVal levelNo As Integer) As Double
```

C#

```
[DllImport("DAQMX100.dll" CharSet=CharSet.Auto,  
EntryPoint="alarmDoubleValueONMX100")]  
public static extern double alarmDoubleValueONMX100(int  
daqmx100, int chNo, int levelNo);
```

Parameters

daqmx100	Specify the device descriptor.
chNo	Specify the channel number.
levelNo	Specify the alarm level.

Description

Gets the specified channel number and alarm level alarm value (ON) from the stored current channel setting data.

- The return value is a floating point number including the decimal point position.
- Returns 0.0 if it does not exist.

Return value

Returns the alarm value (ON value).

Reference

```
CDAQMX100::getClassMXItemConfig  
CDAQMXItemConfig::getDoubleAlarmON
```

alarmHisterisysMX100

Syntax

```
int alarmHisterisysMX100(DAQMX100 daqmx100, int chNo, int levelNo);
```

Declaration

Visual Basic

```
Public Declare Function alarmHisterisysMX100 Lib "DAQMX100"(ByVal daqmx100 As Long, ByVal chNo As Long, ByVal levelNo As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function alarmHisterisysMX100 Lib "DAQMX100"(ByVal daqmx100 As Integer, ByVal chNo As Integer, ByVal levelNo As Integer) As Integer
```

C#

```
[DllImport("DAQMX100.dll" CharSet=CharSet.Auto, EntryPoint="alarmHisterisysMX100")]
public static extern int alarmHisterisysMX100(int daqmx100, int chNo, int levelNo);
```

Parameters

daqmx100	Specify the device descriptor.
chNo	Specify the channel number.
levelNo	Specify the alarm level.

Description

Gets the specified channel number and alarm level hysteresis from the stored current channel setting data.

- The return value is an integer excluding the decimal point position.
- Returns 0 if it does not exist.

Return value

Returns the hysteresis.

Reference

```
CDAQMX100::getClassMXItemConfig
CDAQMXItemConfig::getHisterisys
```

alarmMaxLengthMX100

Syntax

```
int alarmMaxLengthMX100(void);
```

Declaration

Visual Basic

```
Public Declare Function alarmMaxLengthMX100 Lib "DAQMX100"()  
As Long
```

Visual Basic.NET

```
Public Declare Ansi Function alarmMaxLengthMX100 Lib  
"DAQMX100"() As Integer
```

C#

```
[DllImport("DAQMX100.dll" CharSet=CharSet.Auto,  
EntryPoint="alarmMaxLengthMX100")]  
public static extern int alarmMaxLengthMX100();
```

Description

Gets the maximum length of the alarm type.

- The return value does not include the terminator.

Return value

Returns the length of the string.

Reference

CDAQMXDataInfo::getMaxLenAlarmName

alarmTypeMX100

Syntax

```
int alarmTypeMX100(DAQMX100 daqmx100, int chNo, int levelNo);
```

Declaration**Visual Basic**

```
Public Declare Function alarmTypeMX100 Lib "DAQMX100" (ByVal daqmx100 As Long, ByVal chNo As Long, ByVal levelNo As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function alarmTypeMX100 Lib "DAQMX100" (ByVal daqmx100 As Integer, ByVal chNo As Integer, ByVal levelNo As Integer) As Integer
```

C#

```
[DllImport("DAQMX100.dll" CharSet=CharSet.Auto, EntryPoint="alarmTypeMX100")]
public static extern int alarmTypeMX100(int daqmx100, int chNo, int levelNo);
```

Parameters

daqmx100	Specify the device descriptor.
chNo	Specify the channel number.
levelNo	Specify the alarm level.

Description

Gets the specified channel number and alarm level alarm type from the stored current channel setting data.

- If it does not exist, "No alarm" is returned.

Return value

Returns the alarm type.

Reference

```
CDAQMX100::getClassMXItemConfig
CDAQMXChConfig::getAlarmType
CDAQMXItemConfig::getClassMXChConfig
```

alarmValueOFFMX100

Syntax

```
int alarmValueOFFMX100(DAQMx100 daqmx100, int chNo, int levelNo);
```

Declaration

Visual Basic

```
Public Declare Function alarmValueOFFMX100 Lib "DAQMx100" (ByVal daqmx100 As Long, ByVal chNo As Long, ByVal levelNo As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function alarmValueOFFMX100 Lib "DAQMx100" (ByVal daqmx100 As Integer, ByVal chNo As Integer, ByVal levelNo As Integer) As Integer
```

C#

```
[DllImport("DAQMx100.dll", CharSet=CharSet.Auto, EntryPoint="alarmValueOFFMX100")] public static extern int alarmValueOFFMX100(int daqmx100, int chNo, int levelNo);
```

Parameters

daqmx100	Specify the device descriptor.
chNo	Specify the channel number.
levelNo	Specify the alarm level.

Description

Gets the specified channel number and alarm level alarm value (OFF) from the stored current channel setting data.

- The return value is an integer excluding the decimal point position.
- Returns 0 if it does not exist.

Return value

Returns the alarm value (OFF value).

Reference

```
CDAQMx100::getClassMXItemConfig  
CDAQMxChConfig::getAlarmValueOFF  
CDAQMxItemConfig::getClassMXChConfig
```

alarmValueONMX100

Syntax

```
int alarmValueONMX100(DAQMX100 daqmx100, int chNo, int levelNo);
```

Declaration**Visual Basic**

```
Public Declare Function alarmValueONMX100 Lib "DAQMX100" (ByVal daqmx100 As Long, ByVal chNo As Long, ByVal levelNo As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function alarmValueONMX100 Lib "DAQMX100" (ByVal daqmx100 As Integer, ByVal chNo As Integer, ByVal levelNo As Integer) As Integer
```

C#

```
[DllImport("DAQMX100.dll" CharSet=CharSet.Auto, EntryPoint="alarmValueONMX100")]
public static extern int alarmValueONMX100(int daqmx100, int chNo, int levelNo);
```

Parameters

daqmx100	Specify the device descriptor.
chNo	Specify the channel number.
levelNo	Specify the alarm level.

Description

Gets the specified channel number and alarm level alarm value (ON) from the stored current channel setting data.

- The return value is an integer excluding the decimal point position.
- Returns 0 if it does not exist.

Return value

Returns the alarm value (ON value).

Reference

```
CDAQMX100::getClassMXItemConfig
CDAQMXChConfig::getAlarmValueON
CDAQMXItemConfig::getClassMXChConfig
```

channelBalanceValidMX100

Syntax

```
int channelBalanceValidMX100(DAQMX100 daqmx100, int balanceNo);
```

Declaration

Visual Basic

```
Public Declare Function channelBalanceValidMX100 Lib "DAQMX100" (ByVal daqmx100 As Long, ByVal balanceNo As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function channelBalanceValidMX100 Lib "DAQMX100" (ByVal daqmx100 As Integer, ByVal balanceNo As Integer) As Integer
```

C#

```
[DllImport("DAQMX100.dll" CharSet=CharSet.Auto, EntryPoint="channelBalanceValidMX100")] public static extern int channelBalanceValidMX100(int daqmx100, int balanceNo);
```

Parameters

daqmx100	Specify the device descriptor.
balanceNo	Specify the initial balance data number.

Description

Gets valid/invalid for the specified initial balance data number from the stored current channel setting data.

- If it does not exist, Invalid is returned.

Return value

Returns a Boolean value.

Reference

```
CDAQMX100::getClassMXItemConfig  
CDAQMXBalanceData::getBalanceValid  
CDAQMXItemConfig::getClassMXBalanceData
```

channelBalanceValueMX100

Syntax

```
int channelBalanceValueMX100(DAQMX100 daqmx100, int balanceNo);
```

Declaration

Visual Basic

```
Public Declare Function channelBalanceValueMX100 Lib "DAQMX100"(ByVal daqmx100 As Long, ByVal balanceNo As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function channelBalanceValueMX100 Lib "DAQMX100"(ByVal daqmx100 As Integer, ByVal balanceNo As Integer) As Integer
```

C#

```
[DllImport("DAQMX100.dll" CharSet=CharSet.Auto, EntryPoint="channelBalanceValueMX100")]  
public static extern int channelBalanceValueMX100(int daqmx100, int balanceNo);
```

Parameters

daqmx100	Specify the device descriptor.
balanceNo	Specify the initial balance data number.

Description

Gets the initial balance value for the specified initial balance data number from the stored current channel setting data.

- Returns 0 if it does not exist.

Return value

Returns the initial balance value.

Reference

```
CDAQMX100::getClassMXItemConfig  
CDAQMXBalanceData::getBalanceValue  
CDAQMXItemConfig::getClassMXBalanceData
```

channelBurnoutMX100

Syntax

```
int channelBurnoutMX100(DAQMX100 daqmx100, int chNo);
```

Declaration

Visual Basic

```
Public Declare Function channelBurnoutMX100 Lib  
"DAQMX100"(ByVal daqmx100 As Long, ByVal chNo As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function channelBurnoutMX100 Lib  
"DAQMX100"(ByVal daqmx100 As Integer, ByVal chNo As Integer)  
As Integer
```

C#

```
[DllImport("DAQMX100.dll" CharSet=CharSet.Auto,  
EntryPoint="channelBurnoutMX100")]  
public static extern int channelBurnoutMX100(int daqmx100, int  
chNo);
```

Parameters

daqmx100	Specify the device descriptor.
chNo	Specify the channel number.

Description

Gets the specified channel number and burnout type from the stored current channel setting data.

- If it does not exist, Undetected is returned.

Return value

Returns the burnout type.

Reference

CDAQMX100::getClassMXItemConfig
CDAQMXChConfig::getBurnout
CDAQMXItemConfig::getClassMXChConfig

channelDeenergizeMX100

Syntax

```
int channelDeenergizeMX100(DAQMX100 daqmx100, int doNo);
```

Declaration

Visual Basic

```
Public Declare Function channelDeenergizeMX100 Lib  
"DAQMX100"(ByVal daqmx100 As Long, ByVal doNo As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function channelDeenergizeMX100 Lib  
"DAQMX100"(ByVal daqmx100 As Integer, ByVal doNo As Integer)  
As Integer
```

C#

```
[DllImport("DAQMX100.dll" CharSet=CharSet.Auto,  
EntryPoint="channelDeenergizeMX100")]  
public static extern int channelDeenergizeMX100(int daqmx100,  
int doNo);
```

Parameters

daqmx100	Specify the device descriptor.
doNo	Specify the data number.

Description

Gets the de-energize Boolean for the specified DO data number from the stored current channel setting data.

- If it does not exist, Invalid is returned.

Return value

Returns a Boolean value.

Reference

```
CDAQMX100::getClassMXItemConfig  
CDAQMXChConfig::isDeenergize  
CDAQMXItemConfig::getClassMXChConfig
```

channelDisplayMaxMX100

Syntax

```
double channelDisplayMaxMX100(DAQMX100 daqmx100, int chNo);
```

Declaration

Visual Basic

```
Public Declare Function channelDisplayMaxMX100 Lib  
"DAQMX100"(ByVal daqmx100 As Long, ByVal chNo As Long) As  
Double
```

Visual Basic.NET

```
Public Declare Ansi Function channelDisplayMaxMX100 Lib  
"DAQMX100"(ByVal daqmx100 As Integer, ByVal chNo As Integer)  
As Double
```

C#

```
[DllImport("DAQMX100.dll" CharSet=CharSet.Auto,  
EntryPoint="channelDisplayMaxMX100")]  
public static extern double channelDisplayMaxMX100(int  
daqmx100, int chNo);
```

Parameters

daqmx100	Specify the device descriptor.
chNo	Specify the channel number.

Description

Gets the specified channel number and display maximum value from the stored current channel information data.

- Returns 0.0 if it does not exist.

Return value

Returns the display maximum value.

Reference

```
CDAQMX100::getClassMXDataBuffer  
CDAQMXChInfo::getDisplayMax  
CDAQMXDataBuffer::getClassMXChInfo
```

channelDisplayMinMX100

Syntax

```
double channelDisplayMinMX100(DAQMX100 daqmx100, int chNo);
```

Declaration**Visual Basic**

```
Public Declare Function channelDisplayMinMX100 Lib
"DAQMX100"(ByVal daqmx100 As Long, ByVal chNo As Long) As
Double
```

Visual Basic.NET

```
Public Declare Ansi Function channelDisplayMinMX100 Lib
"DAQMX100"(ByVal daqmx100 As Integer, ByVal chNo As Integer)
As Double
```

C#

```
[DllImport("DAQMX100.dll" CharSet=CharSet.Auto,
EntryPoint="channelDisplayMinMX100")]
public static extern double channelDisplayMinMX100(int
daqmx100, int chNo);
```

Parameters

daqmx100	Specify the device descriptor.
chNo	Specify the channel number.

Description

Gets the specified channel number and display minimum value from the stored current channel information data.

- Returns 0.0 if it does not exist.

Return value

Returns the display minimum value.

Reference

```
CDAQMX100::getClassMXDataBuffer
CDAQMXChInfo::getDisplayMin
CDAQMXDataBuffer::getClassMXChInfo
```

channelDoublePresetValueMX100

Syntax

```
double channelDoublePresetValueMX100(DAQMX100 daqmx100, int outputNo);
```

Declaration

Visual Basic

```
Public Declare Function channelDoublePresetValueMX100 Lib "DAQMX100" (ByVal daqmx100 As Long, ByVal outputNo As Long) As Double
```

Visual Basic.NET

```
Public Declare Ansi Function channelDoublePresetValueMX100 Lib "DAQMX100" (ByVal daqmx100 As Integer, ByVal outputNo As Integer) As Double
```

C#

```
[DllImport("DAQMX100.dll" CharSet=CharSet.Auto, EntryPoint="channelDoublePresetValueMX100")] public static extern double channelDoublePresetValueMX100(int daqmx100, int outputNo);
```

Parameters

daqmx100	Specify the device descriptor.
outputNo	Specify the output channel data number.

Description

Gets user specified output value for the specified output channel data number from the stored current channel setting data.

- The return value is a floating point number including the decimal point position.
- Returns 0.0 if it does not exist.

Return value

Returns the user specified output value.

Reference

```
CDAQMX100::getClassMXItemConfig  
CDAQMXItemConfig::getDoublePresetValue
```

channelDoubleScaleMaxMX100

Syntax

```
double channelDoubleScaleMaxMX100(DAQMX100 daqmx100, int
chNo);
```

Declaration**Visual Basic**

```
Public Declare Function channelDoubleScaleMaxMX100 Lib
"DAQMX100"(ByVal daqmx100 As Long, ByVal chNo As Long) As
Double
```

Visual Basic.NET

```
Public Declare Ansi Function channelDoubleScaleMaxMX100 Lib
"DAQMX100"(ByVal daqmx100 As Integer, ByVal chNo As Integer)
As Double
```

C#

```
[DllImport("DAQMX100.dll" CharSet=CharSet.Auto,
EntryPoint="channelDoubleScaleMaxMX100")]
public static extern double channelDoubleScaleMaxMX100(int
daqmx100, int chNo);
```

Parameters

daqmx100	Specify the device descriptor.
chNo	Specify the channel number.

Description

Gets the specified channel number and scale maximum value from the stored current channel setting data.

- The return value is a floating point number including the decimal point position.
- Returns 0.0 if it does not exist.

Return value

Returns the scale maximum.

Reference

```
CDAQMX100::getClassMXItemConfig
CDAQMXItemConfig::getDoubleScaleMax
```

channelDoubleScaleMinMX100

Syntax

```
double channelDoubleScaleMinMX100(DAQMX100 daqmx100, int chNo);
```

Declaration

Visual Basic

```
Public Declare Function channelDoubleScaleMaxMX100 Lib "DAQMX100" (ByVal daqmx100 As Long, ByVal chNo As Long) As Double
```

Visual Basic.NET

```
Public Declare Ansi Function channelDoubleScaleMinMX100 Lib "DAQMX100" (ByVal daqmx100 As Integer, ByVal chNo As Integer) As Double
```

C#

```
[DllImport("DAQMX100.dll" CharSet=CharSet.Auto, EntryPoint="channelDoubleScaleMinMX100")] public static extern double channelDoubleScaleMinMX100(int daqmx100, int chNo);
```

Parameters

daqmx100	Specify the device descriptor.
chNo	Specify the channel number.

Description

Gets the specified channel number and scale minimum value from the stored current channel setting data.

- The return value is a floating point number including the decimal point position.
- Returns 0.0 if it does not exist.

Return value

Returns the scale minimum.

Reference

```
CDAQMX100::getClassMXItemConfig  
CDAQMXItemConfig::getDoubleScaleMin
```

channelDoubleSpanMaxMX100

Syntax

```
double channelDoubleSpanMaxMX100(DAQMx100 daqmx100, int chNo);
```

Declaration

Visual Basic

```
Public Declare Function channelDoubleSpanMaxMX100 Lib  
"DAQMX100"(ByVal daqmx100 As Long, ByVal chNo As Long) As  
Double
```

Visual Basic.NET

```
Public Declare Ansi Function channelDoubleSpanMaxMX100 Lib  
"DAQMX100"(ByVal daqmx100 As Integer, ByVal chNo As Integer)  
As Double
```

C#

```
[DllImport("DAQMX100.dll" CharSet=CharSet.Auto,  
EntryPoint="channelDoubleSpanMaxMX100")]  
public static extern double channelDoubleSpanMaxMX100(int  
daqmx100, int chNo);
```

Parameters

daqmx100	Specify the device descriptor.
chNo	Specify the channel number.

Description

Gets the specified channel number and span maximum value from the stored current channel setting data.

- The return value is a floating point number including the decimal point position.
- Returns 0.0 if it does not exist.

Return value

Returns the span maximum.

Reference

```
CDAQMX100::getClassMXItemConfig  
CDAQMXItemConfig::getDoubleSpanMax
```

channelDoubleSpanMinMX100

Syntax

```
double channelDoubleSpanMinMX100(DAQMx100 daqmx100, int chNo);
```

Declaration

Visual Basic

```
Public Declare Function channelDoubleSpanMinMX100 Lib  
"DAQMX100"(ByVal daqmx100 As Long, ByVal chNo As Long) As  
Double
```

Visual Basic.NET

```
Public Declare Ansi Function channelDoubleSpanMinMX100 Lib  
"DAQMX100"(ByVal daqmx100 As Integer, ByVal chNo As Integer)  
As Double
```

C#

```
[DllImport("DAQMX100.dll" CharSet=CharSet.Auto,  
EntryPoint="channelDoubleSpanMinMX100")]  
public static extern double channelDoubleSpanMinMX100(int  
daqmx100, int chNo);
```

Parameters

daqmx100	Specify the device descriptor.
chNo	Specify the channel number.

Description

Gets the specified channel number and span minimum value from the stored current channel setting.

- The return value is a floating point number including the decimal point position.
- Returns 0.0 if it does not exist.

Return value

Returns the span minimum.

Reference

```
CDAQMX100::getClassMXItemConfig  
CDAQMXItemConfig::getDoubleSpanMin
```

channelErrorChoiceMX100

Syntax

```
int channelErrorChoiceMX100(DAQMX100 daqmx100, int outputNo);
```

Declaration**Visual Basic**

```
Public Declare Function channelErrorChoiceMX100 Lib
"DAQMX100"(ByVal daqmx100 As Long, ByVal outputNo As Long) As
Long
```

Visual Basic.NET

```
Public Declare Ansi Function channelErrorChoiceMX100 Lib
"DAQMX100"(ByVal daqmx100 As Integer, ByVal outputNo As
Integer) As Integer
```

C#

```
[DllImport("DAQMX100.dll" CharSet=CharSet.Auto,
EntryPoint="channelErrorChoiceMX100")]
public static extern int channelErrorChoiceMX100(int daqmx100,
int outputNo);
```

Parameters

daqmx100	Specify the device descriptor.
outputNo	Specify the output channel data number.

Description

Gets selection value during error for the specified output channel data number from the stored current channel setting data.

- If it does not exist, "Previous value" is returned.

Return value

Returns the selected value.

Reference

```
CDAQMX100::getClassMXItemConfig
CDAQMXItemConfig::getClassMXOutputData
CDAQMXOutputData::getErrorChoice
```

channelFIFOIndexMX100

Syntax

```
int channelFIFOIndexMX100(DAQMX100 daqmx100, int chNo);
```

Declaration

Visual Basic

```
Public Declare Function channelFIFOIndexMX100 Lib  
"DAQMX100"(ByVal daqmx100 As Long, ByVal chNo As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function channelFIFOIndexMX100 Lib  
"DAQMX100"(ByVal daqmx100 As Integer, ByVal chNo As Integer)  
As Integer
```

C#

```
[DllImport("DAQMX100.dll" CharSet=CharSet.Auto,  
EntryPoint="channelFIFOIndexMX100")]  
public static extern int channelFIFOIndexMX100(int daqmx100,  
int chNo);
```

Parameters

daqmx100	Specify the device descriptor.
chNo	Specify the channel number.

Description

Gets the FIFO channel sequence number of the specified channel number from the stored current channel information data.

- Returns a negative value if it does not exist.

Return value

Returns the channel sequence number in the FIFO.

Reference

```
CDAQMX100::getClassMXDataBuffer  
CDAQMXChInfo::getFIFOIndex  
CDAQMXDataBuffer::getClassMXChInfo
```

channelFIFONoMX100

Syntax

```
int channelFIFONoMX100(DAQMX100 daqmx100, int chNo);
```

Declaration**Visual Basic**

```
Public Declare Function channelFIFONoMX100 Lib
"DAQMX100"(ByVal daqmx100 As Long, ByVal chNo As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function channelFIFONoMX100 Lib
"DAQMX100"(ByVal daqmx100 As Integer, ByVal chNo As Integer)
As Integer
```

C#

```
[DllImport("DAQMX100.dll" CharSet=CharSet.Auto,
EntryPoint="channelFIFONoMX100")]
public static extern int channelFIFONoMX100(int daqmx100, int
chNo);
```

Parameters

daqmx100	Specify the device descriptor.
chNo	Specify the channel number.

Description

Gets the FIFO number of the specified channel number from the stored current channel information data.

- Returns a negative value if it does not exist.

Return value

Returns the FIFO number.

Reference

```
CDAQMX100::getClassMXDataBuffer
CDAQMXChInfo::getFIFONo
CDAQMXDataBuffer::getClassMXChInfo
```

channelFilterMX100

Syntax

```
int channelFilterMX100(DAQMX100 daqmx100, int chNo);
```

Declaration

Visual Basic

```
Public Declare Function channelFilterMX100 Lib  
"DAQMX100"(ByVal daqmx100 As Long, ByVal chNo As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function channelFilterMX100 Lib  
"DAQMX100"(ByVal daqmx100 As Integer, ByVal chNo As Integer)  
As Integer
```

C#

```
[DllImport("DAQMX100.dll" CharSet=CharSet.Auto,  
EntryPoint="channelFilterMX100")]  
public static extern int channelFilterMX100(int daqmx100, int  
chNo);
```

Parameters

daqmx100	Specify the device descriptor.
chNo	Specify the channel number.

Description

Gets the filter constant of the specified channel number from the stored current channel setting data.

- Returns "Time constant 0" if it does not exist.

Return value

Returns the filter time constant.

Reference

```
CDAQMX100::getClassMXItemConfig  
CDAQMXChConfig::getFilter  
CDAQMXItemConfig::getClassMXChConfig
```

channelHoldMX100

Syntax

```
int channelHoldMX100(DAQMX100 daqmx100, int doNo);
```

Declaration

Visual Basic

```
Public Declare Function channelHoldMX100 Lib
"DAQMX100"(ByVal daqmx100 As Long, ByVal doNo As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function channelHoldMX100 Lib
"DAQMX100"(ByVal daqmx100 As Integer, ByVal doNo As Integer)
As Integer
```

C#

```
[DllImport("DAQMX100.dll" CharSet=CharSet.Auto,
EntryPoint="channelHoldMX100")]
public static extern int channelHoldMX100(int daqmx100, int
doNo);
```

Parameters

daqmx100	Specify the device descriptor.
doNo	Specify the data number.

Description

Gets the hold Boolean for the specified DO data number from the stored current channel setting data.

- If it does not exist, Invalid is returned.

Return value

Returns a Boolean value.

Reference

```
CDAQMX100::getClassMXItemConfig
CDAQMXChConfig::isHold
CDAQMXItemConfig::getClassMXChConfig
```

channelIdleChoiceMX100

Syntax

```
int channelIdleChoiceMX100(DAQMX100 daqmx100, int outputNo);
```

Declaration

Visual Basic

```
Public Declare Function channelIdleChoiceMX100 Lib  
"DAQMX100"(ByVal daqmx100 As Long, ByVal outputNo As Long) As  
Long
```

Visual Basic.NET

```
Public Declare Ansi Function channelIdleChoiceMX100 Lib  
"DAQMX100"(ByVal daqmx100 As Integer, ByVal outputNo As  
Integer) As Integer
```

C#

```
[DllImport("DAQMX100.dll" CharSet=CharSet.Auto,  
EntryPoint="channelIdleChoiceMX100")]  
public static extern int channelIdleChoiceMX100(int daqmx100,  
int outputNo);
```

Parameters

daqmx100	Specify the device descriptor.
outputNo	Specify the output channel data number.

Description

Gets the selection value when idle for the specified output channel data number from the stored current channel setting data.

- If it does not exist, "Previous value" is returned.

Return value

Returns the selected value.

Reference

```
CDAQMX100::getClassMXItemConfig  
CDAQMXItemConfig::getClassMXOutputData  
CDAQMXOutputData::getIdleChoice
```

channelKindMX100

Syntax

```
int channelKindMX100(DAQMX100 daqmx100, int chNo);
```

Declaration**Visual Basic**

```
Public Declare Function channelKindMX100 Lib "DAQMX100" (ByVal daqmx100 As Long, ByVal chNo As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function channelKindMX100 Lib "DAQMX100" (ByVal daqmx100 As Integer, ByVal chNo As Integer) As Integer
```

C#

```
[DllImport("DAQMX100.dll" CharSet=CharSet.Auto, EntryPoint="channelKindMX100")]
public static extern int channelKindMX100(int daqmx100, int chNo);
```

Parameters

daqmx100	Specify the device descriptor.
chNo	Specify the channel number.

Description

Gets the channel type of the specified channel number from the stored current channel setting.

- If it does not exist, "Unused" is returned.

Return value

Returns the channel type.

Reference

```
CDAQMX100::getClassMXItemConfig
CDAQMXChConfig::getKind
CDAQMXItemConfig::getClassMXChConfig
```

channelNumberMX100

Syntax

```
int channelNumberMX100(DAQMX100 daqmx100, int fifoNo, int  
fifoIndex);
```

Declaration

Visual Basic

```
Public Declare Function channelNumberMX100 Lib  
"DAQMX100"(ByVal daqmx100 As Long, ByVal fifoNo As Long, ByVal  
fifoIndex As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function channelNumberMX100 Lib  
"DAQMX100"(ByVal daqmx100 As Integer, ByVal fifoNo As Integer,  
ByVal fifoIndex As Integer) As Integer
```

C#

```
[DllImport("DAQMX100.dll" CharSet=CharSet.Auto,  
EntryPoint="channelNumberMX100")]  
public static extern int channelNumberMX100(int daqmx100, int  
fifoNo, int fifoIndex);
```

Parameters

daqmx100	Specify the device descriptor.
fifoNo	Specify the FIFO number.
fifoIndex	Specify the channel sequence number in the FIFO.

Description

Gets the channel number from the specified FIFO number and FIFO channel sequence number.

- Returns 0 if it does not exist.

Return value

Returns the channel number.

Reference

CDAQMX100::toChNo

channelOutputTypeMX100

Syntax

```
int channelOutputTypeMX100(DAQMX100 daqmx100, int outputNo);
```

Declaration**Visual Basic**

```
Public Declare Function channelOutputTypeMX100 Lib
"DAQMX100"(ByVal daqmx100 As Long, ByVal outputNo As Long) As
Long
```

Visual Basic.NET

```
Public Declare Ansi Function channelOutputTypeMX100 Lib
"DAQMX100"(ByVal daqmx100 As Integer, ByVal outputNo As
Integer) As Integer
```

C#

```
[DllImport("DAQMX100.dll" CharSet=CharSet.Auto,
EntryPoint="channelOutputTypeMX100")]
public static extern int channelOutputTypeMX100(int daqmx100,
int outputNo);
```

Parameters

daqmx100	Specify the device descriptor.
outputNo	Specify the output channel data number.

Description

Gets output type for the specified output channel data number from the stored current channel setting data.

- If it does not exist, "No output" is returned.

Return value

Returns the output type.

Reference

```
CDAQMX100::getClassMXItemConfig
CDAQMXItemConfig::getClassMXOutputData
CDAQMXOutputData::getOutputType
```

channelPointMX100

Syntax

```
int channelPointMX100(DAQMX100 daqmx100, int chNo);
```

Declaration

Visual Basic

```
Public Declare Function channelPointMX100 Lib "DAQMX100" (ByVal daqmx100 As Long, ByVal chNo As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function channelPointMX100 Lib "DAQMX100" (ByVal daqmx100 As Integer, ByVal chNo As Integer) As Integer
```

C#

```
[DllImport("DAQMX100.dll" CharSet=CharSet.Auto, EntryPoint="channelPointMX100")]  
public static extern int channelPointMX100(int daqmx100, int chNo);
```

Parameters

daqmx100	Specify the device descriptor.
chNo	Specify the channel number.

Description

Gets the decimal point position of the specified channel number from the stored current channel setting data.

- Returns 0 if it does not exist.

Return value

Returns the decimal point position.

Reference

CDAQMX100::getClassMXItemConfig
CDAQMXChConfig::getPoint
CDAQMXItemConfig::getClassMXChConfig

channelPresetValueMX100

Syntax

```
int channelPresetValueMX100(DAQMX100 daqmx100, int outputNo);
```

Declaration**Visual Basic**

```
Public Declare Function channelPresetValueMX100 Lib
"DAQMX100"(ByVal daqmx100 As Long, ByVal outputNo As Long) As
Long
```

Visual Basic.NET

```
Public Declare Ansi Function channelPresetValueMX100 Lib
"DAQMX100"(ByVal daqmx100 As Integer, ByVal outputNo As
Integer) As Integer
```

C#

```
[DllImport("DAQMX100.dll" CharSet=CharSet.Auto,
EntryPoint="channelPresetValueMX100")]
public static extern int channelPresetValueMX100(int daqmx100,
int outputNo);
```

Parameters

daqmx100	Specify the device descriptor.
outputNo	Specify the output channel data number.

Description

Gets the user specified output value for the specified output channel data number from the stored current channel setting data.

- The return value is an integer excluding the decimal point position.
- Returns 0 if it does not exist.

Return value

Returns the user specified output value.

Reference

```
CDAQMX100::getClassMXItemConfig
CDAQMXItemConfig::getClassMXOutputData
CDAQMXOutputData::getPresetValue
```

channelPulseTimeMX100

Syntax

```
int channelPulseTimeMX100(DAQMX100 daqmx100, int outputNo);
```

Declaration

Visual Basic

```
Public Declare Function channelPulseTimeMX100 Lib  
"DAQMX100"(ByVal daqmx100 As Long, ByVal outputNo As Long) As  
Long
```

Visual Basic.NET

```
Public Declare Ansi Function channelPulseTimeMX100 Lib  
"DAQMX100"(ByVal daqmx100 As Integer, ByVal outputNo As  
Integer) As Integer
```

C#

```
[DllImport("DAQMX100.dll" CharSet=CharSet.Auto,  
EntryPoint="channelPulseTimeMX100")]  
public static extern int channelPulseTimeMX100(int daqmx100,  
int outputNo);
```

Parameters

daqmx100	Specify the device descriptor.
outputNo	Specify the output channel data number.

Description

Gets pulse interval integer multiple for the specified output channel data number from the stored current channel setting data.

- Returns 1 (minimum value) if it does not exist.

Return value

Returns the integer multiple of the pulse interval.

Reference

```
CDAQMX100::getClassMXItemConfig  
CDAQMXItemConfig::getClassMXOutputData  
CDAQMXOutputData::getPulseTime
```

channelRangeMX100

Syntax

```
int channelRangeMX100(DAQMX100 daqmx100, int chNo);
```

Declaration**Visual Basic**

```
Public Declare Function channelRangeMX100 Lib "DAQMX100" (ByVal daqmx100 As Long, ByVal chNo As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function channelRangeMX100 Lib "DAQMX100" (ByVal daqmx100 As Integer, ByVal chNo As Integer) As Integer
```

C#

```
[DllImport("DAQMX100.dll" CharSet=CharSet.Auto, EntryPoint="channelRangeMX100")]
public static extern int channelRangeMX100(int daqmx100, int chNo);
```

Parameters

daqmx100	Specify the device descriptor.
chNo	Specify the channel number.

Description

Gets the range type of the specified channel number from the stored current channel setting.

- If it does not exist, the SKIP (unused) range type gets the same handling. See the channel status.
- Returns 0 (20 mV) if it does not exist.

Return value

Returns the range type.

Reference

```
CDAQMX100::getClassMXItemConfig
CDAQMXChConfig::getRange
CDAQMXItemConfig::getClassMXChConfig
```

channelRealMaxMX100

Syntax

```
double channelRealMaxMX100(DAQMX100 daqmx100, int chNo);
```

Declaration

Visual Basic

```
Public Declare Function channelRealMaxMX100 Lib  
"DAQMX100"(ByVal daqmx100 As Long, ByVal chNo As Long) As  
Double
```

Visual Basic.NET

```
Public Declare Ansi Function channelRealMaxMX100 Lib  
"DAQMX100"(ByVal daqmx100 As Integer, ByVal chNo As Integer)  
As Double
```

C#

```
[DllImport("DAQMX100.dll" CharSet=CharSet.Auto,  
EntryPoint="channelRealMaxMX100")]  
public static extern double channelRealMaxMX100(int daqmx100,  
int chNo);
```

Parameters

daqmx100	Specify the device descriptor.
chNo	Specify the channel number.

Description

Gets the actual range maximum value of the specified channel number from the stored current channel information data.

- Returns 0.0 if it does not exist.

Return value

Returns the actual range maximum value.

Reference

```
CDAQMX100::getClassMXDataBuffer  
CDAQMXChInfo::getRealMax  
CDAQMXDataBuffer::getClassMXChInfo
```

channelRealMinMX100

Syntax

```
double channelRealMinMX100(DAQMX100 daqmx100, int chNo);
```

Declaration

Visual Basic

```
Public Declare Function channelRealMinMX100 Lib  
"DAQMX100"(ByVal daqmx100 As Long, ByVal chNo As Long) As  
Double
```

Visual Basic.NET

```
Public Declare Ansi Function channelRealMinMX100 Lib  
"DAQMX100"(ByVal daqmx100 As Integer, ByVal chNo As Integer)  
As Double
```

C#

```
[DllImport("DAQMX100.dll" CharSet=CharSet.Auto,  
EntryPoint="channelRealMinMX100")]  
public static extern double channelRealMinMX100(int daqmx100,  
int chNo);
```

Parameters

daqmx100	Specify the device descriptor.
chNo	Specify the channel number.

Description

Gets the actual range minimum value of the specified channel number from the stored current channel information data.

- Returns 0.0 if it does not exist.

Return value

Returns the actual range minimum value.

Reference

```
CDAQMX100::getClassMXDataBuffer  
CDAQMXChInfo::getRealMin  
CDAQMXDataBuffer::getClassMXChInfo
```

channelRefAlarmMX100

Syntax

```
int channelRefAlarmMX100(DAQMX100 daqmx100, int doNo, int refChNo, int levelNo);
```

Declaration

Visual Basic

```
Public Declare Function channelRefAlarmMX100 Lib "DAQMX100" (ByVal daqmx100 As Long, ByVal doNo As Long, ByVal refChNo As Long, ByVal levelNo As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function channelRefAlarmMX100 Lib "DAQMX100" (ByVal daqmx100 As Integer, ByVal doNo As Integer, ByVal refChNo As Integer, ByVal levelNo As Integer) As Integer
```

C#

```
[DllImport("DAQMX100.dll" CharSet=CharSet.Auto, EntryPoint="channelRefAlarmMX100")]  
public static extern int channelRefAlarmMX100(int daqmx100, int doNo, int refChNo, int levelNo);
```

Parameters

daqmx100	Specify the device descriptor.
doNo	Specify the DO data number.
refChNo	Specify the reference channel using a channel number.
levelNo	Specify the alarm level.

Description

Gets the reference alarm for the specified DO data number from the stored current channel setting data.

- The reference alarm is specified by channel number and alarm level.
- If it does not exist, Invalid is returned.

Return value

Returns a Boolean value.

Reference

```
CDAQMX100::getClassMXItemConfig  
CDAQMXChConfig::isRefAlarm  
CDAQMXItemConfig::getClassMXChConfig
```

channelRefChNoMX100

Syntax

```
int channelRefChNoMX100(DAQMX100 daqmx100, int chNo);
```

Declaration**Visual Basic**

```
Public Declare Function channelRefChNoMX100 Lib
"DAQMX100"(ByVal daqmx100 As Long, ByVal chNo As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function channelRefChNoMX100 Lib
"DAQMX100"(ByVal daqmx100 As Integer, ByVal chNo As Integer)
As Integer
```

C#

```
[DllImport("DAQMX100.dll" CharSet=CharSet.Auto,
EntryPoint="channelRefChNoMX100")]
public static extern int channelRefChNoMX100(int daqmx100, int
chNo);
```

Parameters

daqmx100	Specify the device descriptor.
chNo	Specify the channel number.

Description

Gets the reference channel number of the specified channel number from the stored current channel setting data.

- Returns the constant for “undefined reference channel number” if it does not exist.

Return value

Returns the reference channel number.

Reference

```
CDAQMX100::getClassMXItemConfig
CDAQMXChConfig::getRefChNo
CDAQMXItemConfig::getClassMXChConfig
```

channelRJCTypeMX100

Syntax

```
int channelRJCTypeMX100(DAQMX100 daqmx100, int chNo);
```

Declaration

Visual Basic

```
Public Declare Function channelRJCTypeMX100 Lib  
"DAQMX100"(ByVal daqmx100 As Long, ByVal chNo As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function channelRJCTypeMX100 Lib  
"DAQMX100"(ByVal daqmx100 As Integer, ByVal chNo As Integer)  
As Integer
```

C#

```
[DllImport("DAQMX100.dll" CharSet=CharSet.Auto,  
EntryPoint="channelRJCTypeMX100")]  
public static extern int channelRJCTypeMX100(int daqmx100, int  
chNo);
```

Parameters

daqmx100	Specify the device descriptor.
chNo	Specify the channel number.

Description

Gets the RJC type of the specified channel number from the stored current channel setting data.

- If it does not exist, Internal is returned.

Return value

Returns the RJC type.

Reference

```
CDAQMX100::getClassMXItemConfig  
CDAQMXChConfig::getRJCType  
CDAQMXItemConfig::getClassMXChConfig
```

channelRJCVoltMX100

Syntax

```
int channelRJCVoltMX100(DAQMX100 daqmx100, int chNo);
```

Declaration**Visual Basic**

```
Public Declare Function channelRJCVoltMX100 Lib
"DAQMX100"(ByVal daqmx100 As Long, ByVal chNo As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function channelRJCVoltMX100 Lib
"DAQMX100"(ByVal daqmx100 As Integer, ByVal chNo As Integer)
As Integer
```

C#

```
[DllImport("DAQMX100.dll" CharSet=CharSet.Auto,
EntryPoint="channelRJCVoltMX100")]
public static extern int channelRJCVoltMX100(int daqmx100, int
chNo);
```

Parameters

daqmx100	Specify the device descriptor.
chNo	Specify the channel number.

Description

Gets the RJC voltage value of the specified channel number from the stored current channel setting data.

- Returns 0 if it does not exist.

Return value

Returns the RJC voltage.

Reference

```
CDAQMX100::getClassMXItemConfig
CDAQMXChConfig::getRJCVolt
CDAQMXItemConfig::getClassMXChConfig
```

channelScaleMaxMX100

Syntax

```
int channelScaleMaxMX100(DAQMX100 daqmx100, int chNo);
```

Declaration

Visual Basic

```
Public Declare Function channelScaleMaxMX100 Lib  
"DAQMX100"(ByVal daqmx100 As Long, ByVal chNo As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function channelScaleMaxMX100 Lib  
"DAQMX100"(ByVal daqmx100 As Integer, ByVal chNo As Integer)  
As Integer
```

C#

```
[DllImport("DAQMX100.dll" CharSet=CharSet.Auto,  
EntryPoint="channelScaleMaxMX100")]  
public static extern int channelScaleMaxMX100(int daqmx100,  
int chNo);
```

Parameters

daqmx100	Specify the device descriptor.
chNo	Specify the channel number.

Description

Gets the scale maximum value of the specified channel number from the stored current channel setting data.

- The return value is an integer excluding the decimal point position.
- Returns 0 if it does not exist.

Return value

Returns the scale maximum.

Reference

```
CDAQMX100::getClassMXItemConfig  
CDAQMXChConfig::getScaleMax  
CDAQMXItemConfig::getClassMXChConfig
```

channelScaleMinMX100

Syntax

```
int channelScaleMinMX100(DAQMX100 daqmx100, int chNo);
```

Declaration

Visual Basic

```
Public Declare Function channelScaleMinMX100 Lib  
"DAQMX100"(ByVal daqmx100 As Long, ByVal chNo As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function channelScaleMinMX100 Lib  
"DAQMX100"(ByVal daqmx100 As Integer, ByVal chNo As Integer)  
As Integer
```

C#

```
[DllImport("DAQMX100.dll" CharSet=CharSet.Auto,  
EntryPoint="channelScaleMinMX100")]  
public static extern int channelScaleMinMX100(int daqmx100,  
int chNo);
```

Parameters

daqmx100	Specify the device descriptor.
chNo	Specify the channel number.

Description

Gets the scale minimum value of the specified channel number from the stored current channel setting data.

- The return value is an integer excluding the decimal point position.
- Returns 0 if it does not exist.

Return value

Returns the scale minimum.

Reference

```
CDAQMX100::getClassMXItemConfig  
CDAQMXChConfig::getScaleMin  
CDAQMXItemConfig::getClassMXChConfig
```

channelScaleTypeMX100

Syntax

```
int channelScaleTypeMX100(DAQMX100 daqmx100, int chNo);
```

Declaration

Visual Basic

```
Public Declare Function channelScaleTypeMX100 Lib  
"DAQMX100"(ByVal daqmx100 As Long, ByVal chNo As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function channelScaleTypeMX100 Lib  
"DAQMX100"(ByVal daqmx100 As Integer, ByVal chNo As Integer)  
As Integer
```

C#

```
[DllImport("DAQMX100.dll" CharSet=CharSet.Auto,  
EntryPoint="channelScaleTypeMX100")]  
public static extern int channelScaleTypeMX100(int daqmx100,  
int chNo);
```

Parameters

daqmx100	Specify the device descriptor.
chNo	Specify the channel number.

Description

Gets the scale type of the specified channel number from the stored current channel setting data.

- If it does not exist, "No scale" is returned.

Return value

Returns the scale type.

Reference

```
CDAQMX100::getClassMXItemConfig  
CDAQMXChConfig::getScale  
CDAQMXItemConfig::getClassMXChConfig
```

channelSpanMaxMX100

Syntax

```
int channelSpanMaxMX100(DAQMX100 daqmx100, int chNo);
```

Declaration**Visual Basic**

```
Public Declare Function channelSpanMaxMX100 Lib
"DAQMX100"(ByVal daqmx100 As Long, ByVal chNo As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function channelSpanMaxMX100 Lib
"DAQMX100"(ByVal daqmx100 As Integer, ByVal chNo As Integer)
As Integer
```

C#

```
[DllImport("DAQMX100.dll" CharSet=CharSet.Auto,
EntryPoint="channelSpanMaxMX100")]
public static extern int channelSpanMaxMX100(int daqmx100, int
chNo);
```

Parameters

daqmx100	Specify the device descriptor.
chNo	Specify the channel number.

Description

Gets the span maximum value of the specified channel number from the stored current channel setting data.

- The return value is an integer excluding the decimal point position.
- Returns 0 if it does not exist.

Return value

Returns the span maximum.

Reference

```
CDAQMX100::getClassMXItemConfig
CDAQMXChConfig::getSpanMax
CDAQMXItemConfig::getClassMXChConfig
```

channelSpanMinMX100

Syntax

```
int channelSpanMinMX100(DAQMX100 daqmx100, int chNo);
```

Declaration

Visual Basic

```
Public Declare Function channelSpanMinMX100 Lib  
"DAQMX100"(ByVal daqmx100 As Long, ByVal chNo As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function channelSpanMinMX100 Lib  
"DAQMX100"(ByVal daqmx100 As Integer, ByVal chNo As Integer)  
As Integer
```

C#

```
[DllImport("DAQMX100.dll" CharSet=CharSet.Auto,  
EntryPoint="channelSpanMinMX100")]  
public static extern int channelSpanMinMX100(int daqmx100, int  
chNo);
```

Parameters

daqmx100	Specify the device descriptor.
chNo	Specify the channel number.

Description

Gets the span maximum value of the specified channel number from the stored current channel setting data.

- The return value is an integer excluding the decimal point position.
- Returns 0 if it does not exist.

Return value

Returns the span minimum.

Reference

```
CDAQMX100::getClassMXItemConfig  
CDAQMXChConfig::getSpanMin  
CDAQMXItemConfig::getClassMXChConfig
```

channelValidMX100

Syntax

```
int channelValidMX100(DAQMX100 daqmx100, int chNo);
```

Declaration

Visual Basic

```
Public Declare Function channelValidMX100 Lib "DAQMX100" (ByVal daqmx100 As Long, ByVal chNo As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function channelValidMX100 Lib "DAQMX100" (ByVal daqmx100 As Integer, ByVal chNo As Integer) As Integer
```

C#

```
[DllImport("DAQMX100.dll" CharSet=CharSet.Auto, EntryPoint="channelValidMX100")]  
public static extern int channelValidMX100(int daqmx100, int chNo);
```

Parameters

daqmx100	Specify the device descriptor.
chNo	Specify the channel number.

Description

Gets the channel status of the specified channel number from the stored current channel setting data as a Boolean.

- If it does not exist, Invalid is returned.

Return value

Returns a Boolean value.

Reference

```
CDAQMX100::getClassMXItemConfig  
CDAQMXChConfig::isValid  
CDAQMXItemConfig::getClassMXChConfig
```

currentAOPWMValidMX100

Syntax

```
int currentAOPWMValidMX100(DAQMX100 daqmx100, int aopwmNo);
```

Declaration

Visual Basic

```
Public Declare Function currentAOPWMValidMX100 Lib  
"DAQMX100"(ByVal daqmx100 As Long, ByVal aopwmNo As Long) As  
Long
```

Visual Basic.NET

```
Public Declare Ansi Function currentAOPWMValidMX100 Lib  
"DAQMX100"(ByVal daqmx100 As Integer, ByVal aopwmNo As  
Integer) As Integer
```

C#

```
[DllImport("DAQMX100.dll" CharSet=CharSet.Auto,  
EntryPoint="currentAOPWMValidMX100")]  
public static extern int currentAOPWMValidMX100(int daqmx100,  
int aopwmNo);
```

Parameters

daqmx100	Specify the device descriptor.
aopwmNo	Specify the AO/PWM data number.

Description

Gets valid/invalid for the specified AO/PWM data number from the stored current AO/PWM data as a Boolean value.

- If it does not exist, Invalid is returned.

Return value

Returns a Boolean value.

Reference

```
CDAQMX100::getClassMXAOPWMList  
CDAQMXAOPWMData::getAOPWMValid  
CDAQMXAOPWMList::getCurrent
```

currentAOPWMValueMX100

Syntax

```
int currentAOPWMValueMX100(DAQMX100 daqmx100, int aopwmNo);
```

Declaration

Visual Basic

```
Public Declare Function currentAOPWMValueMX100 Lib  
"DAQMX100"(ByVal daqmx100 As Long, ByVal aopwmNo As Long) As  
Long
```

Visual Basic.NET

```
Public Declare Ansi Function currentAOPWMValueMX100 Lib  
"DAQMX100"(ByVal daqmx100 As Integer, ByVal aopwmNo As  
Integer) As Integer
```

C#

```
[DllImport("DAQMX100.dll" CharSet=CharSet.Auto,  
EntryPoint="currentAOPWMValidMX100")]  
public static extern int currentAOPWMValueMX100(int daqmx100,  
int aopwmNo);
```

Parameters

daqmx100	Specify the device descriptor.
aopwmNo	Specify the AO/PWM data number.

Description

Gets the output data value of the specified AO/PWM data number from the stored current AO/PWM data.

- Returns 0 if it does not exist.

Return value

Returns the output data value.

Reference

```
CDAQMX100::getClassMXAOPWMList  
CDAQMXAOPWMData::getAOPWMValue  
CDAQMXAOPWMList::getCurrent
```

currentBalanceResultMX100

Syntax

```
int currentBalanceResultMX100(DAQMX100 daqmx100, int balanceNo);
```

Declaration

Visual Basic

```
Public Declare Function currentBalanceResultMX100 Lib "DAQMX100" (ByVal daqmx100 As Long, ByVal balanceNo As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function currentBalanceResultMX100 Lib "DAQMX100" (ByVal daqmx100 As Integer, ByVal balanceNo As Integer) As Integer
```

C#

```
[DllImport("DAQMX100.dll" CharSet=CharSet.Auto, EntryPoint="currentBalanceResultMX100")] public static extern int currentBalanceResultMX100(int daqmx100, int balanceNo);
```

Parameters

daqmx100	Specify the device descriptor.
balanceNo	Specify the initial balance data number.

Description

Gets the initial balance value for the specified initial balance data number from the stored current initial balance data.

- Returns the result from the last-executed initial balance setting function.
- If it does not exist, Unspecified is returned.

Return value

Returns the initial balance result.

Reference

```
CDAQMX100::getClassMXBalanceList  
CDAQMXBalanceList::getCurrent  
CDAQMXBalanceResult::getResult
```

currentBalanceValidMX100

Syntax

```
int currentBalanceValidMX100(DAQMX100 daqmx100, int
balanceNo);
```

Declaration**Visual Basic**

```
Public Declare Function currentBalanceValidMX100 Lib
"DAQMX100"(ByVal daqmx100 As Long, ByVal balanceNo As Long) As
Long
```

Visual Basic.NET

```
Public Declare Ansi Function currentBalanceValidMX100 Lib
"DAQMX100"(ByVal daqmx100 As Integer, ByVal balanceNo As
Integer) As Integer
```

C#

```
[DllImport("DAQMX100.dll" CharSet=CharSet.Auto,
EntryPoint="currentBalanceValidMX100")]
public static extern int currentBalanceValidMX100(int
daqmx100, int balanceNo);
```

Parameters

daqmx100	Specify the device descriptor.
balanceNo	Specify the initial balance data number.

Description

Gets the Invalid/Valid Boolean value for the specified initial balance data number from the stored current initial balance data.

- If it does not exist, Invalid is returned.

Return value

Returns a Boolean value.

Reference

```
CDAQMX100::getClassMXBalanceList
CDAQMXBalanceList::getCurrent
CDAQMXBalanceResult::getBalanceValid
```

currentBalanceValueMX100

Syntax

```
int currentBalanceValueMX100(DAQMX100 daqmx100, int balanceNo);
```

Declaration

Visual Basic

```
Public Declare Function currentBalanceValueMX100 Lib "DAQMX100" (ByVal daqmx100 As Long, ByVal balanceNo As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function currentBalanceValueMX100 Lib "DAQMX100" (ByVal daqmx100 As Integer, ByVal balanceNo As Integer) As Integer
```

C#

```
[DllImport("DAQMX100.dll" CharSet=CharSet.Auto, EntryPoint="currentBalanceValueMX100")]  
public static extern int currentBalanceValueMX100(int daqmx100, int balanceNo);
```

Parameters

daqmx100	Specify the device descriptor.
balanceNo	Specify the initial balance data number.

Description

Gets the initial balance value for the specified initial balance data number from the stored current initial balance data.

- Returns 0 if it does not exist.

Return value

Returns the initial balance value.

Reference

```
CDAQMX100::getClassMXBalanceList  
CDAQMXBalanceList::getCurrent  
CDAQMXBalanceResult::getBalanceValue
```

currentDoubleAOPWMValueMX100

Syntax

```
double currentDoubleAOPWMValueMX100(DAQMX100 daqmx100, int
aopwmNo);
```

Declaration**Visual Basic**

```
Public Declare Function currentDoubleAOPWMValueMX100 Lib
"DAQMX100"(ByVal daqmx100 As Long, ByVal aopwmNo As Long) As
Double
```

Visual Basic.NET

```
Public Declare Ansi Function currentDoubleAOPWMValueMX100 Lib
"DAQMX100"(ByVal daqmx100 As Integer, ByVal aopwmNo As
Integer) As Double
```

C#

```
[DllImport("DAQMX100.dll" CharSet=CharSet.Auto,
EntryPoint="currentDoubleAOPWMValueMX100")]
public static extern double currentDoubleAOPWMValueMX100(int
daqmx100, int aopwmNo);
```

Parameters

daqmx100	Specify the device descriptor.
aopwmNo	Specify the AO/PWM data number.

Description

Gets the actual output value of the output data value of the specified AO/PWM data number from the stored current AO/PWM data.

- Returns 0.0 if it does not exist.

Return value

Returns the actual output value.

Reference

CDAQMX100::currentDoubleAOPWMValue

currentDOValidMX100

Syntax

```
int currentDOValidMX100(DAQMX100 daqmx100, int doNo);
```

Declaration

Visual Basic

```
Public Declare Function currentDOValidMX100 Lib  
"DAQMX100"(ByVal daqmx100 As Long, ByVal doNo As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function currentDOValidMX100 Lib  
"DAQMX100"(ByVal daqmx100 As Integer, ByVal doNo As Integer)  
As Integer
```

C#

```
[DllImport("DAQMX100.dll" CharSet=CharSet.Auto,  
EntryPoint="currentDOValidMX100")]  
public static extern int currentDOValidMX100(int daqmx100, int  
doNo);
```

Parameters

daqmx100	Specify the device descriptor.
doNo	Specify the data number.

Description

Gets the Invalid/Valid Boolean value for the specified DO data number from the stored current DO data.

- If it does not exist, Invalid is returned.

Return value

Returns a Boolean value.

Reference

```
CDAQMX100::getClassMXDOList  
CDAQMXDOData::getDOValid  
CDAQMXDOList::getCurrent
```

currentDOValueMX100

Syntax

```
int currentDOValueMX100(DAQMX100 daqmx100, int doNo);
```

Declaration

Visual Basic

```
Public Declare Function currentDOValueMX100 Lib
"DAQMX100"(ByVal daqmx100 As Long, ByVal doNo As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function currentDOValueMX100 Lib
"DAQMX100"(ByVal daqmx100 As Integer, ByVal doNo As Integer)
As Integer
```

C#

```
[DllImport("DAQMX100.dll" CharSet=CharSet.Auto,
EntryPoint="currentDOValueMX100")]
public static extern int currentDOValueMX100(int daqmx100, int
doNo);
```

Parameters

daqmx100	Specify the device descriptor.
doNo	Specify the data number.

Description

Gets ON/OFF for the specified DO data number from the stored current DO data as a Boolean value.

- If it does not exist, Invalid is returned.

Return value

Returns a Boolean value.

Reference

```
CDAQMX100::getClassMXDOList
CDAQMXDOData::getDOONOFF
CDAQMXDOList::getCurrent
```

currentTransmitMX100

Syntax

```
int currentTransmitMX100(DAQMX100 daqmx100, int aopwmNo);
```

Declaration

Visual Basic

```
Public Declare Function currentTransmitMX100 Lib  
"DAQMX100"(ByVal daqmx100 As Long, ByVal aopwmNo As Long) As  
Long
```

Visual Basic.NET

```
Public Declare Ansi Function currentTransmitMX100 Lib  
"DAQMX100"(ByVal daqmx100 As Integer, ByVal aopwmNo As  
Integer) As Integer
```

C#

```
[DllImport("DAQMX100.dll" CharSet=CharSet.Auto,  
EntryPoint="currentTransmitMX100")]  
public static extern int currentTransmitMX100(int daqmx100,  
int aopwmNo);
```

Parameters

daqmx100	Specify the device descriptor.
aopwmNo	Specify the AO/PWM data number.

Description

Gets the transfer status of the specified AO/PWM data number from the stored current transmission output data.

- If it does not exist, "Unspecified (Unknown)" is returned.

Return value

Returns the transmission status.

Reference

```
CDAQMX100::getClassMXTransmitList  
CDAQMXTransmit::getTransmit  
CDAQMXTransmitList::getCurrent
```

dataAlarmMX100

Syntax

```
int dataAlarmMX100(DAQMX100 daqmx100, int chNo, int levelNo);
```

Declaration

Visual Basic

```
Public Declare Function dataAlarmMX100 Lib "DAQMX100" (ByVal daqmx100 As Long, ByVal chNo As Long, ByVal levelNo As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function dataAlarmMX100 Lib "DAQMX100" (ByVal daqmx100 As Integer, ByVal chNo As Integer, ByVal levelNo As Integer) As Integer
```

C#

```
[DllImport("DAQMX100.dll" CharSet=CharSet.Auto, EntryPoint="dataAlarmMX100")]
public static extern int dataAlarmMX100(int daqmx100, int chNo, int levelNo);
```

Parameters

daqmx100	Specify the device descriptor.
chNo	Specify the channel number.
levelNo	Specify the alarm level.

Description

Gets the valid/invalid value of the alarm corresponding to the alarm level for the specified channel number from the stored current measured data.

- If it does not exist, Invalid is returned.

Return value

Returns a Boolean value.

Reference

```
CDAQMX100::getClassMXDataBuffer
CDAQMXDataBuffer::currentDataInfo
CDAQMXDataInfo::isAlarm
```

dataDayMX100

Syntax

```
int dataDayMX100(DAQMX100 daqmx100, int chNo);
```

Declaration

Visual Basic

```
Public Declare Function dataDayMX100 Lib "DAQMX100" (ByVal daqmx100 As Long, ByVal chNo As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function dataDayMX100 Lib "DAQMX100" (ByVal daqmx100 As Integer, ByVal chNo As Integer) As Integer
```

C#

```
[DllImport("DAQMX100.dll" CharSet=CharSet.Auto, EntryPoint="dataDayMX100")]  
public static extern int dataDayMX100(int daqmx100, int chNo);
```

Parameters

daqmx100	Specify the device descriptor.
chNo	Specify the channel number.

Description

Gets the day of the specified channel number from the stored current time information data.

- The day is a number from 1 to 31.
- Returns 0 if it does not exist.

Return value

Returns the day value.

Reference

```
CDAQMX100::getClassMXDataBuffer  
CDAQMXDataBuffer::currentDateTime  
CDAQMXDateTime::toLocalDateTime
```

dataDoubleValueMX100

Syntax

```
double dataDoubleValueMX100(DAQMX100 daqmx100, int chNo);
```

Declaration**Visual Basic**

```
Public Declare Function dataDoubleValueMX100 Lib
"DAQMX100"(ByVal daqmx100 As Long, ByVal chNo As Long) As
Double
```

Visual Basic.NET

```
Public Declare Ansi Function dataDoubleValueMX100 Lib
"DAQMX100"(ByVal daqmx100 As Integer, ByVal chNo As Integer)
As Double
```

C#

```
[DllImport("DAQMX100.dll" CharSet=CharSet.Auto,
EntryPoint="dataDoubleValueMX100")]
public static extern double dataDoubleValueMX100(int daqmx100,
int chNo);
```

Parameters

daqmx100	Specify the device descriptor.
chNo	Specify the channel number.

Description

Gets the measured value of the specified channel number from the stored current measured data.

- Returns 0.0 if it does not exist.

Return value

Returns the measured value as a double-precision floating point number.

Reference

```
CDAQMX100::getClassMXDataBuffer
CDAQMXDataBuffer::currentDataInfo
CDAQMXDataInfo::getDoubleValue
```

dataHourMX100

Syntax

```
int dataHourMX100(DAQMX100 daqmx100, int chNo);
```

Declaration

Visual Basic

```
Public Declare Function dataHourMX100 Lib "DAQMX100" (ByVal daqmx100 As Long, ByVal chNo As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function dataHourMX100 Lib "DAQMX100" (ByVal daqmx100 As Integer, ByVal chNo As Integer) As Integer
```

C#

```
[DllImport("DAQMX100.dll" CharSet=CharSet.Auto, EntryPoint="dataHourMX100")]  
public static extern int dataHourMX100(int daqmx100, int chNo);
```

Parameters

daqmx100	Specify the device descriptor.
chNo	Specify the channel number.

Description

Gets the hour of the specified channel number from the stored current time information data.

- The hour is a number from 0 to 23.
- Returns 0 if it does not exist.

Return value

Returns the hour value.

Reference

```
CDAQMX100::getClassMXDataBuffer  
CDAQMXDataBuffer::currentDateTime  
CDAQMXDateTime::toLocalDateTime
```

dataMilliSecMX100

Syntax

```
int dataMilliSecMX100(DAQMX100 daqmx100, int chNo);
```

Declaration

Visual Basic

```
Public Declare Function dataMilliSecMX100 Lib "DAQMX100" (ByVal daqmx100 As Long, ByVal chNo As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function dataMilliSecMX100 Lib "DAQMX100" (ByVal daqmx100 As Integer, ByVal chNo As Integer) As Integer
```

C#

```
[DllImport("DAQMX100.dll" CharSet=CharSet.Auto, EntryPoint="dataMilliSecMX100")]  
public static extern int dataMilliSecMX100(int daqmx100, int chNo);
```

Parameters

daqmx100	Specify the device descriptor.
chNo	Specify the channel number.

Description

Gets the milliseconds of the specified channel number from the stored current time information data.

- Returns 0 if it does not exist.

Return value

Returns the milliseconds value.

Reference

```
CDAQMX100::getClassMXDataBuffer  
CDAQMXDataBuffer::currentDateTime  
CDAQMXDateTime::getMilliSecond
```

dataMinuteMX100

Syntax

```
int dataMinuteMX100(DAQMX100 daqmx100, int chNo);
```

Declaration

Visual Basic

```
Public Declare Function dataMinuteMX100 Lib "DAQMX100" (ByVal daqmx100 As Long, ByVal chNo As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function dataMinuteMX100 Lib "DAQMX100" (ByVal daqmx100 As Integer, ByVal chNo As Integer) As Integer
```

C#

```
[DllImport("DAQMX100.dll" CharSet=CharSet.Auto, EntryPoint="dataMinuteMX100")]  
public static extern int dataMinuteMX100(int daqmx100, int chNo);
```

Parameters

daqmx100	Specify the device descriptor.
chNo	Specify the channel number.

Description

Gets the minutes of the specified channel number from the stored current time information data.

- The minute is a number from 0 to 59.
- Returns 0 if it does not exist.

Return value

Returns the minute value.

Reference

```
CDAQMX100::getClassMXDataBuffer  
CDAQMXDataBuffer::currentDateTime  
CDAQMXDateTime::toLocalDateTime
```

dataMonthMX100

Syntax

```
int dataMonthMX100(DAQMX100 daqmx100, int chNo);
```

Declaration

Visual Basic

```
Public Declare Function dataMonthMX100 Lib "DAQMX100" (ByVal daqmx100 As Long, ByVal chNo As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function dataMonthMX100 Lib "DAQMX100" (ByVal daqmx100 As Integer, ByVal chNo As Integer) As Integer
```

C#

```
[DllImport("DAQMX100.dll" CharSet=CharSet.Auto, EntryPoint="dataMonthMX100")]  
public static extern int dataMonthMX100(int daqmx100, int chNo);
```

Parameters

daqmx100	Specify the device descriptor.
chNo	Specify the channel number.

Description

Gets the month of the specified channel number from the stored current time information data.

- The month is a number from 1 to 12.
- Returns 0 if it does not exist.

Return value

Returns the month value.

Reference

```
CDAQMX100::getClassMXDataBuffer  
CDAQMXDataBuffer::currentDateTime  
CDAQMXDateTime::toLocalDateTime
```

dataNumChMX100

Syntax

```
int dataNumChMX100(DAQMX100 daqmx100, int chNo);
```

Declaration

Visual Basic

```
Public Declare Function dataNumChMX100 Lib "DAQMX100" (ByVal daqmx100 As Long, ByVal chNo As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function dataNumChMX100 Lib "DAQMX100" (ByVal daqmx100 As Integer, ByVal chNo As Integer) As Integer
```

C#

```
[DllImport("DAQMX100.dll" CharSet=CharSet.Auto, EntryPoint="dataNumChMX100")]  
public static extern int dataNumChMX100(int daqmx100, int chNo);
```

Parameters

daqmx100	Specify the device descriptor.
chNo	Specify the channel number.

Description

Gets the specified number of remaining data from the current status data of the specified channel number from among the data that is acquired and stored by a data retrieval function.

- Returns 0 if it does not exist.

Return value

Returns the remaining number of data.

Reference

```
CDAQMX100::getClassMXDataBuffer  
CDAQMXDataBuffer::getDataNum
```

dataNumFIFOMX100

Syntax

```
int dataNumFIFOMX100(DAQMX100 daqmx100, int fifoNo);
```

Declaration

Visual Basic

```
Public Declare Function dataNumFIFOMX100 Lib "DAQMX100" (ByVal daqmx100 As Long, ByVal fifoNo As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function dataNumFIFOMX100 Lib "DAQMX100" (ByVal daqmx100 As Integer, ByVal fifoNo As Integer) As Integer
```

C#

```
[DllImport("DAQMX100.dll" CharSet=CharSet.Auto, EntryPoint="dataNumFIFOMX100")]
public static extern int dataNumFIFOMX100(int daqmx100, int fifoNo);
```

Parameters

daqmx100	Specify the device descriptor.
fifoNo	Specify the FIFO number.

Description

Gets the specified number of remaining data from the current status data of the specified FIFO number from among the data that is acquired and stored by a data retrieval function.

- Returns the minimum value of the channels in the FIFO.
- Returns 0 if it does not exist.

Return value

Returns the remaining number of data.

Reference

CDAQMX100::getDataNum

dataSecondMX100

Syntax

```
int dataSecondMX100(DAQMX100 daqmx100, int chNo);
```

Declaration

Visual Basic

```
Public Declare Function dataSecondMX100 Lib "DAQMX100" (ByVal daqmx100 As Long, ByVal chNo As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function dataSecondMX100 Lib "DAQMX100" (ByVal daqmx100 As Integer, ByVal chNo As Integer) As Integer
```

C#

```
[DllImport("DAQMX100.dll" CharSet=CharSet.Auto, EntryPoint="dataSecondMX100")]  
public static extern int dataSecondMX100(int daqmx100, int chNo);
```

Parameters

daqmx100	Specify the device descriptor.
chNo	Specify the channel number.

Description

Gets the seconds of the specified channel number from the stored current time information data.

- The second is a number from 0 to 59.
- Returns 0 if it does not exist.

Return value

Returns the seconds value.

Reference

```
CDAQMX100::getClassMXDataBuffer  
CDAQMXDataBuffer::currentDateTime  
CDAQMXDateTime::toLocalDateTime
```

dataStatusMX100

Syntax

```
int dataStatusMX100(DAQMX100 daqmx100, int chNo);
```

Declaration

Visual Basic

```
Public Declare Function dataStatusMX100 Lib "DAQMX100" (ByVal daqmx100 As Long, ByVal chNo As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function dataStatusMX100 Lib "DAQMX100" (ByVal daqmx100 As Integer, ByVal chNo As Integer) As Integer
```

C#

```
[DllImport("DAQMX100.dll" CharSet=CharSet.Auto, EntryPoint="dataStatusMX100")]  
public static extern int dataStatusMX100(int daqmx100, int chNo);
```

Parameters

daqmx100	Specify the device descriptor.
chNo	Specify the channel number.

Description

Gets the data status value of the specified channel number from the stored current measured data.

- If it does not exist, "Unknown status" is returned.

Return value

Returns the data status value.

Reference

```
CDAQMX100::getClassMXDataBuffer  
CDAQMXDataBuffer::currentDataInfo  
CDAQMXDataInfo::getStatus
```

dataStringValueMX100

Syntax

```
int dataStringValueMX100(DAQMX100 daqmx100, int chNo, char *
strValue, int lenValue);
```

Declaration

Visual Basic

```
Public Declare Function dataStringValueMX100 Lib
"DAQMX100"(ByVal daqmx100 As Long, ByVal chNo As Long, ByVal
strValue As String, ByVal lenValue As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function dataStringValueMX100 Lib
"DAQMX100"(ByVal daqmx100 As Integer, ByVal chNo As Integer,
ByVal strValue As String, ByVal lenValue As Integer) As
Integer
```

C#

```
[DllImport("DAQMX100.dll" CharSet=CharSet.Auto,
EntryPoint="dataStringValueMX100")]
public static extern int dataStringValueMX100(int daqmx100,
int chNo, byte[] strValue, int lenValue);
```

Parameters

daqmx100	Specify the device descriptor.
chNo	Specify the channel number.
strValue	Specify the field where the string is to be stored.
lenValue	Specify the byte size of the field where the string is to be stored.

Description

Gets the measured value of the specified channel number from the stored current measured data.

- Converts into a string and stores it in the specified field.
- The string stored in the field includes the terminator (NULL).
- The return value is the length of the actual string. The return value does not include the terminator.
- Returns 0 if it does not exist.
- The strings that can be stored are, in general, ASCII strings.

Return value

Returns the length of the string.

Reference

```
CDAQMX100::getClassMXDataBuffer
CDAQMXDataBuffer::currentDataInfo
CDAQMXDataInfo::getStringValue
```

dataTimeMX100

Syntax

```
int dataTimeMX100(DAQMX100 daqmx100, int chNo);
```

Declaration**Visual Basic**

```
Public Declare Function dataTimeMX100 Lib "DAQMX100" (ByVal daqmx100 As Long, ByVal chNo As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function dataTimeMX100 Lib "DAQMX100" (ByVal daqmx100 As Integer, ByVal chNo As Integer) As Integer
```

C#

```
[DllImport("DAQMX100.dll" CharSet=CharSet.Auto, EntryPoint="dataTimeMX100")]
public static extern int dataTimeMX100(int daqmx100, int chNo);
```

Parameters

daqmx100	Specify the device descriptor.
chNo	Specify the channel number.

Description

Gets the seconds of the specified channel number from the stored current time information data.

- This is the number of seconds from the reference date/time (Jan. 1, 1970).
- Returns 0 if it does not exist.

Return value

Returns seconds.

Reference

```
CDAQMX100::getClassMXDataBuffer
CDAQMXDataBuffer::currentDateTime
CDAQMXDateTime::getTime
```

dataValidMX100

Syntax

```
int dataValidMX100(DAQMX100 daqmx100, int chNo);
```

Declaration

Visual Basic

```
Public Declare Function dataValidMX100 Lib "DAQMX100" (ByVal daqmx100 As Long, ByVal chNo As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function dataValidMX100 Lib "DAQMX100" (ByVal daqmx100 As Integer, ByVal chNo As Integer) As Integer
```

C#

```
[DllImport("DAQMX100.dll" CharSet=CharSet.Auto, EntryPoint="dataValidMX100")]  
public static extern int dataValidMX100(int daqmx100, int chNo);
```

Parameters

daqmx100	Specify the device descriptor.
chNo	Specify the channel number.

Description

Gets valid/invalid for the measured data of the specified channel number from the stored current measured data as a Boolean.

- If it does not exist, Invalid is returned.

Return value

Returns a Boolean value.

Reference

CDAQMX100::getClassMXDataBuffer
CDAQMXDataBuffer::isCurrent

dataValueMX100

Syntax

```
int dataValueMX100(DAQMX100 daqmx100, int chNo);
```

Declaration

Visual Basic

```
Public Declare Function dataValueMX100 Lib "DAQMX100" (ByVal daqmx100 As Long, ByVal chNo As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function dataValueMX100 Lib "DAQMX100" (ByVal daqmx100 As Integer, ByVal chNo As Integer) As Integer
```

C#

```
[DllImport("DAQMX100.dll" CharSet=CharSet.Auto, EntryPoint="dataValueMX100")]  
public static extern int dataValueMX100(int daqmx100, int chNo);
```

Parameters

daqmx100	Specify the device descriptor.
chNo	Specify the channel number.

Description

Gets the data value of the specified channel number from the stored current measured data.

- Returns 0 if it does not exist.

Return value

Returns the data value.

Reference

```
CDAQMX100::getClassMXDataBuffer  
CDAQMXDataBuffer::currentDataInfo  
CDAQMXDataInfo::getValue
```

dataYearMX100

Syntax

```
int dataYearMX100(DAQMX100 daqmx100, int chNo);
```

Declaration

Visual Basic

```
Public Declare Function dataYearMX100 Lib "DAQMX100" (ByVal daqmx100 As Long, ByVal chNo As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function dataYearMX100 Lib "DAQMX100" (ByVal daqmx100 As Integer, ByVal chNo As Integer) As Integer
```

C#

```
[DllImport("DAQMX100.dll" CharSet=CharSet.Auto, EntryPoint="dataYearMX100")]  
public static extern int dataYearMX100(int daqmx100, int chNo);
```

Parameters

daqmx100	Specify the device descriptor.
chNo	Specify the channel number.

Description

Gets the year of the specified channel number from the stored current time information data.

- The year is a 4-digit number.
- Returns 0 if it does not exist.

Return value

Returns the year value.

Reference

```
CDAQMX100::getClassMXDataBuffer  
CDAQMXDataBuffer::currentDateTime  
CDAQMXDateTime::toLocalDateTime
```

errorMaxLengthMX100

Syntax

```
int errorMaxLengthMX100(void);
```

Declaration

Visual Basic

```
Public Declare Function errorMaxLengthMX100 Lib "DAQMX100"()  
As Long
```

Visual Basic.NET

```
Public Declare Ansi Function errorMaxLengthMX100 Lib  
"DAQMX100"() As Integer
```

C#

```
[DllImport("DAQMX100.dll" CharSet=CharSet.Auto,  
EntryPoint="errorMaxLengthMX100")]  
public static extern int errorMaxLengthMX100();
```

Description

Gets the maximum length of the error message string.

- The return value does not include the terminator.

Return value

Returns the length of the string.

Reference

CDAQMX100::getMaxLenErrorMessage

getAlarmNameMX100

[Visual C only]

Syntax

```
const char * getAlarmNameMX100(int iAlarmType);
```

Parameters

iAlarmType Specify the alarm type.

Description

Gets the string corresponding to the specified alarm type.

- If it does not exist, returns the pointer to the string corresponding to “No alarm.”

Return value

Returns a pointer to the string.

Reference

CDAQMXDataInfo::getAlarmName

getChannelCommentMX100**[Visual C only]****Syntax**

```
const char * getChannelCommentMX100(DAQMX100 daqmx100, int  
chNo);
```

Parameters

daqmx100	Specify the device descriptor.
chNo	Specify the channel number.

Description

Gets the specified channel number and comment from the stored current channel setting data.

- Returns NULL if it does not exist.

Return value

Returns a pointer to the string.

Reference

```
CDAQMX100::getClassMXItemConfig  
CDAQMXChConfig::getComment  
CDAQMXItemConfig::getClassMXChConfig
```

getChannelTagMX100

[Visual C only]

Syntax

```
const char * getChannelTagMX100(DAQMX100 daqmx100, int chNo);
```

Parameters

daqmx100	Specify the device descriptor.
chNo	Specify the channel number.

Description

Gets the specified channel number and tag from the stored current channel setting data.

- Returns NULL if it does not exist.

Return value

Returns a pointer to the string.

Reference

```
CDAQMX100::getClassMXItemConfig  
CDAQMXChConfig::getTag  
CDAQMXItemConfig::getClassMXChConfig
```

getChannelUnitMX100 [Visual C only]

Syntax

```
const char * getChannelUnitMX100(DAQMX100 daqmx100, int chNo);
```

Parameters

daqmx100	Specify the device descriptor.
chNo	Specify the channel number.

Description

Gets the specified channel number and unit name from the stored current channel setting data.

- Returns NULL if it does not exist.

Return value

Returns a pointer to the string.

Reference

```
CDAQMX100::getClassMXItemConfig  
CDAQMXChConfig::getUnit  
CDAQMXItemConfig::getClassMXChConfig
```

getErrorMessageMX100 **[Visual C only]**

Syntax

```
const char * getErrorMessageMX100(int errorCode);
```

Parameters

errorCode Specify the error number.

Description

Gets the error message string corresponding to the specified error number.

- Returns a pointer to the string [Unknown] if it does not exist.

Return value

Returns a pointer to the string.

Reference

CDAQMX100::getErrorMessage

getModuleSerialMX100 [Visual C only]

Syntax

```
const char * getModuleSerialMX100(DAQMX100 daqmx100, int  
moduleNo);
```

Parameters

daqmx100	Specify the device descriptor.
moduleNo	Specify the module number.

Description

Gets the serial number of the specified module number from the stored current system configuration data.

- Returns NULL if it does not exist.

Return value

Returns a pointer to the string.

Reference

```
CDAQMX100::getClassMXItemConfig  
CDAQMXItemConfig::getClassMXSysInfo  
CDAQMXSysInfo::getModuleSerial
```

getNetHostMX100 **[Visual C only]**

Syntax

```
const char * getNetHostMX100(DAQMX100 daqmx100);
```

Parameters

daqmx100 Specify the device descriptor.

Description

Gets the host name from the stored current network information data.

- Returns NULL if it does not exist.

Return value

Returns a pointer to the string.

Reference

CDAQMX100::getClassMXItemConfig
CDAQMXItemConfig::getClassMXNetInfo
CDAQMXNetInfo::getHost

getUnitPartNoMX100**[Visual C only]****Syntax**

```
const char * getUnitPartNoMX100(DAQMX100 daqmx100);
```

Parameters

daqmx100 Specify the device descriptor.

Description

Gets the part number from the stored current system configuration data.

- Returns NULL if it does not exist.

Return value

Returns a pointer to the string.

Reference

CDAQMX100::getClassMXItemConfig

CDAQMXItemConfig::getClassMXSysInfo

CDAQMXSysInfo::getPartNo

getUnitSerialMX100

[Visual C only]

Syntax

```
const char * getUnitSerialMX100(DAQMX100 daqmx100);
```

Parameters

daqmx100 Specify the device descriptor.

Description

Gets the unit's serial number from the stored current system configuration data.

- Returns NULL if it does not exist.

Return value

Returns a pointer to the string.

Reference

CDAQMX100::getClassMXItemConfig
CDAQMXItemConfig::getClassMXSysInfo
CDAQMXSysInfo::getUnitSerial

itemErrorMX100

Syntax

```
int itemErrorMX100(DAQMX100 daqmx100);
```

Declaration

Visual Basic

```
Public Declare Function itemErrorMX100 Lib "DAQMX100" (ByVal daqmx100 As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function itemErrorMX100 Lib "DAQMX100" (ByVal daqmx100 As Integer) As Integer
```

C#

```
[DllImport("DAQMX100.dll" CharSet=CharSet.Auto, EntryPoint="itemErrorMX100")]  
public static extern int itemErrorMX100(int daqmx100);
```

Parameters

daqmx100 Specify the device descriptor.

Description

Gets the setup item number on which an error was last detected.

- If it does not exist, "Unknown" is returned.

Return value

Returns the setting item number.

Reference

CDAQMX100::getItemError

itemMaxLengthMX100

Syntax

```
int itemMaxLengthMX100(void);
```

Declaration

Visual Basic

```
Public Declare Function itemMaxLengthMX100 Lib "DAQMX100"() As Long
```

Visual Basic.NET

```
Public Declare Ansi Function itemMaxLengthMX100 Lib "DAQMX100"() As Integer
```

C#

```
[DllImport("DAQMX100.dll" CharSet=CharSet.Auto, EntryPoint="itemMaxLengthMX100")]  
public static extern int itemMaxLengthMX100();
```

Description

Gets the maximum length of the name string corresponding to the setup item.

- The return value does not include the terminator.

Return value

Returns the length of the string.

Reference

CDAQMXItemConfig::getMaxLenItemName

lastErrorMX100

Syntax

```
int lastErrorMX100(DAQMX100 daqmx100);
```

Declaration

Visual Basic

```
Public Declare Function lastErrorMX100 Lib "DAQMX100" (ByVal daqmx100 As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function lastErrorMX100 Lib "DAQMX100" (ByVal daqmx100 As Integer) As Integer
```

C#

```
[DllImport("DAQMX100.dll", CharSet=CharSet.Auto, EntryPoint="lastErrorMX100")]  
public static extern int lastErrorMX100(int daqmx100);
```

Parameters

daqmx100 Specify the device descriptor.

Description

Gets the MX-specific error received in the last communication.

- Returns 0 if it does not exist.

Return value

Returns the MX-specific error.

Reference

CDAQMX100::getLastError

moduleChNumMX100

Syntax

```
int moduleChNumMX100(DAQMX100 daqmx100, int moduleNo);
```

Declaration

Visual Basic

```
Public Declare Function moduleChNumMX100 Lib "DAQMX100" (ByVal daqmx100 As Long, ByVal moduleNo As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function moduleChNumMX100 Lib "DAQMX100" (ByVal daqmx100 As Integer, ByVal moduleNo As Integer) As Integer
```

C#

```
[DllImport("DAQMX100.dll" CharSet=CharSet.Auto, EntryPoint="moduleChNumMX100")]  
public static extern int moduleChNumMX100(int daqmx100, int moduleNo);
```

Parameters

daqmx100	Specify the device descriptor.
moduleNo	Specify the module number.

Description

Gets the number of channels of the specified module number from the stored current system configuration data.

- Returns 0 if it does not exist.

Return value

Returns the number of channels.

Reference

```
CDAQMX100::getClassMXItemConfig  
CDAQMXItemConfig::getClassMXSysInfo  
CDAQMXSysInfo::getChNum
```

moduleFIFONoMX100

Syntax

```
int moduleFIFONoMX100(DAQMX100 daqmx100, int moduleNo);
```

Declaration**Visual Basic**

```
Public Declare Function moduleFIFONoMX100 Lib "DAQMX100" (ByVal daqmx100 As Long, ByVal moduleNo As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function moduleFIFONoMX100 Lib "DAQMX100" (ByVal daqmx100 As Integer, ByVal moduleNo As Integer) As Integer
```

C#

```
[DllImport("DAQMX100.dll" CharSet=CharSet.Auto, EntryPoint="moduleFIFONoMX100")]
public static extern int moduleFIFONoMX100(int daqmx100, int moduleNo);
```

Parameters

daqmx100	Specify the device descriptor.
moduleNo	Specify the module number.

Description

Gets the FIFO number of the specified module number from the stored current system configuration data.

- Returns a negative number if it does not exist.

Return value

Returns the FIFO number.

Reference

```
CDAQMX100::getClassMXItemConfig
CDAQMXItemConfig::getClassMXSysInfo
CDAQMXSysInfo::getFIFONo
```

moduleIntegralMX100

Syntax

```
int moduleIntegralMX100(DAQMX100 daqmx100, int moduleNo);
```

Declaration

Visual Basic

```
Public Declare Function moduleIntegralMX100 Lib  
"DAQMX100"(ByVal daqmx100 As Long, ByVal moduleNo As Long) As  
Long
```

Visual Basic.NET

```
Public Declare Ansi Function moduleIntegralMX100 Lib  
"DAQMX100"(ByVal daqmx100 As Integer, ByVal moduleNo As  
Integer) As Integer
```

C#

```
[DllImport("DAQMX100.dll" CharSet=CharSet.Auto,  
EntryPoint="moduleIntegralMX100")]  
public static extern int moduleIntegralMX100(int daqmx100, int  
moduleNo);
```

Parameters

daqmx100	Specify the device descriptor.
moduleNo	Specify the module number.

Description

Gets the A/D integral time type of the specified module number from the stored current system configuration data.

- If it does not exist, "Automatic" is returned.

Return value

Returns the type of AD integral time.

Reference

```
CDAQMX100::getClassMXItemConfig  
CDAQMXItemConfig::getClassMXSysInfo  
CDAQMXSysInfo::getIntegral
```

moduleIntervalMX100

Syntax

```
int moduleIntervalMX100(DAQMX100 daqmx100, int moduleNo);
```

Declaration**Visual Basic**

```
Public Declare Function moduleIntervalMX100 Lib
"DAQMX100"(ByVal daqmx100 As Long, ByVal moduleNo As Long) As
Long
```

Visual Basic.NET

```
Public Declare Ansi Function moduleIntervalMX100 Lib
"DAQMX100"(ByVal daqmx100 As Integer, ByVal moduleNo As
Integer) As Integer
```

C#

```
[DllImport("DAQMX100.dll" CharSet=CharSet.Auto,
EntryPoint="moduleIntervalMX100")]
public static extern int moduleIntervalMX100(int daqmx100, int
moduleNo);
```

Parameters

daqmx100	Specify the device descriptor.
moduleNo	Specify the module number.

Description

Gets the interval type of the specified module number from the stored current system configuration data.

- Returns 0 if it does not exist.

Return value

Returns the interval type.

Reference

```
CDAQMX100::getClassMXItemConfig
CDAQMXItemConfig::getClassMXSysInfo
CDAQMXSysInfo::getInterval
```

moduleRealTypeMX100

Syntax

```
int moduleRealTypeMX100(DAQMX100 daqmx100, int moduleNo);
```

Declaration

Visual Basic

```
Public Declare Function moduleRealTypeMX100 Lib  
"DAQMX100"(ByVal daqmx100 As Long, ByVal moduleNo As Long) As  
Long
```

Visual Basic.NET

```
Public Declare Ansi Function moduleRealTypeMX100 Lib  
"DAQMX100"(ByVal daqmx100 As Integer, ByVal moduleNo As  
Integer) As Integer
```

C#

```
[DllImport("DAQMX100.dll" CharSet=CharSet.Auto,  
EntryPoint="moduleRealTypeMX100")]  
public static extern int moduleRealTypeMX100(int daqmx100, int  
moduleNo);
```

Parameters

daqmx100	Specify the device descriptor.
moduleNo	Specify the module number.

Description

Gets the actual module type of the specified module number from the stored current system configuration data.

- If it does not exist, "No module" is returned.

Return value

Returns the module type.

Reference

```
CDAQMX100::getClassMXItemConfig  
CDAQMXItemConfig::getClassMXSysInfo  
CDAQMXSysInfo::getRealType
```

moduleStandbyTypeMX100

Syntax

```
int moduleStandbyTypeMX100(DAQMX100 daqmx100, int moduleNo);
```

Declaration**Visual Basic**

```
Public Declare Function moduleStandbyTypeMX100 Lib
"DAQMX100"(ByVal daqmx100 As Long, ByVal moduleNo As Long) As
Long
```

Visual Basic.NET

```
Public Declare Ansi Function moduleStandbyTypeMX100 Lib
"DAQMX100"(ByVal daqmx100 As Integer, ByVal moduleNo As
Integer) As Integer
```

C#

```
[DllImport("DAQMX100.dll" CharSet=CharSet.Auto,
EntryPoint="moduleStandbyTypeMX100")]
public static extern int moduleStandbyTypeMX100(int daqmx100,
int moduleNo);
```

Parameters

daqmx100	Specify the device descriptor.
moduleNo	Specify the module number.

Description

Gets the startup module type of the specified module number from the stored current system configuration data.

- If it does not exist, "No module" is returned.

Return value

Returns the module type.

Reference

```
CDAQMX100::getClassMXItemConfig
CDAQMXItemConfig::getClassMXSysInfo
CDAQMXSysInfo::getStandbyType
```

moduleTerminalMX100

Syntax

```
int moduleTerminalMX100(DAQMX100 daqmx100, int moduleNo);
```

Declaration

Visual Basic

```
Public Declare Function moduleTerminalMX100 Lib  
"DAQMX100"(ByVal daqmx100 As Long, ByVal moduleNo As Long) As  
Long
```

Visual Basic.NET

```
Public Declare Ansi Function moduleTerminalMX100 Lib  
"DAQMX100"(ByVal daqmx100 As Integer, ByVal moduleNo As  
Integer) As Integer
```

C#

```
[DllImport("DAQMX100.dll" CharSet=CharSet.Auto,  
EntryPoint="moduleTerminalMX100")]  
public static extern int moduleTerminalMX100(int daqmx100, int  
moduleNo);
```

Parameters

daqmx100	Specify the device descriptor.
moduleNo	Specify the module number.

Description

Gets the terminal type of the specified module number from the stored current system configuration data.

- If it does not exist, "Screw" is returned.

Return value

Returns the terminal type.

Reference

```
CDAQMX100::getClassMXItemConfig  
CDAQMXItemConfig::getClassMXSysInfo  
CDAQMXSysInfo::getTerminalType
```

moduleTypeMX100

Syntax

```
int moduleTypeMX100(DAQMX100 daqmx100, int moduleNo);
```

Declaration

Visual Basic

```
Public Declare Function moduleTypeMX100 Lib "DAQMX100" (ByVal daqmx100 As Long, ByVal moduleNo As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function moduleTypeMX100 Lib "DAQMX100" (ByVal daqmx100 As Integer, ByVal moduleNo As Integer) As Integer
```

C#

```
[DllImport("DAQMX100.dll" CharSet=CharSet.Auto, EntryPoint="moduleTypeMX100")]  
public static extern int moduleTypeMX100(int daqmx100, int moduleNo);
```

Parameters

daqmx100	Specify the device descriptor.
moduleNo	Specify the module number.

Description

Gets the module type of the specified module number from the stored current system configuration data.

- If it does not exist, "No module" is returned.

Return value

Returns the module type.

Reference

```
CDAQMX100::getClassMXItemConfig  
CDAQMXItemConfig::getClassMXSysInfo  
CDAQMXSysInfo::getModuleType
```

moduleValidMX100

Syntax

```
int moduleValidMX100(DAQMX100 daqmx100, int moduleNo);
```

Declaration

Visual Basic

```
Public Declare Function moduleValidMX100 Lib "DAQMX100" (ByVal daqmx100 As Long, ByVal moduleNo As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function moduleValidMX100 Lib "DAQMX100" (ByVal daqmx100 As Integer, ByVal moduleNo As Integer) As Integer
```

C#

```
[DllImport("DAQMX100.dll" CharSet=CharSet.Auto, EntryPoint="moduleValidMX100")]  
public static extern int moduleValidMX100(int daqmx100, int moduleNo);
```

Parameters

daqmx100	Specify the device descriptor.
moduleNo	Specify the module number.

Description

Gets valid/invalid of the specified module number from the stored current system configuration data as a Boolean value.

- If it does not exist, Invalid is returned.

Return value

Returns a Boolean value.

Reference

```
CDAQMX100::getClassMXItemConfig  
CDAQMXItemConfig::getClassMXSysInfo  
CDAQMXSysInfo::isModuleValid
```

moduleVersionMX100

Syntax

```
int moduleVersionMX100(DAQMX100 daqmx100, int moduleNo);
```

Declaration**Visual Basic**

```
Public Declare Function moduleVersionMX100 Lib
"DAQMX100"(ByVal daqmx100 As Long, ByVal moduleNo As Long) As
Long
```

Visual Basic.NET

```
Public Declare Ansi Function moduleVersionMX100 Lib
"DAQMX100"(ByVal daqmx100 As Integer, ByVal moduleNo As
Integer) As Integer
```

C#

```
[DllImport("DAQMX100.dll" CharSet=CharSet.Auto,
EntryPoint="moduleVersionMX100")]
public static extern int moduleVersionMX100(int daqmx100, int
moduleNo);
```

Parameters

daqmx100	Specify the device descriptor.
moduleNo	Specify the module number.

Description

Gets the module version of the specified module number from the stored current system configuration data.

- Returns 0 if it does not exist.

Return value

Returns the version.

Reference

```
CDAQMX100::getClassMXItemConfig
CDAQMXItemConfig::getClassMXSysInfo
CDAQMXSysInfo::getModuleVersion
```

netAddressMX100

Syntax

```
unsigned int netAddressMX100(DAQMX100 daqmx100);
```

Declaration

Visual Basic

```
Public Declare Function netAddressMX100 Lib "DAQMX100" (ByVal daqmx100 As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function netAddressMX100 Lib "DAQMX100" (ByVal daqmx100 As Integer) As Integer
```

C#

```
[DllImport("DAQMX100.dll" CharSet=CharSet.Auto, EntryPoint="netAddressMX100")]  
public static extern int netAddressMX100(int daqmx100);
```

Parameters

daqmx100 Specify the device descriptor.

Description

Gets the IP address from the stored current network information data.

- Returns 0 if it does not exist.

Return value

Returns the IP address

Reference

```
CDAQMX100::getClassMXItemConfig  
CDAQMXItemConfig::getClassMXNetInfo  
CDAQMXNetInfo::getAddress
```

netGatewayMX100

Syntax

```
unsigned int netGatewayMX100(DAQMX100 daqmx100);
```

Declaration

Visual Basic

```
Public Declare Function netGatewayMX100 Lib "DAQMX100" (ByVal daqmx100 As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function netGatewayMX100 Lib "DAQMX100" (ByVal daqmx100 As Integer) As Integer
```

C#

```
[DllImport("DAQMX100.dll", CharSet=CharSet.Auto, EntryPoint="netGatewayMX100")]  
public static extern int netGatewayMX100(int daqmx100);
```

Parameters

daqmx100 Specify the device descriptor.

Description

Gets the Gateway address from the stored current network information data.

- Returns 0 if it does not exist.

Return value

Returns the Gateway address.

Reference

```
CDAQMX100::getClassMXItemConfig  
CDAQMXItemConfig::getClassMXNetInfo  
CDAQMXNetInfo::getGateway
```

netPortMX100

Syntax

```
unsigned int netPortMX100(DAQMX100 daqmx100);
```

Declaration

Visual Basic

```
Public Declare Function netPortMX100 Lib "DAQMX100" (ByVal daqmx100 As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function netPortMX100 Lib "DAQMX100" (ByVal daqmx100 As Integer) As Integer
```

C#

```
[DllImport("DAQMX100.dll" CharSet=CharSet.Auto, EntryPoint="netPortMX100")]  
public static extern int netPortMX100(int daqmx100);
```

Parameters

daqmx100 Specify the device descriptor.

Description

Gets the port number from the stored current network information data.

- Returns 0 if it does not exist.

Return value

Returns the port number.

Reference

```
CDAQMX100::getClassMXItemConfig  
CDAQMXItemConfig::getClassMXNetInfo  
CDAQMXNetInfo::getPort
```

netSubmaskMX100

Syntax

```
unsigned int netSubmaskMX100(DAQMX100 daqmx100);
```

Declaration

Visual Basic

```
Public Declare Function netSubmaskMX100 Lib "DAQMX100" (ByVal daqmx100 As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function netSubmaskMX100 Lib "DAQMX100" (ByVal daqmx100 As Integer) As Integer
```

C#

```
[DllImport("DAQMX100.dll", CharSet=CharSet.Auto, EntryPoint="netSubmaskMX100")]  
public static extern int netSubmaskMX100(int daqmx100);
```

Parameters

daqmx100 Specify the device descriptor.

Description

Gets the subnet mask from the stored current network information data.

- Returns 0 if it does not exist.

Return value

Returns the subnet mask.

Reference

```
CDAQMX100::getClassMXItemConfig  
CDAQMXItemConfig::getClassMXNetInfo  
CDAQMXNetInfo::getSubMask
```

rangePointMX100

Syntax

```
int rangePointMX100(DAQMX100 daqmx100, int iRange);
```

Declaration

Visual Basic

```
Public Declare Function rangePointMX100 Lib "DAQMX100" (ByVal daqmx100 As Long, ByVal iRange As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function rangePointMX100 Lib "DAQMX100" (ByVal daqmx100 As Integer, ByVal iRange As Integer) As Integer
```

C#

```
[DllImport("DAQMX100.dll" CharSet=CharSet.Auto, EntryPoint="rangePointMX100")]  
public static extern int rangePointMX100(int daqmx100, int iRange);
```

Parameters

daqmx100	Specify the device descriptor.
iRange	Specify the range type.

Description

Gets the decimal point position of the specified range type.

- For a range type of digital input (DI), specify the detailed range of the digital input. Specification cannot be made without the module type.
- Returns 0 if it does not exist.

Return value

Returns the decimal point position.

Reference

CDAQMX100::getClassMXItemConfig
CDAQMXItemConfig::getRangePoint

revisionAPIMX100

Syntax

```
const int revisionAPIMX100(void);
```

Declaration

Visual Basic

```
Public Declare Function revisionAPIMX100 Lib "DAQMX100"() As Long
```

Visual Basic.NET

```
Public Declare Ansi Function revisionAPIMX100 Lib "DAQMX100"()  
As Integer
```

C#

```
[DllImport("DAQMX100.dll" CharSet=CharSet.Auto,  
EntryPoint="revisionAPIMX100")]  
public static extern int revisionAPIMX100();
```

Description

Gets the revision number of this API.

Return value

Returns the revision number.

Reference

CDAQMX100::getRevisionAPIMX

statusBackupMX100

Syntax

```
int statusBackupMX100(DAQMX100 daqmx100);
```

Declaration

Visual Basic

```
Public Declare Function statusBackupMX100 Lib "DAQMX100" (ByVal daqmx100 As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function statusBackupMX100 Lib "DAQMX100" (ByVal daqmx100 As Integer) As Integer
```

C#

```
[DllImport("DAQMX100.dll" CharSet=CharSet.Auto, EntryPoint="statusBackupMX100")]  
public static extern int statusBackupMX100(int daqmx100);
```

Parameters

daqmx100 Specify the device descriptor.

Description

Gets the Boolean value for the backup specification from the stored current status data.

- If it does not exist, Invalid is returned.

Return value

Returns a Boolean value.

Reference

```
CDAQMX100::getClassMXItemConfig  
CDAQMXItemConfig::getClassMXStatus  
CDAQMXStatus::isBackup
```

statusCFMX100

Syntax

```
int statusCFMX100(DAQMX100 daqmx100);
```

Declaration

Visual Basic

```
Public Declare Function statusCFMX100 Lib "DAQMX100" (ByVal daqmx100 As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function statusCFMX100 Lib "DAQMX100" (ByVal daqmx100 As Integer) As Integer
```

C#

```
[DllImport("DAQMX100.dll", CharSet=CharSet.Auto, EntryPoint="statusCFMX100")]  
public static extern int statusCFMX100(int daqmx100);
```

Parameters

daqmx100 Specify the device descriptor.

Description

Gets the CF status type from the stored current status data.

- Returns "All Off" if it does not exist.

Return value

Returns the CF status type.

Reference

CDAQMX100::getClassMXItemConfig
CDAQMXItemConfig::getClassMXStatus
CDAQMXStatus::getCFStatus

statusCFRemainMX100

Syntax

```
int statusCFRemainMX100(DAQMX100 daqmx100);
```

Declaration

Visual Basic

```
Public Declare Function statusCFRemainMX100 Lib  
"DAQMX100"(ByVal daqmx100 As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function statusCFRemainMX100 Lib  
"DAQMX100"(ByVal daqmx100 As Integer) As Integer
```

C#

```
[DllImport("DAQMX100.dll" CharSet=CharSet.Auto,  
EntryPoint="statusCFRemainMX100")]  
public static extern int statusCFRemainMX100(int daqmx100);
```

Parameters

daqmx100 Specify the device descriptor.

Description

Gets the remaining capacity of the CF from the stored current status data.

- The unit is KB.
- Returns 0 if it does not exist.

Return value

Returns the remaining size.

Reference

```
CDAQMX100::getClassMXItemConfig  
CDAQMXItemConfig::getClassMXStatus  
CDAQMXStatus::getCFRemain
```

statusCFSizeMX100

Syntax

```
int statusCFSizeMX100(DAQMX100 daqmx100);
```

Declaration

Visual Basic

```
Public Declare Function statusCFSizeMX100 Lib "DAQMX100" (ByVal daqmx100 As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function statusCFSizeMX100 Lib "DAQMX100" (ByVal daqmx100 As Integer) As Integer
```

C#

```
[DllImport("DAQMX100.dll" CharSet=CharSet.Auto, EntryPoint="statusCFSizeMX100")]  
public static extern int statusCFSizeMX100(int daqmx100);
```

Parameters

daqmx100 Specify the device descriptor.

Description

Gets the size of the CF from the stored current status data.

- The unit is KB.
- Returns 0 if it does not exist.

Return value

Returns the size.

Reference

```
CDAQMX100::getClassMXItemConfig  
CDAQMXItemConfig::getClassMXStatus  
CDAQMXStatus::getCFSize
```

statusDayMX100

Syntax

```
int statusDayMX100(DAQMX100 daqmx100);
```

Declaration

Visual Basic

```
Public Declare Function statusDayMX100 Lib "DAQMX100" (ByVal daqmx100 As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function statusDayMX100 Lib "DAQMX100" (ByVal daqmx100 As Integer) As Integer
```

C#

```
[DllImport("DAQMX100.dll" CharSet=CharSet.Auto, EntryPoint="statusDayMX100")]  
public static extern int statusDayMX100(int daqmx100);
```

Parameters

daqmx100 Specify the device descriptor.

Description

Returns the day from the stored current status data.

- Converts the number of seconds from the reference date/time and returns the result.
- The day is a number from 1 to 31.
- Returns 0 if it does not exist.

Return value

Returns the day value.

Reference

statusTimeMX100
CDAQMXDateTime::toLocalDateTime

statusFIFOIntervalMX100

Syntax

```
int statusFIFOIntervalMX100(DAQMX100 daqmx100, int fifoNo);
```

Declaration**Visual Basic**

```
Public Declare Function statusFIFOIntervalMX100 Lib
"DAQMX100"(ByVal daqmx100 As Long, ByVal fifoNo As Long) As
Long
```

Visual Basic.NET

```
Public Declare Ansi Function statusFIFOIntervalMX100 Lib
"DAQMX100"(ByVal daqmx100 As Integer, ByVal fifoNo As Integer)
As Integer
```

C#

```
[DllImport("DAQMX100.dll" CharSet=CharSet.Auto,
EntryPoint="statusFIFOIntervalMX100")]
public static extern int statusFIFOIntervalMX100(int daqmx100,
int fifoNo);
```

Parameters

daqmx100	Specify the device descriptor.
fifoNo	Specify the FIFO number.

Description

Gets the interval type of the specified FIFO number from the stored current status data.

- Returns 0 if it does not exist.

Return value

Returns the interval type.

Reference

```
CDAQMX100::getClassMXItemConfig
CDAQMXItemConfig::getClassMXStatus
CDAQMXStatus::getInterval
```

statusFIFOMX100

Syntax

```
int statusFIFOMX100(DAQMX100 daqmx100, int fifoNo);
```

Declaration

Visual Basic

```
Public Declare Function statusFIFOMX100 Lib "DAQMX100" (ByVal daqmx100 As Long, ByVal fifoNo As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function statusFIFOMX100 Lib "DAQMX100" (ByVal daqmx100 As Integer, ByVal fifoNo As Integer) As Integer
```

C#

```
[DllImport("DAQMX100.dll" CharSet=CharSet.Auto, EntryPoint="statusFIFOMX100")]  
public static extern int statusFIFOMX100(int daqmx100, int fifoNo);
```

Parameters

daqmx100	Specify the device descriptor.
fifoNo	Specify the FIFO number.

Description

Gets the FIFO status value of the specified FIFO number from the stored current status data.

- If it does not exist, "Unknown" is returned.

Return value

Returns the FIFO status value.

Reference

```
CDAQMX100::getClassMXItemConfig  
CDAQMXItemConfig::getClassMXStatus  
CDAQMXStatus::getFIFOStatus
```

statusFIFONumMX100

Syntax

```
int statusFIFONumMX100(DAQMX100 daqmx100);
```

Declaration

Visual Basic

```
Public Declare Function statusFIFONumMX100 Lib  
"DAQMX100"(ByVal daqmx100 As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function statusFIFONumMX100 Lib  
"DAQMX100"(ByVal daqmx100 As Integer) As Integer
```

C#

```
[DllImport("DAQMX100.dll" CharSet=CharSet.Auto,  
EntryPoint="statusFIFONumMX100")]  
public static extern int statusFIFONumMX100(int daqmx100);
```

Parameters

daqmx100 Specifies the device descriptor.

Description

Gets the number of valid FIFOs from the stored current status data.

- Returns 0 if it does not exist.

Return value

Returns the valid number of FIFOs.

Reference

CDAQMX100::getClassMXItemConfig
CDAQMXItemConfig::getClassMXStatus
CDAQMXStatus::getFIFONum

statusHourMX100

Syntax

```
int statusHourMX100(DAQMX100 daqmx100);
```

Declaration

Visual Basic

```
Public Declare Function statusHourMX100 Lib "DAQMX100" (ByVal daqmx100 As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function statusHourMX100 Lib "DAQMX100" (ByVal daqmx100 As Integer) As Integer
```

C#

```
[DllImport("DAQMX100.dll" CharSet=CharSet.Auto, EntryPoint="statusHourMX100")]  
public static extern int statusHourMX100(int daqmx100);
```

Parameters

daqmx100 Specify the device descriptor.

Description

Gets the hour from the stored current status data.

- Converts the number of seconds from the reference date/time and returns the result.
- The hour is a number from 0 to 23.
- Returns 0 if it does not exist.

Return value

Returns the hour value.

Reference

statusTimeMX100
CDAQMXDateTime::toLocalDateTime

statusMilliSecMX100

Syntax

```
int statusMilliSecMX100(DAQMX100 daqmx100);
```

Declaration

Visual Basic

```
Public Declare Function statusMilliSecMX100 Lib  
"DAQMX100"(ByVal daqmx100 As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function statusMilliSecMX100 Lib  
"DAQMX100"(ByVal daqmx100 As Integer) As Integer
```

C#

```
[DllImport("DAQMX100.dll" CharSet=CharSet.Auto,  
EntryPoint="statusMilliSecMX100")]  
public static extern int statusMilliSecMX100(int daqmx100);
```

Parameters

daqmx100 Specify the device descriptor.

Description

Gets the milliseconds from the stored current status data.

- Returns 0 if it does not exist.

Return value

Returns the milliseconds value.

Reference

```
CDAQMX100::getClassMXItemConfig  
CDAQMXItemConfig::getClassMXStatus  
CDAQMXStatus::getMilliSecond
```

statusMinuteMX100

Syntax

```
int statusMinuteMX100(DAQMX100 daqmx100);
```

Declaration

Visual Basic

```
Public Declare Function statusMinuteMX100 Lib "DAQMX100" (ByVal daqmx100 As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function statusMinuteMX100 Lib "DAQMX100" (ByVal daqmx100 As Integer) As Integer
```

C#

```
[DllImport("DAQMX100.dll" CharSet=CharSet.Auto, EntryPoint="statusMinuteMX100")]  
public static extern int statusMinuteMX100(int daqmx100);
```

Parameters

daqmx100 Specify the device descriptor.

Description

Gets the minutes from the stored current status data.

- Converts the number of seconds from the reference date/time and returns the result.
- The minute is a number from 0 to 59.
- Returns 0 if it does not exist.

Return value

Returns the minute value.

Reference

```
statusTimeMX100  
CDAQMXDateTime::toLocalDateTime
```

statusMonthMX100

Syntax

```
int statusMonthMX100(DAQMX100 daqmx100);
```

Declaration

Visual Basic

```
Public Declare Function statusMonthMX100 Lib "DAQMX100" (ByVal daqmx100 As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function statusMonthMX100 Lib "DAQMX100" (ByVal daqmx100 As Integer) As Integer
```

C#

```
[DllImport("DAQMX100.dll", CharSet=CharSet.Auto, EntryPoint="statusMonthMX100")]  
public static extern int statusMonthMX100(int daqmx100);
```

Parameters

daqmx100 Specify the device descriptor.

Description

Gets the month from the stored current status data.

- Converts the number of seconds from the reference date/time and returns the result.
- The month is a number from 1 to 12.
- Returns 0 if it does not exist.

Return value

Returns the month value.

Reference

statusTimeMX100
CDAQMXDateTime::toLocalDateTime

statusSecondMX100

Syntax

```
int statusSecondMX100(DAQMX100 daqmx100);
```

Declaration

Visual Basic

```
Public Declare Function statusSecondMX100 Lib "DAQMX100" (ByVal daqmx100 As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function statusSecondMX100 Lib "DAQMX100" (ByVal daqmx100 As Integer) As Integer
```

C#

```
[DllImport("DAQMX100.dll" CharSet=CharSet.Auto, EntryPoint="statusSecondMX100")]  
public static extern int statusSecondMX100(int daqmx100);
```

Parameters

daqmx100 Specify the device descriptor.

Description

Gets the seconds from the stored current status data.

- Converts the number of seconds from the reference date/time and returns the result.
- The second is a number from 0 to 59.
- Returns 0 if it does not exist.

Return value

Returns the seconds value.

Reference

statusTimeMX100
CDAQMXDateTime::toLocalDateTime

statusTimeMX100

Syntax

```
int statusTimeMX100(DAQMX100 daqmx100);
```

Declaration

Visual Basic

```
Public Declare Function statusTimeMX100 Lib "DAQMX100" (ByVal daqmx100 As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function statusTimeMX100 Lib "DAQMX100" (ByVal daqmx100 As Integer) As Integer
```

C#

```
[DllImport("DAQMX100.dll" CharSet=CharSet.Auto, EntryPoint="statusTimeMX100")]  
public static extern int statusTimeMX100(int daqmx100);
```

Parameters

daqmx100 Specify the device descriptor.

Description

Gets the seconds from the stored current status data.

- This is the number of seconds from the reference date/time (Jan. 1, 1970).
- Returns 0 if it does not exist.

Return value

Returns seconds.

Reference

```
CDAQMX100::getClassMXItemConfig  
CDAQMXItemConfig::getClassMXStatus  
CDAQMXStatus::getTime
```

statusUnitMX100

Syntax

```
int statusUnitMX100(DAQMX100 daqmx100);
```

Declaration

Visual Basic

```
Public Declare Function statusUnitMX100 Lib "DAQMX100" (ByVal daqmx100 As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function statusUnitMX100 Lib "DAQMX100" (ByVal daqmx100 As Integer) As Integer
```

C#

```
[DllImport("DAQMX100.dll" CharSet=CharSet.Auto, EntryPoint="statusUnitMX100")]  
public static extern int statusUnitMX100(int daqmx100);
```

Parameters

daqmx100 Specify the device descriptor.

Description

Gets the unit status value from the stored current status data.

- If it does not exist, "Unknown" is returned.

Return value

Returns the unit status value.

Reference

```
CDAQMX100::getClassMXItemConfig  
CDAQMXItemConfig::getClassMXStatus  
CDAQMXStatus::getUnitStatus
```

statusYearMX100

Syntax

```
int statusYearMX100(DAQMX100 daqmx100);
```

Declaration

Visual Basic

```
Public Declare Function statusYearMX100 Lib "DAQMX100" (ByVal daqmx100 As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function statusYearMX100 Lib "DAQMX100" (ByVal daqmx100 As Integer) As Integer
```

C#

```
[DllImport("DAQMX100.dll", CharSet=CharSet.Auto, EntryPoint="statusYearMX100")]  
public static extern int statusYearMX100(int daqmx100);
```

Parameters

daqmx100 Specify the device descriptor.

Description

Gets the year from the stored current status data.

- Converts the number of seconds from the reference date/time and returns the result.
- The year is a 4-digit number.
- Returns 0 if it does not exist.

Return value

Returns the year value.

Reference

```
statusTimeMX100  
CDAQMXDateTime::toLocalDateTime
```

toAlarmNameMX100

Syntax

```
int toAlarmNameMX100(int iAlarmType, char * strAlarm, int lenAlarm);
```

Declaration

Visual Basic

```
Public Declare Function toAlarmNameMX100 Lib "DAQMX100" (ByVal iAlarmType As Long, ByVal strAlarm As String, ByVal lenAlarm As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function toAlarmNameMX100 Lib "DAQMX100" (ByVal iAlarmType As Integer, ByVal strAlarm As String, ByVal lenAlarm As Integer) As Integer
```

C#

```
[DllImport("DAQMX100.dll" CharSet=CharSet.Auto, EntryPoint="toAlarmNameMX100")]  
public static extern int toAlarmNameMX100(int iAlarmType, byte[] strAlarm, int lenAlarm);
```

Parameters

iAlarmType	Specify the alarm type.
strAlarm	Specify the field where the string is to be stored.
lenAlarm	Specify the byte size of the field where the string is to be stored.

Description

Stores the string corresponding to the specified alarm type to the specified field.

- The string stored in the field includes the terminator (NULL).
- The return value is the length of the actual string. The return value does not include the terminator.
- The strings that can be stored are, in general, ASCII strings.

Return value

Returns the length of the string.

Reference

getAlarmNameMX100

toAOPWMValueMX100

Syntax

```
int toAOPWMValueMX100(double realValue, int iRangeAOPWM);
```

Declaration

Visual Basic

```
Public Declare Function toAOPWMValueMX100 Lib "DAQMX100" (ByVal  
realValue As Double, ByVal iRangeAOPWM As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function toAOPWMValueMX100 Lib  
"DAQMX100" (ByVal realValue As Double, ByVal iRangeAOPWM As  
Integer) As Integer
```

C#

```
[DllImport("DAQMX100.dll" CharSet=CharSet.Auto,  
EntryPoint="toAOPWMValueMX100")]  
public static extern int toAOPWMValueMX100(double realValue,  
int iRangeAOPWM);
```

Parameters

realValue	Specify the actual output value.
iRangeAOPWM	Specify the range type.

Description

Converts the actual output values to AO/PWM data output data values according to the specified range type.

- Valid range types are AO and PWM.
- Returns 0 if it does not exist.

Return value

Returns the output data value.

Reference

CDAQMXAOPWMData::toAOPWMValue

toChannelCommentMX100

Syntax

```
int toChannelCommentMX100(DAQMX100 daqmx100, int chNo, char *
strComment, int lenComment);
```

Declaration

Visual Basic

```
Public Declare Function toChannelCommentMX100 Lib
"DAQMX100"(ByVal daqmx100 As Long, ByVal chNo As Long, ByVal
strComment As String, ByVal lenComment As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function toChannelCommentMX100 Lib
"DAQMX100"(ByVal daqmx100 As Integer, ByVal chNo As Integer,
ByVal strComment As String, ByVal lenComment As Integer) As
Integer
```

C#

```
[DllImport("DAQMX100.dll" CharSet=CharSet.Auto,
EntryPoint="toChannelCommentMX100")]
public static extern int toChannelCommentMX100(int daqmx100,
int chNo, byte[] strComment, int lenComment);
```

Parameters

daqmx100	Specify the device descriptor.
chNo	Specify the channel number.
strComment	Specify the field where the string is to be stored.
lenComment	Specify the byte size of the field where the string is to be stored.

Description

Gets the specified channel number and comment from the stored current channel setting data.

- Stores the string in the specified storage destination.
- The string stored in the field includes the terminator (NULL).
- The return value is the length of the actual string. The return value does not include the terminator.
- Returns 0 if it does not exist.
- The strings that can be stored are, in general, ASCII strings.

Return value

Returns the length of the actual string.

Reference

getChannelCommentMX100

toChannelTagMX100

Syntax

```
int toChannelTagMX100(DAQMX100 daqmx100, int chNo, char *
strTag, int lenTag);
```

Declaration**Visual Basic**

```
Public Declare Function toChannelTagMX100 Lib "DAQMX100" (ByVal
daqmx100 As Long, ByVal chNo As Long, ByVal strTag As String,
ByVal lenTag As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function toChannelTagMX100 Lib
"DAQMX100" (ByVal daqmx100 As Integer, ByVal chNo As Integer,
ByVal strTag As String, ByVal lenTag As Integer) As Integer
```

C#

```
[DllImport("DAQMX100.dll" CharSet=CharSet.Auto,
EntryPoint="toChannelTagMX100")]
public static extern int toChannelTagMX100(int daqmx100, int
chNo, byte[] strTag, int lenTag);
```

Parameters

daqmx100	Specify the device descriptor.
chNo	Specify the channel number.
strTag	Specify the field where the string is to be stored.
lenTag	Specify the byte size of the field where the string is to be stored.

Description

Gets the specified channel number and tag from the stored current channel setting data.

- Stores the string in the specified storage destination.
- The string stored in the field includes the terminator (NULL).
- The return value is the length of the actual string. The return value does not include the terminator.
- Returns 0 if it does not exist.
- The strings that can be stored are, in general, ASCII strings.

Return value

Returns the length of the actual string.

Reference

getChannelTagMX100

toChannelUnitMX100

Syntax

```
int toChannelUnitMX100(DAQMX100 daqmx100, int chNo, char *
strUnit, int lenUnit);
```

Declaration

Visual Basic

```
Public Declare Function toChannelUnitMX100 Lib
"DAQMX100"(ByVal daqmx100 As Long, ByVal chNo As Long, ByVal
strUnit As String, ByVal lenUnit As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function toChannelUnitMX100 Lib
"DAQMX100"(ByVal daqmx100 As Integer, ByVal chNo As Integer,
ByVal strUnit As String, ByVal lenUnit As Integer) As Integer
```

C#

```
[DllImport("DAQMX100.dll" CharSet=CharSet.Auto,
EntryPoint="toChannelUnitMX100")]
public static extern int toChannelUnitMX100(int daqmx100, int
chNo, byte[] strUnit, int lenUnit);
```

Parameters

daqmx100	Specify the device descriptor.
chNo	Specify the channel number.
strUnit	Specify the field where the string is to be stored.
lenUnit	Specify the byte size of the field where the string is to be stored.

Description

Gets the specified channel number and unit name from the stored current channel setting data.

- Stores the string in the specified storage destination.
- The string stored in the field includes the terminator (NULL).
- The return value is the length of the actual string. The return value does not include the terminator.
- Returns 0 if it does not exist.
- The strings that can be stored are, in general, ASCII strings.

Return value

Returns the length of the actual string.

Reference

getChannelUnitMX100

toDoubleValueMX100

Syntax

```
double toDoubleValueMX100(int dataValue, int point);
```

Declaration

Visual Basic

```
Public Declare Function toDoubleValueMX100 Lib  
"DAQMX100"(ByVal dataValue As Long, ByVal point As Long) As  
Double
```

Visual Basic.NET

```
Public Declare Ansi Function toDoubleValueMX100 Lib  
"DAQMX100"(ByVal dataValue As Integer, ByVal point As Integer)  
As Double
```

C#

```
[DllImport("DAQMX100.dll" CharSet=CharSet.Auto,  
EntryPoint="toDoubleValueMX100")]  
public static extern double toDoubleValueMX100(int dataValue,  
int point);
```

Parameters

dataValue	Specify the data value.
point	Specify the decimal point position.

Description

Generates the measured value from the specified data value and decimal point position.

Return value

Returns the measured value as a double-precision floating number.

Reference

CDAQMXDataInfo::toDoubleValue

toErrorMessageMX100

Syntax

```
int toErrorMessageMX100(int errorCode, char * errStr, int errLen);
```

Declaration

Visual Basic

```
Public Declare Function toErrorMessageMX100 Lib "DAQMX100" (ByVal errorCode As Long, ByVal errStr As String, ByVal errLen As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function toErrorMessageMX100 Lib "DAQMX100" (ByVal errorCode As Integer, ByVal errStr As String, ByVal errLen As Integer) As Integer
```

C#

```
[DllImport("DAQMX100.dll" CharSet=CharSet.Auto, EntryPoint="toErrorMessageMX100")]  
public static extern int toErrorMessageMX100(int errorCode, byte[] errStr, int errLen);
```

Parameters

errorCode	Specify the error number.
errStr	Specify the field where the string is to be stored.
errLen	Specify the byte size of the field where the string is to be stored.

Description

Stores the error message string corresponding to the error number to the specified field.

- The string stored in the field includes the terminator (NULL).
- The return value is the length of the actual string. The return value does not include the terminator.
- The strings that can be stored are, in general, ASCII strings.

Return value

Returns the length of the string.

Reference

getErrorMessageMX100

toItemNameMX100

Syntax

```
int toItemNameMX100(int itemNo, char * strItem, int lenItem);
```

Declaration**Visual Basic**

```
Public Declare Function toItemNameMX100 Lib "DAQMX100" (ByVal
itemNo As Long, ByVal strItem As String, ByVal lenItem As
Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function toItemNameMX100 Lib
"DAQMX100" (ByVal itemNo As Integer, ByVal strItem As String,
ByVal lenItem As Integer) As Integer
```

C#

```
[DllImport("DAQMX100.dll" CharSet=CharSet.Auto,
EntryPoint="toItemNameMX100")]
public static extern int toItemNameMX100(int itemNo, byte[]
strItem, int lenItem);
```

Parameters

itemNo	Specify the setup item number.
strItem	Specify the field where the string is to be stored.
lenItem	Specify the byte size of the field where the string is to be stored.

Description

Stores the name string corresponding to the specified setup item number to the specified field.

- The string stored in the field includes the terminator (NULL).
- The return value is the length of the actual string. The return value does not include the terminator.
- Returns 0 if it does not exist.
- The strings that can be stored are, in general, ASCII strings.

Return value

Returns the length of the string.

Reference

CDAQMXItemConfig::toItemName

toItemNoMX100

Syntax

```
int toItemNoMX100(const char * strItem);
```

Declaration

Visual Basic

```
Public Declare Function toItemNoMX100 Lib "DAQMX100" (ByVal  
strItem As String) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function toItemNoMX100 Lib  
"DAQMX100" (ByVal strItem As String) As Integer
```

C#

```
[DllImport("DAQMX100.dll" CharSet=CharSet.Auto,  
EntryPoint="toItemNoMX100")]  
public static extern int toItemNoMX100(byte[] strItem);
```

Parameters

strItem	Gets the name string corresponding to the setup item number.
---------	--

Description

Gets the setup item number corresponding to the specified string.

- It is case-sensitive.
- The specified string is, in general, an ASCII string.
- If it does not exist, "Unknown" is returned.

Return value

Returns the setting item number.

Reference

CDAQMXItemConfig::toItemNo

toModuleSerialMX100

Syntax

```
int toModuleSerialMX100(DAQMX100 daqmx100, int moduleNo, char
* strSerial, int lenSerial);
```

Declaration**Visual Basic**

```
Public Declare Function toModuleSerialMX100 Lib
"DAQMX100"(ByVal daqmx100 As Long, ByVal moduleNo As Long,
ByVal strSerial As String, ByVal lenSerial As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function toModuleSerialMX100 Lib
"DAQMX100"(ByVal daqmx100 As Integer, ByVal moduleNo As
Integer, ByVal strSerial As String, ByVal lenSerial As
Integer) As Integer
```

C#

```
[DllImport("DAQMX100.dll" CharSet=CharSet.Auto,
EntryPoint="toModuleSerialMX100")]
public static extern int toModuleSerialMX100(int daqmx100, int
moduleNo, byte[] strSerial, int lenSerial);
```

Parameters

daqmx100	Specify the device descriptor.
moduleNo	Specify the module number.
strSerial	Specify the field where the string is to be stored.
lenSerial	Specify the byte size of the field where the string is to be stored.

Description

Gets the serial number of the specified module number from the stored current system configuration data.

- Stores the string in the specified storage destination.
- The string stored in the field includes the terminator (NULL).
- The return value is the length of the actual string. The return value does not include the terminator.
- Returns 0 if it does not exist.
- The strings that can be stored are, in general, ASCII strings.

Return value

Returns the length of the string.

Reference

getModuleSerialMX100

toNetHostMX100

Syntax

```
int toNetHostMX100(DAQMX100 daqmx100, char * strHost, int lenHost);
```

Declaration

Visual Basic

```
Public Declare Function toNetHostMX100 Lib "DAQMX100" (ByVal daqmx100 As Long, ByVal strHost As String, ByVal lenHost As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function toNetHostMX100 Lib "DAQMX100" (ByVal daqmx100 As Integer, ByVal strHost As String, ByVal lenHost As Integer) As Integer
```

C#

```
[DllImport("DAQMX100.dll" CharSet=CharSet.Auto, EntryPoint="toNetHostMX100")]  
public static extern int toNetHostMX100(int daqmx100, byte[] strHost, int lenHost);
```

Parameters

daqmx100	Specify the device descriptor.
strHost	Specify the field where the string is to be stored.
lenHost	Specify the byte size of the field where the string is to be stored.

Description

Gets the host name from the stored current network information data.

- Stores the string in the specified storage destination.
- The string stored in the field includes the terminator (NULL).
- The return value is the length of the actual string. The return value does not include the terminator.
- Returns 0 if it does not exist.
- The strings that can be stored are, in general, ASCII strings.

Return value

Returns the length of the string.

Reference

getNetHostMX100

toRealValueMX100

Syntax

```
double toRealValueMX100(int iAOPWMValue, int iRangeAOPWM);
```

Declaration

Visual Basic

```
Public Declare Function toRealValueMX100 Lib "DAQMX100" (ByVal iAOPWMValue As Long, ByVal iRangeAOPWM As Long) As Double
```

Visual Basic.NET

```
Public Declare Ansi Function toRealValueMX100 Lib "DAQMX100" (ByVal iAOPWMValue As Integer, ByVal iRangeAOPWM As Integer) As Double
```

C#

```
[DllImport("DAQMX100.dll" CharSet=CharSet.Auto, EntryPoint="toRealValueMX100")]  
public static extern double toRealValueMX100(int iAOPWMValue, int iRangeAOPWM);
```

Parameters

iAOPWMValue	Specify the output data value.
iRangeAOPWM	Specify the range type.

Description

Converts the output data of AO/PWM data to actual output values according to the specified range type.

- Valid range types are AO and PWM.
- Returns 0.0 if it does not exist.

Return value

Returns the actual output value.

Reference

CDAQMXAOPWMData::toRealValue

toStringValueMX100

Syntax

```
int toStringValueMX100(int dataValue, int point, char *
strValue, int lenValue);
```

Declaration

Visual Basic

```
Public Declare Function toStringValueMX100 Lib
"DAQMX100"(ByVal dataValue As Long, ByVal point As Long, ByVal
strValue As String, ByVal lenValue As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function toStringValueMX100 Lib
"DAQMX100"(ByVal dataValue As Integer, ByVal point As Integer,
ByVal strValue As String, ByVal lenValue As Integer) As
Integer
```

C#

```
[DllImport("DAQMX100.dll" CharSet=CharSet.Auto,
EntryPoint="toStringValueMX100")]
public static extern int toStringValueMX100(int dataValue, int
point, byte[] strValue, int lenValue);
```

Parameters

dataValue	Specify the data value.
point	Specify the decimal point position.
strValue	Specify the field where the string is to be stored.
lenValue	Specify the byte size of the field where the string is to be stored.

Description

Generates the measured value from the specified data value and decimal point position.

- Converts the generated measured value into a string and stores to the specified field.
- The string stored in the field includes the terminator (NULL).
- The return value is the length of the actual string. The return value does not include the terminator.
- The strings that can be stored are, in general, ASCII strings.

Return value

Returns the length of the string.

Reference

CDAQMXDataInfo::toStringValue

toUnitPartNoMX100

Syntax

```
int toUnitPartNoMX100(DAQMX100 daqmx100, char * strPartNo, int lenPartNo);
```

Declaration

Visual Basic

```
Public Declare Function toUnitPartNoMX100 Lib "DAQMX100" (ByVal daqmx100 As Long, ByVal strPartNo As String, ByVal lenPartNo As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function toUnitPartNoMX100 Lib "DAQMX100" (ByVal daqmx100 As Integer, ByVal strPartNo As String, ByVal lenPartNo As Integer) As Integer
```

C#

```
[DllImport("DAQMX100.dll", CharSet=CharSet.Auto, EntryPoint="toUnitPartNoMX100")]  
public static extern int toUnitPartNoMX100(int daqmx100, byte[] strPartNo, int lenPartNo);
```

Parameters

daqmx100	Specify the device descriptor.
strPartNo	Specify the field where the string is to be stored.
lenPartNo	Specify the byte size of the field where the string is to be stored.

Description

Gets the part number from the stored current system configuration data.

- Stores the string in the specified storage destination.
- The string stored in the field includes the terminator (NULL).
- The return value is the length of the actual string. The return value does not include the terminator.
- Returns 0 if it does not exist.
- The strings that can be stored are, in general, ASCII strings.

Return value

Returns the length of the string.

Reference

getUnitPartNoMX100

toUnitSerialMX100

Syntax

```
int toUnitSerialMX100(DAQMX100 daqmx100, char * strSerial, int lenSerial);
```

Declaration

Visual Basic

```
Public Declare Function toUnitSerialMX100 Lib "DAQMX100" (ByVal daqmx100 As Long, ByVal strSerial As String, ByVal lenSerial As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function toUnitSerialMX100 Lib "DAQMX100" (ByVal daqmx100 As Integer, ByVal strSerial As String, ByVal lenSerial As Integer) As Integer
```

C#

```
[DllImport("DAQMX100.dll" CharSet=CharSet.Auto, EntryPoint="toUnitSerialMX100")]  
public static extern int toUnitSerialMX100(int daqmx100, byte[] strSerial, int lenSerial);
```

Parameters

daqmx100	Specify the device descriptor.
strSerial	Specify the field where the string is to be stored.
lenSerial	Specify the byte size of the field where the string is to be stored.

Description

Gets the unit's serial number from the stored current system configuration data.

- Stores the string in the specified storage destination.
- The string stored in the field includes the terminator (NULL).
- The return value is the length of the actual string. The return value does not include the terminator.
- Returns 0 if it does not exist.
- The strings that can be stored are, in general, ASCII strings.

Return value

Returns the length of the string.

Reference

getUnitSerialMX100

unitCFWriteModeMX100

Syntax

```
int unitCFWriteModeMX100(DAQMX100 daqmx100);
```

Declaration

Visual Basic

```
Public Declare Function unitCFWriteModeMX100 Lib  
"DAQMX100"(ByVal daqmx100 As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function unitCFWriteModeMX100 Lib  
"DAQMX100"(ByVal daqmx100 As Integer) As Integer
```

C#

```
[DllImport("DAQMX100.dll" CharSet=CharSet.Auto,  
EntryPoint="unitCFWriteModeMX100")]  
public static extern int unitCFWriteModeMX100(int daqmx100);
```

Parameters

daqmx100 Specify the device descriptor.

Description

Gets the CF write mode from the stored current system configuration data.

- Returns "No overwrite" if it does not exist.

Return value

Returns the CF write mode.

Reference

```
CDAQMX100::getClassMXItemConfig  
CDAQMXItemConfig::getClassMXSysInfo  
CDAQMXSysInfo::getCFWriteMode
```

unitFrequencyMX100

Syntax

```
int unitFrequencyMX100(DAQMX100 daqmx100);
```

Declaration

Visual Basic

```
Public Declare Function unitFrequencyMX100 Lib  
"DAQMX100"(ByVal daqmx100 As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function unitFrequencyMX100 Lib  
"DAQMX100"(ByVal daqmx100 As Integer) As Integer
```

C#

```
[DllImport("DAQMX100.dll" CharSet=CharSet.Auto,  
EntryPoint="unitFrequencyMX100")]  
public static extern int unitFrequencyMX100(int daqmx100);
```

Parameters

daqmx100 Specify the device descriptor.

Description

Gets the power supply frequency from the stored current system configuration data.

- Returns 0 if it does not exist.

Return value

Returns the power supply frequency.

Reference

```
CDAQMX100::getClassMXItemConfig  
CDAQMXItemConfig::getClassMXSysInfo  
CDAQMXSysInfo::getFrequency
```

unitMACMX100

Syntax

```
int unitMACMX100(DAQMX100 daqmx100, int index);
```

Declaration

Visual Basic

```
Public Declare Function unitMACMX100 Lib "DAQMX100" (ByVal daqmx100 As Long, ByVal index As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function unitMACMX100 Lib "DAQMX100" (ByVal daqmx100 As Integer, ByVal index As Integer) As Integer
```

C#

```
[DllImport("DAQMX100.dll" CharSet=CharSet.Auto, EntryPoint="unitMACMX100")]  
public static extern int unitMACMX100(int daqmx100, int index);
```

Parameters

daqmx100	Specify the device descriptor.
index	Specify the byte position.

Description

Gets the byte position of the MAC address from the stored current system configuration data.

- Returns the number of bytes of the specified byte position.
- The byte position is specified with the index value (from 0) of the specified number of "MAC address elements."
- Returns 0 if it does not exist.

Return value

Returns the byte value.

Reference

```
CDAQMX100::getClassMXItemConfig  
CDAQMXItemConfig::getClassMXSysInfo  
CDAQMXSysInfo::getMAC
```

unitNoMX100

Syntax

```
int unitNoMX100(DAQMX100 daqmx100);
```

Declaration

Visual Basic

```
Public Declare Function unitNoMX100 Lib "DAQMX100" (ByVal daqmx100 As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function unitNoMX100 Lib "DAQMX100" (ByVal daqmx100 As Integer) As Integer
```

C#

```
[DllImport("DAQMX100.dll" CharSet=CharSet.Auto, EntryPoint="unitNoMX100")]  
public static extern int unitNoMX100(int daqmx100);
```

Parameters

daqmx100 Specify the device descriptor.

Description

Gets the unit number from the stored current system configuration data.

- Returns 0 if it does not exist.

Return value

Returns the unit number.

Reference

CDAQMX100::getClassMXItemConfig
CDAQMXItemConfig::getClassMXSysInfo
CDAQMXSysInfo::getUnitNo

unitOptionMX100

Syntax

```
int unitOptionMX100(DAQMX100 daqmx100);
```

Declaration

Visual Basic

```
Public Declare Function unitOptionMX100 Lib "DAQMX100" (ByVal daqmx100 As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function unitOptionMX100 Lib "DAQMX100" (ByVal daqmx100 As Integer) As Integer
```

C#

```
[DllImport("DAQMX100.dll" CharSet=CharSet.Auto, EntryPoint="unitOptionMX100")]  
public static extern int unitOptionMX100(int daqmx100);
```

Parameters

daqmx100 Specify the device descriptor.

Description

Gets the option from the stored current system configuration.

- If it does not exist, "No option" is returned.

Return value

Returns the option.

Reference

```
CDAQMX100::getClassMXItemConfig  
CDAQMXItemConfig::getClassMXSysInfo  
CDAQMXSysInfo::getOption
```

unitStyleMX100

Syntax

```
int unitStyleMX100(DAQMX100 daqmx100);
```

Declaration

Visual Basic

```
Public Declare Function unitStyleMX100 Lib "DAQMX100" (ByVal daqmx100 As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function unitStyleMX100 Lib "DAQMX100" (ByVal daqmx100 As Integer) As Integer
```

C#

```
[DllImport("DAQMX100.dll" CharSet=CharSet.Auto, EntryPoint="unitStyleMX100")]  
public static extern int unitStyleMX100(int daqmx100);
```

Parameters

daqmx100 Specify the device descriptor.

Description

Gets the style from the stored current system configuration data.

- Returns 0 if it does not exist.

Return value

Returns the style value.

Reference

CDAQMX100::getClassMXItemConfig
CDAQMXItemConfig::getClassMXSysInfo
CDAQMXSysInfo::getStyle

unitTempMX100

Syntax

```
int unitTempMX100(DAQMX100 daqmx100);
```

Declaration

Visual Basic

```
Public Declare Function unitTempMX100 Lib "DAQMX100" (ByVal daqmx100 As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function unitTempMX100 Lib "DAQMX100" (ByVal daqmx100 As Integer) As Integer
```

C#

```
[DllImport("DAQMX100.dll" CharSet=CharSet.Auto, EntryPoint="unitTempMX100")]  
public static extern int unitTempMX100(int daqmx100);
```

Parameters

daqmx100 Specify the device descriptor.

Description

Gets the temperature unit type from the stored current system configuration data.

- If it does not exist, "°C" is returned.

Return value

Returns the temperature unit type.

Reference

```
CDAQMX100::getClassMXItemConfig  
CDAQMXItemConfig::getClassMXSysInfo  
CDAQMXSysInfo::getTempUnit
```

unitTypeMX100

Syntax

```
int unitTypeMX100(DAQMX100 daqmx100);
```

Declaration

Visual Basic

```
Public Declare Function unitTypeMX100 Lib "DAQMX100" (ByVal daqmx100 As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function unitTypeMX100 Lib "DAQMX100" (ByVal daqmx100 As Integer) As Integer
```

C#

```
[DllImport("DAQMX100.dll" CharSet=CharSet.Auto, EntryPoint="unitTypeMX100")]  
public static extern int unitTypeMX100(int daqmx100);
```

Parameters

daqmx100 Specify the device descriptor.

Description

Gets the unit type from the stored current system configuration data.

- If it does not exist, "Unknown" is returned.

Return value

Returns the unit type.

Reference

CDAQMX100::getClassMXItemConfig
CDAQMXItemConfig::getClassMXSysInfo
CDAQMXSysInfo::getUnitType

userAOPWMValidMX100

Syntax

```
int userAOPWMValidMX100(DAQMX100 daqmx100, int idAOPWM, int aopwmNo);
```

Declaration

Visual Basic

```
Public Declare Function userAOPWMValidMX100 Lib "DAQMX100" (ByVal daqmx100 As Long, ByVal idAOPWM As Long, ByVal aopwmNo As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function userAOPWMValidMX100 Lib "DAQMX100" (ByVal daqmx100 As Integer, ByVal idAOPWM As Integer, ByVal aopwmNo As Integer) As Integer
```

C#

```
[DllImport("DAQMX100.dll" CharSet=CharSet.Auto, EntryPoint="userAOPWMValidMX100")]  
public static extern int userAOPWMValidMX100(int daqmx100, int idAOPWM, int aopwmNo);
```

Parameters

daqmx100	Specify the device descriptor.
idAOPWM	Specify the AO/PWM data identifier.
aopwmNo	Specify the AO/PWM data number.

Description

Gets valid/invalid for the specified AO/PWM data number from the AO/PWM data of the specified AO/PWM data identifier as a Boolean value.

- If it does not exist, Invalid is returned.

Return value

Returns a Boolean value.

Reference

```
CDAQMX100::getClassMXAOPWMList  
CDAQMXAOPWMData::getAOPWMValid  
CDAQMXAOPWMList::getClassMXAOPWMData
```

userAOPWMValueMX100

Syntax

```
int userAOPWMValueMX100(DAQMX100 daqmx100, int idAOPWM, int
aopwmNo);
```

Declaration**Visual Basic**

```
Public Declare Function userAOPWMValueMX100 Lib
"DAQMX100"(ByVal daqmx100 As Long, ByVal idAOPWM As Long,
ByVal aopwmNo As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function userAOPWMValueMX100 Lib
"DAQMX100"(ByVal daqmx100 As Integer, ByVal idAOPWM As
Integer, ByVal aopwmNo As Integer) As Integer
```

C#

```
[DllImport("DAQMX100.dll" CharSet=CharSet.Auto,
EntryPoint="userAOPWMValueMX100")]
public static extern int userAOPWMValueMX100(int daqmx100, int
idAOPWM, int aopwmNo);
```

Parameters

daqmx100	Specify the device descriptor.
idAOPWM	Specify the AO/PWM data identifier.
aopwmNo	Specify the AO/PWM data number.

Description

Gets the output data value of the specified AO/PWM data number from the AO/PWM data of the specified AO/PWM data identifier.

- Returns 0 if it does not exist.

Return value

Returns the output data value.

Reference

```
CDAQMX100::getClassMXAOPWMList
CDAQMXAOPWMData::getAOPWMValue
CDAQMXAOPWMList::getClassMXAOPWMData
```

userBalanceValidMX100

Syntax

```
int userBalanceValidMX100(DAQMX100 daqmx100, int idBalance,  
int balanceNo);
```

Declaration

Visual Basic

```
Public Declare Function userBalanceValidMX100 Lib  
"DAQMX100"(ByVal daqmx100 As Long, ByVal idBalance As Long,  
ByVal balanceNo As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function userBalanceValidMX100 Lib  
"DAQMX100"(ByVal daqmx100 As Integer, ByVal idBalance As  
Integer, ByVal balanceNo As Integer) As Integer
```

C#

```
[DllImport("DAQMX100.dll" CharSet=CharSet.Auto,  
EntryPoint="userBalanceValidMX100")]  
public static extern int userBalanceValidMX100(int daqmx100,  
int idBalance, int balanceNo);
```

Parameters

daqmx100	Specify the device descriptor.
idBalance	Specify the initial balance data identifier.
balanceNo	Specify the initial balance data number.

Description

Gets valid/invalid of the specified initial balance data number from the initial balance data of the specified initial balance data identifier as a Boolean value.

- If it does not exist, Invalid is returned.

Return value

Returns a Boolean value.

Reference

```
CDAQMX100::getClassMXBalanceList  
CDAQMXBalanceData::getBalanceValid  
CDAQMXBalanceList::getClassMXBalanceData
```

userBalanceValueMX100

Syntax

```
int userBalanceValueMX100(DAQMX100 daqmx100, int idBalance,
int balanceNo);
```

Declaration**Visual Basic**

```
Public Declare Function userBalanceValueMX100 Lib
"DAQMX100"(ByVal daqmx100 As Long, ByVal idBalance As Long,
ByVal balanceNo As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function userBalanceValueMX100 Lib
"DAQMX100"(ByVal daqmx100 As Integer, ByVal idBalance As
Integer, ByVal balanceNo As Integer) As Integer
```

C#

```
[DllImport("DAQMX100.dll" CharSet=CharSet.Auto,
EntryPoint="userBalanceValueMX100")]
public static extern int userBalanceValueMX100(int daqmx100,
int idBalance, int balanceNo);
```

Parameters

daqmx100	Specify the device descriptor.
idBalance	Specify the initial balance data identifier.
balanceNo	Specify the initial balance data number.

Description

Gets initial balance value of the specified initial balance data number from the initial balance data of the specified initial balance data identifier.

- Returns 0 if it does not exist.

Return value

Returns the initial balance value.

Reference

```
CDAQMX100::getClassMXBalanceList
CDAQMXBalanceData::getBalanceValue
CDAQMXBalanceList::getClassMXBalanceData
```

userDoubleAOPWMValueMX100

Syntax

```
double userDoubleAOPWMValueMX100(DAQMX100 daqmx100, int  
idAOPWM, int aopwmNo);
```

Declaration

Visual Basic

```
Public Declare Function userDoubleAOPWMValueMX100 Lib  
"DAQMX100"(ByVal daqmx100 As Long, ByVal idAOPWM As Long,  
ByVal aopwmNo As Long) As Double
```

Visual Basic.NET

```
Public Declare Ansi Function userDoubleAOPWMValueMX100 Lib  
"DAQMX100"(ByVal daqmx100 As Integer, ByVal idAOPWM As  
Integer, ByVal aopwmNo As Integer) As Double
```

C#

```
[DllImport("DAQMX100.dll" CharSet=CharSet.Auto,  
EntryPoint="userDoubleAOPWMValueMX100")]  
public static extern double userDoubleAOPWMValueMX100(int  
daqmx100, int idAOPWM, int aopwmNo);
```

Parameters

daqmx100	Specify the device descriptor.
idAOPWM	Specify the AO/PWM data identifier.
aopwmNo	Specify the AO/PWM data number.

Description

Retrieves the output data value of the specified AO/PWM data number from the AO/PWM data of the specified AO/PWM data identifier as the actual output value.

- Returns 0.0 if it does not exist.

Return value

Returns the actual output value.

Reference

CDAQMX100::userDoubleAOPWMValue

userDOValidMX100

Syntax

```
int userDOValidMX100(DAQMX100a daqmx100, int idDO, int doNo);
```

Declaration

Visual Basic

```
Public Declare Function userDOValidMX100 Lib "DAQMX100" (ByVal daqmx100 As Long, ByVal idDO As Long, ByVal doNo As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function userDOValidMX100 Lib "DAQMX100" (ByVal daqmx100 As Integer, ByVal idDO As Integer, ByVal doNo As Integer) As Integer
```

C#

```
[DllImport("DAQMX100.dll" CharSet=CharSet.Auto, EntryPoint="userDOValidMX100")]  
public static extern int userDOValidMX100(int daqmx100, int idDO, int doNo);
```

Parameters

daqmx100	Specify the device descriptor.
idDO	Specify the DO data identifier.
doNo	Specify the DO data number.

Description

Gets valid/invalid of the specified DO data number from the DO data of the specified DO data identifier as a Boolean value.

- If it does not exist, Invalid is returned.

Return value

Returns a Boolean value.

Reference

```
CDAQMX100::getClassMXDOList  
CDAQMXDOData::getDOValid  
CDAQMXDOList::getClassMXDOData
```

userDOValueMX100

Syntax

```
int userDOValueMX100(DAQMX100 daqmx100, int idDO, int doNo);
```

Declaration

Visual Basic

```
Public Declare Function userDOValueMX100 Lib "DAQMX100" (ByVal daqmx100 As Long, ByVal idDO As Long, ByVal doNo As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function userDOValueMX100 Lib "DAQMX100" (ByVal daqmx100 As Integer, ByVal idDO As Integer, ByVal doNo As Integer) As Integer
```

C#

```
[DllImport("DAQMX100.dll" CharSet=CharSet.Auto, EntryPoint="userDOValueMX100")]  
public static extern int userDOValueMX100(int daqmx100, int idDO, int doNo);
```

Parameters

daqmx100	Specify the device descriptor.
idDO	Specify the DO data identifier.
doNo	Specify the DO data number.

Description

Gets ON/OFF for the specified DO data number from the DO data of the specified DO data identifier as a Boolean value.

- If it does not exist, Invalid is returned.

Return value

Returns a Boolean value.

Reference

```
CDAQMX100::getClassMXDOList  
CDAQMXDOData::getDOONOFF  
CDAQMXDOList::getClassMXDOData
```

userTransmitMX100

Syntax

```
int userTransmitMX100(DAQMX100 daqmx100, int idTrans, int
aopwmNo);
```

Declaration**Visual Basic**

```
Public Declare Function userTransmitMX100 Lib "DAQMX100" (ByVal
daqmx100 As Long, ByVal idTrans As Long, ByVal aopwmNo As
Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function userTransmitMX100 Lib
"DAQMX100" (ByVal daqmx100 As Integer, ByVal idTrans As
Integer, ByVal aopwmNo As Integer) As Integer
```

C#

```
[DllImport("DAQMX100.dll" CharSet=CharSet.Auto,
EntryPoint="userTransmitMX100")]
public static extern int userTransmitMX100(int daqmx100, int
idTrans, int aopwmNo);
```

Parameters

daqmx100	Specify the device descriptor.
idTrans	Specify the transmission output data identifier.
aopwmNo	Specify the AO/PWM data number.

Description

Gets the transmission status of the specified AO/PWM data number from the output transmission output data of the specified transmission output data identifier.

- If it does not exist, "Unspecified (Unknown)" is returned.

Return value

Returns the transmission status.

Reference

```
CDAQMX100::getClassMXTransmitList
CDAQMXTransmit::getTransmit
CDAQMXTransmitList::getClassMXTransmit
```

versionAPIMX100

Syntax

```
const int versionAPIMX100(void);
```

Declaration

Visual Basic

```
Public Declare Function versionAPIMX100 Lib "DAQMX100"() As Long
```

Visual Basic.NET

```
Public Declare Ansi Function versionAPIMX100 Lib "DAQMX100"() As Integer
```

C#

```
[DllImport("DAQMX100.dll" CharSet=CharSet.Auto, EntryPoint="versionAPIMX100")]  
public static extern int versionAPIMX100();
```

Description

Gets the version number of this API.

Return value

Returns the version number.

Reference

CDAQMX100::getVersionAPI

18.1 Overview of the MX100 Constants

This Extended API provides the following types of constants.

The data types below are provided. In Visual C/Visual C++, the constants from chapter 6 are inherited. Also, constants and range type constants have been added for the Extended API. See section 18.2. The constants for Visual Basic and Visual Basic.NET/C# are listed in section 18.2.

Type	Description	Page
Number of items	Number of modules, etc.	6-3, 18-4
Maximum values	Maximum length of the tag string, etc.	6-3, 18-4
Constants	Data numbers for specifying the instantaneous value, etc.	6-4, 18-3, 18-5
Boolean value	Valid (ON) setting or Invalid (OFF) setting	6-4, 18-5
Data status values	Status of the measured data	6-4, 18-5
Alarm types	Upper-limit alarm, etc.	6-5, 18-6
Channel kinds	Universal input, digital input, etc.	6-5, 18-6
Scale types	No scaling or linear scale	6-6, 18-7
Module types	4-CH universal input, etc.	6-6, 18-7
Numbers of channels	4 or 10	6-6, 18-7
Interval types	10 ms to 60000 ms	6-6, 18-7
Filter coefficient	Input filter coefficient	6-7, 18-8
RJC types	Internal RJC or external RJC	6-7, 18-8
Burnout types	Off/Up/Down	6-7, 18-8
Unit types	MX100	6-7, 18-8
Terminal types	Screw terminal or clamp terminal	6-7, 18-8
A/D integral time types	Auto, 50 Hz or 60 Hz	6-7, 18-8
Temperature unit types	Celsius or Fahrenheit	6-8, 18-9
CF write modes	Data write mode to the CF card	6-8, 18-9
CF status types	CF status	6-8, 18-9
Unit status values	Unit status	6-8, 18-9
FIFO status values	FIFO status	6-8, 18-9
Display format values	Display format of the 7-segment LED	6-8, 18-9
Output types	Output range type	6-9, 18-10
Selected values	Output value selection	6-9, 18-10
Transmission statuses	Transmission output status	6-9, 18-10
Initial balance results	Result of execution of initial balancing	6-9, 18-10
Options	Presence/absence of options	6-9, 18-10

18.1 Overview of the MX100 Constants

Type	Description	Page
Range types		
Reference range	See the measurement range of the channels undergoing difference between channels computation for the measurement range of the reference channel.	6-9, 18-11
Skip	Not used	18-3, 18-11
DC voltage range types	20 mV, etc.	6-10, 18-11
TC range types	Type R, etc.	6-10, 18-12
RTD (1 mA) range	Pt100, etc.	6-11, 18-12
RTD (2 mA) range	Pt100, etc.	6-13, 18-14
RTD (other) range	Pt500,Pt1000	6-14, 18-15
Resistance range	20 Ω , 200 Ω , or 2 k Ω (0.25 mA)	6-14, 18-15
Digital input (DI) range	Level or contact input	18-3, 18-11
Digital input (DI) detailed ranges	Contact input of the 4-CH Universal Input Module and others: detailed range	6-14, 18-15
Strain range	2000 μ strain, 20000 μ strain, or 200000 μ strain	6-15, 18-16
AO range	V output or mA output	6-15, 18-16
PWM range	PWM output resolution 1 ms or 10 ms	6-15, 18-16

18.2 MX100 Constants

This section describes the mnemonic and the meaning of the constants. For the details on the MX100 functions, see the relevant user's manual.

Visual C/Visual C++ Constants

In Visual C/Visual C++, the constants from chapter 6 are inherited. The following constants have been added.

Constant

Mnemonic	Description
DAQMX_LIST_ALL	Specifies all data identifiers.
DAQMX_LIST_CURRENT	Specifies the current data when copying.

See also "Constants" in section 6.2.

Range Types

This Extended API defines bits that differentiate between the specially-defined ranges and the existing ranges. Differentiations can be made using logical operations.

Mnemonic	Description
DAQMX_RANGETYPE_DI	Special range type for digital input
DAQMX_RANGETYPE_SKIP	Other special range type

See also "Range Types" in section 6.2.

Digital Input (DI) Range Types

The detailed range of digital input is used for the digital input range. If specification is made regardless of the module range, the following is used.

Mnemonic	Description
DAQMX_RANGE_DI_LEVEL	Less than 2.4 V, Greater than or equal to 2.4 V
DAQMX_RANGE_DI_CONTACT	0:open, 1:close

Skip

You can specify the following definition for the special range setting.

Mnemonic	Description
DAQMX_RANGE_SKIP	SKIP (not used)

Constants for Visual Basic and Visual Basic.NET/C#

This section describes the mnemonics for and meanings of the constants. For the details on the MX100 functions, see the relevant user's manual.

In C#, it is the constant data for the DAQMX100 class. Prefix each constant with CDAQMX100. (Ex.: CDAQMX100.DAQMX_COMMPORT).

Numbers

Mnemonic	Description
DAQMX100_NUMMODULE	The number of modules.
DAQMX100_NUMCHANNEL	The number of channels.
DAQMX100_NUMDO	The number of DO data sets.
DAQMX100_NUMFIFO	The number of FIFOs.
DAQMX100_NUMALARM	The number of alarms.
DAQMX100_NUMSEGMENT	The number of 7-segment LEDs.
DAQMX100_NUMMACADDR	The number of MAC address elements (byte count).
DAQMX100_NUMAOPWM	The number of AO/PWM data.
DAQMX100_NUMBALANCE	Number of initial balance data.
DAQMX100_NUMOUTPUT	Number of output channel data.

Maximum Values

Mnemonic	Description
DAQMX100_MAXHOSTNAMELEN	Maximum length of the host name string.
DAQMX100_MAXUNITLEN	Maximum length of the unit name string.
DAQMX100_MAXTAGLEN	Maximum length of the tag string.
DAQMX100_MAXCOMMENTLEN	Maximum length of the comment string.
DAQMX100_MAXSERIALLEN	Maximum length of the MX100 serial number string.
DAQMX100_MAXPARTNOLEN	Maximum length of the part number (firmware part number) string.
DAQMX100_MAXDECIMALPOINT	Maximum value of the decimal point position.
DAQMX100_MAXDISPTIME	Maximum value of the 7-segment LED display time.
DAQMX100_MAXPULSETIME	Maximum value of the integer multiple of the pulse interval.

The maximum length of the string does not include the terminator (NULL).

Constants

Mnemonic	Description
DAQMX100_INSTANTANEOUS	Data number when specifying the retrieval of the instantaneous value.
DAQMX100_REFCHNO_ALL	Specification of all reference channel numbers.
DAQMX100_LEVELNO_ALL	Specification of all alarm level numbers.
DAQMX100_DONO_ALL	Specification of all DO numbers.
DAQMX100_SEGMENTNO_ALL	Specification of all segment numbers of the 7-segment LED.
DAQMX100_CHNO_ALL	Specification of all channel numbers.
DAQMX100_MODULENO_ALL	Specification of all module numbers.
DAQMX100_FIFONO_ALL	Specification of all FIFO numbers.
DAQMX100_AOPWMNO_ALL	Specification of all AO/PWM data numbers.
DAQMX100_BALANCENO_ALL	Specification of all initial balance numbers.
DAQMX100_OUTPUTNO_ALL	Specification of all output data numbers.
DAQMX100_REFCHNO_NONE	Undefined reference channel numbers.
DAQMX100_LIST_ALL	Specifies all data identifiers.
DAQMX100_LIST_CURRENT	Specifies the current data when copying.

Boolean Value (valid/invalid)

Mnemonic	Description
DAQMX100_VALID_OFF	Invalid (OFF) value.
DAQMX100_VALID_ON	Valid (ON) value.

Data Status Values

Mnemonic	Description
DAQMX100_DATA_UNKNOWN	Unknown.
DAQMX100_DATA_NORMAL	Normal.
DAQMX100_DATA_PLUSOVER	Positive overrange.
DAQMX100_DATA_MINUSOVER	Negative overrange.
DAQMX100_DATA_SKIP	SKIP (not used).
DAQMX100_DATA_ILLEGAL	Illegal data status.
DAQMX100_DATA_NODATA	No data status.
DAQMX100_DATA_LACK	Data dropout status.
DAQMX100_DATA_INVALID	Invalid status.

Alarm Types

◇ indicates a space.

Mnemonic	Description	String
DAQMX100_ALARM_NONE	Alarm OFF	◇◇
DAQMX100_ALARM_UPPER	Upper limit alarm	H◇
DAQMX100_ALARM_LOWER	Lower limit alarm	L◇
DAQMX100_ALARM_UPDIFF	Difference upper limit alarm	dH
DAQMX100_ALARM_LOWDIFF	Difference lower limit alarm	dL

Channel Kinds

Mnemonic	Description
DAQMX100_CHKIND_NONE	Not used
DAQMX100_CHKIND_AI	AI*
DAQMX100_CHKIND_AIDIFF	AI* (difference computation between channels specification)
DAQMX100_CHKIND_AIRJC	AI* (remote RJC channel)
DAQMX100_CHKIND_DI	DI*
DAQMX100_CHKIND_DIDIFF	DI* (difference between channel computation designation)
DAQMX100_CHKIND_DO	DO* (alarm output designation)
DAQMX100_CHKIND_DOCOM	DO* (command DO designation)
DAQMX100_CHKIND_DOFAIL	DO* (system failure output designation)
DAQMX100_CHKIND_DOERR	DO* (system error output designation)
DAQMX100_CHKIND_AO	AO* (transmission output)
DAQMX100_CHKIND_AOCOM	AO* (command AO)
DAQMX100_CHKIND_PWM	PWM* (transmission output)
DAQMX100_CHKIND_PWMCOM	PWM* (command PWM)

- * AI: Analog input, DC voltage input, TC input, etc.
- AO: Analog output, analog output module
- DI: Digital input, digital input module
- DO: Digital output, digital output module
- PWM: Pulse Width Modulation, PWM output module

For input channels, establish the channel types according to the range type in the range settings.

For output channels, set the channel type in the channel settings.

Scale Types

Mnemonic	Description
DAQMX100_SCALE_NONE	No scale
DAQMX100_SCALE_LINEAR	Linear scale

Module Types

Mnemonic	Description
DAQMX100_MODULE_NONE	None
DAQMX100_MODULE_MX110UNVH04	4-CH, High-Speed Universal Input Module
DAQMX100_MODULE_MX110UNVM10	10-CH, Medium-Speed Universal Input Module
DAQMX100_MODULE_MX115D05H10	10-CH, High-Speed Digital Input Module
DAQMX100_MODULE_MX125MKCM10	10-CH, Medium-Speed Digital Output Module
DAQMX100_MODULE_MX110V4RM06	6-CH, Medium-Speed 4-wire RTD Resistance Input Module
DAQMX100_MODULE_MX112NDIM04	4-CH Strain Input Module (NDIS)
DAQMX100_MODULE_MX112B35M04	4-CH Strain Input Module (350 OHM)
DAQMX100_MODULE_MX112B12M04	4-CH Strain Input Module (20 OHM)
DAQMX100_MODULE_MX115D24H10	10-CH, High-Speed Digital Input Module (DC24 V)
DAQMX100_MODULE_MX120VAOM08	8-CH, Medium-Speed Analog Output Module
DAQMX100_MODULE_MX120PWMM08	8-CH, Medium-Speed PWM Output Module

Number of Channels

Mnemonic	Description
DAQMX100_CHNUM_0	0
DAQMX100_CHNUM_4	4
DAQMX100_CHNUM_6	6
DAQMX100_CHNUM_8	8
DAQMX100_CHNUM_10	10

Interval Types

Mnemonic	Description
DAQMX100_INTERVAL_10	10 msec
DAQMX100_INTERVAL_50	50 msec
DAQMX100_INTERVAL_100	100 msec
DAQMX100_INTERVAL_200	200 msec
DAQMX100_INTERVAL_500	500 msec
DAQMX100_INTERVAL_1000	1000 msec
DAQMX100_INTERVAL_2000	2000 msec
DAQMX100_INTERVAL_5000	5000 msec
DAQMX100_INTERVAL_10000	10000 msec
DAQMX100_INTERVAL_20000	20000 msec
DAQMX100_INTERVAL_30000	30000 msec
DAQMX100_INTERVAL_60000	60000msec

Filter Coefficient

Mnemonic	Description
DAQMX100_FILTER_0	Coefficient 0
DAQMX100_FILTER_5	Coefficient 5
DAQMX100_FILTER_10	Coefficient 10
DAQMX100_FILTER_20	Coefficient 20
DAQMX100_FILTER_25	Coefficient 25
DAQMX100_FILTER_40	Coefficient 40
DAQMX100_FILTER_50	Coefficient 50
DAQMX100_FILTER_100	Coefficient 100

RJC Types

Mnemonic	Description
DAQMX100_RJC_INTERNAL	RJC function of the MX100
DAQMX100_RJC_EXTERNAL	External RJC function

Burnout Types

Mnemonic	Description
DAQMX100_BURNOUT_OFF	No burnout detection function
DAQMX100_BURNOUT_UP	Display +OVER when burnout is detected
DAQMX100_BURNOUT_DOWN	Display -OVER when burnout is detected

Unit Type Logic

Can be synthesized using OR computations

Mnemonic	Description
DAQMX100_UNITTYPE_NONE	Unknown
DAQMX100_UNITTYPE_MX100	MX100

Terminal Types

Mnemonic	Description
DAQMX100_TERMINAL_SCREW	Screw terminal
DAQMX100_TERMINAL_CLAMP	Clamp terminal
DAQMX100_TERMINAL_NDIS	NDIS

A/D Integral Time Types

Mnemonic	Description
DAQMX100_INTEGRAL_AUTO	Auto (The MX100 automatically sets 50 or 60 Hz)
DAQMX100_INTEGRAL_50HZ	50 Hz
DAQMX100_INTEGRAL_60HZ	60 Hz

Temperature Unit Types

Mnemonic	Description
DAQMX100_TEMPUNIT_C	°C
DAQMX100_TEMPUNIT_F	°F

CF Write Modes

Mnemonic	Description
DAQMX100_CFWRITEMODE_ONCE	No overwriting (stops writing when there is no more free space)
DAQMX100_CFWRITEMODE_FIFO	Repeat (overwrite from the oldest data)

CF Status Types

Can be synthesized using logical OR computations

Mnemonic	Description
DAQMX100_CFSTATUS_NONE	All OFF
DAQMX100_CFSTATUS_EXIST	Presence or absence
DAQMX100_CFSTATUS_USE	CF card is usable
DAQMX100_CFSTATUS_FORMAT	CF card is being formatted

Unit Status Values

Mnemonic	Description
DAQMX100_UNITSTAT_NONE	Unknown
DAQMX100_UNITSTAT_INIT	Initializing
DAQMX100_UNITSTAT_STOP	Stopped
DAQMX100_UNITSTAT_RUN	Measuring
DAQMX100_UNITSTAT_BACKUP	Measuring (backing up)

FIFO Status Values

Mnemonic	Description
DAQMX100_FIFOSTAT_NONE	Unknown
DAQMX100_FIFOSTAT_INIT	Initializing
DAQMX100_FIFOSTAT_STOP	Stopped
DAQMX100_FIFOSTAT_RUN	Measuring
DAQMX100_FIFOSTAT_BACKUP	Measuring (backing up)

Display Format Values

Mnemonic	Description
DAQMX100_DISPTYPE_NONE	Undefined
DAQMX100_DISPTYPE_ON	ON
DAQMX100_DISPTYPE_BLINK	Blinking

Output Types

Mnemonic	Description	Setting range
DAQMX100_OUTPUT_NONE	No output	
DAQMX100_OUTPUT_AO_10V	V output	-11.000 to 11.000 V
DAQMX100_OUTPUT_AO_20MA	mA output	0 to 22.000 mA
DAQMX100_OUTPUT_PWM_1MS	PWM output resolution 1 ms	0 to 100.000 %
DAQMX100_OUTPUT_PWM_10MS	PWM output resolution 10 ms	0 to 100.000 %

Corresponds to the output range type.

Selected Values

Mnemonic	Description
DAQMX100_CHOICE_PREV	Previous value
DAQMX100_CHOICE_PRESET	Specified value

Transmission Statuses

Mnemonic	Description
DAQMX100_TRANSMIT_NONE	No specification (unknown)
DAQMX100_TRANSMIT_RUN	Output start (outputting)
DAQMX100_TRANSMIT_STOP	Output stop

Initial Balance Results

Mnemonic	Description
DAQMX100_BALANCE_NONE	No specification
DAQMX100_BALANCE_DONE	Concluded successfully
DAQMX100_BALANCE_NG	Out of range
DAQMX100_BALANCE_ERROR	Error

Options

Mnemonic	Description
DAQMX100_OPTION_NONE	No option
DAQMX100_OPTION_DS	Dual Save (/DS option)

Range Types

This Extended API defines bits that differentiate between the specially-defined ranges and the existing ranges. Differentiations can be made using logical operations.

Mnemonic	Description
DAQMX100_RANGETYPE_DI	Special range type for digital input
DAQMX100_RANGETYPE_SKIP	Other special range type

Reference Ranges

If this constant is specified as the measurement range of the difference computation channel, the measurement range of the difference computation channel is set to the same range as the measurement range of the reference channel.

The reference range is used for specification when you want to set the same range as the reference channels to be referenced in difference computation and other calculations.

Mnemonic	Description
DAQMX100_RANGE_REFERENCE	Measurement range of the reference channel.

Digital Input (DI) Range Types

The detailed range of digital input is used for the digital input range. If specification is made regardless of the module range, the following is used.

Mnemonic	Description	Setting range
DAQMX_RANGE_DI_LEVEL	Level	0: Less than 2.4 V, 1: Greater than or equal to 2.4 V
DAQMX_RANGE_DI_CONTACT	Contact input	0: open, 1: close

Skip

You can specify the following definition for the special range setting.

Mnemonic	Description
DAQMX100_RANGE_SKIP	SKIP (not used)

DC Voltage Range Types

Mnemonic	Description	Setting range
DAQMX100_RANGE_VOLT_20MV	20 mV	-20.000 to 20.000 mV
DAQMX100_RANGE_VOLT_60MV	60 mV	-60.00 to 60.00 mV
DAQMX100_RANGE_VOLT_200MV	200 mV	-200.00 to 200.00 mV
DAQMX100_RANGE_VOLT_2V	2 V	-2.0000 to 2.0000 V
DAQMX100_RANGE_VOLT_6V	6 V	-6.000 to 6.000 V
DAQMX100_RANGE_VOLT_20V	20 V	-20.000 to 20.000 V
DAQMX100_RANGE_VOLT_100V	100 V	-100.00 to 100.00 V
DAQMX100_RANGE_VOLT_60MVH	60mV:High resolution	0.000 to 60.000 mV
DAQMX100_RANGE_VOLT_1V	1V	-10000 to 1.0000 V
DAQMX100_RANGE_VOLT_6VH	6V:High resolution	0.0000 to 6.0000 V

TC Ranges

Mnemonic	Description	Setting range	
DAQMX100_RANGE_TC_R	R	0.0 to 1760.0°C	32 to 3200°F
DAQMX100_RANGE_TC_S	S	0.0 to 1760.0°C	32 to 3200°F
DAQMX100_RANGE_TC_B	B	0.0 to 1820.0°C	32 to 3308°F
DAQMX100_RANGE_TC_K	K	-200.0 to 1370.0°C	-328 to 2498°F
DAQMX100_RANGE_TC_E	E	-200.0 to 800.0°C	-328.0 to 1472.0°F
DAQMX100_RANGE_TC_J	J	-200.0 to 1100.0°C	-328.0 to 2012.0°F
DAQMX100_RANGE_TC_T	T	-200.0 to 400.0°C	-328.0 to 752.0°F
DAQMX100_RANGE_TC_N	N	0.0 to 1300.0°C	32 to 237°F
DAQMX100_RANGE_TC_W	W	0.0 to 2315.0°C	32 to 4199°F
DAQMX100_RANGE_TC_L	L	-200.0 to 900.0°C	-328.0 to 1652.0°F
DAQMX100_RANGE_TC_U	U	-200.0 to 400.0°C	-328.0 to 752.0°F
DAQMX100_RANGE_TC_KP	KpAu7Fe	0.0 to 300.0K	0.0 to 300.0K
DAQMX100_RANGE_TC_PL	PLATINEL	0.0 to 1400.0°C	32 to 2552°F
DAQMX100_RANGE_TC_PR	PR40-20	0.0 to 1900.0°C	32 to 3452°F
DAQMX100_RANGE_TC_NNM	NiNiMo	0.0 to 1310.0°C	32 to 2390°F
DAQMX100_RANGE_TC_WR	WRe3-25	0.0 to 2400.0°C	32 to 4352°F
DAQMX100_RANGE_TC_WWR	W/WRe26	0.0 to 2400.0°C	32 to 4352°F
DAQMX100_RANGE_TC_AWG	Type-N(AWG14)	0.0 to 1300.0°C	32 to 2372°F

RTD (1 mA) Range Types

Mnemonic	Description	Setting range
DAQMX100_RANGE_RTD_1MAPT	Pt100	-200.0 to 600.0°C -328.0 to 1112.0°F
DAQMX100_RANGE_RTD_1MAJPT	JPt100	-200.0 to 550.0°C -328.0 to 1022.0°F
DAQMX100_RANGE_RTD_1MAPTH	Pt100: high resolution	-140.00 to 150.00°C -220.0 to 302.0°F
DAQMX100_RANGE_RTD_1MAJPTH	JPt100: high resolution	-140.00 to 150.00°C -220.0 to 302.0°F
DAQMX100_RANGE_RTD_1MANIS	Ni100:SAMA	-200.0 to 250.0°C -328.0 to 482.0°F
DAQMX100_RANGE_RTD_1MANID	Ni100:DIN	-60.0 to 180.0°C -76.0 to 356.0°F
DAQMX100_RANGE_RTD_1MANI120	Ni120	-70.0 to 200.0°C -94.0 to 392.0°F
DAQMX100_RANGE_RTD_1MAPT50	Pt50	-200.0 to 550.0°C -328.0 to 1022.0°F
DAQMX100_RANGE_RTD_1MACU10GE	Cu10:GE	-200.0 to 300.0°C -328.0 to 572.0°F

Mnemonic	Description	Setting range
DAQMX100_RANGE_RTD_1MACU10LN	Cu10:L&N	-200.0 to 300.0°C -328.0 to 572.0°F
DAQMX100_RANGE_RTD_1MACU10WEED	Cu10:WEED	-200.0 to 300.0°C -328.0 to 572.0°F
DAQMX100_RANGE_RTD_1MACU10BAILEY	Cu10:BAILEY	-200.0 to 300.0°C -328.0 to 572.0°F
DAQMX100_RANGE_RTD_1MAJ263B	J263*B	0.0 to 300.0K 0.0 to 300.0K
DAQMX100_RANGE_RTD_1MACU10A392	Cu10 at 20°C a=0.00392	-200.0 to 300.0°C -328.0 to 572.0°F
DAQMX100_RANGE_RTD_1MACU10A393	Cu10 at 20°C a=0.00393	-200.0 to 300.0°C -328.0 to 572.0°F
DAQMX100_RANGE_RTD_1MACU25	Cu25 at 0°C a=0.00425	-200.0 to 300.0°C -328.0 to 572.0°F
DAQMX100_RANGE_RTD_1MACU53	Cu53 at 0°C a=0.00426035	-50.0 to 150.0°C 58.0 to 302.0°F
DAQMX100_RANGE_RTD_1MACU100	Cu100 at 0°C a=0.00425	-50.0 to 150.0°C -58.0 to 302.0°F
DAQMX100_RANGE_RTD_1MAPT25	Pt25	-200.0 to 550.0°C -328.0 to 1022.0°F
DAQMX100_RANGE_RTD_1MACU10GEH	Cu10:GE :High resolution	-200.0 to 300.0°C -328.0 to 572.0°F
DAQMX100_RANGE_RTD_1MACU10LNH	Cu10:L&N :High resolution	-500.0 to 500.0°C -328.0 to 572.0°F
DAQMX100_RANGE_RTD_1MACU10WEEDH	Cu10:WEED :High resolution	-500.0 to 500.0°C -328.0 to 572.0°F
DAQMX100_RANGE_RTD_1MACU10BAILEYH	Cu10:BAILEY :High resolution	-500.0 to 500.0°C -328.0 to 572.0°F
DAQMX100_RANGE_RTD_1MAPTN	Pt100 :high noise resistance	-800.0 to 800.0°C -328.0 to 1112.0°F
DAQMX100_RANGE_RTD_1MAJPTN	Jpt100 :high noise resistance	-750.0 to 750.°C -328.0 to 1022.0°F

RTD (2 mA) Ranges

Mnemonic	Description	Setting range
DAQMX100_RANGE_RTD_2MAPT	Pt100	-200.0 to 250.0°C -328.0 to 482.0°F
DAQMX100_RANGE_RTD_2MAJPT	JPt100	-200.0 to 250.0°C -328.0 to 482.0°F
DAQMX100_RANGE_RTD_2MAPTH	Pt100 :High resolution	-140.00 to 150.00°C -220.0 to 302.0°F
DAQMX100_RANGE_RTD_2MAJPTH	JPt100 :High resolution	-140.00 to 150.00°C -220.0 to 302.0°F
DAQMX100_RANGE_RTD_2MAPT50	Pt50	-200.0 to 550.0°C -328.0 to 1022.0°F
DAQMX100_RANGE_RTD_2MACU10GE	CU10:GE	-200.0 to 300.0°C -328.0 to 572.0°F
DAQMX100_RANGE_RTD_2MACU10LN	Cu10:L&N	-200.0 to 300.0°C -328.0 to 572.0°F
DAQMX100_RANGE_RTD_2MACU10WEED	Cu10:WEED	-200.0 to 300.0°C -328.0 to 572.0°F
DAQMX100_RANGE_RTD_2MACU10BAILEY	Cu10:BAILEY	-200.0 to 300.0°C -328.0 to 572.0°F
DAQMX100_RANGE_RTD_2MAJ263B	J263*B	0.0 to 300.0K 0.0 to 300.0K
DAQMX100_RANGE_RTD_2MACU10A392	Cu10 at 20°C a=0.00392	-200.0 to 300.0°C -328.0 to 572.0°F
DAQMX100_RANGE_RTD_2MACU10A393	Cu10 at 20°C a=0.00393	-200.0 to 300.0°C -328.0 to 572.0°F
DAQMX100_RANGE_RTD_2MACU25	Cu25 at 0°C a=0.00425	-200.0 to 300.0°C -328.0 to 572.0°F
DAQMX100_RANGE_RTD_2MACU53	Cu53 at 0°C a=0.00426035	-50.0 to 150.0°C -58.0 to 302.0°F
DAQMX100_RANGE_RTD_2MACU100	Cu100 at 0[DEG]Ca=0.00425	-50.0 to 150.0°C -58.0 to 302.0°F
DAQMX100_RANGE_RTD_2MAPT25	Pt25	-200.0 to 550.0°C -328.0 to 1022.0°F
DAQMX100_RANGE_RTD_2MACU10GEH	CU10:GE :High resolution	-200.0 to 300.0°C -328.0 to 572.0°F
DAQMX100_RANGE_RTD_2MACU10LNH	Cu10:L&N :High resolution	-200.0 to 300.0°C -328.0 to 572.0°F

Mnemonic	Description	Setting range
DAQMX100_RANGE_RTD_2MACU10WEEDH	Cu10:WEED :High resolution	-200.0 to 300.0°C -328.0 to 572.0°F
DAQMX100_RANGE_RTD_2MACU10BAILEYH	Cu10:BAILEY :High resolution	-200.0 to 300.0°C -328.0 to 572.0°F
DAQMX100_RANGE_RTD_2MAPTN	Pt100 :high noise resistance	-200.0 to 250.0°C -328.0 to 482.0°F
DAQMX100_RANGE_RTD_2MAJPTN	Jpt100 :high noise resistance	-200.0 to 250.0°C -328.0 to 482.0°F

RTD (other) Ranges

Mnemonic	Description	Setting range
DAQMX100_RANGE_RTD_025MAPT500	0.25 mA Pt500	-200.0 to 600.0°C -328.0 to 1112.0°F
DAQMX100_RANGE_RTD_025MAPT1K	0.25 mA Pt1000	-200.0 to 600.0°C -328.0 to 1112.0°F

Resistance Ranges

Mnemonic	Description	Setting range
DAQMX100_RANGE_RES_20	20 Ω	0 to 20.000
DAQMX100_RANGE_RES_200	200 Ω	0 to 200.00
DAQMX100_RANGE_RES_2K	2 kΩ (0.25mA)	0 to 2000.0

Digital Input (DI) Detailed Range Types

Mnemonic	Description
DAQMX100_RANGE_DI_LEVEL_AI	DI/Level of the universal input module
DAQMX100_RANGE_DI_CONTACT_AI4	DI/Contact input of the 4-CH, Universal Input Module
DAQMX100_RANGE_DI_CONTACT_AI10	DI/Contact input of the 10-CH, Universal Input Module
DAQMX100_RANGE_DI_LEVEL_DI	DI/Level of the digital input module
DAQMX100_RANGE_DI_CONTACT_DI	DI/Contact input of the digital input module
DAQMX100_RANGE_DI_LEVEL_DI5V [*] DI 5V	DI/contact input of the digital input module (5 V)
DAQMX100_RANGE_DI_LEVEL_DI24V DI 24V	DI/contact input of the digital input module (24 V)

* Separate name for DAQMX100_RANGE_DI_LEVEL_DI. Defined to differentiate from 24 V.

Strain Ranges

Mnemonic	Description	Setting range
DAQMX100_RANGE_STRAIN_2K	2000 μ STR	-2000.0 to 2000.0 \pm 563200000
DAQMX100_RANGE_STRAIN_20K	20000 μ STR	-20000 to 20000 \pm 56320000
DAQMX100_RANGE_STRAIN_200K	200000 μ STR	-200000 to 200000 \pm 5632000

AO Ranges

Mnemonic	Description	Setting range
DAQMX100_RANGE_AO_10V	V output	-10.000 to 10.000 V
DAQMX100_RANGE_AO_20MA	mA output	4.000 to 20.000 mA

PWM Ranges

Mnemonic	Description	Setting range
DAQMX100_RANGE_PWM_1MS	PWM output resolution 1 ms	0 to 100.000 %
DAQMX100_RANGE_PWM_10MS	PWM output resolution 10 ms	0 to 100.000 %

18.3 MX100 Setting Item Numbers

See section 6.3.

18.4 MX100 Types

Detailed Explanation of Types

DAQMX100

This is a type for storing the device descriptor for these functions.
Handled as a Long type in Visual Basic and as int in Visual C. Handled as an Integer type in Visual Basic.NET. Handled as the int type in C#.

Callback

Type	Description
Callback type	Add prefix "DLL" to the function name and write in uppercase. Example: callback type of the openMX100 function: DLLOPENMX100

The callback type is used to link the executable module (.dll) when using Visual C.

19.1 DARWIN Class

The extension API consists of the following additional API classes.

- CDAQHandler
 - CDAQDARWIN
 - ▲ CDAQDA100
 - ◆ CDAQDA100Reader
- ▲ CDAQDARWINDataBuffer

- : Class common to the MX100 and the DARWIN.
- : Class dedicated to the MX100.
- ▲ : Class added for the DA100 (extended API)
- ◆ : Class added for the DA100 for reader (loading instantaneous value data)

CDAQDA100

The extension API handler class.

CDAQDA100Reader

The handler class for loading instantaneous value data.

CDAQDARWINDataBuffer

The channel data class.

Note

Data types and retrieval method

The retrieval of the DARWIN data is handled by a dedicated class. The setup data is not handled by a dedicated class, since the data is retrieved at the line level.

19.2 Correspondence between the Functions and Class/Member Functions - DARWIN -

This section indicates functions and classes that this extension API supports.

Note

This extension API provides a portion of the functions common to the DARWIN series instruments. Model-specific functions, setup functions of the setup mode, and A/D calibration functions are not given. The functions can be added by using the commands of the DARWIN communication function.

The word “command” in the table signifies the command of the DARWIN communication function. For the details on the command, see the Communication Interface User’s Manual.

There are two types of functions, status transition functions and retrieval functions. Status transition functions control DARWIN.

With retrieval functions, the parameter values of the current status are retrieved. When using retrieval functions, the data value of the stored current status is returned (the status of the extension API does not change).

Status Transition Functions

Communication Functions

Function	Command	Class and Member Function
Communication connection	-	CDAQDA100:: open
Comm. cut	-	CDAQDA100:: close
Send in units of lines	-	CDAQDA100:: sendLine
Receive in units of lines	-	CDAQDA100:: receiveLine
Receive in units of bytes	-	CDAQDA100:: receiveByte
Send trigger	(ESC T)	CDAQDA100:: sendTrigger
Update status	(ESC S)	CDAQDA100:: updateStatus
Execute command	-	CDAQDA100:: runCommand

Except during connection or status updates, communication functions do not perform status updates of stored data.

Control Functions

Function	Command	Class and Member Function
Switch operation mode	DS	CDAQDA100:: switchMode
Retrieve code type Switch (binary/ASCII code)	-	CDAQDA100:: switchCode
Reconfigure	RS	CDAQDA100:: reconstruct
Initialize settings	RC	CDAQDA100:: initSetValue
Reset alarms	AR	CDAQDA100:: ackAlarm
Set the date/time	SD	CDAQDA100:: setDateTime
Calculation start/stop	EX	CDAQDA100:: switchCompute
Report start/stop	DR	CDAQDA100:: switchReport
Establish setup mode	XE	CDAQDA100:: establish

Generally, the status is updated at the end of the process.

Status is not updated when establishing the setup mode.

Setting (Operation Mode) Functions

Function	Command	Class and Member Function
Range	SKIP (not used)	CDAQDA100:: setRange
	DC voltage input	CDAQDA100:: setRange
	Thermocouple input	CDAQDA100:: setRange
	RTD input	CDAQDA100:: setRange
	Contact input (DI)	CDAQDA100:: setRange
	DC Current	CDAQDA100:: setRange
	Strain	CDAQDA100:: setRange
	Pulse	CDAQDA100:: setRange
	Power monitor	CDAQDA100:: setRange
	Difference computation between channels	CDAQDA100:: setChDELTA
	Remote RJC	CDAQDA100:: setChRRJC
Scaling unit	SN	CDAQDA100:: setChUnit
Alarm	SA	CDAQDA100:: setChAlarm

Becomes the channel unit setting.

The status is updated after the setting is entered.

Data Retrieval Functions

Function		Command	Class and Member Function
Meas data (inst value)	Meas ch	TS, FM	CDAQDA100:: measInstCh
	Comp ch	TS, FM	CDAQDA100:: mathInstCh
Ch info data	Meas ch	TS, LF	CDAQDA100:: measInfoCh
	Comp ch	TS, LF	CDAQDA100:: mathInfoCh
System configuration data		TS, CF	CDAQDA100:: updateSystemConfig
Report status		TS,RF	CDAQDA100:: updateReportStatus
Setup data			
Declare Op. mode	Sngl spec	TS, LF	CDAQDA100:: talkOperationChData
	Specify rng	TS, LF	CDAQDA100:: talkOperationData
Setup mode	Sngl spec	TS, LF	CDAQDA100:: talkSetupChData
	Specify range	TS, LF	CDAQDA100:: talkSetupData
Cal mode	Sngl spec	TS, LF	CDAQDA100:: talkCalibrationChData
	Specify range	TS, LF	CDAQDA100:: talkCalibrationData
Get data in units of lines		-	CDAQDA100:: getSetDataByLine

Setup data is not stored, so it is retrieved in the same order as mentioned in sections 7.2 and 7.3. In this case, status is not updated.

Channel information data and system configuration data are stored internally, but the user can explicitly perform acquisition.

The report status is stored internally, but is not updated unless the user explicitly acquires it.

Retrieval Functions

Measured Data

Data Name		Class and Member Function
Data value		CDAQDARWINDataInfo:: getValue
Data status values		CDAQDARWINDataInfo:: getStatus
Alarm (presence/absence)		CDAQDARWINDataBuffer:: isDataAlarm
Meas val	Dbl. integer	CDAQDARWINDataInfo:: getDoubleValue
	String	CDAQDARWINDataInfo:: getStringValue
Time	Year	CDAQDARWINDateTime:: getFullYear
	Month	CDAQDARWINDateTime:: getMonth
	Day	CDAQDARWINDateTime:: getDay
	Hour	CDAQDARWINDateTime:: getHour
	Minute	CDAQDARWINDateTime:: getMinute
	Second	CDAQDARWINDateTime:: getSecond
Alarm type		CDAQDARWINDataInfo:: getAlarm

The data is retrieved from CDAQDA100::getClassDataBuffer through CDAQDARWINDataBuffer::getClassDARWINDataInfo and CDAQDARWINDataBuffer::getClassDARWINDateTime.

Channel Information

Data Name		Class and Member Function
Decimal point position		CDAQDARWINChInfo:: getPoint
Channel status		CDAQDARWINChInfo:: getChStatus
Unit name		CDAQDARWINChInfo:: getUnit

The data is retrieved from CDAQDA100::getClassDataBuffer through CDAQDARWINDataBuffer::getClassDARWINChInfo.

System Configuration Data

Data Name		Class and Member Function
Measurement interval		CDAQDARWINSysInfo:: getInterval
Unit	Presence or absence	CDAQDARWINSysInfo:: isExist
Module	Internal code	CDAQDA100:: getModuleCode
	Module name	CDAQDARWINSysInfo:: getModuleName

The data is retrieved from CDAQDA100::getClassSysInfo.

Status Data

Data Name	Class and Member Function
Status byte	CDAQDA100:: getByte
Retrieve code type (binary/ASCII code)	CDAQDA100:: getCode
Report status	CDAQDA100:: getReport

Utilities

Function/Data Name	Class and Member Function
Meas val Chng to double integer	CDAQDARWINDataInfo:: toDoubleValue
Convert into string.	CDAQDARWINDataInfo:: toStringValue
Alarm Alarm type string	CDAQDARWINDataInfo:: getAlarmName
Get the maximum length of the string	CDAQDARWINDataInfo::getMaxLenAlarmName
The version number of this API	CDAQDA100:: getVersionAPI
The revision number of this API	CDAQDA100:: getRevisionAPI
Error Error message string	CDAQDA100:: getErrorMessage
Error message string maximum length	CDAQDA100:: getMaxLenErrorMessage

19.3 Programming - DARWIN/Visual C++ -

Adding the Path to the Include File

Add the path of the include file (DAQDA100.h) to the project. The method of adding the include file varies depending on the environment used.

Declaration in the Source File

Write the declaration in the source file.

```
#include "DAQDA100.h"
```

Note

The include file of the common section (DAQHandler.h, DARWIN.h) and that for DARWIN (DAQDARWIN.h) is referenced from the include file described above. Thus, declaration for it is not necessary.

Library Designation

Adds libraries (DAQDA100.lib, DAQDARWIN.lib, and DAQHandler.lib) to the project. The method of adding the include file varies depending on the environment used. This enables the use of all classes. It also enables the use of all Visual C functions.

Data Retrieval

Program Example

```
////////////////////////////////////  
// DA100 sample for measurement  
#include <stdio.h>  
#include "DAQDA100.h"  
////////////////////////////////////  
int main(int argc, char* argv[])  
{  
    int rc; //return code  
    CDAQDA100 daqda100; //class  
    int value;  
    //connect  
    rc = daqda100.open("192.168.1.11");  
    //get  
    rc = daqda100.measInstCh(0, 1);  
    value = ((daqda100.getClassDataBuffer(0, 1))  
->getClassDARWINDataInfo()).getValue();  
    //disconnect  
    rc = daqda100.close();  
    return rc;  
}  
////////////////////////////////////
```

19.4 Functions and Class Members for Loading Instantaneous Value Data

This section indicates the correspondence between the functions that are supported by the instantaneous value loading functions and the class.

There are two types of functions, status transition functions and retrieval functions. Status transition functions control DARWIN. When using status transition functions, if measured data is retrieved with a data retrieval function, the measured data increments by only interval's worth of data (the status of the extension API changes).

The retrieval function returns the parameter value. When data is retrieved, the data value of the current status is returned (the status of the extension API does not change).

Status Transition Functions

Communication Functions

Function	Command	Class and Member Function
Comm. open	-	CDAQDA100Reader:: open
Comm. cut	-	CDAQDA100Reader:: close

Data Retrieval Functions

Function	Command	Class and Member Function
Meas data Meas channels	EF	CDAQDA100Reader:: measInstCh
(inst val) Comp channels	EF	CDAQDA100Reader:: mathInstCh
Channel information data		
Meas channels	EL	CDAQDA100Reader:: measInfoCh
Comp channels	EL	CDAQDA100Reader:: mathInfoCh

Channel information data is stored internally, but the user can explicitly perform acquisition.

Retrieval Functions

Measured Data

Data Name		Class and Member Function
Data value		CDAQDARWINDataInfo:: getValue
Data status values		CDAQDARWINDataInfo:: getStatus
Alarm (presence/absence)		CDAQDARWINDataBuffer:: isAlarm
Meas val	Double integer	CDAQDARWINDataInfo:: getDoubleValue
	String	CDAQDARWINDataInfo:: getStringValue
Time	Year	CDAQDARWINDateTime:: getFoilYear
	Month	CDAQDARWINDateTime:: getMonth
	Day	CDAQDARWINDateTime:: getDay
	Hour	CDAQDARWINDateTime:: getHour
	Minute	CDAQDARWINDateTime:: getMinute
	Second	CDAQDARWINDateTime:: getSecond
	Milliseconds	CDAQDARWINDateTime:: getMilliSecond
Alarm type		CDAQDARWINDataInfo:: getAlarm

The data is retrieved from CDAQDA100Reader::getClassDataBuffer through CDAQDARWINDataBuffer::getClassDARWINDataInfo and CDAQDARWINDataBuffer::getClassDARWINDateTime.

Channel Information Data

Data Name		Class and Member Function
Decimal point position		CDAQDARWINChInfo:: getPoint
Channel status		CDAQDARWINChInfo:: getChStatus
Unit name		CDAQDARWINChInfo:: getUnit

The data is retrieved from CDAQDA100Reader::getClassDataBuffer through CDAQDARWINDataBuffer::getClassDARWINChInfo.

Utilities

Function/Data Name		Class and Member Function
Measured values	Chng to dbl integer	CDAQDARWINDataInfo:: toDoubleValue
	Convert into string	CDAQDARWINDataInfo:: toStringValue
Alarm	Gets the alarm type string	CDAQDARWINDataInfo:: getAlarmName
	Get max lngth of alarm strg	CDAQDARWINDataInfo:: getMaxLenAlarmName
Get the version number of this API		CDAQDA100Reader:: getVersionAPI
Get the revision number of this API		CDAQDA100Reader:: getRevisionAPI
Error	Get error message string	CDAQDA100Reader:: getErrorMessage
	Get max lngth error message string	CDAQDA100Reader:: getMaxLenErrorMessage

19.5 Program for Loading Instantaneous Value Data - DARWIN/Visual C++ -

Adding the Path to the Include File

Add the path of the include file (DAQDA100Reader.h) to the project. The method of adding the include file varies depending on the environment used.

Declaration in the Source File

Write the declaration in the source file.

```
#include "DAQDA100Reader.h"
```

Note

The include files of the common section and for DARWIN (DAQHandler.h, DAQDARWIN.h, and DAQDA100.h) is referenced from the include file described above. Thus, declaration for it is not necessary.

Library Designation

Adds libraries (DAQDA100.lib, DAQDARWIN.lib, and DAQHandler.lib) to the project. The method of adding the include file varies depending on the environment used. This enables the use of all classes. It also enables the use of all Visual C functions.

Retrieval of the Measured Data

Program Example

```
//////////////////////////////////////////////////////////////////  
// DA100 sample for measurement  
#include <stdio.h>  
#include "DAQDA100.h"  
//////////////////////////////////////////////////////////////////  
int main(int argc, char* argv[])  
{  
    int rc; //return code  
    CDAQDA100 daqda100; //class  
    int value;  
    //connect  
    rc = daqda100.open("192.168.1.11");  
    //get  
    rc = daqda100.measInstCh(0, 1);  
    value = ((daqda100.getClassDataBuffer(0, 1))  
->getClassDARWINDataInfo()).getValue();  
    //disconnect  
    rc = daqda100.close();  
    return rc;  
}  
//////////////////////////////////////////////////////////////////
```


19.6 Details of the DARWIN Class

The classes are listed in alphabetical order by the class name.

CDAQDA100 Class

- CDAQHandler
 - CDAQDARWIN
 - CDAQDA100

This class communicates with DARWIN, and stores the retrieved data.

The following data can be stored.

- Status bytes
- System configuration data
- Channel information data
- Time information data
- Measured data

When each function is executed, the data is updated as necessary. Also, the user can update the data explicitly.

Public Members

Construct/Destruct

- | | |
|------------|-----------------------|
| CDAQDA100 | Constructs an object. |
| ~CDAQDA100 | Destructs an object. |

Communication Functions

- | | |
|--------------|--------------------------|
| updateStatus | Updates the status byte. |
|--------------|--------------------------|

Control Functions

- | | |
|---------------|--------------------------------------|
| switchMode | Switches the configuration mode. |
| switchCode | Switches the retrieve code type. |
| reconstruct | Executes reconfiguration. |
| initSetValue | Initializes operation mode settings. |
| ackAlarm | Resets alarms. |
| switchCompute | Switches computation. |
| switchReport | Switches the report execution type. |

Setup Functions

setRange	Sets the range.
setChDELTA	Sets difference computation between channels.
setChRRJC	Sets remote RJC.
setChUnit	Sets the unit name.
setChAlarm	Sets the alarm value.

Data Retrieval Functions

measInstCh	Gets the measured data of the measurement channel.
mathInstCh	Gets the measured data of the computation channel.
measInfoCh	Gets the channel information data of the measurement channel.
mathInfoCh	Gets the channel information data of the computation channel.
updateSystemConfig	Updates the system configuration data.
updateReportStatus	Updates the report status.
talkOperationChData	Declares the retrieval of the setup data of operation mode on independent channels.
talkSetupChData	Declares the retrieval of the setup data of basic setting mode on independent channels.
talkCalibrationChData	Declares the retrieval of the setup data of calibration mode on independent channels.

Member Data Manipulation

getClassSysInfo	Gets the system configuration data.
getClassDataBuffer	Gets the type information for each channel.
getCode	Gets the retrieve code type.
getByte	Gets the status byte.
getReport	Gets the report status.

- **Overridden Members**

Communication Functions

open	Establishes connection.
------	-------------------------

Data Retrieval Functions

getData	Gets the measured data.
getChannel	Gets the channel information data.

Control Functions

setDateTime	Sets the date/time.
-------------	---------------------

Utilities

isObject Checks an object.

• Inherited Members

See CDAQHandler.

close getErrorMessage getMaxLengthErrorMessage getRevisionAPI
getVersionAPI receiveLine sendLine setTimeout

See CDAQDARWIN

compute establish getChDataByASCII getChDataByBinary getChInfo
getReportStatus getSetDataByLine getStatusByte getSystemConfig
initSystem receiveByte reporting runCommand sendTrigger
setAlarm setDELTA setDI setRRJC setRTD setPOWER setPULSE
setScalingUnit setSKIP setSTRAIN setTC setVOLT
talkCalibrationData talkChInfo talkDataByASCII
talkDataByBinary talkOperationData talkSetupData transMode

Protected Members

Data Members

m_code	Field for storing the retrieve code type.
m_statusByte	Field for storing the status byte.
m_reportStatus	Field for storing the report status.
m_cSysInfo	Field for storing the system configuration data.
m_cMeasData	Field for storing various types of information on measurement channels.
m_cMathData	Field for storing various types of information on computation channels.

Communication Functions

updateAll	Updates all status and information data.
updateRenew	Updates the status.
updateChInfo	Updates all channel information data.

Data Retrieval

getInstChBINARY	Gets the measured data in binary mode.
getInstChASCII	Gets the measured data in ASCII mode.
getInfoCh	Gets the channel information data.

Member Data Manipulation

measClear	Initializes the data member for retrieval of the measured data.
-----------	---

Utilities

chNumMax	Gets the number of channels.
chNumMaxReport	Gets the number of report channels.
getVersionDA100DLL	Gets the version number of the DLL.
getRevisionDA100DLL	Gets the revision number of the DLL.

• Inherited Members

See CDAQHandler.

m_comm m_nRemainSize receive receiveRemain send

See CDAQDARWIN.

checkAck getVersionDLL getRevisionDLL startTalker

Private Members

None.

Member Functions (Alphabetical Order)**CDAQDA100::ackAlarm****Syntax**

```
int ackAlarm(void);
```

Description

Executes the Alarm set system control type.

Updates the status if successful.

Return value

Returns an error number.

Reference

initSystem updateRenew

CDAQDA100::CDAQDA100

Syntax

```
CDAQDA100(void);  
CDAQDA100(const char * strAddress, unsigned int uiPort =  
DAQDARWIN_COMMPORT, nt * errCode = NULL);  
virtual ~CDAQDA100(void);
```

Parameters

strAddress	Specify the IP address as a string.
uiPort	Specify the port number.
errCode	Specify the destination where the error number is to be returned.

Description

Constructs or destructs an object.

When constructing, the data member is initialized. When the parameters are specified, a connection is established during construction. If the return destination is specified, the error number during connection is returned.

When destructing, the data member field is released. The connection is dropped when the communication descriptor exists. The error number is not returned.

Reference

```
measClear open CDAQDARWIN::CDAQDARWIN
```

CDAQDA100::chNumMax

Syntax

```
int chNumMax(int chType);
```

Parameters

chType	Specify the channel type.
--------	---------------------------

Description

Retrieves the maximum number of channels within the specified channel type.

Identifies the model (standalone or expandable) with system configuration data and returns the value.

Return value

Returns the maximum number of channels.

Reference

```
chNumMaxReport getClassSysInfo CDAQDARWINSysInfo::isExist
```

CDAQDA100::chNumMaxReport

Syntax

```
virtual int chNumMaxReport(void);
```

Description

Retrieves the maximum number of channels when the channel type is report.
For the DR130, the value is not correct because identification cannot be made.
Override if necessary.

Return value

Returns the maximum number of channels.

CDAQDA100::getByte

Syntax

```
int getByte(void);
```

Description

Gets the value of the status byte field from the data member.

Return value

Returns the status byte.

CDAQDA100::getChannel

Syntax

```
virtual int getChannel(int chType, int chNo, CDAQChInfo &
    cChInfo);
```

Parameters

chType	Specify the channel type.
chNo	Specify the channel number.
cChInfo	Specify the destination where the channel information data is to be returned.

Description

This function gets the channel information data by channels.
Gets the channel information data of the specified channel.

Return value

Returns an error number.

Reference

```
getClassDataBuffer measInstCh
CDAQDARWINDataBuffer::getClassDARWINChInfo
```

CDAQDA100::getClassDataBuffer

Syntax

```
CDAQDARWINDataBuffer * getClassDataBuffer(int chType, int chNo);
```

Parameters

chType	Specify the channel type.
chNo	Specify the channel number.

Description

Gets various kinds of information data of the specified channel from the data member.

Returns NULL if it does not exist.

Return value

Returns a pointer to the object.

CDAQDA100::getClassSysInfo

Syntax

```
CDAQDARWINSysInfo & getClassSysInfo(void);
```

Description

Returns the object of the system configuration data from the data member.

Return value

Returns a reference to the object.

CDAQDA100::getCode

Syntax

```
int getCode(void);
```

Description

Gets the value of the retrieve code type field from the data member.

Return value

Returns the retrieve code type.

CDAQDA100::getData

Syntax

```
virtual int getData(int chType, int chNo, CDAQDateTime &
    cDateTime, CDAQDataInfo & cDataInfo);
```

Parameters

chType	Specify the channel type.
chNo	Specify the channel number.
cDateTime	Specify the destination where the time information data is to be returned.
cDataInfo	Specify the destination where the measured data is to be returned.

Description

This function gets the instantaneous values in units of channels.
Gets the measured data of the specified channel in binary code.

Return value

Returns an error number.

Reference

```
getClassDataBuffer measInstCh
CDAQDARWINDataBuffer::getClassDARWINDateTime
CDAQDARWINDataBuffer::getClassDARWINDataInfo
```

CDAQDA100::getInfoCh

Syntax

```
virtual int getInfoCh(int sChType, int sChNo, int eChType, int
    eChNo);
```

Parameters

sChType	Specify the start channel type.
sChNo	Specify the start channel number.
eChType	Specify the end channel type.
eChNo	Specify the end channel number.

Description

Gets the channel information data of the specified channel range.
The specified data is stored in the data member.

Return value

Returns an error number.

Reference

```
talkChInfo getChInfo getClassDataBuffer
CDAQDARWINDataBuffer::setChInfo
```


CDAQDA100::getInstChASCII

Syntax

```
int getInstChASCII(int sChType, int sChNo, int eChType, int eChNo);
```

Parameters

sChType	Specify the start channel type.
sChNo	Specify the start channel number.
eChType	Specify the end channel type.
eChNo	Specify the end channel number.

Description

Gets the measured data in ASCII mode.
The specified data is stored in the data member.

Return value

Returns an error number.

Reference

```
getChDataByASCII getClassDataBuffer talkDataByASCII  
CDAQDARWINDataBuffer::setDataInfo  
CDAQDARWINDataBuffer::setDateTime
```

CDAQDA100::getInstChBINARY

Syntax

```
int getInstChBINARY(int sChType, int sChNo, int eChType, int eChNo);
```

Parameters

sChType	Specify the start channel type.
sChNo	Specify the start channel number.
eChType	Specify the end channel type.
eChNo	Specify the end channel number.

Description

Gets the measured data in binary mode.
The specified data is stored in the data member.

Return value

Returns an error number.

Reference

```
getChDataByBinary getClassDataBuffer talkDataByBinary  
CDAQDARWINDataBuffer::setDataInfo  
CDAQDARWINDataBuffer::setDateTime
```

CDAQDA100::getReport

Syntax

```
int getReport(void);
```

Description

Gets the report status field from the data member.

Return value

Returns the report status.

CDAQDA100::getRevisionDA100DLL

Syntax

```
static const int getRevisionDA100DLL(void);
```

Description

Gets the revision number of this DLL.

Return value

Returns the revision number of this DLL.

CDAQDA100::getVersionDA100DLL

Syntax

```
static const int getVersionDA100DLL(void);
```

Description

Gets the version number of this DLL.

Return value

Returns the version number of this DLL.

CDAQDA100::initSetValue

Syntax

```
int initSetValue(void);
```

Description

Executes the "Initialize operation mode settings" system control type.

Updates the status if successful.

Return value

Returns an error number.

Reference

```
initSystem updateAll
```

CDAQDA100::isObject

Syntax

```
virtual int isObject(const char * classname = "CDAQDA100");
```

Parameters

classname Specify the class name using a string.

Description

Checks whether the specified class name was inherited.

If parameters are omitted, checks whether it is this class.

Classes that inherit this class must be overridden in order to check their own classes.

Returns true (valid) if the class was inherited. Otherwise, returns false (invalid).

If different from this class, checks the parent class.

Return value

Returns a Boolean value.

Reference

CDAQDARWIN::isObject

CDAQDA100::mathInfoCh

Syntax

```
virtual int mathInfoCh(int chNo = DAQDA100_CHNO_ALL);
```

Description

Gets the channel information data of the specified computation channel range.

If channel numbers is set to the constant for “specify all channel numbers,” all computation channels are processed.

Return value

Returns an error number.

Reference

measInfoCh

CDAQDA100::mathInstCh

Syntax

```
virtual int mathInstCh(int chNo = DAQDA100_CHNO_ALL);
```

Parameters

chNo Specify the channel number.

Description

Gets the measured data of the specified computation channel.

If the channel number is set to the constant for “specify all channel numbers,” all computation channels are processed.

Return value

Returns an error number.

Reference

measInstCh

CDAQDA100::measClear

Syntax

```
void measClear(void);
```

Description

Initializes the data member for retrieval of the measured data.

Sets various types of channel information to the default values.

Reference

getClassSysInfo

CDAQDARWINChInfo::setChType

CDAQDARWINChInfo::setChNo

CDAQDARWINDataBuffer::initialize

CDAQDARWINDataBuffer::getClassDARWINChInfo

CDAQDARWINSysInfo::initialize

CDAQDA100::measInfoCh

Syntax

```
virtual int measInfoCh(int chType = DAQDA100_CHTYPE_MEASALL,  
int chNo = DAQDA100_CHNO_ALL);
```

Parameters

chType	Specify the channel type.
chNo	Specify the channel number.

Description

Gets the channel information data of the specified channel.

The specified data is stored in the various channel information fields of the data member. If the channel type is set to the constant for “Specify all measurement channel types,” all subunits are processed.

If the channel number is set to the constant for “Specify all channel numbers,” all channels within the channel type are processed.

Updates the status if successful.

Return value

Returns an error number.

Reference

chNumMax getClassSysInfo getInfoCh updateRenew
CDAQDARWINSysInfo::isExist

CDAQDA100::measInstCh

Syntax

```
virtual int measInstCh(int chType = DAQDA100_CHTYPE_MEASALL,
int chNo = DAQDA100_CHNO_ALL);
```

Parameters

chType	Specify the channel type.
chNo	Specify the channel number.

Description

Gets the measured data of the specified channel. The specified data is stored in the various channel information fields of the data member. If the channel type is set to the constant for “Specify all measurement channel types,” all subunits are processed. If the channel number is set to the constant for “Specify all channel numbers,” all channels within the channel type are processed.

Updates the status if successful.

Return value

Returns an error number.

Error

Not support Unsupported retrieve code type.

Reference

```
chNumMax getClassSysInfo getCode getInstChASCII
getInstChBINARY updateRenew CDAQDARWINSysInfo::isExist
```

CDAQDA100::open

Syntax

```
virtual int open(const char * strAddress, unsigned int
uiPort);
```

Parameters

strAddress	Specify the IP address as a string.
uiPort	Specify the port number.

Description

Connects to the device with the IP address and port number specified by the parameters. The port number can be omitted. If omitted, it is set to the communication constant, “DARWIN communication port number.”

When initializing the data member for retrieval of the measured data and connection is successful, those items are retrieved and stored.

The communication timeout is set to 3 minutes.

Return value

Returns an error number.

Reference

```
close measClear setTimeOut updateAll CDAQDARWIN::open
```

CDAQDA100::reconstruct

Syntax

```
int reconstruct(void);
```

Description

Executes the “System reconfiguration” system control type.

Updates the status if successful.

Return value

Returns an error number.

Reference

```
initSystem updateAll
```

CDAQDA100::setChAlarm

Syntax

```
int setChAlarm(int chType, int chNo, int levelNo, int  
iAlarmType = DAQDARWIN_ALARM_NONE, int value = 0, int  
relayType = 0, int relayNo = 0);
```

Parameters

chType	Specify the channel type.
chNo	Specify the channel number.
levelNo	Specify the alarm level.
iAlarmType	Specify the alarm type.
value	Specify the alarm value.
relayType	Specify the relay type.
relayNo	Specify the relay number.

Description

Sets the alarm.

If the channel type is set to the constant for “Specify all measurement channel types,” all subunits are processed.

If the channel number is set to the constant for “Specify all channel numbers,” all channels within the channel type are processed.

If the alarm level is set to the constant for “Specify all alarm level numbers,” all alarm levels within the channels are processed.

Updates the status if successful.

Return value

Returns an error number.

Reference

```
chNumMax getClassSysInfo measInfoCh setAlarm  
CDAQDARWINSysInfo::isExist
```

CDAQDA100::setChDELTA

Syntax

```
int setChDELTA(int chType, int chNo, int refChNo, int spanMin  
= 0, int spanMax = 0);
```

Parameters

chType	Specify the channel type.
chNo	Specify the channel number.
refChNo	Specify the channel number of the reference channel.
spanMin	Specify the left value of the span.
spanMax	Specify the right value of the span.

Description

Sets the difference computation for the specified reference channels.
If the left and right values are the same, the span designation is considered omitted.

If the channel type is set to the constant for “Specify all measurement channel types,” all subunits are processed.

If the channel number is set to the constant for “Specify all channel numbers,” all channels within the channel type are processed.

Updates the status if successful.

Return value

Returns an error number.

Reference

```
chNumMax getClassSysInfo measInfoCh setDELTA  
CDAQDARWINSysInfo::isExist
```

CDAQDA100::setChRRJC

Syntax

```
int setChRRJC(int chType, int chNo, int refChNo, int spanMin = 0, int spanMax = 0);
```

Parameters

chType	Specify the channel type.
chNo	Specify the channel number.
refChNo	Specify the channel number of the reference channel.
spanMin	Specify the left value of the span.
spanMax	Specify the right value of the span.

Description

Sets the remote RJC for the specified reference channel.

If the left and right values are the same, the span designation is considered omitted.

If the channel type is set to the constant for "Specify all measurement channel types," all subunits are processed.

If the channel number is set to the constant for "Specify all channel numbers," all channels within the channel type are processed.

Updates the status if successful.

Return value

Returns an error number.

Reference

```
chNumMax getClassSysInfo measInfoCh setRRJC  
CDAQDARWINSysInfo::isExist
```

CDAQDA100::setChUnit

Syntax

```
int setChUnit(int chType, int chNo, const char * strUnit);
```

Parameters

chType	Specify the channel type.
chNo	Specify the channel number.
strUnit	Specify the unit name using a string.

Description

Sets the specified unit name.

If the channel type is set to the constant for “Specify all measurement channel types,” all subunits are processed.

If the channel number is set to the constant for “Specify all channel numbers,” all channels within the channel type are processed.

Updates the status if successful.

Return value

Returns an error number.

Reference

```
chNumMax getClassSysInfo measInfoCh setScalingUnit
CDAQDARWINSysInfo::isExist
```

CDAQDA100::setDateTime

Syntax

```
virtual int setDateTime(CDAQDARWINDateTime * pcDARWINDateTime
= NULL);
```

Parameters

pcDARWINDateTime	Specify the time information data.
------------------	------------------------------------

Description

Sets time information data on the device.

If NULL is specified, the current date/time of the PC is used.

Updates the status if successful.

Return value

Returns an error number.

Reference

```
updateRenew CDAQDARWIN::setDateTime
```

CDAQDA100::setRange

Syntax

```
int setRange(int chType, int chNo, int iRange, int spanMin =
0, int spanMax = 0, int scaleMin = 0, int scaleMax = 0, int
scalePoint = 0, int bFilter = DAQDARWIN_VALID_OFF, int iItem =
DAQDARWIN_POWERITEM_P1, int iWire = DAQDARWIN_WIRE_1PH2W);
```

Parameters

chType	Specify the channel type.
chNo	Specify the channel number.
iRange	Specify the range type.
spanMin	Specify the left value of the span.
spanMax	Specify the right value of the span.
scaleMin	Specify the left value of the scale.
scaleMax	Specify the right value of the scale.
scalePoint	Specify the decimal point position for scaling.
bFilter	Specify ON/OFF for the filter using a Boolean value.
iItem	Specify the power measurement parameter.
iWire	Specify the power connection method.

Description

Sets the range.

If the left and right values are the same, the span or scale designation is considered omitted. The ON/OFF specification for the filter parameter is only valid for the pulse range. The power measurement item and power connection method parameters are only valid for the power monitoring range. If the channel type is set to the constant for "Specify all measurement channel types," all subunits are processed. If the channel number is set to the constant for "Specify all channel numbers," all channels within the channel type are processed. Updates the status if successful.

Return value

Returns an error number.

Error:

Not support Unsupported range type.

Reference

```
chNumMax getClassSysInfo measInfoCh setDI setMA setPOWER
setPULSE setRTD setSKIP setSTRAIN setTC setVOLT
CDAQDARWINSysInfo::isExist
```

CDAQDA100::switchCode

Syntax

```
int switchCode(int iCode);
```

Parameters

iCode Gets the retrieve code type.

Description

Stores the specified value in the retrieve code type field of the data member.

Updates the status if successful.

Return value

Returns an error number.

Reference

updateRenew

CDAQDA100::switchCompute

Syntax

```
int switchCompute(int iCompute);
```

Parameters

iCompute Specify the computation.

Description

Executes the specified computation.

Updates the status if successful.

Return value

Returns an error number.

Reference

compute updateRenew

CDAQDA100::switchMode

Syntax

```
int switchMode(int iMode);
```

Parameters

iMode Specify the mode.

Description

Switches to the specified mode.

Updates the channel information data when switching to the operation mode.

Updates the status if successful.

Return value

Returns an error number.

Reference

transMode updateChInfo updateRenew

CDAQDA100::switchReport

Syntax

```
int switchReport(int iReportRun);
```

Parameters

iReportRun Specify the report execution type.

Description

Executes the specified report execution type.

Updates the status if successful.

Return value

Returns an error number.

Reference

reporting updateRenew

CDAQDA100::talkCalibrationChData

Syntax

```
int talkCalibrationChData(int chType =  
DAQDA100_CHTYPE_MEASALL, int chNo = DAQDA100_CHNO_ALL);
```

Parameters

chType Specify the channel type.

chNo Specify the channel number.

Description

Executes declaration of the retrieval of the setup data of calibration mode on the specified channels.

If the channel type is set to the constant for "Specify all measurement channel types," all subunits are processed.

If the channel number is set to the constant for "Specify all channel numbers," all channels within the channel type are processed.

Return value

Returns an error number.

Reference

talkCalibrationData

CDAQDA100::talkOperationChData

Syntax

```
int talkOperationChData(int chType = DAQDA100_CHTYPE_MEASALL,  
int chNo = DAQDA100_CHNO_ALL);
```

Parameters

chType	Specify the channel type.
chNo	Specify the channel number.

Description

Executes declaration of the retrieval of the setup data of operation mode on the specified channels.

If the channel type is set to the constant for “Specify all measurement channel types,” all subunits are processed.

If the channel number is set to the constant for “Specify all channel numbers,” all channels within the channel type are processed.

Return value

Returns an error number.

Reference

talkOperationData

CDAQDA100::talkSetupChData

Syntax

```
int talkSetupChData(int chType = DAQDA100_CHTYPE_MEASALL, int  
chNo = DAQDA100_CHNO_ALL);
```

Parameters

chType	Specify the channel type.
chNo	Specify the channel number.

Description

Executes declaration of the retrieval of the setup data of basic setting mode on the specified channels.

If the channel type is set to the constant for “Specify all measurement channel types,” all subunits are processed.

If the channel number is set to the constant for “Specify all channel numbers,” all channels within the channel type are processed.

Return value

Returns an error number.

Reference

talkSetupData

CDAQDA100::updateAll

Syntax

```
int updateAll(void);
```

Description

Updates all information data of the data member.

Gets the System configuration data, channel information data, and status byte and stores it.

Return value

Returns an error number.

Reference

```
updateChInfo  
updateRenew  
updateSystemConfig
```

CDAQDA100::updateChInfo

Syntax

```
int updateChInfo(void);
```

Description

Updates all channel information data.

Return value

Returns an error number.

Reference

```
mathInfoCh  
measInfoCh
```

CDAQDA100::updateRenew

Syntax

```
int updateRenew(void);
```

Description

Updates the data member status.

Gets the status byte and stores it.

Return value

Returns an error number.

Reference

```
updateStatus
```

CDAQDA100::updateReportStatus

Syntax

```
int updateReportStatus(void);
```

Description

Gets the report status.

Stores the retrieved data in the report status field of the data member.

Return value

Returns an error number.

Reference

getReportStatus

CDAQDA100::updateStatus

Syntax

```
int updateStatus(void);
```

Description

Gets the status byte.

Stores the retrieved status byte in the data status field of the data member.

Return value

Returns an error number.

Reference

getStatusByte

CDAQDA100::updateSystemConfig

Syntax

```
int updateSystemConfig(void);
```

Description

Gets the system configuration data.

Stores the retrieved data in the system configuration field of the data member.

Return value

Returns an error number.

Reference

getClassSysInfo

getSystemConfig

CDAQDA100 Reader Class

- CDAQHandler
 - CDAQDARWIN
 - CDAQDA100
 - CDAQDA100Reader

This class communicates with DARWIN to load instantaneous value data and stores the retrieved data.

This class uses instantaneous value data loading commands to override the channel information data and measured data retrieval functions.

The following data can be stored.

- Channel Information Data
- Time Information Data
- Measured Data

Functions other than those for retrieval may not operate properly.

Retrieval of measured data is only supported if alarm data exists.

Public Members

Construct/Destruct

- | | |
|------------------|-----------------------|
| CDAQDA100 Reader | Constructs an object. |
| CDAQDA100Reader | Destructs an object. |

• Overridden Members

Communication Functions

- | | |
|------|-------------------------|
| open | Establishes connection. |
|------|-------------------------|

Data Retrieval Functions

- | | |
|------------|------------------------------------|
| measInstCh | Gets the measured data. |
| measInfoCh | Gets the channel information data. |

Utilities

- | | |
|----------|-------------------|
| isObject | Checks an object. |
|----------|-------------------|

• Inherited Members

See CDAQHandler.

`close` `getErrorMessage` `getMaxLenErrorMessage` `getRevisionAPI`
`getVersionAPI` `receiveLine` `sendLine` `setTimeout`

See CDAQDARWIN

compute establish getChDataByASCII getChDataByBinary getChInfo
 getReportStatus getSetDataByLine getStatusByte getSystemConfig
 initSystem receiveByte reporting runCommand sendTrigger
 setAlarm setDELTA setDI setRRJC setRTD setPOWER setPULSE
 setScallingUnit setSKIP setSTRAIN setTC setVOLT
 talkCalibrationData talkChInfo talkDataByASCII
 talkDataByBinary talkOperationData talkSetupData transMode

See CDAQDA100

ackAlarm getByte getClassDataBuffer getClassSysInfo getCode
 getReport initSetValue mathInstCh mathInfoCh reconstruct
 setChAlarm setChDELTA setChRRJC setChUnit setRange switchCode
 switchCompute switchMode switchReport talkCalibrationChData
 talkOperationChData talkSetupChData updateReportStatus
 updateStatus updateSystemConfig

Protected Members**Data Retrieval**

getInstCh Gets the measured data.

• Overridden Members**Data Retrieval**

getInfoCh Gets the channel information data.

• Inherited Members

See CDAQHandler.

m_comm m_nRemainSize receive receiveRemain send

See CDAQDARWIN

checkAck
 getVersionDLL getRevisionDLL startTalker

See CDAQDA100.

chNumMax chNumMaxReport getInfoCh getInstChASCII
 getInstChBINARY getRevisionDA100DLL
 getVersionDA100DLL m_cMathData m_cMeasData
 m_code m_cSysInfo m_reportStatus m_statusByte measClear
 updateAll updateChInfo updateRenew

Private Members

None.

Member Functions (Alphabetical Order)

CDAQDA100Reader::CDAQDA100Reader

Syntax

```
CDAQDA100Reader(void);  
CDAQDA100Reader(const char * strAddress, unsigned int uiPort =  
DAQDA100READER_DATAPORT, int * errCode = NULL);  
virtual ~CDAQDA100Reader(void);
```

Parameters

strAddress	Specify the IP address as a string.
uiPort	Specify the port number.
errCode	Specify the destination where the error number is to be returned.

Description

Constructs or destructs an object.

When constructing, the data member is initialized. When the parameters are specified, a connection is established during construction. If the return destination is specified, the error number during connection is returned. When destructing, the data member field is released. The connection is dropped when the communication descriptor exists. The error number is not returned.

Reference

open CDAQDA100::CDAQDA100

CDAQDA100Reader::getInfoCh

Syntax

```
virtual int getInfoCh(int sChType, int sChNo, int eChType, int
eChNo);
```

Parameters

sChType	Specify the start channel type.
sChNo	Specify the start channel number.
eChType	Specify the end channel type.
eChNo	Specify the end channel number.

Description

Gets the channel information data of the specified channel range.

The specified data is stored in the data member.

This function executes the EL command of the communication interface.

Return value

Returns an error number.

Reference

```
getChInfo getClassDataBuffer send CDAQDARWINChInfo::toChName
CDAQDARWINDataBuffer::setChInfo
```

CDAQDA100Reader::getInstCh

Syntax

```
int getInstCh(int sChType, int sChNo, int eChType, int eChNo);
```

Parameters

sChType	Specify the start channel type.
sChNo	Specify the start channel number.
eChType	Specify the end channel type.
eChNo	Specify the end channel number.

Description

Gets the measured data of the measured channel range.

The specified data is stored in the data member.

This function executes the EB and EF commands of the communication interface.

Return value

Returns an error number.

Error:

Not data	Received data insufficient.
----------	-----------------------------

Reference

```
getChDataByBinary getClassDataBuffer receive runCommand send
CDAQDARWINChInfo::toChName CDAQDARWINDataBuffer::setDataInfo
CDAQDARWINDataBuffer::setDateTime CDAQDARWINDateTime::setByte
```

CDAQDA100Reader::isObject

Syntax

```
virtual int isObject(const char * classname =  
"CDAQDA100Reader");
```

Parameters

classname Specify the class name using a string.

Description

Checks whether the specified class name was inherited.

If parameters are omitted, checks whether it is this class.

Classes that inherit this class must be overridden in order to check their own classes.

Returns true (valid) if the class was inherited. Otherwise, returns false (invalid).

If different from this class, checks the parent class.

Return value

Returns a Boolean value.

Reference

CDAQDA100::isObject

CDAQDA100Reader::measInfoCh

Syntax

```
virtual int measInfoCh(int chType = DAQDA100_CHTYPE_MEASALL,  
int chNo = DAQDA100_CHNO_ALL);
```

Parameters

chType Specify the channel type.

chNo Specify the channel number.

Description

Gets the channel information data of the specified channel.

The specified data is stored in the various channel information fields of the data member.

If the channel type is set to the constant for "Specify all measurement channel types," all subunits are processed.

If the channel number is set to the constant for "Specify all channel numbers," all channels within the channel type are processed.

Return value

Returns an error number.

Reference

getInfoCh

CDAQDA100Reader::measInstCh

Syntax

```
virtual int measInstCh(int chType = DAQDA100_CHTYPE_MEASALL,
int chNo = DAQDA100_CHNO_ALL);
```

Parameters

chType	Specify the channel type.
chNo	Specify the channel number.

Description

Gets the measured data of the specified channel.

The specified data is stored in the various channel information fields of the data member. If the channel type is set to the constant for “Specify all measurement channel types,” all subunits are processed.

If the channel number is set to the constant for “Specify all channel numbers,” all channels within the channel type are processed.

Return value

Returns an error number.

Reference

getInstCh

CDAQDA100Reader::open

Syntax

```
virtual int open(const char * strAddress, unsigned int uiPort
= DAQDA100READER_DATAPORT);
```

Parameters

strAddress	Specify the IP address as a string.
uiPort	Specify the port number.

Description

Connects to the device with the IP address and port number specified by the parameters.

The port number can be omitted. If omitted, it is set to the communication constant, “DARWIN instantaneous value loading port number.”

When initializing the data member for retrieval of the measured data and connection is successful, channel information data is retrieved and stored.

The communication timeout is set to 3 minutes.

Return value

Returns an error number.

Reference

close measClear setTimeout updateChInfo CDAQDARWIN::open

CDAQDARWINDataBuffer Class

This class stores each type of information for each channel of the DARWIN in a group.

The following data can be stored.

- Channel information data
- Time information data
- Setup data

Public Members

Construct/Destruct

CDAQDARWINDataBuffer	Constructs an object.
~CDAQDARWINDataBuffer	Destructs an object.

Member Data Manipulation

initialize	Initializes the data member.
getClassDARWINChInfo	Gets the channel information data.
getClassDARWINDateTime	Gets the time information data.
getClassDARWINDataInfo	Gets the measured data.
setChInfo	Set the channel information data.
setDateTime	Sets the time information data.
setDataInfo	Sets the measured data.
isAlarm	Gets the presence or absence of alarms.

Protected Members

Data Members

m_cChInfo	Field for storing the channel information data.
m_cTimeBuf	Field for storing the time information data.
m_cDataBuf	Field for storing the measured data.

Private Members

None.

Member Functions (Alphabetical Order)

CDAQDARWINDataBuffer::CDAQDARWINDataBuffer

Syntax

```
CDAQDARWINDataBuffer(void);
virtual ~CDAQDARWINDataBuffer(void);
```

Description

Constructs or destructs an object.
When constructing, the data member is initialized.

Reference

initialize

CDAQDARWINDataBuffer::getClassDARWINChInfo

Syntax

```
CDAQDARWINChInfo & getClassDARWINChInfo(void);
```

Description

Gets the object of the channel information data from the data member.

Return value

Returns a reference to the object.

CDAQDARWINDataBuffer::getClassDARWINDataInfo

Syntax

```
CDAQDARWINDataInfo & getClassDARWINDataInfo(void);
```

Description

Returns the object of the measured data from the data member.

Return value

Returns a reference to the object.

CDAQDARWINDataBuffer::getClassDARWINDateTime

Syntax

```
CDAQDARWINDateTime & getClassDARWINDateTime(void);
```

Description

Returns the object of the time information data from the data member.

Return value

Returns a reference to the object.

CDAQDARWINDataBuffer::initialize

Syntax

```
virtual void initialize(void);
```

Description

Initializes the data member.

The default value as a general rule is 0.

Sets the association with the channel information data of the measured data.

Reference

```
getClassDARWINChInfo getClassDARWINDataInfo  
getClassDARWINDateTime CDAQDARWINChInfo::initialize  
CDAQDARWINDataInfo::initialize  
CDAQDARWINDataInfo::setClassDARWINChInfo  
CDAQDARWINDateTime::initialize
```

CDAQDARWINDataBuffer::isAlarm

Syntax

```
int isAlarm(int levelNo);
```

Parameters

levelNo Specify the alarm level.

Description

Gets the presence/absence of alarms of the specified alarm level.

Returns "Invalid" if the alarm type is "No alarm."

Return value

Returns a Boolean value.

Reference

```
getClassDARWINDataInfo  
CDAQDARWINDataInfo::getAlarm
```

CDAQDARWINDataBuffer::setChInfo

Syntax

```
void setChInfo(CDAQDARWINChInfo & cDARWINChInfo);
```

Parameters

cDARWINChInfo Specify the channel information data.

Description

Copies the specified data to the data member.

CDAQDARWINDataBuffer::setDataInfo

Syntax

```
void setDataInfo(CDAQDARWINDataInfo & cDARWINDataInfo);
```

Parameters

cDARWINDataInfo Specify the measured data.

Description

Copies the specified data to the data member.

The association with the channel information data becomes the data member of this class.

Reference

```
getClassDARWINChInfo    getClassDARWINDataInfo  
CDAQDARWINDataInfo::setClassDARWINChInfo
```

CDAQDARWINDataBuffer::setDateTime

Syntax

```
void setDateTime(CDAQDARWINDateTime & cDARWINDateTime);
```

Parameters

cDARWINDateTime Specify the time information data.

Description

Copies the specified data to the data member.

20.1 Functions and Their Functionalities - DARWIN/ Visual C -

This section indicates the correspondence between the functionalities that the extension API supports and the Visual C functions.

Note

This extension API provides a portion of the functions common to the DARWIN series instruments. Model-specific functions, setup functions of the setup mode, and A/D calibration functions are not given. The functions can be added by using the commands of the DARWIN communication function.

The word “command” in the table signifies the command of the DARWIN communication function. For the details on the commands, see the Communication Interface User’s Manual.

There are two types of functions, status transition functions and retrieval functions. Status transition functions control DARWIN.

With retrieval functions, the parameter values of the current status are retrieved. When using retrieval functions, the data value of the stored current status is returned (the status of the extension API does not change).

Status Transition Functions

Communication Functions

Function	Command	Function
Connect to the DARWIN.	-	openDA100
Disconnect from the DARWIN.	-	closeDA100
Send data by line. Used when controlling the data reception in a special way.	-	sendLineDA100
Receive data by line. Used when controlling the data reception in a special way.	-	receiveLineDA100
Receives data by bytes. Used when controlling the data reception in a special way.	-	receiveByteDA100
Send the command and receive the response. Used when implementing function commands.	-	runCommandDA100
Get the status byte. Sends the status byte output command and receives the response.	(ESC S)	updateStatusDA100
Send a trigger command (ESC T), and receive the response. Used when implementing a new talker function.	(ESC T)	sendTriggerDA100

Except during connection or status updates, communication functions do not perform status updates of stored data.

Control Functions

Function	Command	Function
Switch operation mode	DS	switchModeDA100
Retrieve code type switch (binary/ASCII code)	-	switchCodeDA100
Reconfigure	RS	reconstructDA100
Initialize settings	RC	initSetValueDA100
Reset alarms	AR	ackAlarmDA100
Set date/time (current time)	SD	setDateTimeNowDA100
Calculation start/stop	EX	switchComputeDA100
Report start/stop	DR	switchReportDA100
Establish setup mode	XE	establishDA100

Generally, the status is updated at the end of the process.
Status is not updated when establishing the setup mode.

Setting (Operation Mode) Functions

Function	Command	Function
Range	SKIP (not used)	setRangeDA100
	DC voltage input	setRangeDA100
	Thermocouple input	SetRangeDA100
	RTD input	SetRangeDA100
	Contact input (DI)	SetRangeDA100
	DC current	SetRangeDA100
	Strain	SetRangeDA100
	Pulse	SetRangeDA100
	Power monitor	SetRangeDA100
	Difference computation between channels	setChDELTA100
	Remote RJC	setChRRJCDA100
Unit name	SN	setChUnitDA100
Alarm	SA	setChAlarmDA100

Becomes the channel unit setting.
The status is updated after the setting is entered.

Data Retrieval Functions

Function		Command	Function
Measured data	Meas ch	TS, FM	measInstChDA100
(instantaneous value)	Comp ch	TS, FM	mathInstChDA100
Channel	Meas ch	TS, LF	measInfoChDA100
information data	Comp ch	TS, LF	mathInfoChDA100
System configuration data		TS, CF	updateSystemConfigDA100
Report status		TS,RF	updateReportStatusDA100
Setup data			
Declar. Op mode	Sngl spec	TS, LF	talkOperationChDataDA100
	Specify range	TS, LF	talkOperationDataDA100
Setup mode	Sngl spec	TS, LF	talkSetupChDataDA100
	Specify range	TS, LF	talkSetupDataDA100
Cal mode	Sngl spec	TS, LF	talkCalibrationChDataDA100
	Specify range	TS, LF	talkCalibrationDataDA100
Get data in units of lines		-	getSetDataByLineDA100

Setup data is not stored, so it is retrieved in the same order as mentioned in sections 7.2 and 7.3. In this case, status is not updated.

Channel information data and system configuration data are stored internally, but the user can explicitly perform acquisition.

The report status is stored internally, but is not updated unless the user explicitly acquires it.

Retrieval Functions

Measured Data

Data Name		Function
Data value		dataValueDA100
Data status values		dataStatusDA100
Alarm (presence/absence)		dataAlarmDA100
Measured value	Double integer	dataDoubleValueDA100
	String	dataStringValueDA100
Time	Year	dataYearDA100
	Month	dataMonthDA100
	Day	dataDayDA100
	Hour	dataHourDA100
	Minute	dataMinuteDA100
	Second	dataSecondDA100
Alarm type		alarmTypeDA100

Channel Information

Data Name		Function
Decimal point position		channelPointDA100
Channel status		channelStatusDA100
Unit name		toChannelUnitDA100 getChannelUnitDA100

System Configuration Data

Data Name		Function
Measurement interval		unitIntervalDA100
Unit	Presence or absence	unitValidDA100
Module	Internal code	moduleCodeDA100
	Module name	toModuleNameDA100
		getModuleNameDA100

Status Data

Data Name		Function
Status byte		statusByteDA100
Retrieve code type (binary/ASCII code)		statusCodeDA100
Report status		statusReportDA100

Utilities

Function/Data Name		Function
Meas val	Change to double integer	toDoubleValueDA100
	Convert into string	toStringValueDA100
Alarm	Get the alarm type string	toAlarmNameDA100 getAlarmNameDA100
	Get max length of string	alarmMaxLengthMX100
Get version number of this API		versionAPIDA100
Get revision number of this API		revisionAPIDA100
Error	Get error message string	toErrorMessageDA100 getErrorMessageDA100
	Get error message string maximum length	errorMaxLengthDA100

20.2 Programming - DARWIN/Visual C -

Adding the Path to the Include File

Add the path of the include file (DAQDA100.h) to the project. The method of adding the include file varies depending on the environment used.

Declaration in the Source File

Write the declaration in the source file.

```
#include "DAQDA100.h"
```

Note

The include file of the common section (DAQHandler.h, DAQDARWIN.h) is referenced from the include file described above. Thus, declaration for it is not necessary.

Load Library Statement

The statement below is added so that the executable module (.dll) of the extension API can link to the process.

The executable module (.dll) of the extension API is mapped within the address space (LoadLibrary). Next, the address of the export function in the executable module is retrieved (GetProcAddress).

The callback type of the function pointer is the function name with a prefix "DLL" added and converted to uppercase. It is defined in the include file of the extension API.

```
HMODULE pDll = LoadLibrary("DAQDA100");  
DLLOPENDA100 openDA100 = (DLLOPENDA100)GetProcAddress(pDll,  
"openDA100");
```


Retrieval of the Measured Data

Program Example

```

////////////////////////////////////
// DA100 sample for measurement
#include <stdio.h>
#include "DAQDA100.h"
////////////////////////////////////
int main(int argc, char* argv[])
{
    int rc; //return code
    DAQDA100 comm; //discriptor
    int value;
#ifdef WIN32
    HMODULE pDll; //DLL handle
    //callback
    DLLOPENDA100 openDA100;
    DLLCLOSEDA100 closeDA100;
    DLLMEASINSTCHDA100 measInstChDA100;
    DLLDATAVALUEDA100 dataValueDA100;
    //load
    pDll = LoadLibrary("DAQDA100");
    //get address
    openDA100 = (DLLOPENDA100)GetProcAddress(pDll, "openDA100");
    closeDA100 = (DLLCLOSEDA100)GetProcAddress(pDll,
"closeDA100");
    measInstChDA100 = (DLLMEASINSTCHDA100)GetProcAddress(pDll,
"measInstChDA100");
    dataValueDA100 = (DLLDATAVALUEDA100)GetProcAddress(pDll,
"dataValueDA100");
#endif //WIN32
    //connect
    comm = openDA100("192.168.1.11", &rc);
    //get
    rc = measInstChDA100(comm, 0, 1);
    value = dataValueDA100(comm, 0, 1);
    //disconnect
    rc = closeDA100(comm);
#ifdef WIN32
    FreeLibrary(pDll);
#endif
    return rc;
}
////////////////////////////////////

```

20.3 Correspondence between Functions for Instantaneous Value Data Loading and Functions - DARWIN/Visual C -

This section indicates the correspondence between the functionalities that the extension API supports and the Visual C functions.

Note

This extension API provides a portion of the functions common to the DARWIN series instruments. Model-specific functions, setup functions of the setup mode, and A/D calibration functions are not given. The functions can be added by using the commands of the DARWIN communication function.

The word “command” in the table signifies the command of the DARWIN communication function. For the details on the commands, see the Communication Interface User’s Manual.

There are two types of functions, status transition functions and retrieval functions. Status transition functions control DARWIN.

With retrieval functions, the parameter values of the current status are retrieved.

When using retrieval functions, the data value of the stored current status is returned (the status of the extension API does not change).

Status Transition Functions

Communication Functions

Function	Command	Function
Comm. open	-	openDA100Reader
Comm. cut	-	closeDA100Reader

Data Retrieval Functions

Function	Command	Function	
Meas data (inst val)	Measurement channels Computation channels	EF EF	measInstChDA100Reader mathInstChDA100Reader
Channel info data	Measurement Channel Computation channels	EL EL	measInfoChDA100Reader mathInfoChDA100Reader

Channel information data is stored internally, but the user can explicitly perform acquisition.

Retrieval Functions

Measured Data

Data Name		Function
Data value		dataValueDA100Reader
Data status values		dataStatusDA100Reader
Alarm (presence/absence)		dataAlarmDA100Reader
Measured values	Double integer	dataDoubleValueDA100Reader
	String	dataStringValueDA100Reader
Time	Year	dataYearDA100Reader
	Month	dataMonthDA100Reader
	Day	dataDayDA100Reader
	Hour	dataHourDA100Reader
	Minute	dataMinuteDA100Reader
	Second	dataSecondDA100Reader
	Milliseconds	dataMilliSecDA100Reader
Alarm type		alarmTypeDA100Reader

Channel Information Data

Data Name		Function
Decimal point position		channelPointDA100Reader
Channel status		channelStatusDA100Reader
Unit name		toChannelUnitDA100Reader getChannelUnitDA100Reader

Utilities

Function/Data Name		Function
Meas val	Change to dbl integer	toDoubleValueDA100Reader
	Convert into string	toStringValueDA100Reader
Alarm	Get the alarm type string	toAlarmNameDA100Reader getAlarmNameDA100Reader
	Get max length of the alarm string	alarmMaxLengthDA100Reader
Get the version number of this API		versionAPIDA100Reader
Get the revision number of this API		revisionAPIDA100Reader
Error	Get the error message string	toErrorMessageDA100Reader getErrorMessageDA100Reader
	Get the error message string maximum length	errorMaxLengthDA100Reader

20.4 Program for Loading Instantaneous Value Data - DARWIN/Visual C -

Adding the Path to the Include File

Add the path of the include file (DAQDA100.h) to the project. The method of adding the include file varies depending on the environment used.

Declaration in the Source File

Write the declaration in the source file.

```
#include "DAQDA100Reader.h"
```

Note

The include files of the common section (DAQHandler.h, DAQDARWIN, and DAQDA100) are referenced from the include file described above. Thus, declaration for it is not necessary.

Load Library Statement

The statement below is added so that the executable module (.dll) of the extension API can link to the process.

The executable module (.dll) of the extension API is mapped within the address space (LoadLibrary). Next, the address of the export function in the executable module is retrieved (GetProcAddress).

The callback type of the function pointer is the function name with a prefix "DLL" added and converted to uppercase. It is defined in the include file of the extension API.

```
HMODULE pDll = LoadLibrary("DAQDA100");  
DLLOPENDA100READER openDA100Reader =  
(DLLOPENDA100READER)GetProcAddress(pDll, "openDA100Reader");
```

Retrieval of the Measured Data

Program Example

```

////////////////////////////////////
// DA100Reaer sample for measurement
#include <stdio.h>
#include "DAQDA100Reader.h"
////////////////////////////////////
int main(int argc, char* argv[])
{
    int rc; //return code
    DAQDA100READER comm; //discriptor
    int value;
#ifdef WIN32
    HMODULE pDll; //DLL handle
    //callback
    DLOPENDA100READER openDA100Reader;
    DLLCLOSEDA100READER closeDA100Reader;
    DLLMEASINSTCHDA100READER measInstChDA100Reader;
    DLLDATAVALUEDA100READER dataValueDA100Reader;
    //laod
    pDll = LoadLibrary("DAQDA100");
    //get address
    openDA100Reader = (DLOPENDA100READER)GetProcAddress(pDll,
"openDA100Reader");
    closeDA100Reader = (DLLCLOSEDA100READER)GetProcAddress(pDll,
"closeDA100Reader");
    measInstChDA100Reader =
(DLLMEASINSTCHDA100READER)GetProcAddress(pDll,
"measInstChDA100Reader");
    dataValueDA100Reader =
(DLLDATAVALUEDA100READER)GetProcAddress(pDll,
"dataValueDA100Reader");
#endif //WIN32
    //connect
    comm = openDA100Reader("192.168.1.11" &rc);
    //get
    rc = measInstChDA100Reader(comm, 0, 1);
    value = dataValueDA100Reader(comm, 0, 1);
    //disconnect
    rc = closeDA100Reader(comm);
#ifdef WIN32
    FreeLibrary(pDll);
#endif
    return rc;
}
////////////////////////////////////

```

21.1 Correspondence between the Functions and Class/Member Functions - DARWIN/Visual Basic -

This section indicates the correspondence between the functionalities that the extension API supports and the Visual Basic functions.

Note

This extension API provides a portion of the functions common to the DARWIN series instruments. Model-specific functions, setup functions of the setup mode, and A/D calibration functions are not given. The functions can be added by using the commands of the DARWIN communication function.

The word “command” in the table signifies the command of the DARWIN communication function. For the details on the commands, see the Communication Interface User’s Manual.

There are two types of functions, status transition functions and retrieval functions. Status transition functions control DARWIN.

With retrieval functions, the parameter values of the current status are retrieved.

When using retrieval functions, the data value of the stored current status is returned (the status of the extension API does not change).

Status Transition Functions

Communication Functions

Function	Command	Function
Connect to DARWIN.	-	openDA100
Disconnect from DARWIN.	-	closeDA100
Send data by line. Used when controlling the data reception in a special way.	-	sendLineDA100
Receive data line by line. Used when controlling the data reception in a special way.	-	receiveLineDA100
Receive data by bytes. Used when controlling the data reception in a special way.	-	receiveByteDA100
Send the command and receive the response. Used when implementing function commands.	-	runCommandDA100
Get the status byte. Sends the status byte output command and receives the response.	(ESC S)	updateStatusDA100
Send trigger command (ESC T), and recv response Used when implementing a new talker function.	(ESC T)	sendTriggerDA100

Except during connection or status updates, communication functions do not perform status updates of stored data.

Control Functions

Function	Command	Function
Switch operation mode	DS	switchModeDA100
Retrieve code type switch (binary/ASCII code)	-	switchCodeDA100
Reconfigure	RS	reconstructDA100
Initialize settings	RC	initSetValueDA100
Reset alarms	AR	ackAlarmDA100
Set date/time (current time)	SD	setDateTimeNowDA100
Calculation start/stop	EX	switchComputeDA100
Report start/stop	DR	switchReportDA100
Establish setup mode	XE	establishDA100

Generally, the status is updated at the end of the process.
Status is not updated when establishing the setup mode.

Setting (Operation Mode) Functions

Function	Command	Function	
Range	SKIP (not used)	SR	setRangeDA100
	DC voltage input	SR	setRangeDA100
	Thermocouple input	SR	SetRangeDA100
	RTD input	SR	SetRangeDA100
	Contact input (DI)	SR	SetRangeDA100
	DC current	SR	SetRangeDA100
	Strain	SR	SetRangeDA100
	Pulse	SR	SetRangeDA100
	Power monitor	SR	SetRangeDA100
	Difference computation between channels	SR	setChDELTADA100
	Remote RJC	SR	setChRRJCDA100
Unit name	SN	setChUnitDA100	
Alarm	SA	setChAlarmDA100	

Becomes the channel unit setting.
The status is updated after the setting is entered.

Data Retrieval Functions

Function		Command	Function
Measured data (instantaneous value)	Meas ch	TS, FM	measInstChDA100
	Comp ch	TS, FM	mathInstChDA100
Channel information data	Meas ch	TS, LF	measInfoChDA100
	Comp ch	TS, LF	mathInfoChDA100
System configuration data		TS, CF	updateSystemConfigDA100
Report status		TS, RF	updateReportStatusDA100
Setup data			
Declaration Op e mod	Sngl spec	TS, LF	talkOperationChDataDA100
	Specify range	TS, LF	talkOperationDataDA100
Setup mode	Sngl spec	TS, LF	talkSetupChDataDA100
	Specify range	TS, LF	talkSetupDataDA100
Cal mode	Single spec	TS, LF	talkCalibrationChDataDA100
	Specify range	TS, LF	talkCalibrationDataDA100
Get data in units of lines		-	getSetDataByLineDA100

Setup data is not stored, so it is retrieved in the same order as mentioned in sections 7.2 and 7.3. In this case, status is not updated.

Channel information data and system configuration data are stored internally, but the user can explicitly perform acquisition.

The report status is stored internally, but is not updated unless the user explicitly acquires it.

Retrieval Functions

Measured Data

Data Name		Function
Data value		dataValueDA100
Data status values		dataStatusDA100
Alarm (presence/absence)		dataAlarmDA100
Measured value	Double integer	dataDoubleValueDA100
	String	dataStringValueDA100
Time	Year	dataYearDA100
	Month	dataMonthDA100
	Day	dataDayDA100
	Hour	dataHourDA100
	Minute	dataMinuteDA100
	Second	dataSecondDA100
Alarm type		alarmTypeDA100

Channel Information

Data Name		Function
Decimal point position		channelPointDA100
Channel status		channelStatusDA100
Unit name		toChannelUnitDA100

System Configuration Data

Data Name		Function
Measurement interval		unitIntervalDA100
Unit	Presence or absence	unitValidDA100
Module	Internal code	moduleCodeDA100
	Module name	toModuleNameDA100

Status Data

Data Name		Function
Status byte		statusByteDA100
Retrieve code type (binary/ASCII code)		statusCodeDA100
Report status		statusReportDA100

Utilities

Function/Data Name		Function
Measured	Chng to dble integ	toDoubleValueDA100
value	Convert into string	toStringValueDA100
Alarm	Get the alarm type string	toAlarmNameDA100
	Get max length of the string	alarmMaxLengthDA100
Get the version number of this API		versionAPIDA100
Get the revision number of this API		revisionAPIDA100
Error	Error message string Get	toErrorMessageDA100
	Error message string maximum length	errorMaxLengthDA100

21.2 Programming - DARWIN/Visual Basic -

Declaration of Types, Functions, and Constants

To use types, functions, and constants for Visual Basic, they must be declared in advance. The following methods of declaration statements are available.

Statement of All Declarations

Adding the standard module library file for Visual Basic (DAQDA100.bas) to the project is equivalent to declaring all types, functions, and constants.

Statement of Selective Declarations

The API Viewer that comes with Visual Studio can be used to copy the declaration statements of arbitrary types, functions, and constants. Load the text file for the API Viewer (DAQDA100.txt) on the API Viewer to use this function.

For a description of how to use the API Viewer, read the operation manual for Visual Studio.

Writing Declarations Directly

Below is an example of a declaration statement.

```
Public Declare Function openDA100 Lib "DAQDA100" (ByVal  
strAddress As String, ByVal errorCode As Long) As Long
```

Retrieval of the Measured Data

Program Example

```
Attribute VB_Name = "Module1"
Public Function Main()
    'connect
    comm = openDA100("192.168.1.11", rc)
    'get
    rc = measInstChDA100(comm, 0, 1)
    value = dataValueDA100(comm, 0, 1)
    'disconnect
    rc = closeDA100(comm)
End Function
```

21.3 Correspondence between Functions for Instantaneous Value Data Loading and Member Functions - DARWIN/Visual Basic -

This section indicates the correspondence between the functionalities that the extension API supports and the Visual Basic functions.

Note

This extension API provides a portion of the functions common to the DARWIN series instruments. Model-specific functions, setup functions of the setup mode, and A/D calibration functions are not given. The functions can be added by using the commands of the DARWIN communication function.

The word "command" in the table signifies the command of the DARWIN communication function. For the details on the commands, see the Communication Interface User's Manual.

There are two types of functions, status transition functions and retrieval functions. Status transition functions control DARWIN.

With retrieval functions, the parameter values of the current status are retrieved.

When using retrieval functions, the data value of the stored current status is returned (the status of the extension API does not change).

Status Transition Function

Communication Functions

Function	Command	Function
Comm. open	-	CDAQDA100Reader:: open
Comm. cut	-	CDAQDA100Reader:: close

Data Retrieval Functions

Function	Command	Function
Meas data (inst value)	Measurement channel Computation channels	EF EF
Channel info data	Measurement channel Computation channels	EL EL
		measInstChDA100Reader mathInstChDA100Reader measInfoChDA100Reader mathInfoChDA100Reader

Channel information data is stored internally, but the user can explicitly perform acquisition.

Retrieval Functions

Measured Data

Data Name	Function	
Data value	dataValueDA100Reader	
Data status values	dataStatusDA100Reader	
Alarm (presence/absence)	dataAlarmDA100Reader	
Meas value	Double integer	dataDoubleValueDA100Reader
	String	dataStringValueDA100Reader
Time	Year	dataYearDA100Reader
	Month	dataMonthDA100Reader
	Day	dataDayDA100Reader
	Hour	dataHourDA100Reader
	Minute	dataMinuteDA100Reader
	Second	dataSecondDA100Reader
	Milliseconds	dataMilliSecDA100Reader
Alarm type	alarmTypeDA100Reader	

Channel Information Data

Data Name	Function
Decimal point position	channelPointDA100Reader
Channel status	channelStatusDA100Reader
Unit name	toChannelUnitDA100Reader

Utilities

Function/Data Name	Function	
Measured value	Chng to double integer	toDoubleValueDA100Reader
	Convert into string	toStringValueDA100Reader
Alarm	Get the alarm type string	toAlarmNameDA100Reader
	Get max length of the alarm string	alarmMaxLengthDA100Reader
Get the version number of this API	versionAPIDA100Reader	
Get the revision number of this API	revisionAPIDA100Reader	
Error	Get the error message string.	toErrorMessageDA100Reader
	Get max length of error message string	errorMaxLengthDA100Reader

21.4 Program for Loading Instantaneous Value Data - DARWIN/Visual Basic -

To use types, functions, and constants for Visual Basic, they must be declared in advance. The following methods of declaration statements are available.

Statement of All Declarations

Adding the standard module library file for Visual Basic (DAQDA100Reader.bas) to the project is equivalent to declaring all types, functions, and constants.

Statement of Selective Declarations

The API Viewer that comes with Visual Studio can be used to copy the declaration statements of arbitrary types, functions, and constants. Load the text file for the API Viewer (DAQDA100Reader.txt) on the API Viewer to use this function.

For a description of how to use the API Viewer, read the operation manual for Visual Studio.

Writing Declarations Directly

Below is an example of a declaration statement.

```
Public Declare Function openDA100Reader Lib "DAQDA100" (ByVal  
strAddress As String, ByRef errorCode As Long) As Long
```

Retrieval of the Measured Data

Program Example

```
Attribute VB_Name = "Module1"  
Public Function Main()  
    'connect  
    comm = openDA100Reader("192.168.1.11", rc)  
    'get  
    rc = measInstChDA100Reader(comm, 0, 1)  
    Value = dataValueDA100Reader(comm, 0, 1)  
    'disconnect  
    rc = closeDA100Reader(comm)  
End Function
```


22.1 Functions and Their Functionalities - DARWIN/ Visual Basic.NET -

This section indicates functions and classes that this extension API supports.

Note

This extension API provides a portion of the functions common to the DARWIN series instruments. Model-specific functions, setup functions of the setup mode, and A/D calibration functions are not given. The functions can be added by using the commands of the DARWIN communication function.

The word “command” in the table signifies the command of the DARWIN communication function. For the details on the commands, see the Communication Interface User’s Manual.

There are two types of functions, status transition functions and retrieval functions. Status transition functions control DARWIN.

With retrieval functions, the parameter values of the current status are retrieved.

When using retrieval functions, the data value of the stored current status is returned (the status of the extension API does not change).

Status Transition Functions

Communication Functions

Function	Command	Function
Connect to DARWIN.	-	openDA100
Disconnect from DARWIN.	-	closeDA100
Send data by line. Used when controlling the data reception in a special way.	-	sendLineDA100
Receive data line by line. Used when controlling the data reception in a special way.	-	receiveLineDA100
Receive data by bytes. Used when controlling the data reception in a special way.	-	receiveByteDA100
Send the command and receive the response. Used when implementing function commands.	-	runCommandDA100
Get the status byte. Sends the status byte output command and receives the response.	(ESC S)	updateStatusDA100
Send a trigger command (ESC T), and receive the response. Used when implementing a new talker function.	(ESC T)	sendTriggerDA100

Except during connection or status updates, communication functions do not perform status updates of stored data.

Control Functions

Function	Command	Function
Switch operation mode	DS	switchModeDA100
Retrieve code type switch (binary/ASCII code)	-	switchCodeDA100
Reconfigure	RS	reconstructDA100
Initialize settings	RC	initSetValueDA100
Reset alarms	AR	ackAlarmDA100
Set date/time (current time)	SD	setDateTimeNowDA100
Calculation start/stop	EX	switchComputeDA100
Report start/stop	DR	switchReportDA100
Establish setup mode	XE	establishDA100

Generally, the status is updated at the end of the process.
 Status is not updated when establishing the setup mode.

Setting (Operation Mode) Functions

Function	Command	Function
range	SKIP (not used)	setRangeDA100
	DC voltage input	setRangeDA100
	Thermocouple input	SetRangeDA100
	RTD input	SetRangeDA100
	Contact input (DI)	SetRangeDA100
	DC current	SetRangeDA100
	Strain	SetRangeDA100
	Pulse	SetRangeDA100
	Power monitor	SetRangeDA100
	Difference computation between channels	setChDELTA100
	Remote RJC	setChRRJCDA100
Unit name	SN	setChUnitDA100
Alarm	SA	setChAlarmDA100

Becomes the channel unit setting.
 The status is updated after the setting is entered.

Data Retrieval Functions

Function		Command	Function
Meas data (inst value)	Measurement channel	TS, FM	measInstChDA100
	Computation channels	TS, FM	mathInstChDA100
Channel info data	Measurement channel	TS, LF	measInfoChDA100
	Computation channels	TS, LF	mathInfoChDA100
System configuration data		TS, CF	updateSystemConfigDA100
Report status		TS,RF	updateReportStatusDA100
Setup data			
Declare Op mode	Sngl spec	TS, LF	talkOperationChDataDA100
	Specify range	TS, LF	talkOperationDataDA100
Setup mode	Sngl spec	TS, LF	talkSetupChDataDA100
	Specify range	TS, LF	talkSetupDataDA100
Cal mode	Sngl spec	TS, LF	talkCalibrationChDataDA100
	Specify range	TS, LF	talkCalibrationDataDA100
Get data in units of lines		-	getSetDataByLineDA100

Setup data is not stored, so it is retrieved in the same order as mentioned in sections 7.2 and 7.3. In this case, status is not updated.

Channel information data and system configuration data are stored internally, but the user can explicitly perform acquisition.

The report status is stored internally, but is not updated unless the user explicitly acquires it.

Retrieval Functions

Measured Data

Data Name		Function
Data value		dataValueDA100
Data status values		dataStatusDA100
Alarm (presence/absence)		dataAlarmDA100
Measured value	Double integer	dataDoubleValueDA100
	String	dataStringValueDA100
Time	Year	dataYearDA100
	Month	dataMonthDA100
	Day	dataDayDA100
	Hour	dataHourDA100
	Minute	dataMinuteDA100
	Second	dataSecondDA100
Alarm type		alarmTypeDA100

Channel Information

Data Name		Function
Decimal point position		channelPointDA100
Channel status		channelStatusDA100
Unit name		toChannelUnitDA100 getChannelUnitDA100

System Configuration Data

Data Name		Function
Measurement interval		unitIntervalDA100
Unit	Presence or absence	unitValidDA100
Module	Internal code	moduleCodeDA100
	Module name	toModuleNameDA100

Status Data

Data Name		Function
Status byte		statusByteDA100
Retrieve code type (binary/ASCII code)		statusCodeDA100
Report status		statusReportDA100

Utilities

Function/Data Name	Function	
Meas val	Change to double integer	toDoubleValueDA100
	Convert into string	toStringValueDA100
Alarm	Get the alarm type string	toAlarmNameDA100 getAlarmNameDA100
	Get max length of string	alarmMaxLengthDA100
Get the version number of this API	versionAPIDA100	
Get the revision number of this API	revisionAPIDA100	
Error	Error message string Get	toErrorMessageDA100 getErrorMessageDA100
	Error message string maximum length	errorMaxLengthDA100

22.2 Programming - DARWIN/Visual Basic.NET -

Statement of Declarations

Adding the module for Visual Basic.NET to the project is equivalent to declaring all functions and constants.

Retrieval of the Measured Data

Program Example

```
Module Module1
    Public Function Meas()
        Dim comm As Integer
        Dim rc As Integer
        Dim value As Integer
        'connect
        comm = openDA100("192.168.1.11", rc)
        'get
        rc = measInstChDA100(comm, 0, 1)
        value = dataValueDA100(comm, 0, 1)
        'disconnect
        rc = closeDA100(comm)
    End Function
End Module
```

22.3 Correspondence between Functions for Instantaneous Value Data Loading and Functions - DARWIN/Visual Basic.NET -

This section indicates the correspondence between the functionalities that the extension API supports and the Visual C functions.

Note

This extension API provides a portion of the functions common to the DARWIN series instruments. Model-specific functions, setup functions of the setup mode, and A/D calibration functions are not given. The functions can be added by using the commands of the DARWIN communication function.

The word “command” in the table signifies the command of the DARWIN communication function. For the details on the commands, see the Communication Interface User’s Manual.

There are two types of functions, status transition functions and retrieval functions. Status transition functions control DARWIN.

With retrieval functions, the parameter values of the current status are retrieved.

When using retrieval functions, the data value of the stored current status is returned (the status of the extension API does not change).

Status Retrieval Function

Communication Functions

Function	Command	Function
Comm. open	-	openDA100Reader
Comm. cut	-	closeDA100Reader

Data Retrieval Functions

Function	Command	Function	
Meas data	Measurement channel	EF	measInstChDA100Reader
(inst value)	Computation channel	EF	mathInstChDA100Reader
Channel	Measurement channel	EL	measInfoChDA100Reader
info data	Computation channels	EL	mathInfoChDA100Reader

Channel information data is stored internally, but the user can explicitly perform acquisition.

Retrieval Functions

Measured Data

Data Name		Function
Data value		dataValueDA100Reader
Data status values		dataStatusDA100Reader
Alarm (presence/absence)		dataAlarmDA100Reader
Measured value	Double integer	dataDoubleValueDA100Reader
	String	dataStringValueDA100Reader
Time	Year	dataYearDA100Reader
	Month	dataMonthDA100Reader
	Day	dataDayDA100Reader
	Hour	dataHourDA100Reader
	Minute	dataMinuteDA100Reader
	Second	dataSecondDA100Reader
	Milliseconds	dataMilliSecDA100Reader
Alarm type		alarmTypeDA100Reader

Channel Information Data

Data Name	Function
Decimal point position	channelPointDA100Reader
Channel status	channelStatusDA100Reader
Unit name	toChannelUnitDA100Reader

Utilities

Function/Data Name	Function	
Measured value	Change to double integer	toDoubleValueDA100Reader
	Convert into string	toStringValueDA100Reader
Alarm	Get the alarm type string	toAlarmNameDA100Reader
	Get max length of the alarm string	alarmMaxLengthDA100Reader
Get the version number of this API		versionAPIDA100Reader
Get the revision number of this API		revisionAPIDA100Reader
Error	Get the error message string	toErrorMessageDA100Reader
	Get max length of error message string	errorMaxLengthDA100Reader

22.4 Program for Loading Instantaneous Value Data -DARWIN/Visual Basic.NET-

Declaration Statements

Adding the module for Visual Basic.NET to the project is equivalent to declaring all functions and constants.

Retrieval of the Measured Data

Program Example

```
Module Module1
    Public Function Meas()
        Dim comm As Integer
        Dim rc As Integer
        Dim value As Integer
        'connect
        comm = openDA100Reader("192.168.1.11", rc)
        'get
        rc = measInstChDA100Reader(comm, 0, 1)
        value = dataValueDA100Reader(comm, 0, 1)
        'disconnect
        rc = closeDA100Reader(comm)
    End Function
End Module
```

23.1 Functions and Their Functionalities - DARWIN/C# -

This section indicates functions that this extension API supports.

Note

This extension API provides a portion of the functions common to the DARWIN series instruments. Model-specific functions, setup functions of the setup mode, and A/D calibration functions are not given. The functions can be added by using the commands of the DARWIN communication function.

The word command in the table signifies the command of the DARWIN communication function. For the details on the commands, see the Communication Interface User's Manual.

There are two types of functions, status transition functions and retrieval functions. Status transition functions control DARWIN.

With retrieval functions, the parameter values of the current status are retrieved.

When using retrieval functions, the data value of the stored current status is returned (the status of the extension API does not change).

Status Transition Functions

Communication Functions

Function	Command	Function
Connect to DARWIN.	-	DAQDA100. openDA100
Disconnect from DARWIN.	-	DAQDA100. closeDA100
Send data by line. Used when controlling the data reception in a special way.	-	DAQDA100. sendLineDA100
Receive data by line. Used when controlling the data reception in a special way.	-	DAQDA100. receiveLineDA100
Receives data by bytes. Used when controlling the data reception in a special way.	-	DAQDA100. receiveByteDA100
Send command and receive the response. Used when implementing function commands.	-	DAQDA100. runCommandDA100
Get the status byte. Sends the status byte output command and receives the response.	(ESC S)	DAQDA100. updateStatusDA100
Send a trigger command (ESC T), and receive the response. Used when implementing a new talker function.	(ESC T)	DAQDA100. sendTriggerDA100

Except during connection or status updates, communication functions do not perform status updates of stored data.

Control Functions

Function	Command	Function
Switch operation mode	DS	DAQDA100. switchModeDA100
Retrieve code type Switch (binary/ASCII code)	-	DAQDA100. switchCodeDA100
Reconfigure	RS	DAQDA100. reconstructDA100
Initialize settings	RC	DAQDA100. initSetValueDA100
Reset alarms	AR	DAQDA100. ackAlarmDA100
Set date/time (current)	SD	DAQDA100. setDateTimeNowDA100
Calculation start/stop	EX	DAQDA100. switchComputeDA100
Report start/stop	DR	DAQDA100. switchReportDA100
Establish setup mode	XE	DAQDA100. establishDA100

Generally, the status is updated at the end of the process.
Status is not updated when establishing the setup mode.

Setting (Operation Mode) Functions

Function	Command	Function
Range	SR	DAQDA100. setRangeDA100
SKIP (not used)		
DC voltage input	SR	DAQDA100. setRangeDA100
Thermocouple input	SR	DAQDA100. setRangeDA100
RTD input	SR	DAQDA100. setRangeDA100
Contact input (DI)	SR	DAQDA100. setRangeDA100
DC current	SR	DAQDA100. setRangeDA100
Strain	SR	DAQDA100. setRangeDA100
Pulse	SR	DAQDA100. setRangeDA100
Power monitor	SR	DAQDA100. setRangeDA100
Diff comp between ch	SR	DAQDA100. setChDELTA100
Remote RJC	SR	DAQDA100. setChRRJCDA100
Unit name	SN	DAQDA100. setChUnitDA100
Alarm	SA	DAQDA100. setChAlarmDA100

Becomes the channel unit setting.
The status is updated after the setting is entered.

List of Data Retrieval Functions

Function	Command	Function	
Measured	Meas ch	TS, FM	DAQDA100. measInstChDA100
Data (Inst val)	Comp ch	TS, FM	DAQDA100. mathInstChDA100
Channel info data	Meas ch	TS, LF	DAQDA100. measInfoChDA100
	Comp ch	TS, LF	DAQDA100. mathInfoChDA100
System configuration data		TS,CF	CDAQDA100. updateSystemConfigDA100
Report status		TS,RF	CDAQDA100. updateReportStatusDA100
Setup data			
Declare Op	Sngl spec	TS, LF	CDAQDA100. talkOperationChDataDA100
	Specify rng	TS, LF	CDAQDA100. talkOperationDataDA100
Setup Mode	Sngl spec	TS, LF	CDAQDA100. talkSetupChDataDA100
	Specify rng	TS, LF	CDAQDA100. talkSetupDataDA100
Calibrate Mode	Sngl spec	TS, LF	CDAQDA100. talkCalibrationChDataDA100
	Specify rng	TS, LF	CDAQDA100. talkCalibrationDataDA100
Get data by lines		-	CDAQDA100. getSetDataByLineDA100

Setup data is not stored, so it is retrieved in the same order as mentioned in sections 7.2 and 7.3. In this case, status is not updated.

Channel information data and system configuration data are stored internally, but the user can explicitly perform acquisition.

The report status is stored internally, but is not updated unless the user explicitly acquires it.

Retrieval Functions

Measured Data

Data Name	Function	
Data value	DAQDA100. dataValueDA100	
Data status values	DAQDA100. dataStatusDA100	
Alarm (presence/absence)	DAQDA100. dataAlarmDA100	
Measured values	Dbl intg	DAQDA100. dataDoubleValueDA100
	String	DAQDA100. dataStringValueDA100
Time	Year	DAQDA100. dataYearDA100
	Month	DAQDA100. dataMonthDA100
	Day	DAQDA100. dataDayDA100
	Hour	DAQDA100. dataHourDA100
	Minute	DAQDA100. dataMinuteDA100
	Second	DAQDA100. dataSecondDA100
Alarm type	DAQDA100. alarmTypeDA100	

Channel Information Data

Data Name	Function
Decimal point position	DAQDA100. channelPointDA100
Channel status	DAQDA100. channelStatusDA100
Unit name	DAQDA100. toChannelUnitDA100

System Configuration Data

Data Name	Function	
Measurement interval	DAQDA100. unitIntervalDA100	
Unit	Presence or absence	DAQDA100. unitValidDA100
Module	Internal code	DAQDA100. moduleCodeDA100
	Module name	DAQDA100. toModuleNameDA100

Status Data

Data Name	Function
Status byte	DAQDA100. statusByteDA100
Retrieve code type (binary/ASCII code)	DAQDA100. statusCodeDA100
Report status	DAQDA100. statusReportDA100

Utilities

Function/Data Name		Class and Member Function
Meas val	Chnge to dbl intg	DAQDA100. toDoubleValueDA100
	Convert into string	DAQDA100.toStringValueDA100
Alarm	Alarm type string	DAQDA100. toAlarmNameDA100
	Get max length of alarm string	DAQDA100. alarmMaxLengthDA100
The version number of this API		DAQDA100. versionAPIDA100
The revision number of this API		DAQDA100. revisionAPIDA100
Error	Error message string	DAQDA100. toErrorMessageDA100
	Get the length of the error message string	DAQDA100. errorMaxLengthDA100

23.2 Programming - DARWIN/C# -

Declaration Statements

Adding the class file C# to the project is equivalent to declaring all functions and constants.

Retrieval of the Measured Data

Program Example

```
using System;
using System.Text;
using System.Runtime.InteropServices;

namespace MeasCS
{
    class Class1
    {
        [STAThread]
        static void Main(string[] args)
        {
            int rc;
            Encoding enc = Encoding.GetEncoding ("ascii");
            String address = "192.168.1.11";
            //connect
            int comm =
            DAQDA100.openDA100(enc.GetBytes(address), out rc);
            //get
            rc = DAQDA100.measInstChDA100(comm, 0, 1);
            int val = DAQDA100.dataValueDA100(comm, 0, 1);
            //disconnect
            DAQDA100.closeDA100(comm);
        }
    }
}
```

23.3 Correspondence between Functions and Instantaneous Value Data Loading Functions - DARWIN/Visual C# -

This section indicates functions that this extension API supports.

Note

This extension API provides a portion of the functions common to the DARWIN series instruments. Model-specific functions, setup functions of the setup mode, and A/D calibration functions are not given. The functions can be added by using the commands of the DARWIN communication function.

The word command in the table signifies the command of the DARWIN communication function. For the details on the commands, see the Communication Interface User's Manual.

There are two types of functions, status transition functions and retrieval functions. Status transition functions control DARWIN.

With retrieval functions, the parameter values of the current status are retrieved.

When using retrieval functions, the data value of the stored current status is returned (the status of the extension API does not change).

Status Transition Functions

Communication Functions

Function	Command	Function
Comm connection	-	DAQDA100Reader.openDA100Reader
Comm disconnection	-	DAQDA100Reader.closeDA100Reader

Data Retrieval Functions

Function	Command	Function
Meas data (inst val)	EF	DAQDA100Reader.measInstChDA100Reader
Meas ch	EF	DAQDA100Reader.mathInstChDA100Reader
Comp ch	EL	DAQDA100Reader.measInfoChDA100Reader
Channel info data	EL	DAQDA100Reader.mathInfoChDA100Reader
Meas ch		
Comp ch		

Channel information data is stored internally, but the user can explicitly perform acquisition.

Retrieval Functions

Measured Data

Data Name	Function	
Data value	DAQDA100Reader. dataValueDA100Reader	
Data status values	DAQDA100Reader. dataStatusDA100Reader	
Alarm (presence/absence)	DAQDA100Reader. dataAlarmDA100Reader	
Measured value	Double integer	DAQDA100Reader. dataDoubleValueDA100Reader
	String	DAQDA100Reader. dataStringValueDA100Reader
Time	Year	DAQDA100Reader. dataYearDA100Reader
	Month	DAQDA100Reader. dataMonthDA100Reader
	Day	DAQDA100Reader. dataDayDA100Reader
	Hour	DAQDA100Reader. dataHourDA100Reader
	Minute	DAQDA100Reader. dataMinuteDA100Reader
	Second	DAQDA100Reader. dataSecondDA100Reader
	Milliseconds	DAQDA100Reader. dataMilliSecDA100Reader
Alarm type	DAQDA100Reader. alarmTypeDA100Reader	

Channel Information Data

Data Name	Function
Decimal Point Position	DAQDA100Reader. channelPointDA100Reader
Channel Status	DAQDA100Reader. channelStatusDA100Reader
Unit name	DAQDA100Reader. toChannelUnitDA100Reader

Utilities

Function/Data Name	Function	
Measured values	Chng to dbl integ	DAQDA100Reader. toDoubleValueDA100Reader
	Convert into string	DAQDA100Reader. toStringValueDA100Reader
Alarm Retrieval	Get alarm type strg	DAQDA100Reader. toAlarmNameDA100Reader
	Get max lngth alm strng	DAQDA100Reader. alarmMaxLengthDA100Reader
Get the version number of this API	DAQDA100Reader. versionAPIDA100Reader	
Get the revision number of this API	DAQDA100Reader. revisionAPIDA100Reader	
Error	Get err msg string	DAQDA100Reader. toErrorMessageDA100Reader
	Get max length of err msg string	DAQDA100Reader. errorMaxLengthDA100Reader

23.4 Program for Loading Instantaneous Value Data -DARWIN/C#-

Declaration Statements

Adding the class file (DAQDA100Reader.cs) C# to the project is equivalent to declaring all functions and constants.

Retrieval of the Measured Data

Program Example

```
using System;
using System.Text;
using System.Runtime.InteropServices;

namespace MeasCS
{
    class Class1
    {
        [STAThread]
        static void Main(string[] args)
        {
            int rc;
            Encoding enc = Encoding.GetEncoding ("ascii");
            String address = "192.168.1.11";
            //connect
            int comm =
DAQDA100Reader.openDAQDA100Reader(enc.GetBytes(address), out rc);
            //get
            rc = DAQDA100Reader.measInstChDAQDA100Reader(comm,
0, 1);
            int val =
DAQDA100Reader.dataValueDAQDA100Reader(comm, 0, 1);
            //disconnect
            DAQDA100Reader.closeDAQDA100Reader(comm);
        }
    }
}
```

24.1 Details of Functions - DARWIN (Visual C/Visual Basic/Visual Basic.NET/C#) - Status Transition Functions

This section describes the DARWIN functions that are used in Visual C, Visual Basic, Visual Basic.NET, and C#. The functions are listed in alphabetical order by the function name.

For details on constants and types, see chapter 25.

For DARWIN terminology, see appendix 2.

Most functions return an error number as a return value. Error number 0 is returned if there is no error.

ackAlarmDA100

Syntax

```
int ackAlarmDA100(DAQDA100 daqda100);
```

Declaration

Visual Basic

```
Public Declare Function ackAlarmDA100 Lib "DAQDA100" (ByVal daqda100 As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function ackAlarmDA100 Lib "DAQDA100" (ByVal daqda100 As Integer) As Integer
```

C#

```
[DllImport("DAQDA100.dll" CharSet=CharSet.Auto, EntryPoint="ackAlarmDA100")]  
public static extern int ackAlarmDA100(int daqda100);
```

Parameters

daqda100 Specify the device descriptor.

Description

Resets alarms.

- The status is updated at the end of the process.
- This function executes the AR command of the DARWIN communication function.

Return value

Returns an error number.

Error:

Not descriptor No device descriptor.

Reference

CDAQDA100::ackAlarm

closeDA100

Syntax

```
int closeDA100(DAQDA100 daqda100);
```

Declaration**Visual Basic**

```
Public Declare Function closeDA100 Lib "DAQDA100" (ByVal daqda100 As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function closeDA100 Lib "DAQDA100" (ByVal daqda100 As Integer) As Integer
```

C#

```
[DllImport("DAQDA100.dll", CharSet=CharSet.Auto, EntryPoint="closeDA100")]
public static extern int closeDA100(int daqda100);
```

Parameters

daqda100 Specify the device descriptor.

Description

Disconnects the communication using the specified device descriptor.

- When the communication is disconnected, the value of the device descriptor is meaningless.
- After disconnection, do not use the value of the device descriptor.

Return value

Returns an error number.

Error:

Not descriptor No device descriptor.

Reference

CDAQDA100::close

establishDA100

Syntax

```
int establishDA100(DAQDA100 daqda100, int iSetup);
```

Declaration

Visual Basic

```
Public Declare Function establishDA100 Lib "DAQDA100" (ByVal daqda100 As Long, ByVal iSetup As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function establishDA100 Lib "DAQDA100" (ByVal daqda100 As Integer, ByVal iSetup As Integer) As Integer
```

C#

```
[DllImport("DAQDA100.dll" CharSet=CharSet.Auto, EntryPoint="establishDA100")]  
public static extern int establishDA100(int daqda100, int iSetup);
```

Parameters

daqda100	Specify the device descriptor.
iSetup	Specify establishment of setup.

Description

Establishes setting contents for setup mode.

- It is only valid in setup mode.
- This function executes the EX command of the DARWIN communication function.

Return value

Returns an error number.

Error:

Not descriptor No device descriptor.

Reference

CDAQDA100::establish

getSetDataByLineDA100

Syntax

```
int getSetDataByLineDA100(DAQDA100 daqda100, char * strLine,
int maxLine, int * lenLine, int * pFlag);
```

Declaration

Visual Basic

```
Public Declare Function getSetDataByLineDA100 Lib
"DAQDA100"(ByVal daqda100 As Long, ByVal strLine As String,
ByVal maxLine As Long, ByRef lenLine As Long, ByRef pFlag As
Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function getSetDataByLineDA100 Lib
"DAQDA100"(ByVal daqda100 As Integer, ByVal strLine As String,
ByVal maxLine As Integer, ByRef lenLine As Integer, ByRef
pFlag As Integer) As Integer
```

C#

```
[DllImport("DAQDA100.dll" CharSet=CharSet.Auto,
EntryPoint="getSetDataByLineDA100")]
public static extern int getSetDataByLineDA100(int daqda100,
byte[] strLine, int maxLine, out int lenLine, out int pFlag);
```

Parameters

daqda100	Specify the device descriptor.
strLine	Specify the field where the string received by lines is to be stored.
maxLine	Specify the byte size of the field where the string received by lines is to be stored.
lenLine	Specify the destination where the byte size of the actual string received is returned.
pFlag	Specify the destination where the flag is to be returned.

Description

Gets the output from the talker function in units of lines after execution of the declaration for the retrieval of setup data.

- Stores the received string excluding line feeds.
- When the last set of data is retrieved, the flag status is set. The flag status is also set when the function ends in error.
- Do not perform communications using other functions until the data retrieval is completed. Other functions may not operate properly.

Return value

Returns an error number.

Error:

Not descriptor No device descriptor.

Reference

CDAQDA100::getSetDataByLine

initSetValueDA100

Syntax

```
int initSetValueDA100(DAQDA100 daqda100);
```

Declaration

Visual Basic

```
Public Declare Function initSetValueDA100 Lib "DAQDA100" (ByVal daqda100 As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function initSetValueDA100 Lib "DAQDA100" (ByVal daqda100 As Integer) As Integer
```

C#

```
[DllImport("DAQDA100.dll" CharSet=CharSet.Auto, EntryPoint="initSetValueDA100")]  
public static extern int initSetValueDA100(int daqda100);
```

Parameters

daqda100 Specify the device descriptor.

Description

Executes initial balancing of settings.

- The status is updated at the end of the process.
- This function executes the RC command of the DARWIN communication function.

Return value

Returns an error number.

Error:

Not descriptor No device descriptor.

Reference

CDAQDA100::initSetValue

mathInfoChDA100

Syntax

```
int mathInfoChDA100(DAQDA100 daqda100, int chNo);
```

Declaration**Visual Basic**

```
Public Declare Function mathInfoChDA100 Lib "DAQDA100" (ByVal daqda100 As Long, ByVal chNo As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function mathInfoChDA100 Lib "DAQDA100" (ByVal daqda100 As Integer, ByVal chNo As Integer) As Integer
```

C#

```
[DllImport("DAQDA100.dll" CharSet=CharSet.Auto, EntryPoint="mathInfoChDA100")]
public static extern int mathInfoChDA100(int daqda100, int chNo);
```

Parameters

daqda100	Specify the device descriptor.
chNo	Specify the channel number.

Description

Gets the channel information data of the specified computation channel range.

- If the constant for "Specify all channel numbers" is specified for the channel numbers, all computation channels are processed.
- Specify measurement channels and computation channels separately.
- The status is updated at the end of the process.
- This function executes the TS,LF command of the DARWIN communication function.

Return value

Returns an error number.

Error:

Not descriptor No device descriptor.

Reference

CDAQDA100::mathInfoCh

mathInstChDA100

Syntax

```
int mathInstChDA100(DAQDA100 daqda100, int chNo);
```

Declaration

Visual Basic

```
Public Declare Function mathInstChDA100 Lib "DAQDA100" (ByVal daqda100 As Long, ByVal chNo As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function mathInstChDA100 Lib "DAQDA100" (ByVal daqda100 As Integer, ByVal chNo As Integer) As Integer
```

C#

```
[DllImport("DAQDA100.dll" CharSet=CharSet.Auto, EntryPoint="mathInstChDA100")]  
public static extern int mathInstChDA100(int daqda100, int chNo);
```

Parameters

daqda100	Specify the device descriptor.
chNo	Specify the channel number.

Description

Gets the measured data of the specified computation channel.

- If the constant for "Specify all channel numbers" is specified for the channel numbers, all computation channels are processed.
- Specify measurement channels and computation channels separately.
- The status is updated at the end of the process.
- This function executes the TS,FM command of the DARWIN communication function.

Return value

Returns an error number.

Error:

Not descriptor No device descriptor.

Reference

CDAQDA100::mathInstCh

measInfoChDA100

Syntax

```
int measInfoChDA100(DAQDA100 daqda100, int chType, int chNo);
```

Declaration**Visual Basic**

```
Public Declare Function measInfoChDA100 Lib "DAQDA100" (ByVal daqda100 As Long, ByVal chType As Long, ByVal chNo As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function measInfoChDA100 Lib "DAQDA100" (ByVal daqda100 As Integer, ByVal chType As Integer, ByVal chNo As Integer) As Integer
```

C#

```
[DllImport("DAQDA100.dll" CharSet=CharSet.Auto, EntryPoint="measInfoChDA100")]
public static extern int measInfoChDA100(int daqda100, int chType, int chNo);
```

Parameters

daqda100	Specify the device descriptor.
chType	Specify the channel type.
chNo	Specify the channel number.

Description

Gets the channel information data of the specified measurement channel (specified with the channel type and channel number).

- If the channel type is set to the constant for "Specify all measurement channel types," all subunits are processed.
- If the channel number is set to the constant for "Specify all channel numbers," all channels within the channel type are processed.
- Specify measurement channels and computation channels separately.
- The status is updated at the end of the process.
- This function executes the TS,LF command of the DARWIN communication function.

Return value

Returns an error number.

Error:

Not descriptor No device descriptor.

Reference

CDAQDA100::measInfoCh

measInstChDA100

Syntax

```
int measInstChDA100(DAQDA100 daqda100, int chType, int chNo);
```

Declaration

Visual Basic

```
Public Declare Function measInstChDA100 Lib "DAQDA100" (ByVal daqda100 As Long, ByVal chType As Long, ByVal chNo As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function measInstChDA100 Lib "DAQDA100" (ByVal daqda100 As Integer, ByVal chType As Integer, ByVal chNo As Integer) As Integer
```

C#

```
[DllImport("DAQDA100.dll" CharSet=CharSet.Auto, EntryPoint="measInstChDA100")]  
public static extern int measInstChDA100(int daqda100, int chType, int chNo);
```

Parameters

daqda100	Specify the device descriptor.
chType	Specify the channel type.
chNo	Specify the channel number.

Description

Gets the measured data of the specified measurement channel (specified with the channel type and channel number).

- If the channel type is set to the constant for "Specify all measurement channel types," all subunits are processed.
- If the channel number is set to the constant for "Specify all channel numbers," all channels within the channel type are processed.
- Specify measurement channels and computation channels separately.
- The status is updated at the end of the process.
- This function executes the TS,FM command of the DARWIN communication function.

Return value

Returns an error number.

Error:

Not descriptor No device descriptor.

Reference

CDAQDA100::measInstCh

openDA100

Syntax

```
DAQDA100 openDA100(const char * strAddress, int * errorCode);
```

Declaration

Visual Basic

```
Public Declare Function openDA100 Lib "DAQDA100" (ByVal strAddress As String, ByRef errorCode As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function openDA100 Lib "DAQDA100" (ByVal strAddress As String, ByRef errorCode As Integer) As Integer
```

C#

```
[DllImport("DAQDA100.dll", CharSet=CharSet.Auto, EntryPoint="openDA100")]
public static extern int openDA100(byte[] strAddress, out int errorCode);
```

Parameters

strAddress	Specify the IP address as a string.
errorCode	Specify the destination where the error number is to be returned.

Description

Connects to the device with the address specified by the parameters.

- Creates a device descriptor and returns the value as a return value.
- Stores the error number if the return destination is specified.
- The port number is fixed, and set to the communication constant "communication.port number."
- Initializes the stored data. Gets the system configuration data, channel information data, and status byte and stores it.
- The specified string is, in general, an ASCII string.
- If unsuccessful, returns NULL in Visual C, or otherwise 0.

Return value

Returns the device descriptor.

Error:

Creating descriptor is failure	Failed to create the device descriptor.
--------------------------------	---

Reference

CDAQDA100::open

receiveByteDA100

Syntax

```
int receiveByteDA100(DAQDA100 daqda100, unsigned char *  
byteData, int maxData, int * lenData) );
```

Declaration

Visual Basic

```
Public Declare Function receiveByteDA100 Lib "DAQDA100" (ByVal  
daqda100 As Long, ByVal byteData As Byte, ByVal maxData As  
Long, ByVal lenData As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function receiveByteDA100 Lib  
"DAQDA100" (ByVal daqda100 As Integer, ByVal byteData As Byte,  
ByVal maxData As Integer, ByVal lenData As Integer) As Integer
```

C#

```
[DllImport("DAQDA100.dll" CharSet=CharSet.Auto,  
EntryPoint="receiveByteDA100")]  
public static extern int receiveByteDA100(int daqda100, byte[]  
byteData, int maxData, out int lenData);
```

Parameters

daqda100	Specify the device descriptor.
byteData	Specify the field where the received byte data is to be stored.
maxData	Specify the byte size of the received data.
lenData	Specify the destination where the byte size of the actual data received is returned.

Description

Stores the received data to the field specified by the parameter up to the specified byte size.

- Returns the byte size of the actual data received if the return destination is specified.
- If multiple bytes of data exist, repeat the function.
- Do not perform communications using other functions until the data retrieval is completed. Other functions may not operate properly.
- The user must carry out determination of the data end.
- Used for receiving binary output when implementing a model-specific talker function.

Return value

Returns an error number.

Error:

Not descriptor	No device descriptor.
----------------	-----------------------

Reference

CDAQDA100::receiveByte

receiveLineDA100

Syntax

```
int receiveLineDA100(DAQDA100 daqda100, char * strLine, int
maxLine, int * lenLine);
```

Declaration**Visual Basic**

```
Public Declare Function receiveLineDA100 Lib "DAQDA100" (ByVal
daqda100 As Long, ByVal strLine As String, ByVal maxLine As
Long, ByRef lenLine As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function receiveLineDA100 Lib
"DAQDA100" (ByVal daqda100 As Integer, ByVal strLine As String,
ByVal maxLine As Integer, ByRef lenLine As Integer) As Integer
```

C#

```
[DllImport("DAQDA100.dll" CharSet=CharSet.Auto,
EntryPoint="receiveLineDA100")]
public static extern int receiveLineDA100(int daqda100, byte[]
strLine, int maxLine, out int lenLine);
```

Parameters

daqda100	Specify the device descriptor.
strLine	Specify the field where the received string is to be stored.
maxLine	Specify the byte size of the field where the received string is to be stored.
lenLine	Specify the destination where the byte size of the actual string received is returned.

Description

Receives data in the field specified for storing received strings by the parameter, until a carriage return is detected or up to the specified byte size.

- Stores the received string excluding line feeds in the storage field.
- Stores in the specified destination the byte size of the actual data received and stored if the return destination is specified.
- If multiple lines of data exist, repeat the function.
- Do not perform communications using other functions until the data retrieval is completed. Other functions may not operate properly.
- The user must carry out determination of the data end.
- The strings that can be stored are, in general, ASCII strings.

Return value

Returns an error number.

Error:

Not descriptor	No device descriptor.
----------------	-----------------------

Reference

CDAQDA100::receiveLine

reconstructDA100

Syntax

```
int reconstructDA100(DAQDA100 daqda100);
```

Declaration

Visual Basic

```
Public Declare Function reconstructDA100 Lib "DAQDA100" (ByVal daqda100 As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function reconstructDA100 Lib "DAQDA100" (ByVal daqda100 As Integer) As Integer
```

C#

```
[DllImport("DAQDA100.dll" CharSet=CharSet.Auto, EntryPoint="reconstructDA100")]  
public static extern int reconstructDA100(int daqda100);
```

Parameters

daqda100 Specify the device descriptor.

Description

Executes system reconfiguration.

- The status is updated at the end of the process.
- This function executes the RS command of the DARWIN communication function.

Return value

Returns an error number.

Error:

Not descriptor No device descriptor.

Reference

CDAQDA100::reconstruct

runCommandDA100

Syntax

```
int runCommandDA100(DAQDA100 daqda100, const char * strCmd);
```

Declaration**Visual Basic**

```
Public Declare Function runCommandDA100 Lib "DAQDA100" (ByVal daqda100 As Long, ByVal strCmd As String) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function runCommandDA100 Lib "DAQDA100" (ByVal daqda100 As Integer, ByVal strCmd As String) As Integer
```

C#

```
[DllImport("DAQDA100.dll" CharSet=CharSet.Auto, EntryPoint="runCommandDA100")]
public static extern int runCommandDA100(int daqda100, byte[] strCmd);
```

Parameters

daqda100	Specify the device descriptor.
strCmd	Specify the command message to be sent.

Description

Sends the specified command message and terminator and receives the response.

- This function adds a terminator to the command message at the time of transmission. Therefore, do not include the terminator in the command message.
- This function does not support simultaneous transmission of multiple commands or command messages that include the terminator.
- Like the data output request command of the talker function, does not support commands that do not send responses.

Return value

Returns an error number.

Error:

Not descriptor	No device descriptor.
----------------	-----------------------

Reference

CDAQDA100::runCommand

sendLineDA100

Syntax

```
int sendLineDA100(DAQDA100 daqda100, const char * strLine);
```

Declaration

Visual Basic

```
Public Declare Function sendLineDA100 Lib "DAQDA100" (ByVal daqda100 As Long, ByVal strLine As String) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function sendLineDA100 Lib "DAQDA100" (ByVal daqda100 As Integer, ByVal strLine As String) As Integer
```

C#

```
[DllImport("DAQDA100.dll" CharSet=CharSet.Auto, EntryPoint="sendLineDA100")]  
public static extern int sendLineDA100(int daqda100, byte[] strLine);
```

Parameters

daqda100	Specify the device descriptor.
strLine	Specify the string to be sent.

Description

Sends the string data specified by the parameter.

- When sending the command, the terminator is also part of the data.
- This function does not receive a response. Receive the returned data using another receive function.
- The specified string is, in general, an ASCII string.

Return value

Returns an error number.

Error:

Not descriptor	No device descriptor.
----------------	-----------------------

Reference

CDAQDA100::sendLine

sendTriggerDA100

Syntax

```
int sendTriggerDA100(DAQDA100 daqda100);
```

Declaration

Visual Basic

```
Public Declare Function sendTriggerDA100 Lib "DAQDA100" (ByVal daqda100 As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function sendTriggerDA100 Lib "DAQDA100" (ByVal daqda100 As Integer) As Integer
```

C#

```
[DllImport("DAQDA100.dll" CharSet=CharSet.Auto, EntryPoint="sendTriggerDA100")]  
public static extern int sendTriggerDA100(int daqda100);
```

Parameters

daqda100 Specify the device descriptor.

Description

Sends a trigger command (ESC T), and receives the response.

Return value

Returns an error number.

Error:

Not descriptor No device descriptor.

Reference

CDAQDA100::sendTrigger

setChAlarmDA100

Syntax

```
int setChAlarmDA100(DAQDA100 daqda100, int chType, int chNo,
int levelNo, int iAlarmType, int value, int relayType, int
relayNo);
```

Declaration

Visual Basic

```
Public Declare Function setChAlarmDA100 Lib "DAQDA100" (ByVal
daqda100 As Long, ByVal chType As Long, ByVal chNo As Long,
ByVal levelNo As Long, ByVal iAlarmType As Long, ByVal value
As Long, ByVal relayType As Long, ByVal relayNo As Long) As
Long
```

Visual Basic.NET

```
Public Declare Ansi Function setChAlarmDA100 Lib
"DAQDA100" (ByVal daqda100 As Integer, ByVal chType As Integer,
ByVal chNo As Integer, ByVal levelNo As Integer, ByVal
iAlarmType As Integer, ByVal value As Integer, ByVal relayType
As Integer, ByVal relayNo As Integer) As Integer
```

C#

```
[DllImport("DAQDA100.dll" CharSet=CharSet.Auto,
EntryPoint="setChAlarmDA100")]
public static extern int setChAlarmDA100(int daqda100, int
chType, int chNo, int levelNo, int iAlarmType, int value, int
relayType, int relayNo);
```

Parameters

daqda100	Specify the device descriptor.
chType	Specify the channel type.
chNo	Specify the channel number.
levelNo	Specify the alarm level.
iAlarmType	Specify the alarm type.
value	Specify the alarm value.
relayType	Specify the relay type.
relayNo	Specify the relay number.

Description

Sets the specified alarm (alarm level and alarm type) and alarm value to the specified channel (specified by channel type and channel number).

- If the channel type is set to the constant for "Specify all measurement channel types," all subunits are processed.
- If the channel number is set to the constant for "Specify all channel numbers," all channels within the channel type are processed.
- If the alarm level is set to the constant for "Specify all alarm level numbers," all alarm levels within the channels are processed.
- With relay specification, when the relay number is less than or equal to 0, the relay is not specified (turned OFF).
- Updates the stored channel information data after the setting.
- This function executes the SA command of the DARWIN communication function.

Return value

Returns an error number.

Error:

Not descriptor

No device descriptor.

Reference

CDAQDA100::setChAlarm

setChDELTADA100

Syntax

```
int setChDELTADA100(DAQDA100 daqda100, int chType, int chNo,
int refChNo, int spanMin, int spanMax);
```

Declaration

Visual Basic

```
Public Declare Function setChDELTADA100 Lib "DAQDA100" (ByVal
daqda100 As Long, ByVal chType As Long, ByVal chNo As Long,
ByVal refChNo As Long, ByVal spanMin As Long, ByVal spanMax As
Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function setChDELTADA100 Lib
"DAQDA100" (ByVal daqda100 As Integer, ByVal chType As Integer,
ByVal chNo As Integer, ByVal refChNo As Integer, ByVal spanMin
As Integer, ByVal spanMax As Integer) As Integer
```

C#

```
[DllImport("DAQDA100.dll" CharSet=CharSet.Auto,
EntryPoint="setChDELTADA100")]
public static extern int setChDELTADA100(int daqda100, int
chType, int chNo, int refChNo, int spanMin, int spanMax);
```

Parameters

daqda100	Specify the device descriptor.
chType	Specify the channel type.
chNo	Specify the channel number.
refChNo	Specify the channel number of the reference channel.
spanMin	Specify the left value of the span.
spanMax	Specify the right value of the span.

Description

Sets the specified reference channel difference computation on the specified channel (specified under channel type and channel number).

- If the channel type is set to the constant for "Specify all measurement channel types," all subunits are processed.
- If the channel number is set to the constant for "Specify all channel numbers," all channels within the channel type are processed.
- With the span specification, if the left and right values are the same, it is considered omitted.
- Updates the stored channel information data after the setting.
- This function executes the SR command of the DARWIN communication function.

Return value

Returns an error number.

Error:

Not descriptor No device descriptor.

Reference

CDAQDA100::setChDELTA

setChRRJCDA100

Syntax

```
int setChRRJCDA100(DAQDA100 daqda100, int chType, int chNo,
int refChNo, int spanMin, int spanMax);
```

Declaration

Visual Basic

```
Public Declare Function setChRRJCDA100 Lib "DAQDA100" (ByVal
daqda100 As Long, ByVal chType As Long, ByVal chNo As Long,
ByVal refChNo As Long, ByVal spanMin As Long, ByVal spanMax As
Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function setChRRJCDA100 Lib
"DAQDA100" (ByVal daqda100 As Integer, ByVal chType As Integer,
ByVal chNo As Integer, ByVal refChNo As Integer, ByVal spanMin
As Integer, ByVal spanMax As Integer) As Integer
```

C#

```
[DllImport("DAQDA100.dll" CharSet=CharSet.Auto,
EntryPoint="setChRRJCDA100")]
public static extern int setChRRJCDA100(int daqda100, int
chType, int chNo, int refChNo, int spanMin, int spanMax);
```

Parameters

daqda100	Specify the device descriptor.
chType	Specify the channel type.
chNo	Specify the channel number.
refChNo	Specify the channel number of the reference channel.
spanMin	Specify the left value of the span.
spanMax	Specify the right value of the span.

Description

Sets the remote RJC specified for the reference channel on the specified channel (specified under channel type and channel number).

- If the channel type is set to the constant for "Specify all measurement channel types," all subunits are processed.
- If the channel number is set to the constant for "Specify all channel numbers," all channels within the channel type are processed.
- With the span specification, if the left and right values are the same, it is considered omitted.
- Updates the stored channel information data after the setting.
- This function executes the SR command of the DARWIN communication function.

Return value

Returns an error number.

Error:

Not descriptor No device descriptor.

Reference

CDAQDA100::setChRRJC

setChUnitDA100

Syntax

```
int setChUnitDA100(DAQDA100 daqda100, int chType, int chNo,  
const char * strUnit);
```

Declaration

Visual Basic

```
Public Declare Function setChUnitDA100 Lib "DAQDA100" (ByVal  
daqda100 As Long, ByVal chType As Long, ByVal chNo As Long,  
ByVal strUnit As String) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function setChUnitDA100 Lib  
"DAQDA100" (ByVal daqda100 As Integer, ByVal chType As Integer,  
ByVal chNo As Integer, ByVal strUnit As String) As Integer
```

C#

```
[DllImport("DAQDA100.dll" CharSet=CharSet.Auto,  
EntryPoint="setChUnitDA100")]  
public static extern int setChUnitDA100(int daqda100, int  
chType, int chNo, byte[] strUnit);
```

Parameters

daqda100	Specify the device descriptor.
chType	Specify the channel type.
chNo	Specify the channel number.
strUnit	Specify the unit name.

Description

Sets the specified unit name on the specified channel (specified under channel type and channel number).

- If the channel type is set to the constant for "Specify all measurement channel types," all subunits are processed.
- If the channel number is set to the constant for "Specify all channel numbers," all channels within the channel type are processed.
- Updates the stored channel information data after the setting.
- This function executes the SN command of the DARWIN communication function.
- The specified string is, in general, an ASCII string.

Return value

Returns an error number.

Error:

Not descriptor	No device descriptor.
----------------	-----------------------

Reference

CDAQDA100::setChUnit

setDateTimeNowDA100

Syntax

```
int setDateTimeNowDA100(DAQDA100 daqda100);
```

Declaration

Visual Basic

```
Public Declare Function setDateTimeNowDA100 Lib
"DAQDA100"(ByVal daqda100 As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function setDateTimeNowDA100 Lib
"DAQDA100"(ByVal daqda100 As Integer) As Integer
```

C#

```
[DllImport("DAQDA100.dll" CharSet=CharSet.Auto,
EntryPoint="setDateTimeNowDA100")]
public static extern int setDateTimeNowDA100(int daqda100);
```

Parameters

daqda100 Specify the device descriptor.

Description

Sets the current date/time of the PC.

- The status is updated at the end of the process.
- This function executes the SD command of the DARWIN communication function.

Return value

Returns an error number.

Error:

Not descriptor No device descriptor.

Reference

CDAQDA100::setDateTime

setRangeDA100

Syntax

```
int setRangeDA100(DAQDA100 daqda100, int chType, int chNo, int iRange, int spanMin, int spanMax, int scaleMin, int scaleMax, int scalePoint, int bFilter, int iItem, int iWire);
```

Declaration

Visual Basic

```
Public Declare Function setRangeDA100 Lib "DAQDA100" (ByVal daqda100 As Long, ByVal chType As Long, ByVal chNo As Long, ByVal iRange As Long, ByVal spanMin As Long, ByVal spanMax As Long, ByVal scaleMin As Long, ByVal scaleMax As Long, ByVal scalePoint As Long, ByVal bFilter As Long, ByVal iItem As Long, ByVal iWire As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function setRangeDA100 Lib "DAQDA100" (ByVal daqda100 As Integer, ByVal chType As Integer, ByVal chNo As Integer, ByVal iRange As Integer, ByVal spanMin As Integer, ByVal spanMax As Integer, ByVal scaleMin As Integer, ByVal scaleMax As Integer, ByVal scalePoint As Integer, ByVal bFilter As Integer, ByVal iItem As Integer, ByVal iWire As Integer) As Integer
```

C#

```
[DllImport("DAQDA100.dll" CharSet=CharSet.Auto, EntryPoint="setRangeDA100")]  
public static extern int setRangeDA100(int daqda100, int chType, int chNo, int iRange, int spanMin, int spanMax, int scaleMin, int scaleMax, int scalePoint, int bFilter, int iItem, int iWire);
```

Parameters

daqda100	Specify the device descriptor.
chType	Specify the channel type.
chNo	Specify the channel number.
iRange	Specify the range type.
spanMin	Specify the left value of the span.
spanMax	Specify the right value of the span.
scaleMin	Specify the left value of the scale.
scaleMax	Specify the right value of the scale.
scalePoint	Specify the decimal point position for scaling.
bFilter	Specify a filter using a Boolean value.
iItem	Specify the power measurement parameter.
iWire	Specify the power connection method.

Description

Sets the specified range type on the specified channel (specified under channel type and channel number).

- If the channel type is set to the constant for “Specify all measurement channel types,” all subunits are processed.
- If the channel number is set to the constant for “Specify all channel numbers,” all channels within the channel type are processed.
- With the span and scale specification, if the left and right values are the same, they are considered omitted.
- The filter specification is only valid for the pulse range.
- The power measurement item and power connection method are only valid for the power monitoring range.
- Updates the stored channel information data after the setting.
- This function executes the SR command of the DARWIN communication function.

Return value

Returns an error number.

Error:

Not descriptor No device descriptor.

Reference

`CDAQDA100::setRange`

switchCodeDA100

Syntax

```
int switchCodeDA100(DAQDA100 daqda100, int iCode);
```

Declaration

Visual Basic

```
Public Declare Function switchCodeDA100 Lib "DAQDA100" (ByVal daqda100 As Long, ByVal iCode As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function switchCodeDA100 Lib "DAQDA100" (ByVal daqda100 As Integer, ByVal iCode As Integer) As Integer
```

C#

```
[DllImport("DAQDA100.dll" CharSet=CharSet.Auto, EntryPoint="switchCodeDA100")]  
public static extern int switchCodeDA100(int daqda100, int iCode);
```

Parameters

daqda100	Specify the device descriptor.
iCode	Gets the retrieve code type.

Description

Switches to the specified retrieval code type.

- The status is updated at the end of the process.

Return value

Returns an error number.

Error:

Not descriptor	No device descriptor.
----------------	-----------------------

Reference

CDAQDA100::switchCode

switchComputeDA100

Syntax

```
int switchComputeDA100(DAQDA100 daqda100, int iCompute);
```

Declaration**Visual Basic**

```
Public Declare Function switchComputeDA100 Lib
"DAQDA100"(ByVal daqda100 As Long, ByVal iReportRun As Long)
As Long
```

Visual Basic.NET

```
Public Declare Ansi Function switchComputeDA100 Lib
"DAQDA100"(ByVal daqda100 As Integer, ByVal iReportRun As
Integer) As Integer
```

C#

```
[DllImport("DAQDA100.dll" CharSet=CharSet.Auto,
EntryPoint="switchComputeDA100")]
public static extern int switchComputeDA100(int daqda100, int
iReportRun);
```

Parameters

daqdarwin	Specify the device descriptor.
iCompute	Specify the computation.

Description

Starts/stops computation.

- Valid with the computation function.
- The status is updated at the end of the process.
- This function executes the EX command of the DARWIN communication function.

Return value

Returns an error number.

Error:

Not descriptor	No device descriptor.
----------------	-----------------------

Reference

CDAQDA100::switchCompute

switchModeDA100

Syntax

```
int switchModeDA100(DAQDA100 daqda100, int iMode);
```

Declaration

Visual Basic

```
Public Declare Function switchModeDA100 Lib "DAQDA100" (ByVal daqda100 As Long, ByVal iMode As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function switchModeDA100 Lib "DAQDA100" (ByVal daqda100 As Integer, ByVal iMode As Integer) As Integer
```

C#

```
[DllImport("DAQDA100.dll" CharSet=CharSet.Auto, EntryPoint="switchModeDA100")]  
public static extern int switchModeDA100(int daqda100, int iMode);
```

Parameters

daqda100	Specify the device descriptor.
iMode	Specify the mode.

Description

Switches to the specified mode.

- Updates the stored channel information data when switching to the operation mode.
- The status is updated at the end of the process.
- This function executes the DS command of the DARWIN communication function.

Return value

Returns an error number.

Error:

Not descriptor	No device descriptor.
----------------	-----------------------

Reference

CDAQDA100::switchMode

talkCalibrationChDataDA100

Syntax

```
int talkCalibrationChDataDA100(DAQDA100 daqda100, int chType,  
int chNo);
```

Declaration

Visual Basic

```
Public Declare Function talkCalibrationChDataDA100 Lib  
"DAQDA100"(ByVal daqda100 As Long, ByVal chType As Long, ByVal  
chNo As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function talkCalibrationChDataDA100 Lib  
"DAQDA100"(ByVal daqda100 As Integer, ByVal chType As Integer,  
ByVal chNo As Integer) As Integer
```

C#

```
[DllImport("DAQDA100.dll" CharSet=CharSet.Auto,  
EntryPoint="talkCalibrationChDataDA100")]  
public static extern int talkCalibrationChDataDA100(int  
daqda100, int chType, int chNo);
```

Parameters

daqda100	Specify the device descriptor.
chType	Specify the channel type.
chNo	Specify the channel number.

Description

Executes declaration of the retrieval of the setup data of A/D calibration mode on the specified channel (specified by channel type and number).

- The operation mode must be switched to A/D calibration mode in advance.
- If the channel type is set to the constant for "Specify all measurement channel types," all subunits are processed.
- If the channel number is set to the constant for "Specify all channel numbers," all channels within the channel type are processed.
- This function executes the TS,LF command of the DARWIN communication function.
- After executing this function, use the getSetDataByLineDA100 function to retrieve the data by line.

Return value

Returns an error number.

Error:

Not descriptor	No device descriptor.
----------------	-----------------------

Reference

CDAQDA100::talkCalibrationChData

talkCalibrationDataDA100

Syntax

```
int talkCalibrationDataDA100(DAQDA100 daqda100, int
startChType, int startChNo, int endChType, int endChNo);
```

Declaration

Visual Basic

```
Public Declare Function talkCalibrationDataDA100 Lib
"DAQDA100"(ByVal daqda100 As Long, ByVal startChType As Long,
ByVal startChNo As Long, ByVal endChType As Long, ByVal
endChNo As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function talkCalibrationDataDA100 Lib
"DAQDA100"(ByVal daqda100 As Integer, ByVal startChType As
Integer, ByVal startChNo As Integer, ByVal endChType As
Integer, ByVal endChNo As Integer) As Integer
```

C#

```
[DllImport("DAQDA100.dll" CharSet=CharSet.Auto,
EntryPoint="talkCalibrationDataDA100")]
public static extern int talkCalibrationDataDA100(int
daqda100, int startChType, int startChNo, int endChType, int
endChNo);
```

Parameters

daqda100	Specify the device descriptor.
startChType	Specify the start channel type.
startChNo	Specify the start channel number.
endChType	Specify the end channel type.
endChNo	Specify the end channel number.

Description

Executes the declaration for retrieving setup data of A/D calibration mode from the start channel (start channel type and start channel number) to the end channel (end channel type and end channel number).

- The operation mode must be switched to A/D calibration mode in advance.
- This function executes the TS,LF command of the DARWIN communication function.
- After executing this function, use the getSetDataByLineDA100 function to retrieve the data by line.

Return value

Returns an error number.

Error:

Not descriptor	No device descriptor.
----------------	-----------------------

Reference

CDAQDA100::talkCalibrationData

talkOperationChDataDA100

Syntax

```
int talkOperationChDataDA100(DAQDA100 daqda100, int chType,  
int chNo);
```

Declaration

Visual Basic

```
Public Declare Function talkOperationChDataDA100 Lib  
"DAQDA100"(ByVal daqda100 As Long, ByVal chType As Long, ByVal  
chNo As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function talkOperationChDataDA100 Lib  
"DAQDA100"(ByVal daqda100 As Integer, ByVal chType As Integer,  
ByVal chNo As Integer) As Integer
```

C#

```
[DllImport("DAQDA100.dll" CharSet=CharSet.Auto,  
EntryPoint="talkOperationChDataDA100")]  
public static extern int talkOperationChDataDA100(int  
daqda100, int chType, int chNo);
```

Parameters

daqda100	Specify the device descriptor.
chType	Specify the channel type.
chNo	Specify the channel number.

Description

Executes declaration of the retrieval of the setup data of operation mode on the specified channel (specified by channel type and number).

- If the channel type is set to the constant for "Specify all measurement channel types,"all subunits are processed.
- If the channel number is set to the constant for "Specify all channel numbers,"all channels within the channel type are processed.
- This function executes the TS and LF commands of the DARWIN communication function.
- After executing this function, use the getSetDataByLineDA100 function to retrieve the data by line.

Return value

Returns an error number.

Error:

Not descriptor	No device descriptor.
----------------	-----------------------

Reference

CDAQDA100::talkOperationChData

talkOperationDataDA100

Syntax

```
int talkOperationDataDA100(DAQDA100 daqda100, int startChType,
int startChNo, int endChType, int endChNo);
```

Declaration

Visual Basic

```
Public Declare Function talkOperationDataDA100 Lib
"DAQDA100"(ByVal daqda100 As Long, ByVal startChType As Long,
ByVal startChNo As Long, ByVal endChType As Long, ByVal
endChNo As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function talkOperationDataDA100 Lib
"DAQDA100"(ByVal daqda100 As Integer, ByVal startChType As
Integer, ByVal startChNo As Integer, ByVal endChType As
Integer, ByVal endChNo As Integer) As Integer
```

C#

```
[DllImport("DAQDA100.dll" CharSet=CharSet.Auto,
EntryPoint="talkOperationDataDA100")]
public static extern int talkOperationDataDA100(int daqda100,
int startChType, int startChNo, int endChType, int endChNo);
```

Parameters

daqda100	Specify the device descriptor.
startChType	Specify the start channel type.
startChNo	Specify the start channel number.
endChType	Specify the end channel type.
endChNo	Specify the end channel number.

Description

Executes the declaration for retrieving setup data of the operation mode from the start channel (start channel type and start channel number) to the end channel (end channel type and end channel number).

- This function executes the TS,LF command of the DARWIN communication function.
- After executing this function, use the getSetDataByLineDA100 function to retrieve the data by line.

Return value

Returns an error number.

Error:

Not descriptor	No device descriptor.
----------------	-----------------------

Reference

CDAQDA100::talkOperationData

talkSetupChDataDA100

Syntax

```
int talkSetupChDataDA100(DAQDA100 daqda100, int chType, int chNo);
```

Declaration

Visual Basic

```
Public Declare Function talkSetupChDataDA100 Lib "DAQDA100" (ByVal daqda100 As Long, ByVal chType As Long, ByVal chNo As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function talkSetupChDataDA100 Lib "DAQDA100" (ByVal daqda100 As Integer, ByVal chType As Integer, ByVal chNo As Integer) As Integer
```

C#

```
[DllImport("DAQDA100.dll" CharSet=CharSet.Auto, EntryPoint="talkSetupChDataDA100")]  
public static extern int talkSetupChDataDA100(int daqda100, int chType, int chNo);
```

Parameters

daqda100	Specify the device descriptor.
chType	Specify the channel type.
chNo	Specify the channel number.

Description

Executes declaration of the retrieval of the setup data of the setup mode of the specified channel (specified by channel type and number).

- If the channel type is set to the constant for "Specify all measurement channel types," all subunits are processed.
- If the channel number is set to the constant for "Specify all channel numbers," all channels within the channel type are processed.
- This function executes the TS,LF command of the DARWIN communication function.
- After executing this function, use the getSetDataByLineDA100 function to retrieve the data by line.

Return value

Returns an error number.

Error:

Not descriptor	No device descriptor.
----------------	-----------------------

Reference

CDAQDA100::talkSetupChData

talkSetupDataDA100

Syntax

```
int talkSetupDataDA100(DAQDA100 daqda100, int startChType, int
startChNo, int endChType, int endChNo);
```

Declaration

Visual Basic

```
Public Declare Function talkSetupDataDA100 Lib
"DAQDA100"(ByVal daqda100 As Long, ByVal startChType As Long,
ByVal startChNo As Long, ByVal endChType As Long, ByVal
endChNo As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function talkSetupDataDA100 Lib
"DAQDA100"(ByVal daqda100 As Integer, ByVal startChType As
Integer, ByVal startChNo As Integer, ByVal endChType As
Integer, ByVal endChNo As Integer) As Integer
```

C#

```
[DllImport("DAQDA100.dll" CharSet=CharSet.Auto,
EntryPoint="talkSetupDataDA100")]
public static extern int talkSetupDataDA100(int daqda100, int
startChType, int startChNo, int endChType, int endChNo);
```

Parameters

daqda100	Specify the device descriptor.
startChType	Specify the start channel type.
startChNo	Specify the start channel number.
endChType	Specify the end channel type.
endChNo	Specify the end channel number.

Description

Executes the declaration for retrieving setup data of the setup mode from the start channel (start channel type and start channel number) to the end channel (end channel type and end channel number).

- This function executes the TS,LF command of the DARWIN communication function.
- After executing this function, use the getSetDataByLineDA100 function to retrieve the data by line.

Return value

Returns an error number.

Error:

Not descriptor	No device descriptor.
----------------	-----------------------

Reference

CDAQDA100::talkSetupData

updateReportStatusDA100

Syntax

```
int updateReportStatusDA100(DAQDA100 daqda100);
```

Declaration

Visual Basic

```
Public Declare Function updateReportStatusDA100 Lib  
"DAQDA100"(ByVal daqda100 As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function updateReportStatusDA100 Lib  
"DAQDA100"(ByVal daqda100 As Integer) As Integer
```

C#

```
[DllImport("DAQDA100.dll" CharSet=CharSet.Auto,  
EntryPoint="updateReportStatusDA100")]  
public static extern int updateReportStatusDA100(int  
daqda100);
```

Parameters

daqda100 Specify the device descriptor.

Description

Gets the report status.

- Valid with the report option.
- This function executes the TS,RF command of the DARWIN communication function.

Return value

Returns an error number.

Error:

Not descriptor No device descriptor.

Reference

CDAQDA100::updateReportStatus

updateStatusDA100

Syntax

```
int updateStatusDA100(DAQDA100 daqda100);
```

Declaration

Visual Basic

```
Public Declare Function updateStatusDA100 Lib "DAQDA100" (ByVal daqda100 As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function updateStatusDA100 Lib "DAQDA100" (ByVal daqda100 As Integer) As Integer
```

C#

```
[DllImport("DAQDA100.dll" CharSet=CharSet.Auto, EntryPoint="updateStatusDA100")]  
public static extern int updateStatusDA100(int daqda100);
```

Parameters

daqda100 Specify the device descriptor.

Description

Sends the status byte output command (ESC S) and receives the status bytes.

Return value

Returns an error number.

Error:

Not descriptor No device descriptor.

Reference

CDAQDA100::updateStatus

updateSystemConfigDA100

Syntax

```
int updateSystemConfigDA100(DAQDA100 daqda100);
```

Declaration

Visual Basic

```
Public Declare Function updateSystemConfigDA100 Lib  
"DAQDA100"(ByVal daqda100 As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function updateSystemConfigDA100 Lib  
"DAQDA100"(ByVal daqda100 As Integer) As Integer
```

C#

```
[DllImport("DAQDA100.dll" CharSet=CharSet.Auto,  
EntryPoint="updateSystemConfigDA100")]  
public static extern int updateSystemConfigDA100(int  
daqda100);
```

Parameters

daqda100 Specify the device descriptor.

Description

Gets the system configuration data.

- This function executes the TS,CF command of the DARWIN communication function.

Return value

Returns an error number.

Error:

Not descriptor No device descriptor.

Reference

CDAQDA100::updateSystemConfig

24.2 Details of Functions - DARWIN (Visual C/Visual Basic/Visual Basic.NET/C#) - Retrieval Functions

This section describes the DARWIN functions that are used in Visual C, Visual Basic, Visual Basic.NET, and C#. The functions are listed in alphabetical order by the function name.

For details on constants and types, see chapter 25.

For DARWIN terminology, see appendix 2.

alarmMaxLengthDA100

Syntax

```
int alarmMaxLengthDA100(void);
```

Declaration

Visual Basic

```
Public Declare Function alarmMaxLengthDA100 Lib "DAQDA100"()  
As Long
```

Visual Basic.NET

```
Public Declare Ansi Function alarmMaxLengthDA100 Lib  
"DAQDA100"() As Integer
```

C#

```
[DllImport("DAQDA100.dll" CharSet=CharSet.Auto,  
EntryPoint="alarmMaxLengthDA100")]  
public static extern int alarmMaxLengthDA100();
```

Description

Gets the maximum length of the alarm type.

- The return value does not include the terminator.

Return value

Returns the length of the string.

Reference

CDAQDARWINDataInfo::getMaxLenAlarmName

alarmTypeDA100

Syntax

```
int alarmTypeDA100(DAQDA100 daqda100, int chType, int chNo,
int levelNo);
```

Declaration**Visual Basic**

```
Public Declare Function alarmTypeDA100 Lib "DAQDA100" (ByVal
daqda100 As Long, ByVal chType As Long, ByVal chNo As Long,
ByVal levelNo As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function alarmTypeDA100 Lib
"DAQDA100" (ByVal daqda100 As Integer, ByVal chType As Integer,
ByVal chNo As Integer, ByVal levelNo As Integer) As Integer
```

C#

```
[DllImport("DAQDA100.dll" CharSet=CharSet.Auto,
EntryPoint="alarmTypeDA100")]
public static extern int alarmTypeDA100(int daqda100, int
chType, int chNo, int levelNo);
```

Parameters

daqda100	Specify the device descriptor.
chType	Specify the channel type.
chNo	Specify the channel number.
levelNo	Specify the alarm level.

Description

Gets the specified channel (channel type and number) and alarm level alarm type from the stored measured data.

- If it does not exist, "No alarm" is returned.

Return value

Returns the alarm type.

Reference

```
CDAQDA100::getClassDataBuffer
CDAQDARWINDataBuffer::getClassDARWINDataInfo
CDAQDARWINDataInfo::getAlarm
```

channelPointDA100

Syntax

```
int channelPointDA100(DAQDA100 daqda100, int chType, int chNo);
```

Declaration

Visual Basic

```
Public Declare Function channelPointDA100 Lib "DAQDA100" (ByVal daqda100 As Long, ByVal chType As Long, ByVal chNo As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function channelPointDA100 Lib "DAQDA100" (ByVal daqda100 As Integer, ByVal chType As Integer, ByVal chNo As Integer) As Integer
```

C#

```
[DllImport("DAQDA100.dll" CharSet=CharSet.Auto, EntryPoint="channelPointDA100")]  
public static extern int channelPointDA100(int daqda100, int chType, int chNo);
```

Parameters

daqda100	Specify the device descriptor.
chType	Specify the channel type.
chNo	Specify the channel number.

Description

Gets the decimal point position of the specified channel (channel type and number) from the stored channel information data.

- Returns 0 if it does not exist.

Return value

Returns the decimal point position.

Reference

```
CDAQDA100::getClassDataBuffer  
CDAQDARWINChInfo::getPoint  
CDAQDARWINDataBuffer::getClassDARWINChInfo
```

channelStatusDA100

Syntax

```
int channelStatusDA100(DAQDA100 daqda100, int chType, int
chNo);
```

Declaration

Visual Basic

```
Public Declare Function channelStatusDA100 Lib
"DAQDA100"(ByVal daqda100 As Long, ByVal chType As Long, ByVal
chNo As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function channelStatusDA100 Lib
"DAQDA100"(ByVal daqda100 As Integer, ByVal chType As Integer,
ByVal chNo As Integer) As Integer
```

C#

```
[DllImport("DAQDA100.dll" CharSet=CharSet.Auto,
EntryPoint="channelStatusDA100")]
public static extern int channelStatusDA100(int daqda100, int
chType, int chNo);
```

Parameters

daqda100	Specify the device descriptor.
chType	Specify the channel type.
chNo	Specify the channel number.

Description

Gets the channel status of the specified channel (channel type and number) from the stored channel information data.

- If it does not exist, "Unknown" is returned.

Return value

Returns the channel status.

Reference

```
CDAQDA100::getClassDataBuffer
CDAQDARWINChInfo::getChStatus
CDAQDARWINDataBuffer::getClassDARWINChInfo
```

dataAlarmDA100

Syntax

```
int dataAlarmDA100(DAQDA100 daqda100, int chType, int chNo,
int levelNo);
```

Declaration

Visual Basic

```
Public Declare Function dataAlarmDA100 Lib "DAQDA100" (ByVal
daqda100 As Long, ByVal chType As Long, ByVal chNo As Long,
ByVal levelNo As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function dataAlarmDA100 Lib
"DAQDA100" (ByVal daqda100 As Integer, ByVal chType As Integer,
ByVal chNo As Integer, ByVal levelNo As Integer) As Integer
```

C#

```
[DllImport("DAQDA100.dll" CharSet=CharSet.Auto,
EntryPoint="dataAlarmDA100")]
public static extern int dataAlarmDA100(int daqda100, int
chType, int chNo, int levelNo);
```

Parameters

daqda100	Specify the device descriptor.
chType	Specify the channel type.
chNo	Specify the channel number.
levelNo	Specify the alarm level.

Description

Gets the Boolean for the valid/invalid value of the alarm corresponding to the alarm level of the specified channel (channel type and number) from the stored measured data.

- If it does not exist, returns Invalid Value.

Return value

Returns a Boolean value.

Reference

```
CDAQDA100::getClassDataBuffer
CDAQDARWINDataBuffer::isAlarm
```

dataDayDA100

Syntax

```
int dataDayDA100(DAQDA100 daqda100, int chType, int chNo);
```

Declaration**Visual Basic**

```
Public Declare Function dataDayDA100 Lib "DAQDA100" (ByVal daqda100 As Long, ByVal chType As Long, ByVal chNo As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function dataDayDA100 Lib "DAQDA100" (ByVal daqda100 As Integer, ByVal chType As Integer, ByVal chNo As Integer) As Integer
```

C#

```
[DllImport("DAQDA100.dll" CharSet=CharSet.Auto, EntryPoint="dataDayDA100")]
public static extern int dataDayDA100(int daqda100, int chType, int chNo);
```

Parameters

daqda100	Specify the device descriptor.
chType	Specify the channel type.
chNo	Specify the channel number.

Description

Gets the day of the specified channel (channel type and number) from the stored time information data.

- The day is a number from 1 to 31.
- Returns 0 if it does not exist.

Return value

Returns the day value.

Reference

```
CDAQDA100::getClassDataBuffer
CDAQDARWINDataBuffer::getClassDARWINDateTime
CDAQDARWINDateTime::getDay
```

dataDoubleValueDA100

Syntax

```
double dataDoubleValueDA100(DAQDA100 daqda100, int chType, int chNo);
```

Declaration

Visual Basic

```
Public Declare Function dataDoubleValueDA100 Lib "DAQDA100" (ByVal daqda100 As Long, ByVal chType As Long, ByVal chNo As Long) As Double
```

Visual Basic.NET

```
Public Declare Ansi Function dataDoubleValueDA100 Lib "DAQDA100" (ByVal daqda100 As Integer, ByVal chType As Integer, ByVal chNo As Integer) As Double
```

C#

```
[DllImport("DAQDA100.dll", CharSet=CharSet.Auto, EntryPoint="dataDoubleValueDA100")]  
public static extern double dataDoubleValueDA100(int daqda100, int chType, int chNo);
```

Parameters

daqda100	Specify the device descriptor.
chType	Specify the channel type.
chNo	Specify the channel number.

Description

Gets the measured value of the specified channel (channel type and number) from the stored measured data.

- Returns 0.0 if it does not exist.

Return value

Returns the measured value as a double-precision floating number.

Reference

```
CDAQDA100::getClassDataBuffer  
CDAQDARWINDataBuffer::getClassDARWINDataInfo  
CDAQDARWINDataInfo::getDoubleValue
```

dataHourDA100

Syntax

```
int dataHourDA100(DAQDA100 daqda100, int chType, int chNo);
```

Declaration**Visual Basic**

```
Public Declare Function dataHourDA100 Lib "DAQDA100" (ByVal daqda100 As Long, ByVal chType As Long, ByVal chNo As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function dataHourDA100 Lib "DAQDA100" (ByVal daqda100 As Integer, ByVal chType As Integer, ByVal chNo As Integer) As Integer
```

C#

```
[DllImport("DAQDA100.dll" CharSet=CharSet.Auto, EntryPoint="dataHourDA100")]
public static extern int dataHourDA100(int daqda100, int chType, int chNo);
```

Parameters

daqda100	Specify the device descriptor.
chType	Specify the channel type.
chNo	Specify the channel number.

Description

Gets the hour of the specified channel (channel type and number) from the stored time information data.

- The hour is a number from 0 to 23.
- Returns 0 if it does not exist.

Return value

Returns the hour value.

Reference

```
CDAQDA100::getClassDataBuffer
CDAQDARWINDataBuffer::getClassDARWINDateTime
CDAQDARWINDateTime::getHour
```

dataMinuteDA100

Syntax

```
int dataMinuteDA100(DAQDA100 daqda100,int chType,int chNo);
```

Declaration

Visual Basic

```
Public Declare Function dataMinuteDA100 Lib "DAQDA100"(ByVal daqda100 As Long, ByVal chType As Long, ByVal chNo As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function dataMinuteDA100 Lib "DAQDA100"(ByVal daqda100 As Integer, ByVal chType As Integer, ByVal chNo As Integer) As Integer
```

C#

```
[DllImport("DAQDA100.dll" CharSet=CharSet.Auto, EntryPoint="dataMinuteDA100")]  
public static extern int dataMinuteDA100(int daqda100, int chType, int chNo);
```

Parameters

daqda100	Specify the device descriptor.
chType	Specify the channel type.
chNo	Specify the channel number.

Description

Gets the minutes of the specified channel (channel type and number) from the stored time information data.

- The minute is a number from 0 to 59.
- Returns 0 if it does not exist.

Return value

Returns the minute value.

Reference

```
CDAQDA100::getClassDataBuffer  
CDAQDARWINDataBuffer::getClassDARWINDateTime  
CDAQDARWINDateTime::getMinute
```

dataMonthDA100

Syntax

```
int dataMonthDA100(DAQDA100 daqda100, int chType, int chNo);
```

Declaration

Visual Basic

```
Public Declare Function dataMonthDA100 Lib "DAQDA100" (ByVal daqda100 As Long, ByVal chType As Long, ByVal chNo As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function dataMonthDA100 Lib "DAQDA100" (ByVal daqda100 As Integer, ByVal chType As Integer, ByVal chNo As Integer) As Integer
```

C#

```
[DllImport("DAQDA100.dll" CharSet=CharSet.Auto, EntryPoint="dataMonthDA100")]
public static extern int dataMonthDA100(int daqda100, int chType, int chNo);
```

Parameters

daqda100	Specify the device descriptor.
chType	Specify the channel type.
chNo	Specify the channel number.

Description

Gets the month of the specified channel (channel type and number) from the stored time information data.

- The month is a number from 1 to 12.
- Returns 0 if it does not exist.

Return value

Returns the month value.

Reference

```
CDAQDA100::getClassDataBuffer
CDAQDARWINDataBuffer::getClassDARWINDateTime
CDAQDARWINDateTime::getMonth
```

dataSecondDA100

Syntax

```
int dataSecondDA100(DAQDA100 daqda100, int chType, int chNo);
```

Declaration

Visual Basic

```
Public Declare Function dataSecondDA100 Lib "DAQDA100" (ByVal daqda100 As Long, ByVal chType As Long, ByVal chNo As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function dataSecondDA100 Lib "DAQDA100" (ByVal daqda100 As Integer, ByVal chType As Integer, ByVal chNo As Integer) As Integer
```

C#

```
[DllImport("DAQDA100.dll" CharSet=CharSet.Auto, EntryPoint="dataSecondDA100")]  
public static extern int dataSecondDA100(int daqda100, int chType, int chNo);
```

Parameters

daqda100	Specify the device descriptor.
chType	Specify the channel type.
chNo	Specify the channel number.

Description

Gets the seconds of the specified channel (channel type and number) from the stored time information data.

- The second is a number from 0 to 59.
- Returns 0 if it does not exist.

Return value

Returns the seconds value.

Reference

```
CDAQDA100::getClassDataBuffer  
CDAQDARWINDataBuffer::getClassDARWINDateTime  
CDAQDARWINDateTime::getSecond
```


dataStatusDA100

Syntax

```
int dataStatusDA100(DAQDA100 daqda100, int chType, int chNo);
```

Declaration

Visual Basic

```
Public Declare Function dataStatusDA100 Lib "DAQDA100" (ByVal daqda100 As Long, ByVal chType As Long, ByVal chNo As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function dataStatusDA100 Lib "DAQDA100" (ByVal daqda100 As Integer, ByVal chType As Integer, ByVal chNo As Integer) As Integer
```

C#

```
[DllImport("DAQDA100.dll" CharSet=CharSet.Auto, EntryPoint="dataStatusDA100")]
public static extern int dataStatusDA100(int daqda100, int chType, int chNo);
```

Parameters

daqda100	Specify the device descriptor.
chType	Specify the channel type.
chNo	Specify the channel number.

Description

Gets the data status value of the specified channel (channel type and number) from the stored measured data.

- If it does not exist, "Unknown" is returned.

Return value

Returns the data status value.

Reference

```
CDAQDA100::getClassDataBuffer
CDAQDARWINDataBuffer::getClassDARWINDataInfo
CDAQDARWINDataInfo::getStatus
```

dataStringValueDA100

Syntax

```
int dataStringValueDA100(DAQDA100 daqda100, int chType, int chNo, char * strValue, int lenValue);
```

Declaration

Visual Basic

```
Public Declare Function dataStringValueDA100 Lib "DAQDA100"(ByVal daqda100 As Long, ByVal chType As Long, ByVal chNo As Long, ByVal strValue As String, ByVal lenValue As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function dataStringValueDA100 Lib "DAQDA100"(ByVal daqda100 As Integer, ByVal chType As Integer, ByVal chNo As Integer, ByVal strValue As String, ByVal lenValue As Integer) As Integer
```

C#

```
[DllImport("DAQDA100.dll" CharSet=CharSet.Auto, EntryPoint="dataStringValueDA100")]  
public static extern int dataStringValueDA100(int daqda100, int chType, int chNo, byte[] strValue, int lenValue);
```

Parameters

daqda100	Specify the device descriptor.
chType	Specify the channel type.
chNo	Specify the channel number.
strValue	Specify the field where the string is to be stored.
lenValue	Specify the byte size of the field where the string is to be stored.

Description

Gets the measured value of the specified channel (channel type and number) from the stored measured data.

- Converts into a string and stores it in the specified field.
- The string stored in the field includes the terminator (NULL).
- The return value is the length of the actual string. The return value does not include the terminator.
- Returns 0 if it does not exist.
- The strings that can be stored are, in general, ASCII strings.

Return value

Returns the length of the string.

Reference

```
CDAQDA100::getClassDataBuffer  
CDAQDARWINDataBuffer::getClassDARWINDataInfo  
CDAQDARWINDataInfo::getStringValue
```

dataValueDA100

Syntax

```
int dataValueDA100(DAQDA100 daqda100, int chType, int chNo);
```

Declaration**Visual Basic**

```
Public Declare Function dataValueDA100 Lib "DAQDA100" (ByVal daqda100 As Long, ByVal chType As Long, ByVal chNo As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function dataValueDA100 Lib "DAQDA100" (ByVal daqda100 As Integer, ByVal chType As Integer, ByVal chNo As Integer) As Integer
```

C#

```
[DllImport("DAQDA100.dll" CharSet=CharSet.Auto, EntryPoint="dataValueDA100")]
public static extern int dataValueDA100(int daqda100, int chType, int chNo);
```

Parameters

daqda100	Specify the device descriptor.
chType	Specify the channel type.
chNo	Specify the channel number.

Description

Gets the data value of the specified channel (channel type and number) from the stored measured data.

- Returns 0 if it does not exist.

Return value

Returns the data value.

Reference

```
CDAQDA100::getClassDataBuffer
CDAQDARWINDataBuffer::getClassDARWINDataInfo
CDAQDARWINDataInfo::getValue
```

dataYearDA100

Syntax

```
int dataYearDA100(DAQDA100 daqda100, int chType, int chNo);
```

Declaration

Visual Basic

```
Public Declare Function dataYearDA100 Lib "DAQDA100" (ByVal daqda100 As Long, ByVal chType As Long, ByVal chNo As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function dataYearDA100 Lib "DAQDA100" (ByVal daqda100 As Integer, ByVal chType As Integer, ByVal chNo As Integer) As Integer
```

C#

```
[DllImport("DAQDA100.dll" CharSet=CharSet.Auto, EntryPoint="dataYearDA100")]  
public static extern int dataYearDA100(int daqda100, int chType, int chNo);
```

Parameters

daqda100	Specify the device descriptor.
chType	Specify the channel type.
chNo	Specify the channel number.

Description

Gets the year of the specified channel (channel type and number) from the stored time information data.

- The year is a 4-digit number.
- Returns 0 if it does not exist.

Return value

Returns the year value.

Reference

```
CDAQDA100::getClassDataBuffer  
CDAQDARWINDataBuffer::getClassDARWINDateTime  
CDAQDARWINDateTime::getFullYear
```

errorMaxLengthDA100

Syntax

```
int errorMaxLengthDA100(void);
```

Declaration

Visual Basic

```
Public Declare Function errorMaxLengthDA100 Lib "DAQDA100"()  
As Long
```

Visual Basic.NET

```
Public Declare Ansi Function errorMaxLengthDA100 Lib  
"DAQDA100"() As Integer
```

C#

```
[DllImport("DAQDA100.dll" CharSet=CharSet.Auto,  
EntryPoint="errorMaxLengthDA100")]  
public static extern int errorMaxLengthDA100();
```

Description

Gets the maximum length of the error message string.

- The return value does not include the terminator.

Return value

Returns the length of the string.

Reference

CDAQDA100::getMaxLenErrorMessage

getAlarmNameDA100 [Visual C only]

Syntax

```
const char * getAlarmNameDA100(int iAlarmType);
```

Parameters

iAlarmType Specify the alarm type.

Description

Gets the string corresponding to the specified alarm type.

- If it does not exist, returns the pointer to the string corresponding to “No alarm.”

Return value

Returns a pointer to the string.

Reference

CDAQDARWINDataInfo::getAlarmName

getChannelUnitDA100 [Visual C only]

Syntax

```
const char * getChannelUnitDA100(DAQDA100 daqda100, int  
chType, int chNo);
```

Parameters

daqda100	Specify the device descriptor.
chType	Specify the channel type.
chNo	Specify the channel number.

Description

Gets the unit name of the specified channel (channel type and number) from the stored channel information data.

- Returns NULL if it does not exist.

Return value

Returns a pointer to the string.

Reference

```
CDAQDA100::getClassDataBuffer  
CDAQDARWINChInfo::getUnit  
CDAQDARWINDataBuffer::getClassDARWINChInfo
```

getErrorMessageDA100 [Visual C only]

Syntax

```
const char * getErrorMessageDA100(int errorCode);
```

Parameters

errorCode Specify the error number.

Description

Gets the error message string corresponding to the specified error number.

- Returns a pointer to the string [Unknown] if it does not exist.

Return value

Returns a pointer to the string.

Reference

CDAQDA100::getErrorMessage

getModuleNameDA100 [Visual C only]

Syntax

```
const char * getModuleNameDA100(DAQDA100 daqda100, int unitNo,  
int slotNo);
```

Parameters

daqda100	Specify the device descriptor.
unitNo	Specify the unit number.
slotNo	Specify the slot number.

Description

Gets the module name in the position indicated by the specified unit number and slot number from the stored system configuration data.

- Returns NULL if it does not exist.

Return value

Returns a pointer to the string.

Reference

CDAQDA100::getClassSysInfo
CDAQDARWINSysInfo::getModuleName

moduleCodeDA100

Syntax

```
int moduleCodeDA100(DAQDA100 daqda100, int unitNo, int slotNo);
```

Declaration

Visual Basic

```
Public Declare Function moduleCodeDA100 Lib "DAQDA100" (ByVal daqda100 As Long, ByVal unitNo As Long, ByVal slotNo As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function moduleCodeDA100 Lib "DAQDA100" (ByVal daqda100 As Integer, ByVal unitNo As Integer, ByVal slotNo As Integer) As Integer
```

C#

```
[DllImport("DAQDA100.dll" CharSet=CharSet.Auto, EntryPoint="moduleCodeDA100")]  
public static extern int moduleCodeDA100(int daqda100, int unitNo, int slotNo);
```

Parameters

daqda100	Specify the device descriptor.
unitNo	Specify the unit number.
slotNo	Specify the slot number.

Description

Gets the internal code in the position indicated by the specified unit number and slot number from the stored system configuration data.

- Returns 0 if it does not exist.

Return value

Returns the internal code.

Reference

```
CDAQDA100::getClassSysInfo  
CDAQDARWINSysInfo::getModuleCode
```

revisionAPIDA100

Syntax

```
const int revisionAPIDA100(void);
```

Declaration

Visual Basic

```
Public Declare Function revisionAPIDA100 Lib "DAQDA100"() As Long
```

Visual Basic.NET

```
Public Declare Ansi Function revisionAPIDA100 Lib "DAQDA100"() As Integer
```

C#

```
[DllImport("DAQDA100.dll" CharSet=CharSet.Auto, EntryPoint="revisionAPIDA100")]  
public static extern int revisionAPIDA100();
```

Description

Gets the revision number of this API.

Return value

Returns the revision number.

Reference

CDAQDA100::getRevisionAPI

statusByteDA100

Syntax

```
int statusByteDA100(DAQDA100 daqda100);
```

Declaration

Visual Basic

```
Public Declare Function statusByteDA100 Lib "DAQDA100" (ByVal daqda100 As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function statusByteDA100 Lib "DAQDA100" (ByVal daqda100 As Integer) As Integer
```

C#

```
[DllImport("DAQDA100.dll" CharSet=CharSet.Auto, EntryPoint="statusByteDA100")]  
public static extern int statusByteDA100(int daqda100);
```

Parameters

daqda100 Specify the device descriptor.

Description

Gets the stored status byte.

- If it does not exist, returns "the value when all status bytes are invalid."

Return value

Returns the status byte.

Reference

CDAQDA100::getBytes

statusCodeDA100

Syntax

```
int statusCodeDA100(DAQDA100 daqda100);
```

Declaration

Visual Basic

```
Public Declare Function statusCodeDA100 Lib "DAQDA100" (ByVal daqda100 As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function statusCodeDA100 Lib "DAQDA100" (ByVal daqda100 As Integer) As Integer
```

C#

```
[DllImport("DAQDA100.dll" CharSet=CharSet.Auto, EntryPoint="statusCodeDA100")]  
public static extern int statusCodeDA100(int daqda100);
```

Parameters

daqda100 Specify the device descriptor.

Description

Gets the stored retrieve code type.

- If it does not exist, "Binary code" is returned.

Return value

Returns the retrieve code type.

Reference

CDAQDA100::getCode

statusReportDA100

Syntax

```
int statusReportDA100(DAQDA100 daqda100);
```

Declaration

Visual Basic

```
Public Declare Function statusReportDA100 Lib "DAQDA100" (ByVal daqda100 As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function statusReportDA100 Lib "DAQDA100" (ByVal daqda100 As Integer) As Integer
```

C#

```
[DllImport("DAQDA100.dll" CharSet=CharSet.Auto, EntryPoint="statusReportDA100")]  
public static extern int statusReportDA100(int daqda100);
```

Parameters

daqda100 Specify the device descriptor.

Description

Gets the stored report status.

- If it does not exist, returns "All Invalid."

Return value

Returns the report status.

Reference

CDAQDA100::getReport

toAlarmNameDA100

Syntax

```
int toAlarmNameDA100(int iAlarmType, char * strAlarm, int lenAlarm);
```

Declaration

Visual Basic

```
Public Declare Function toAlarmNameDA100 Lib "DAQDA100" (ByVal iAlarmType As Long, ByVal strAlarm As String, ByVal lenAlarm As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function toAlarmNameDA100 Lib "DAQDA100" (ByVal iAlarmType As Integer, ByVal strAlarm As String, ByVal lenAlarm As Integer) As Integer
```

C#

```
[DllImport("DAQDA100.dll" CharSet=CharSet.Auto, EntryPoint="toAlarmNameDA100")]  
public static extern int toAlarmNameDA100(int iAlarmType, byte[] strAlarm, int lenAlarm);
```

Parameters

iAlarmType	Specify the alarm type.
strAlarm	Specify the field where the string is to be stored.
lenAlarm	Specify the byte size of the field where the string is to be stored.

Description

Stores the string corresponding to the specified alarm type to the specified field.

- The string stored in the field includes the terminator (NULL).
- The return value is the length of the actual string. The return value does not include the terminator.
- The strings that can be stored are, in general, ASCII strings.

Return value

Returns the length of the string.

Reference

getAlarmNameDA100

toChannelUnitDA100

Syntax

```
int toChannelUnitDA100(DAQDA100 daqda100, int chType, int chNo, char * strUnit, int lenUnit);
```

Declaration

Visual Basic

```
Public Declare Function toChannelUnitDA100 Lib "DAQDA100" (ByVal daqda100 As Long, ByVal chType As Long, ByVal chNo As Long, ByVal strUnit As String, ByVal lenUnit As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function toChannelUnitDA100 Lib "DAQDA100" (ByVal daqda100 As Integer, ByVal chType As Integer, ByVal chNo As Integer, ByVal strUnit As String, ByVal lenUnit As Integer) As Integer
```

C#

```
[DllImport("DAQDA100.dll" CharSet=CharSet.Auto, EntryPoint="toChannelUnitDA100")] public static extern int toChannelUnitDA100(int daqda100, int chType, int chNo, byte[] strUnit, int lenUnit);
```

Parameters

daqda100	Specify the device descriptor.
chType	Specify the channel type.
chNo	Specify the channel number.
strUnit	Specify the field where the string is to be stored.
lenUnit	Specify the byte size of the field where the string is to be stored.

Description

Gets the unit name of the specified channel (channel type and number) from the stored channel information data.

- Stores the string in the specified storage destination.
- The string stored in the field includes the terminator (NULL).
- The return value is the length of the actual string. The return value does not include the terminator.
- The strings that can be stored are, in general, ASCII strings.
- Returns 0 if it does not exist.

Return value

Returns the length of the actual string.

Reference

getChannelUnitDA100

toDoubleValueDA100

Syntax

```
double toDoubleValueDA100(int dataValue, int point);
```

Declaration

Visual Basic

```
Public Declare Function toDoubleValueDA100 Lib  
"DAQDA100"(ByVal dataValue As Long, ByVal point As Long) As  
Double
```

Visual Basic.NET

```
Public Declare Ansi Function toDoubleValueDA100 Lib  
"DAQDA100"(ByVal dataValue As Integer, ByVal point As Integer)  
As Double
```

C#

```
[DllImport("DAQDA100.dll" CharSet=CharSet.Auto,  
EntryPoint="toDoubleValueDA100")]  
public static extern double toDoubleValueDA100(int dataValue,  
int point);
```

Parameters

dataValue	Specify the data value.
point	Specify the decimal point position.

Description

Generates the measured value from the specified data value and decimal point position.

Return value

Returns the measured value as a double-precision floating number.

Reference

CDAQDARWINDataInfo::toDoubleValue

toErrorMessageDA100

Syntax

```
int toErrorMessageDA100(int errCode, char * errStr, int  
errLen);
```

Declaration

Visual Basic

```
Public Declare Function toErrorMessageDA100 Lib  
"DAQDA100"(ByVal errCode As Long, ByVal errStr As String,  
ByVal errLen As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function toErrorMessageDA100 Lib  
"DAQDA100"(ByVal errCode As Integer, ByVal errStr As String,  
ByVal errLen As Integer) As Integer
```

C#

```
[DllImport("DAQDA100.dll" CharSet=CharSet.Auto,  
EntryPoint="toErrorMessageDA100")]  
public static extern int toErrorMessageDA100(int errCode,  
byte[] errStr, int errLen);
```

Parameters

errCode	Specify the error number.
errStr	Specify the field where the string is to be stored.
errLen	Specify the byte size of the field where the string is to be stored.

Description

Stores the error message string corresponding to the error number to the specified field.

- The string stored in the field includes the terminator (NULL).
- The return value is the length of the actual string. The return value does not include the terminator.
- The strings that can be stored are, in general, ASCII strings.

Return value

Returns the length of the string.

Reference

getErrorMessageDA100

toModuleNameDA100

Syntax

```
int toModuleNameDA100(DAQDA100 daqda100, int unitNo, int
slotNo, char * strName, int lenName);
```

Declaration**Visual Basic**

```
Public Declare Function toModuleNameDA100 Lib "DAQDA100" (ByVal
daqda100 As Long, ByVal unitNo As Long, ByVal slotNo As Long,
ByVal strName As String, ByVal lenName As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function toModuleNameDA100 Lib
"DAQDA100" (ByVal daqda100 As Integer, ByVal unitNo As Integer,
ByVal slotNo As Integer, ByVal strName As String, ByVal
lenName As Integer) As Integer
```

C#

```
[DllImport("DAQDA100.dll" CharSet=CharSet.Auto,
EntryPoint="toModuleNameDA100")]
public static extern int toModuleNameDA100(int daqda100, int
unitNo, int slotNo, byte[] strName, int lenName);
```

Parameters

daqda100	Specify the device descriptor.
unitNo	Specify the unit number.
slotNo	Specify the slot number.
strName	Specify the field where the string is to be stored.
lenName	Specify the byte size of the field where the string is to be stored.

Description

Gets the module name in the position indicated by the specified unit number and slot number from the stored system configuration data.

- Stores the string in the specified storage destination.
- The string stored in the field includes the terminator (NULL).
- The return value is the length of the actual string. The return value does not include the terminator.
- Returns 0 if it does not exist.
- The strings that can be stored are, in general, ASCII strings.

Return value

Returns the length of the actual string.

Reference

getModuleNameDA100

toStringValueDA100

Syntax

```
int toStringValueDA100(int dataValue, int point, char *  
strValue, int lenValue);
```

Declaration

Visual Basic

```
Public Declare Function toStringValueDA100 Lib  
"DAQDA100"(ByVal dataValue As Long, ByVal point As Long, ByVal  
strValue As String, ByVal lenValue As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function toStringValueDA100 Lib  
"DAQDA100"(ByVal dataValue As Integer, ByVal point As Integer,  
ByVal strValue As String, ByVal lenValue As Integer) As  
Integer
```

C#

```
[DllImport("DAQDA100.dll" CharSet=CharSet.Auto,  
EntryPoint="toStringValueDA100")]  
public static extern int toStringValueDA100(int dataValue, int  
point, byte[] strValue, int lenValue);
```

Parameters

dataValue	Specify the data value.
point	Specify the decimal point position.
strValue	Specify the field where the string is to be stored.
lenValue	Specify the byte size of the field where the string is to be stored.

Description

Generates the measured value from the specified data value and decimal point position.

- Converts the generated measured value into a string and stores to the specified field.
- The string stored in the field includes the terminator (NULL).
- The return value is the length of the actual string. The return value does not include the terminator.
- The strings that can be stored are, in general, ASCII strings.

Return value

Returns the length of the string.

Reference

CDAQDARWINDataInfo::toStringValue

unitIntervalDA100

Syntax

```
double unitIntervalDA100(DAQDA100 daqda100);
```

Declaration

Visual Basic

```
Public Declare Function unitIntervalDA100 Lib "DAQDA100" (ByVal daqda100 As Long) As Double
```

Visual Basic.NET

```
Public Declare Ansi Function unitIntervalDA100 Lib "DAQDA100" (ByVal daqda100 As Integer) As Double
```

C#

```
[DllImport("DAQDA100.dll" CharSet=CharSet.Auto, EntryPoint="unitIntervalDA100")]  
public static extern double unitIntervalDA100(int daqda100);
```

Parameters

daqda100 Specify the device descriptor.

Description

Gets the measurement interval from the stored current system configuration data.

- Returns 0.0 if it does not exist.

Return value

Returns the scan interval.

Reference

CDAQDA100::getClassSysInfo

CDAQDARWINSysInfo::getInterval

unitValidDA100

Syntax

```
int unitValidDA100(DAQDA100 daqda100, int unitNo);
```

Declaration

Visual Basic

```
Public Declare Function unitValidDA100 Lib "DAQDA100" (ByVal daqda100 As Long, ByVal unitNo As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function unitValidDA100 Lib "DAQDA100" (ByVal daqda100 As Integer, ByVal unitNo As Integer) As Integer
```

C#

```
[DllImport("DAQDA100.dll" CharSet=CharSet.Auto, EntryPoint="unitValidDA100")]  
public static extern int unitValidDA100(int daqda100, int unitNo);
```

Parameters

daqda100	Specify the device descriptor.
unitNo	Specify the unit number.

Description

Gets valid/invalid of the specified unit number from the stored system configuration data as a Boolean value.

- If it does not exist, Invalid is returned.

Return value

Returns a Boolean value.

Reference

CDAQDA100::getClassSysInfo
CDAQDARWINSysInfo::isExist

versionAPIDA100

Syntax

```
const int versionAPIDA100(void);
```

Declaration

Visual Basic

```
Public Declare Function versionAPIDA100 Lib "DAQDA100"() As Long
```

Visual Basic.NET

```
Public Declare Ansi Function versionAPIDA100 Lib "DAQDA100"() As Integer
```

C#

```
[DllImport("DAQDA100.dll" CharSet=CharSet.Auto, EntryPoint="versionAPIDA100")]  
public static extern int versionAPIDA100();
```

Description

Gets the version number of this API.

Return value

Returns the version number.

Reference

CDAQDA100::getVersionAPI

24.3 Details of Functions for Instantaneous Value Loading - DARWIN (Visual C/Visual Basic/Visual Basic.NET/C#) Status Transition Functions

This section describes the DARWIN functions that are used in Visual C, Visual Basic, Visual Basic.NET, and C# when using the instantaneous value data loading port.

Most functions return an error number as a return value. Error number 0 is returned if there is no error.

closeDA100Reader

Syntax

```
int closeDA100Reader(DAQDA100READER daqda100);
```

Declaration

Visual Basic

```
Public Declare Function closeDA100Reader Lib "DAQDA100" (ByVal daqda100Reader As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function closeDA100Reader Lib "DAQDA100" (ByVal daqda100Reader As Integer) As Integer
```

C#

```
[DllImport("DAQDA100.dll" CharSet=CharSet.Auto, EntryPoint="closeDA100Reader")]  
public static extern int closeDA100Reader(int daqda100Reader);
```

Parameters

daqda100Reader Specify the device descriptor.

Description

Disconnects the communication using the specified device descriptor.

- When the communication is disconnected, the value of the device descriptor is meaningless.
- After disconnection, do not use the value of the device descriptor.

Return value

Returns an error number.

Error:

Not descriptor	No device descriptor.
----------------	-----------------------

Reference

CDAQDA100Reader::closes

mathInfoChDA100Reader

Syntax

```
int mathInfoChDA100Reader(DAQDA100READER daqda100reader, int chNo);
```

Declaration

Visual Basic

```
Public Declare Function mathInfoChDA100Reader Lib "DAQDA100" (ByVal daqda100reader As Long, ByVal chNo As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function mathInfoChDA100Reader Lib "DAQDA100" (ByVal daqda100reader As Integer, ByVal chNo As Integer) As Integer
```

C#

```
[DllImport("DAQDA100.dll", CharSet=CharSet.Auto, EntryPoint="mathInfoChDA100Reader")] public static extern int mathInfoChDA100Reader(int daqda100reader, int chNo);
```

Parameters

daqda100Reader	Specify the device descriptor.
chNo	Specify the channel number.

Description

Gets the channel information data of the specified computation channel range.

- If the constant for "Specify all channel numbers" is specified for the channel numbers, all computation channels are processed.
- Specify measurement channels and computation channels separately.
- This function executes the EL command of the DARWIN communication function.

Return value

Returns an error number.

Error:

Not descriptor	No device descriptor.
----------------	-----------------------

Reference

CDAQDA100Reader::mathInfoCh

mathInstChDA100Reader

Syntax

```
int mathInstChDA100Reader(DAQDA100READER daqda100reader, int
chNo);
```

Declaration

Visual Basic

```
Public Declare Function mathInstChDA100Reader Lib
"DAQDA100"(ByVal daqda100reader As Long, ByVal chNo As Long)
As Long
```

Visual Basic.NET

```
Public Declare Ansi Function mathInstChDA100Reader Lib
"DAQDA100"(ByVal daqda100reader As Integer, ByVal chNo As
Integer) As Integer
```

C#

```
[DllImport("DAQDA100.dll" CharSet=CharSet.Auto,
EntryPoint="mathInstChDA100Reader")]
public static extern int mathInstChDA100Reader(int
daqda100reader, int chNo);
```

Parameters

daqda100Reader	Specify the device descriptor.
chNo	Specify the channel number.

Description

Gets the measured data of the specified computation channel.

- If the constant for “Specify all channel numbers” is specified for the channel numbers, all computation channels are processed.
- Gets the measured data and alarm data.
- Specify measurement channels and computation channels separately.
- This function executes the EF command of the DARWIN communication function.

Return value

Returns an error number.

Error:

Not descriptor	No device descriptor.
----------------	-----------------------

Reference

CDAQDA100Reader::mathInstCh

measInfoChDA100Reader

Syntax

```
int measInfoChDA100Reader(DAQDA100READER daqda100reader, int  
chType, int chNo);
```

Declaration

Visual Basic

```
Public Declare Function measInfoChDA100Reader Lib  
"DAQDA100"(ByVal daqda100reader As Long, ByVal chType As Long,  
ByVal chNo As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function measInfoChDA100Reader Lib  
"DAQDA100"(ByVal daqda100reader As Integer, ByVal chType As  
Integer, ByVal chNo As Integer) As Integer
```

C#

```
[DllImport("DAQDA100.dll" CharSet=CharSet.Auto,  
EntryPoint="measInfoChDA100Reader")]  
public static extern int measInfoChDA100Reader(int  
daqda100reader, int chType, int chNo);
```

Parameters

daqda100Reader	Specify the device descriptor.
chType	Specify the channel type.
chNo	Specify the channel number.

Description

Gets the channel information data of the specified measurement channel (specified with the channel type and channel number).

- If the channel type is set to the constant for "Specify all measurement channel types," all subunits are processed.
- If the channel number is set to the constant for "Specify all channel numbers," all channels within the channel type are processed.
- Specify measurement channels and computation channels separately.
- This function executes the EL command of the DARWIN communication function.

Return value

Returns an error number.

Error:

Not descriptor	No device descriptor.
----------------	-----------------------

Reference

CDAQDA100Reader::measInfoCh

measInstChDA100Reader

Syntax

```
int measInstChDA100Reader(DAQDA100READER daqda100reader, int
chType, int chNo);
```

Declaration

Visual Basic

```
Public Declare Function measInstChDA100Reader Lib
"DAQDA100"(ByVal daqda100reader As Long, ByVal chType As Long,
ByVal chNo As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function measInstChDA100Reader Lib
"DAQDA100"(ByVal daqda100reader As Integer, ByVal chType As
Integer, ByVal chNo As Integer) As Integer
```

C#

```
[DllImport("DAQDA100.dll" CharSet=CharSet.Auto,
EntryPoint="measInstChDA100Reader")]
public static extern int measInstChDA100Reader(int
daqda100reader, int chType, int chNo);
```

Parameters

daqda100Reader	Specify the device descriptor.
chType	Specify the channel type.
chNo	Specify the channel number.

Description

Gets the measured data of the specified measurement channel (specified with the channel type and channel number).

- If the channel type is set to the constant for "Specify all measurement channel types," all subunits are processed.
- If the channel number is set to the constant for "Specify all channel numbers," all channels within the channel type are processed.
- Gets the measured data and alarm data.
- Specify measurement channels and computation channels separately.
- This function executes the EF command of the DARWIN communication function.

Return value

Returns an error number.

Error:

Not descriptor	No device descriptor.
----------------	-----------------------

Reference

CDAQDA100Reader::measInstCh

openDA100Reader

Syntax

```
DAQDA100READER openDA100Reader(const char * strAddress, int *  
errorCode);
```

Declaration

Visual Basic

```
Public Declare Function openDA100Reader Lib "DAQDA100" (ByVal  
strAddress As String, ByVal errorCode As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function openDA100Reader Lib  
"DAQDA100" (ByVal strAddress As String, ByVal errorCode As  
Integer) As Integer
```

C#

```
[DllImport("DAQDA100.dll" CharSet=CharSet.Auto,  
EntryPoint="openDA100Reader")]  
public static extern int openDA100Reader(byte[] strAddress,  
out int errorCode);
```

Parameters

strAddress	Specify the IP address as a string.
errorCode	Specify the destination where the error number is to be returned.

Description

Connects to the device with the address specified by the parameters.

- Creates a device descriptor and returns the value as a return value.
- Stores the error number if the return destination is specified.
- The port number is fixed, and it is set to the communication constant, "instantaneous value loading port number."
- Initializes the stored data. Retrieves information about the status of the instrument such as channel information data, and stores the information.
- The specified string is, in general, an ASCII string.
- If unsuccessful, returns NULL in Visual C or 0 in Visual Basic, Visual Basic.NET/C#.

Return value

Returns the device descriptor.

Error:

Creating descriptor is failure	Failed to create the device descriptor.
--------------------------------	---

Reference

CDAQDA100Reader::open

24.4 Details of Instantaneous Value Loading Functions - DARWIN (Visual C/Visual Basic/Visual Basic.NET/C#) Status Transition Functions

This section describes the DARWIN functions that are used in Visual C, Visual Basic, Visual Basic.NET, and C# when using the instantaneous value data loading port.

alarmMaxLengthDA100Reader

Syntax

```
int alarmMaxLengthDA100Reader(void);
```

Declaration

Visual Basic

```
Public Declare Function alarmMaxLengthDA100Reader Lib  
"DAQDA100"() As Long
```

Visual Basic.NET

```
Public Declare Ansi Function alarmMaxLengthDA100Reader Lib  
"DAQDA100"() As Integer
```

C#

```
[DllImport("DAQDA100.dll" CharSet=CharSet.Auto,  
EntryPoint="alarmMaxLengthDA100Reader")]  
public static extern int alarmMaxLengthDA100Reader();
```

Description

Gets the maximum length of the alarm type.

- The return value does not include the terminator.

Return value

Returns the length of the string.

Reference

CDAQDARWINDataInfo::getMaxLenAlarmName

alarmTypeDA100Reader

Syntax

```
int alarmTypeDA100Reader(DAQDA100READER daqda100reader, int
chType, int chNo, int levelNo);
```

Declaration**Visual Basic**

```
Public Declare Function alarmTypeDA100Reader Lib
"DAQDA100"(ByVal daqda100reader As Long, ByVal chType As Long,
ByVal chNo As Long, ByVal levelNo As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function alarmTypeDA100Reader Lib
"DAQDA100"(ByVal daqda100reader As Integer, ByVal chType As
Integer, ByVal chNo As Integer, ByVal levelNo As Integer) As
Integer
```

C#

```
[DllImport("DAQDA100.dll" CharSet=CharSet.Auto,
EntryPoint="alarmTypeDA100Reader")]
public static extern int alarmTypeDA100Reader(int
daqda100reader, int chType, int chNo, int levelNo);
```

Parameters

daqda100Reader	Specify the device descriptor.
chType	Specify the channel type.
chNo	Specify the channel number.
levelNo	Specify the alarm level.

Description

Gets the specified channel (channel type and number) and alarm level alarm type from the stored measured data.

- If it does not exist, "No alarm" is returned.

Return value

Returns the alarm type.

Reference

alarmTypeDA100

channelPointDA100Reader

Syntax

```
int channelPointDA100Reader(DAQDA100READER daqda100reader, int  
chType, int chNo);
```

Declaration

Visual Basic

```
Public Declare Function channelPointDA100Reader Lib  
"DAQDA100"(ByVal daqda100reader As Long, ByVal chType As Long,  
ByVal chNo As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function channelPointDA100Reader Lib  
"DAQDA100"(ByVal daqda100reader As Integer, ByVal chType As  
Integer, ByVal chNo As Integer) As Integer
```

C#

```
[DllImport("DAQDA100.dll" CharSet=CharSet.Auto,  
EntryPoint="channelPointDA100Reader")]  
public static extern int channelPointDA100Reader(int  
daqda100reader, int chType, int chNo);
```

Parameters

daqda100Reader	Specify the device descriptor.
chType	Specify the channel type.
chNo	Specify the channel number.

Description

Gets the decimal point position of the specified channel (channel type and number) from the stored channel information data.

- Returns 0 if it does not exist.

Return value

Returns the decimal point position.

Reference

channelPointDA100

channelStatusDA100Reader

Syntax

```
int channelStatusDA100Reader(DAQDA100READER daqda100reader,  
int chType, int chNo);
```

Declaration

Visual Basic

```
Public Declare Function channelStatusDA100Reader Lib  
"DAQDA100"(ByVal daqda100reader As Long, ByVal chType As Long,  
ByVal chNo As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function channelStatusDA100Reader Lib  
"DAQDA100"(ByVal daqda100reader As Integer, ByVal chType As  
Integer, ByVal chNo As Integer) As Integer
```

C#

```
[DllImport("DAQDA100.dll" CharSet=CharSet.Auto,  
EntryPoint="channelStatusDA100Reader")]  
public static extern int channelStatusDA100Reader(int  
daqda100reader, int chType, int chNo);
```

Parameters

daqda100Reader	Specify the device descriptor.
chType	Specify the channel type.
chNo	Specify the channel number.

Description

Gets the channel status of the specified channel (channel type and number) from the stored channel information data.

- If it does not exist, "Unknown" is returned.

Return value

Returns the channel status.

Reference

channelStatusDA100

dataAlarmDA100Reader

Syntax

```
int dataAlarmDA100Reader(DAQDA100READER daqda100reader, int  
chType, int chNo, int levelNo);
```

Declaration

Visual Basic

```
Public Declare Function dataAlarmDA100Reader Lib  
"DAQDA100"(ByVal daqda100reader As Long, ByVal chType As Long,  
ByVal chNo As Long, ByVal levelNo As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function dataAlarmDA100Reader Lib  
"DAQDA100"(ByVal daqda100reader As Integer, ByVal chType As  
Integer, ByVal chNo As Integer, ByVal levelNo As Integer) As  
Integer
```

C#

```
[DllImport("DAQDA100.dll" CharSet=CharSet.Auto,  
EntryPoint="dataAlarmDA100Reader")]  
public static extern int dataAlarmDA100Reader(int  
daqda100reader, int chType, int chNo, int levelNo);
```

Parameters

daqda100Reader	Specify the device descriptor.
chType	Specify the channel type.
chNo	Specify the channel number.
levelNo	Specify the alarm level.

Description

Gets the Boolean for the valid/invalid value of the alarm corresponding to the alarm level of the specified channel (channel type and number) from the stored measured data.

- If it does not exist, Invalid is returned.

Return value

Returns a Boolean value.

Reference

dataAlarmDA100

dataDayDA100Reader

Syntax

```
int dataDayDA100Reader(DAQDA100READER daqda100reader, int  
chType, int chNo);
```

Declaration

Visual Basic

```
Public Declare Function dataDayDA100Reader Lib  
"DAQDA100"(ByVal daqda100reader As Long, ByVal chType As Long,  
ByVal chNo As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function dataDayDA100Reader Lib  
"DAQDA100"(ByVal daqda100reader As Integer, ByVal chType As  
Integer, ByVal chNo As Integer) As Integer
```

C#

```
[DllImport("DAQDA100.dll" CharSet=CharSet.Auto,  
EntryPoint="dataDayDA100Reader")]  
public static extern int dataDayDA100Reader(int  
daqda100reader, int chType, int chNo);
```

Parameters

daqda100Reader	Specify the device descriptor.
chType	Specify the channel type.
chNo	Specify the channel number.

Description

Gets the day of the specified channel (channel type and number) from the stored time information data.

- The day is a number from 1 to 31.
- Returns 0 if it does not exist.

Return value

Returns the day value.

Reference

dataDayDA100

dataDoubleValueDA100Reader

Syntax

```
double dataDoubleValueDA100Reader(DAQDA100READER  
daqda100reader, int chType, int chNo);
```

Declaration

Visual Basic

```
Public Declare Function dataDoubleValueDA100Reader Lib  
"DAQDA100"(ByVal daqda100reader As Long, ByVal chType As Long,  
ByVal chNo As Long) As Double
```

Visual Basic.NET

```
Public Declare Ansi Function dataDoubleValueDA100Reader Lib  
"DAQDA100"(ByVal daqda100reader As Integer, ByVal chType As  
Integer, ByVal chNo As Integer) As Double
```

C#

```
[DllImport("DAQDA100.dll" CharSet=CharSet.Auto,  
EntryPoint="dataDoubleValueDA100Reader")]  
public static extern double dataDoubleValueDA100Reader(int  
daqda100reader, int chType, int chNo);
```

Parameters

daqda100Reader	Specify the device descriptor.
chType	Specify the channel type.
chNo	Specify the channel number.

Description

Gets the measured value of the specified channel (channel type and number) from the stored measured data.

- Returns 0.0 if it does not exist.

Return value

Returns the measured value as a double-precision floating number.

Reference

dataDoubleValueDA100

dataHourDA100Reader

Syntax

```
int dataHourDA100Reader(DAQDA100READER daqda100reader, int
chType, int chNo);
```

Declaration

Visual Basic

```
Public Declare Function dataHourDA100Reader Lib
"DAQDA100"(ByVal daqda100reader As Long, ByVal chType As Long,
ByVal chNo As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function dataHourDA100Reader Lib
"DAQDA100"(ByVal daqda100reader As Integer, ByVal chType As
Integer, ByVal chNo As Integer) As Integer
```

C#

```
[DllImport("DAQDA100.dll" CharSet=CharSet.Auto,
EntryPoint="dataHourDA100Reader")]
public static extern int dataHourDA100Reader(int
daqda100reader, int chType, int chNo);
```

Parameters

daqda100Reader	Specify the device descriptor.
chType	Specify the channel type.
chNo	Specify the channel number.

Description

Gets the hour of the specified channel (channel type and number) from the stored time information data.

- The hour is a number from 0 to 23.
- Returns 0 if it does not exist.

Return value

Returns the hour value.

Reference

dataHourDA100

dataMilliSecDA100Reader

Syntax

```
int dataMilliSecDA100Reader(DAQDA100READER daqda100reader, int  
chType, int chNo);
```

Declaration

Visual Basic

```
Public Declare Function dataMilliSecDA100Reader Lib  
"DAQDA100"(ByVal daqda100reader As Long, ByVal chType As Long,  
ByVal chNo As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function dataMilliSecDA100Reader Lib  
"DAQDA100"(ByVal daqda100reader As Integer, ByVal chType As  
Integer, ByVal chNo As Integer) As Integer
```

C#

```
[DllImport("DAQDA100.dll" CharSet=CharSet.Auto,  
EntryPoint="dataMilliSecDA100Reader")]  
public static extern int dataMilliSecDA100Reader(int  
daqda100reader, int chType, int chNo);
```

Parameters

daqda100Reader	Specify the device descriptor.
chType	Specify the channel type.
chNo	Specify the channel number.

Description

Gets the milliseconds of the specified channel (channel type and number) from the stored time information data.

- Returns 0 if it does not exist.

Return value

Returns the milliseconds value.

Reference

```
CDAQDA100Reader::getClassDataBuffer  
CDAQDARWINDataBuffer::getClassDARWINDateTime  
CDAQDARWINDateTime::getMilliSecond
```

dataMinuteDA100Reader

Syntax

```
int dataMinuteDA100Reader(DAQDA100READER daqda100reader, int
chType, int chNo);
```

Declaration**Visual Basic**

```
Public Declare Function dataMinuteDA100Reader Lib
"DAQDA100"(ByVal daqda100reader As Long, ByVal chType As Long,
ByVal chNo As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function dataMinuteDA100Reader Lib
"DAQDA100"(ByVal daqda100reader As Integer, ByVal chType As
Integer, ByVal chNo As Integer) As Integer
```

C#

```
[DllImport("DAQDA100.dll" CharSet=CharSet.Auto,
EntryPoint="dataMinuteDA100Reader")]
public static extern int dataMinuteDA100Reader(int
daqda100reader, int chType, int chNo);
```

Parameters

daqda100Reader	Specify the device descriptor.
chType	Specify the channel type.
chNo	Specify the channel number.

Description

Gets the minutes of the specified channel (channel type and number) from the stored time information data.

- The minute is a number from 0 to 59.
- Returns 0 if it does not exist.

Return value

Returns the minute value.

Reference

dataMinuteDA100

dataMonthDA100Reader

Syntax

```
int dataMonthDA100Reader(DAQDA100READER daqda100reader, int  
chType, int chNo);
```

Declaration

Visual Basic

```
Public Declare Function dataMonthDA100Reader Lib  
"DAQDA100"(ByVal daqda100reader As Long, ByVal chType As Long,  
ByVal chNo As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function dataMonthDA100Reader Lib  
"DAQDA100"(ByVal daqda100reader As Integer, ByVal chType As  
Integer, ByVal chNo As Integer) As Integer
```

C#

```
[DllImport("DAQDA100.dll" CharSet=CharSet.Auto,  
EntryPoint="dataMonthDA100Reader")]  
public static extern int dataMonthDA100Reader(int  
daqda100reader, int chType, int chNo);
```

Parameters

daqda100Reader	Specify the device descriptor.
chType	Specify the channel type.
chNo	Specify the channel number.

Description

Gets the month of the specified channel (channel type and number) from the stored time information data.

- The month is a number from 1 to 12.
- Returns 0 if it does not exist.

Return value

Returns the month value.

Reference

dataMonthDA100

dataSecondDA100Reader

Syntax

```
int dataSecondDA100Reader(DAQDA100READER daqda100reader, int
chType, int chNo);
```

Declaration

Visual Basic

```
Public Declare Function dataSecondDA100Reader Lib
"DAQDA100"(ByVal daqda100reader As Long, ByVal chType As Long,
ByVal chNo As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function dataSecondDA100Reader Lib
"DAQDA100"(ByVal daqda100reader As Integer, ByVal chType As
Integer, ByVal chNo As Integer) As Integer
```

C#

```
[DllImport("DAQDA100.dll" CharSet=CharSet.Auto,
EntryPoint="dataSecondDA100Reader")]
public static extern int dataSecondDA100Reader(int
daqda100reader, int chType, int chNo);
```

Parameters

daqda100Reader	Specify the device descriptor.
chType	Specify the channel type.
chNo	Specify the channel number.

Description

Gets the seconds of the specified channel (channel type and number) from the stored time information data.

- The second is a number from 0 to 59.
- Returns 0 if it does not exist.

Return value

Returns the seconds value.

Reference

dataSecondDA100

dataStatusDA100Reader

Syntax

```
int dataStatusDA100Reader(DAQDA100READER daqda100reader, int  
chType, int chNo);
```

Declaration

Visual Basic

```
Public Declare Function dataStatusDA100Reader Lib  
"DAQDA100"(ByVal daqda100reader As Long, ByVal chType As Long,  
ByVal chNo As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function dataStatusDA100Reader Lib  
"DAQDA100"(ByVal daqda100reader As Integer, ByVal chType As  
Integer, ByVal chNo As Integer) As Integer
```

C#

```
[DllImport("DAQDA100.dll" CharSet=CharSet.Auto,  
EntryPoint="dataStatusDA100Reader")]  
public static extern int dataStatusDA100Reader(int  
daqda100reader, int chType, int chNo);
```

Parameters

daqda100Reader	Specify the device descriptor.
chType	Specify the channel type.
chNo	Specify the channel number.

Description

Gets the data status value of the specified channel (channel type and number) from the stored measured data.

- If it does not exist, "Unknown" is returned.

Return value

Returns the data status value.

Reference

dataStatusDA100

dataStringValueDA100Reader

Syntax

```
int dataStringValueDA100Reader(DAQDA100READER daqda100reader,
int chType, int chNo, char * strValue, int lenValue);
```

Declaration**Visual Basic**

```
Public Declare Function dataStringValueDA100Reader Lib
"DAQDA100"(ByVal daqda100reader As Long, ByVal chType As Long,
ByVal chNo As Long, ByVal strValue As String, ByVal lenValue
As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function dataStringValueDA100Reader Lib
"DAQDA100"(ByVal daqda100reader As Integer, ByVal chType As
Integer, ByVal chNo As Integer, ByVal strValue As String,
ByVal lenValue As Integer) As Integer
```

C#

```
[DllImport("DAQDA100.dll" CharSet=CharSet.Auto,
EntryPoint="dataStringValueDA100Reader")]
public static extern int dataStringValueDA100Reader(int
daqda100reader, int chType, int chNo, byte[] strValue, int
lenValue);
```

Parameters

daqda100Reader	Specify the device descriptor.
chType	Specify the channel type.
chNo	Specify the channel number.
strValue	Specify the field where the string is to be stored.
lenValue	Specify the byte size of the field where the string is to be stored.

Description

Gets the measured value of the specified channel (channel type and number) from the stored measured data.

- Converts into a string and stores it in the specified field.
- The string stored in the field includes the terminator (NULL).
- The return value is the length of the actual string. The return value does not include the terminator.
- Returns 0 if it does not exist.
- The strings that can be stored are, in general, ASCII strings.

Return value

Returns the length of the string.

Reference

dataStringValueDA100

dataValueDA100Reader

Syntax

```
int dataValueDA100Reader(DAQDA100READER daqda100reader, int  
chType, int chNo);
```

Declaration

Visual Basic

```
Public Declare Function dataValueDA100Reader Lib  
"DAQDA100"(ByVal daqda100reader As Long, ByVal chType As Long,  
ByVal chNo As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function dataValueDA100Reader Lib  
"DAQDA100"(ByVal daqda100reader As Integer, ByVal chType As  
Integer, ByVal chNo As Integer) As Integer
```

C#

```
[DllImport("DAQDA100.dll" CharSet=CharSet.Auto,  
EntryPoint="dataValueDA100Reader")]  
public static extern int dataValueDA100Reader(int  
daqda100reader, int chType, int chNo);
```

Parameters

daqda100Reader	Specify the device descriptor.
chType	Specify the channel type.
chNo	Specify the channel number.

Description

Gets the data value of the specified channel (channel type and number) from the stored measured data.

- Returns 0 if it does not exist.

Return value

Returns the data value.

Reference

dataValueDA100

dataYearDA100Reader

Syntax

```
int dataYearDA100Reader(DAQDA100READER daqda100reader, int  
chType, int chNo);
```

Declaration

Visual Basic

```
Public Declare Function dataYearDA100Reader Lib  
"DAQDA100"(ByVal daqda100reader As Long, ByVal chType As Long,  
ByVal chNo As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function dataYearDA100Reader Lib  
"DAQDA100"(ByVal daqda100reader As Integer, ByVal chType As  
Integer, ByVal chNo As Integer) As Integer
```

C#

```
[DllImport("DAQDA100.dll" CharSet=CharSet.Auto,  
EntryPoint="dataYearDA100Reader")]  
public static extern int dataYearDA100Reader(int  
daqda100reader, int chType, int chNo);
```

Parameters

daqda100Reader	Specify the device descriptor.
chType	Specify the channel type.
chNo	Specify the channel number.

Description

Gets the year of the specified channel (channel type and number) from the stored time information data.

- The year is a 4-digit number.
- Returns 0 if it does not exist.

Return value

Returns the year value.

Reference

dataYearDA100

errorMaxLengthDA100Reader

Syntax

```
int errorMaxLengthDA100Reader(void);
```

Declaration

Visual Basic

```
Public Declare Function errorMaxLengthDA100Reader Lib  
"DAQDA100"() As Long
```

Visual Basic.NET

```
Public Declare Ansi Function errorMaxLengthDA100Reader Lib  
"DAQDA100"() As Integer
```

C#

```
[DllImport("DAQDA100.dll" CharSet=CharSet.Auto,  
EntryPoint="errorMaxLengthDA100Reader")]  
public static extern int errorMaxLengthDA100Reader();
```

Description

Gets the maximum length of the error message string.

- The return value does not include the terminator.

Return value

Returns the length of the string.

Reference

CDAQDA100Reader::getMaxLenErrorMessage

getAlarmNameDA100Reader [Visual C only]

Syntax

```
const char * getAlarmNameDA100Reader(int iAlarmType);
```

Parameters

iAlarmType Specify the alarm type.

Description

Gets the string corresponding to the specified alarm type.

- If it does not exist, returns the pointer to the string corresponding to "No alarm."

Return value

Returns a pointer to the string.

Reference

CDAQDARWINDataInfo::getAlarmName

getChannelUnitDA100Reader [Visual C only]

Syntax

```
const char * getChannelUnitDA100Reader(DAQDA100READER  
daqda100reader, int chType, int chNo);
```

Parameters

daqda100Reader Specify the device descriptor.

chType Specify the channel type.

chNo Specify the channel number.

Description

Gets the unit name of the specified channel (channel type and number) from the stored channel information data.

- Returns NULL if it does not exist.

Return value

Returns a pointer to the string.

Reference

getChannelUnitDA100

getErrorMessageDA100Reader [Visual C only]

Syntax

```
const char * getErrorMessageDA100Reader(int errCode);
```

Parameters

errCode Specify the error number.

Description

Gets the error message string corresponding to the specified error number.

- Returns a pointer to the string [Unknown] if it does not exist.

Return value

Returns a pointer to the string.

Reference

CDAQDA100Reader::getErrorMessage

revisionAPIDA100Reader

Syntax

```
const int revisionAPIDA100Reader(void);
```

Declaration

Visual Basic

```
Public Declare Function revisionAPIDA100Reader Lib  
"DAQDA100"() As Long
```

Visual Basic.NET

```
Public Declare Ansi Function revisionAPIDA100Reader Lib  
"DAQDA100"() As Integer
```

C#

```
[DllImport("DAQDA100.dll" CharSet=CharSet.Auto,  
EntryPoint="revisionAPIDA100Reader")]  
public static extern int revisionAPIDA100Reader();
```

Description

Gets the revision number of this API.

Return value

Returns the revision number.

Reference

CDAQDA100Reader::getRevisionAPI

toAlarmNameDA100Reader

Syntax

```
int toAlarmNameDA100Reader(int iAlarmType, char * strAlarm,
int lenAlarm);
```

Declaration**Visual Basic**

```
Public Declare Function toAlarmNameDA100Reader Lib
"DAQDA100"(ByVal iAlarmType As Long, ByVal strAlarm As String,
ByVal lenAlarm As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function toAlarmNameDA100Reader Lib
"DAQDA100"(ByVal iAlarmType As Integer, ByVal strAlarm As
String, ByVal lenAlarm As Integer) As Integer
```

C#

```
[DllImport("DAQDA100.dll" CharSet=CharSet.Auto,
EntryPoint="toAlarmNameDA100Reader")]
public static extern int toAlarmNameDA100Reader(int
iAlarmType, byte[] strAlarm, int lenAlarm);
```

Parameters

iAlarmType	Specify the alarm type.
strAlarm	Specify the field where the string is to be stored.
lenAlarm	Specify the byte size of the field where the string is to be stored.

Description

Stores the string corresponding to the specified alarm type to the specified field.

- The string stored in the field includes the terminator (NULL).
- The return value is the length of the actual string. The return value does not include the terminator.
- The strings that can be stored are, in general, ASCII strings.

Return value

Returns the length of the string.

Reference

getAlarmNameDA100Reader

toChannelUnitDA100Reader

Syntax

```
int toChannelUnitDA100Reader(DAQDA100READER daqda100reader,  
int chType, int chNo, char * strUnit, int lenUnit);
```

Declaration

Visual Basic

```
Public Declare Function toChannelUnitDA100Reader Lib  
"DAQDA100"(ByVal daqda100reader As Long, ByVal chType As Long,  
ByVal chNo As Long, ByVal strUnit As String, ByVal lenUnit As  
Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function toChannelUnitDA100Reader Lib  
"DAQDA100"(ByVal daqda100reader As Integer, ByVal chType As  
Integer, ByVal chNo As Integer, ByVal strUnit As String, ByVal  
lenUnit As Integer) As Integer
```

C#

```
[DllImport("DAQDA100.dll" CharSet=CharSet.Auto,  
EntryPoint="toChannelUnitDA100Reader")]  
public static extern int toChannelUnitDA100Reader(int  
daqda100reader, int chType, int chNo, byte[] strUnit, int  
lenUnit);
```

Parameters

daqda100Reader	Specify the device descriptor.
chType	Specify the channel type.
chNo	Specify the channel number.
strUnit	Specify the field where the string is to be stored.
lenUnit	Specify the byte size of the field where the string is to be stored.

Description

Gets the unit name of the specified channel (channel type and number) from the stored channel information data.

- Stores the string in the specified storage destination.
- The string stored in the field includes the terminator (NULL).
- The return value is the length of the actual string. The return value does not include the terminator.
- Returns 0 if it does not exist.
- The strings that can be stored are, in general, ASCII strings.

Return value

Returns the length of the string.

Reference

getChannelUnitDA100Reader

toDoubleValueDA100Reader

Syntax

```
double toDoubleValueDA100Reader(int dataValue, int point);
```

Declaration

Visual Basic

```
Public Declare Function toDoubleValueDA100Reader Lib  
"DAQDA100"(ByVal dataValue As Long, ByVal point As Long) As  
Double
```

Visual Basic.NET

```
Public Declare Ansi Function toDoubleValueDA100Reader Lib  
"DAQDA100"(ByVal dataValue As Integer, ByVal point As Integer)  
As Double
```

C#

```
[DllImport("DAQDA100.dll" CharSet=CharSet.Auto,  
EntryPoint="toDoubleValueDA100Reader")]  
public static extern double toDoubleValueDA100Reader(int  
dataValue, int point);
```

Parameters

dataValue	Specify the data value.
point	Specify the decimal point position.

Description

Generates the measured value from the specified data value and decimal point position.

Return value

Returns the measured value as a double-precision floating point number.

Reference

CDAQDARWINDataInfo::toDoubleValue

toErrorMessageDA100Reader

Syntax

```
int toErrorMessageDA100Reader(int errCode, char * errStr, int errLen);
```

Declaration

Visual Basic

```
Public Declare Function toErrorMessageDA100Reader Lib "DAQDA100" (ByVal errCode As Long, ByVal errStr As String, ByVal errLen As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function toErrorMessageDA100Reader Lib "DAQDA100" (ByVal errCode As Integer, ByVal errStr As String, ByVal errLen As Integer) As Integer
```

C#

```
[DllImport("DAQDA100.dll" CharSet=CharSet.Auto, EntryPoint="toErrorMessageDA100Reader")] public static extern int toErrorMessageDA100Reader(int errCode, byte[] errStr, int errLen);
```

Parameters

errCode	Specify the error number.
errStr	Specify the field where the string is to be stored.
errLen	Specify the byte size of the field where the string is to be stored.

Description

Stores the error message string corresponding to the error number to the specified field.

- The string stored in the field includes the terminator (NULL).
- The return value is the length of the actual string. The return value does not include the terminator.
- The strings that can be stored are, in general, ASCII strings.

Return value

Returns the length of the string.

Reference

getErrorMessageDA100Reader

toStringValueDA100Reader

Syntax

```
int toStringValueDA100Reader(int dataValue, int point, char *
    strValue, int lenValue);
```

Declaration**Visual Basic**

```
Public Declare Function toStringValueDA100Reader Lib
    "DAQDA100"(ByVal dataValue As Long, ByVal point As Long, ByVal
    strValue As String, ByVal lenValue As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function toStringValueDA100Reader Lib
    "DAQDA100"(ByVal dataValue As Integer, ByVal point As Integer,
    ByVal strValue As String, ByVal lenValue As Integer) As
    Integer
```

C#

```
[DllImport("DAQDA100.dll" CharSet=CharSet.Auto,
    EntryPoint="toStringValueDA100Reader" ]
public static extern int toStringValueDA100Reader(int
    dataValue, int point, byte[] strValue, int lenValue);
```

Parameters

dataValue	Specify the data value.
point	Specify the decimal point position.
strValue	Specify the field where the string is to be stored.
lenValue	Specify the byte size of the field where the string is to be stored.

Description

Generates the measured value from the specified data value and decimal point position.

- Converts the generated measured value into a string and stores to the specified field.
- The string stored in the field includes the terminator (NULL).
- The return value is the length of the actual string. The return value does not include the terminator.
- The strings that can be stored are, in general, ASCII strings.

Return value

Returns the length of the string.

Reference

CDAQDARWINDataInfo::toStringValue

versionAPIDA100Reader

Syntax

```
const int versionAPIDA100Reader(void);
```

Declaration

Visual Basic

```
Public Declare Function versionAPIDA100Reader Lib "DAQDA100"()  
As Long
```

Visual Basic.NET

```
Public Declare Ansi Function versionAPIDA100Reader Lib  
"DAQDA100"() As Integer
```

C#

```
[DllImport("DAQDA100.dll" CharSet=CharSet.Auto,  
EntryPoint="versionAPIDA100Reader")]  
public static extern int versionAPIDA100Reader();
```

Description

Gets the version number of this API.

Return value

Returns the version number.

Reference

CDAQDA100Reader::getVersionAPI

25.1 Overview of the DARWIN Constants

This Extended API provides the following types of constants.

In Visual C/Visual C++, the constants from chapter 11.1 are inherited. Also, constants, retrieval code types, range types, and skip range constants have been added for the Extended API. See section 25.2.

The constants for Visual Basic and Visual Basic.NET/C# are listed in section 25.2.

Type	Description	Page
Constants	Channel numbers in the unit, etc.	25-2, 25-5
Retrieve code types	Binary code, ASCII code	25-2, 25-5
Comm constants	Communication port number of DARWIN	11-2, 25-5
Numbers of items	Number of subunits, etc.	11-2, 25-5
Maximum values	Maximum length of the channel name string, etc.	11-2, 25-6
Boolean values	Valid (ON) setting or Invalid (OFF) setting	11-2, 25-6
Flag statuses	Detects the last data set when data is retrieved	11-3, 25-6
Data status values	Status of the measured data	11-3, 25-6
Alarm types	Upper-limit alarm, etc.	11-3, 25-7
Channel/relay types	Channel or relay types	11-4, 25-7
Operation modes	Operation, setup, and A/D calibration mode	11-4, 25-7
Talker function types	Talker supporting output data	11-4, 25-8
Status byte values	Various statuses	11-5, 25-8
Establish setup modes	Destroy, establish	11-5, 25-8
Unit numbers	Expandable model, standalone model	11-5, 25-8
Computations	Calculation start, stop, clear, etc.	11-5, 25-9
Report execution types	Report start/stop	11-5, 25-9
Report types	Hourly, daily, monthly, status	11-6, 25-9
Report statuses	Invalidate all report types, latest info, valid	11-6, 25-9
Range types		
DC voltage range types	20 mV, etc.	11-6, 25-3, 25-10
TC range types	Type R, etc.	11-6, 25-3, 25-10
RTD range types	Pt100: 1 mA, etc.	11-7, 25-3, 25-11
Contact input rng types	Voltage level or contact input	11-7,25-4, 25-11
Strain input ranges	2 k, 20 k, 200 k	11-7, 25-4, 25-11
Pulse ranges	GATE, RATE	11-7,25-4, 25-11
Power monitor ranges	25 V 0.5 A, 25 V 5 A, 250 V 0.5 A, 250 V 5 A	11-8,25-4, 25-12
Power connection methods	Single-phase two-wire, etc.	11-8, 25-12
DC current ranges	20 mA	11-8, 25-4, 25-12
SKIP ranges	Skip	25-4, 25-12
Power connection methods	Single-phase two-wire, etc.	11-8, 25-12
Power meas parameters	Effective current, etc.	11-9, 25-13

25.2 DARWIN Constants

This section describes the mnemonics for and the meanings of the constants. For DARWIN terminology, see appendix 2.

Visual C/Visual C++ Constants

In Visual C/Visual C++, the constants from chapter 11.2 are inherited. The following constants have been added.

Constants

Mnemonic	Description
DAQDA100_NUMCH_BYUNIT	Channel numbers in the unit
DAQDA100_CHTYPE_MEASALL	Specifies all measurement channels (computation channels not included)
DAQDA100_CHNO_ALL	Specification of all channel numbers
DAQDA100_LEVELNO_ALL	Specification of all alarm levels

Retrieve Code Types

Mnemonic	Description
DAQDA100_CODE_BINARY	Binary code
DAQDA100_CODE_ASCII	ASCII code

The retrieval code type is the output format type when retrieving measured data.

Range Types

The Extended API provides definitions for identifying unique ranges. It is synthesized with logical operations.

Mnemonic	Description
DAQDA100_RANGETYPE_VOLT	DC voltage range types
DAQDA100_RANGETYPE_DI	Contact range
DAQDA100_RANGETYPE_TC	TC range
DAQDA100_RANGETYPE_RTD	RTD range
DAQDA100_RANGETYPE_SKIP	SKIP range
DAQDA100_RANGETYPE_MA	DC current range
DAQDA100_RANGETYPE_POWER	Power monitor range
DAQDA100_RANGETYPE_STRAIN	Strain input range
DAQDA100_RANGETYPE_PULSE	Pulse range

When specifying the range, specify the above types and the synthesized unique range types shown below.

DC Voltage Range Types

Mnemonic	Description	Setting range
DAQDA100_RANGE_VOLT_20MV	20 mV	-20.000 to 20.000 mV
DAQDA100_RANGE_VOLT_60MV	60 mV	-60.00 to 60.00 mV
DAQDA100_RANGE_VOLT_200MV	200 mV	-200.00 to 200.00 mV
DAQDA100_RANGE_VOLT_2V	2 V	-2.0000 to 2.0000 V
DAQDA100_RANGE_VOLT_6V	6 V	-6.000 to 6.000 V
DAQDA100_RANGE_VOLT_20V	20 V	-20.000 to 20.000 V
DAQDA100_RANGE_VOLT_50V	50 V	-50.00 to 50.00 V

TC Range Types

Mnemonic	Description	Setting range
DAQDA100_RANGE_TC_R	R	0.0 to 1760.0°C
DAQDA100_RANGE_TC_S	S	0.0 to 1760.0°C
DAQDA100_RANGE_TC_B	B	0.0 to 1820.0°C
DAQDA100_RANGE_TC_K	K	-200.0 to 1370.0°C
DAQDA100_RANGE_TC_E	E	-200.0 to 800.0°C
DAQDA100_RANGE_TC_J	J	-200.0 to 1100.0°C
DAQDA100_RANGE_TC_T	T	-200.0 to 400.0°C
DAQDA100_RANGE_TC_N	N	0.0 to 1300.0°C
DAQDA100_RANGE_TC_W	W	0.0 to 2315.0°C
DAQDA100_RANGE_TC_L	L	-200.0 to 900.0°C
DAQDA100_RANGE_TC_U	U	-200.0 to 400.0°C
DAQDA100_RANGE_TC_KP	KpAu7Fe	0.0 to 300.0 K

RTD Range Types

Mnemonic	Description	Setting range
DAQDA100_RANGE_RTD_1MAPT	Pt100:1mA	-200.0 to 600.0°C
DAQDA100_RANGE_RTD_2MAPT	Pt100:2mA	-200.0 to 250.0°C
DAQDA100_RANGE_RTD_1MAJPT	JPt100:1mA	-200.0 to 550.0°C
DAQDA100_RANGE_RTD_2MAJPT	JPt100:2mA	-200.0 to 250.0°C
DAQDA100_RANGE_RTD_2MAPT50	Pt50:2mA	-200.0 to 550.0°C
DAQDA100_RANGE_RTD_1MAPTH	Pt100:1mA-H	-140.00 to 150.00°C
DAQDA100_RANGE_RTD_2MAPTH	Pt100:2mA-H	-70.00 to 70.00°C
DAQDA100_RANGE_RTD_1MAJPTH	JPt100:1mA-H	-140.00 to 150.00°C
DAQDA100_RANGE_RTD_2MAJPTH	JPt100:2mA-H	-70.00 to 70.00°C
DAQDA100_RANGE_RTD_1MANS	Ni100:1mA-S	-200.0 to 250.0°C
DAQDA100_RANGE_RTD_1MANID	Ni100:1mA-D	-60.0 to 180.0°C
DAQDA100_RANGE_RTD_1MANI120	Ni120:1mA	-70.0 to 200.0°C
DAQDA100_RANGE_RTD_CU10GE	Cu10:GE	-200.0 to 300.0°C
DAQDA100_RANGE_RTD_CU10LN	Cu10:L&N	-200.0 to 300.0°C
DAQDA100_RANGE_RTD_CU10WEED	Cu10:WEED	-200.0 to 300.0°C
DAQDA100_RANGE_RTD_CU10BAILEY	Cu10:BAILEY	-200.0 to 300.0°C
DAQDA100_RANGE_RTD_J263B	J263*B	-0.0 to 300.0 K

Contact Input Ranges

Mnemonic	Description	Setting range
DAQDA100_RANGE_DI_LEVEL	Voltage input	0: Less than 2.4 V, 1: Greater than or equal to 2.4 V
DAQDA100_RANGE_DI_CONTACT	Contact input	0:open,1:close

Strain Input Ranges

Mnemonic	Description	Setting range
DAQDA100_RANGE_STRAIN_2K	2 k	-2000 to 2000 μ Strain ^{*1} -1000 to 1000 μ strain ^{*2} -500 to 500 μ strain ^{*3}
DAQDA100_RANGE_STRAIN_20K	20 k	-20000 to 20000 μ Strain ^{*1} -10000 to 10000 μ strain ^{*2} -5000 to 5000 μ strain ^{*3}
DAQDA100_RANGE_STRAIN_200K	200 k	-200000 to 200000 μ Strain ^{*1} -100000 to 100000 μ strain ^{*2} -50000 to 50000 μ strain ^{*3}

*1: 1-Gauge Method, *2: 2-Gauge Method, *3: 4-Gauge Method

Pulse Ranges

Mnemonic	Description	Setting range
DAQDA100_RANGE_PULSE_RATE	RATE	0 - 30000
DAQDA100_RANGE_PULSE_GATE	GATE	0 - 30000

Power Monitor Ranges

Mnemonic	Description	Setting range
DAQDA100_RANGE_POWER_25V05A	25 V 0.5 A	Voltage 25 V, current 0.5 A
DAQDA100_RANGE_POWER_25V5A	25 V 5 A	Voltage 25 V, current 5 A
DAQDA100_RANGE_POWER_250V05A	250 V 0.5 A	Voltage 250 V, current 0.5 A
DAQDA100_RANGE_POWER_250V5A	250 V 5 A	Voltage 250 V, current 5 A

DC Current Ranges

Mnemonic	Description	Setting range
DAQDA100_RANGE_MA_20MA	20 mA	-20.000 to 20.000 mA

SKIP Range

Mnemonic	Description
DAQDA100_RANGE_SKIP SKIP	SKIP (not used) range

Constants for Visual Basic and Visual Basic.NET/C#

This section describes the mnemonics for and the meanings of the constants. For the details on the DARWIN functions, see the relevant user's manual.

In C#, it is the constant data for the DAQDA100 class. Prefix each constant with DAQDA100 (Ex.: DAQDA100.DAQDA100_NUMCHANNEL).

Constants

Mnemonic	Description
DAQDA100_NUMCH_BYUNIT	Channel numbers in the unit
DAQDA100_CHTYPE_MEASALL	Specifies all measurement channels (computation channels are not included)
DAQDA100_CHNO_ALL	Specification of all channel numbers
DAQDA100_LEVELNO_ALL	Specification of all alarm levels

Retrieve Code Types

Mnemonic	Description
DAQDA100_CODE_BINARY	Binary code
DAQDA100_CODE_ASCII	ASCII code

The retrieval code type is the output format type when retrieving measured data.

Communication Constants

Mnemonic	Description
DAQDA100_COMMPORT	Communication port number of DARWIN.

Number of Items

Mnemonic	Description
DAQDA100_NUMCHANNEL	The number of channels.
DAQDA100_NUMALARM	The number of alarms.
DAQDA100_NUMUNIT	The number of subunits.
DAQDA100_NUMSLOT	The number of slots per subunit.
DAQDA100_NUMTERM	The number of terminals per slot (module).

Sets the number of items such as the number of modules or units.

Maximum Value

Mnemonic	Description
DAQDA100_MAXCHNAMELEN	Maximum length of the channel name string.
DAQDA100_MAXCHRANGLLEN	Maximum length of the channel range name string.
DAQDA100_MAXUNITLEN	Maximum length of the unit name string.
DAQDA100_MAXMODULELEN	Maximum length of the module name string.
DAQDA100_MAXRELAYLEN	Maximum length of the relay name string. Same as maximum length of the channel name string. Relay refers to the output relay of the alarm output module or the DI/DO module.
DAQDA100_MAXDECIMALPOINT	Maximum value of the decimal point position.

The maximum length of the string does not include the terminator (NULL).

Boolean values (Valid/Invalid Value)

Mnemonic	Description
DAQDA100_VALID_OFF	Invalid (OFF) value.
DAQDA100_VALID_ON	Valid (ON) value.

Flag Status

Mnemonic	Description
DAQDA100_FLAG_OFF	All OFF.
DAQDA100_FLAG_ENDDATA	The data line retrieved using ASCII codes or in units of lines is at the last data set.

Can be synthesized using logical OR operators.

Data Status Values

Mnemonic	Description
DAQDA100_UNKNWON	The data status is not set.
DAQDA100_DATA_NORMAL	Normal.
DAQDA100_DATA_DIFFINPUT	Difference computation between channels being performed.
DAQDA100_DATA_PLUSOVER	Positive overrange.
DAQDA100_DATA_MINUSOVER	Negative overrange.
DAQDA100_DATA_SKIP	SKIP (not used).
DAQDA100_DATA_ILLEGAL	Illegal data status.
DAQDA100_DATA_ABNORMAL	Abnormal data status.
DAQDA100_DATA_NODATA	No data status.
DAQDA100_DATA_READER	Status when loading and communicating instantaneous value data

When using the communication port for loading instantaneous value data, the status when loading and communicating instantaneous value data is the channel status in which channel information data is retrieved.

Alarm Types

◇ indicates a space.

Mnemonic	Description	String
DAQDA100_ALARM_NONE	No alarm (alarm OFF)	◇◇
DAQDA100_ALARM_UPPER	Upper limit alarm	H◇
DAQDA100_ALARM_LOWER	Lower limit alarm	L◇
DAQDA100_ALARM_UPDIFF	Difference upper limit alarm	dH
DAQDA100_ALARM_LOWDIFF	Difference lower limit alarm	dL
DAQDA100_ALARM_INCRATE	High limit on rate-of-change alarm	RH
DAQDA100_ALARM_DECRATE	Low limit on rate-of-change alarm	RL

Channel/Relay Types

Type values for channels, relays, communication input, and computation constants. Can be used to specify the channel or relay.

Mnemonic	Description	Channel	Relay
DAQDA100_CHTYPE_MAINUNIT	Expandable model main unit.	-	Yes
DAQDA100_CHTYPE_STANDALONE	Value representing the standalone unit. Same as subunit number of 0.	Yes	Yes
DAQDA100_CHTYPE_MATHTYPE	Value representing the computation channel.	Yes	-
DAQDA100_CHTYPE_SWITCH	Value representing the internal switch.	-	Yes
DAQDA100_CHTYPE_COMMDATA	Value representing the communication input.	-	-
DAQDA100_CHTYPE_CONSTANT	Value representing the computation constant.	-	-
DAQDA100_CHTYPE_REPORT	Value representing the report.	-	-

Yes: Channel/Relay of the type exists.

- : Channel/Relay of the type does not exist.

Note

The subunit number used to identify the subunit that is connected to the expandable model is also a type number. The subunit number is an integer value between 0 and 5. See appendix 2.

Operation Modes

Mnemonic	Description
DAQDA100_MODE_OPE	Operation mode
DAQDA100_MODE_SETUP	Setup mode
DAQDA100_MODE_CALIB	A/D calibration mode

Talker Function Types

Mnemonic	Description
DAQDA100_TALK_MEASUREDDATA	Outputs measured and computed data
DAQDA100_TALK_OPEDATA	Outputs the setup data of operation mode.
DAQDA100_TALK_CHINFODATA	Outputs the channel information data
DAQDA100_TALK_SYSINFODATA	Outputs the system configuration data
DAQDA100_TALK_CALIBDATA	Outputs the calibration data (setup data of calibration mode)
DAQDA100_TALK_SETUPDATA	Outputs the setup data of setup mode
DAQDA100_TALK_REPORTDATA	Outputs the report status

Status Byte Values

Can be synthesized using logical OR operators.

Mnemonic	Description
DAQDA100_STATUS_OFF	Value when all status bytes are invalid
DAQDA100_STATUS_ADCONV	A/D conversion complete
DAQDA100_STATUS_SYNTAX	Command syntax error
DAQDA100_STATUS_TIMER	Internal timer start/report creation
DAQDA100_STATUS_MEDIA	Access to medium (DC100)
DAQDA100_STATUS_RELEASE	Measurement dropout during computation
DAQDA100_STATUS_ALL	Mask value that enables all status bytes
DAQDA100_STATUS_SRQ	SRQ

For details on the meaning of the status byte value, see the communication interface user's manual for the DARWIN instrument.

Establish Setup Mode

Mnemonic	Description
DAQDA100_SETUP_ABORT	Destroy
DAQDA100_SETUP_STORE	Establish

Unit Numbers

Mnemonic	Description
DAQDA100_UNITNO_MAINUNIT	Main unit of the extension model
DAQDA100_UNITNO_STANDALONE	Standalone model unit

The subunit number is numerical. See Channel/Relay types.

Computation

Mnemonic	Description
DAQDA100_COMPUTE_START	Computation start
DAQDA100_COMPUTE_STOP	Computation stop
DAQDA100_COMPUTE_RESTART	After clearing computation data, restart
DAQDA100_COMPUTE_CLEAR	Clears computation data
DAQDA100_COMPUTE_RELEASE	Clears status display of measurement dropouts

Report Execution Types

Mnemonic	Description
DAQDA100_REPORT_RUN_START	Report start
DAQDA100_REPORT_RUN_STOP	Report stop

Report Types

Mnemonic	Description
DAQDA100_REPORT_HOURLY	Hourly
DAQDA100_REPORT_DAILY	Daily
DAQDA100_REPORT_MONTHLY	Monthly
DAQDA100_REPORT_STATUS	Status

Report Statuses

Can be synthesized using logical OR operators.

Mnemonic	Description
DAQDA100_REPSTATUS_NONE	All invalid
DAQDA100_REPSTATUS_HOURLY_NEW	Newest hourly
DAQDA100_REPSTATUS_HOURLY_VALID	Valid hourly
DAQDA100_REPSTATUS_DAILY_NEW	Newest daily
DAQDA100_REPSTATUS_DAILY_VALID	Valid daily
DAQDA100_REPSTATUS_MONTHLY_NEW	Newest monthly
DAQDA100_REPSTATUS_MONTHLY_VALID	Valid monthly

Range Types

The Extended API provides definitions for identifying unique ranges. It is synthesized with logical operations.

Mnemonic	Description
DAQDA100_RANGETYPE_VOLT	DC voltage range
DAQDA100_RANGETYPE_DI	Contact range
DAQDA100_RANGETYPE_TC	TC range
DAQDA100_RANGETYPE_RTD	RTD range
DAQDA100_RANGETYPE_SKIP	SKIP range
DAQDA100_RANGETYPE_MA	DC current range
DAQDA100_RANGETYPE_POWER	Power monitor range
DAQDA100_RANGETYPE_STRAIN	Strain input range
DAQDA100_RANGETYPE_PULSE	Pulse range

When specifying the range, specify the above types and the synthesized unique range types shown below.

DC Voltage Range Types

Mnemonic	Description	Setting range
DAQDA100_RANGE_VOLT_20MV	20 mV	-20.000 to 20.000 mV
DAQDA100_RANGE_VOLT_60MV	60 mV	-60.00 to 60.00 mV
DAQDA100_RANGE_VOLT_200MV	200 mV	-200.00 to 200.00 mV
DAQDA100_RANGE_VOLT_2V	2 V	-2.0000 to 2.0000 V
DAQDA100_RANGE_VOLT_6V	6 V	-6.000 to 6.000 V
DAQDA100_RANGE_VOLT_20V	20 V	-20.000 to 20.000 V
DAQDA100_RANGE_VOLT_50V	50 V	-50.00 to 50.00 V

TC Ranges

Mnemonic	Description	Setting range
DAQDA100_RANGE_TC_R	R	0.0 to 1760.0°C
DAQDA100_RANGE_TC_S	S	0.0 to 1760.0°C
DAQDA100_RANGE_TC_B	B	0.0 to 1820.0°C
DAQDA100_RANGE_TC_K	K	-200.0 to 1370.0°C
DAQDA100_RANGE_TC_E	E	-200.0 to 800.0°C
DAQDA100_RANGE_TC_J	J	-200.0 to 1100.0°C
DAQDA100_RANGE_TC_T	T	-200.0 to 400.0°C
DAQDA100_RANGE_TC_N	N	0.0 to 1300.0°C
DAQDA100_RANGE_TC_W	W	0.0 to 2315.0°C
DAQDA100_RANGE_TC_L	L	-200.0 to 900.0°C
DAQDA100_RANGE_TC_U	U	-200.0 to 400.0°C
DAQDA100_RANGE_TC_KP	KpAu7Fe	0.0 to 300.0 K

RTD Ranges

Mnemonic	Description	Setting range
DAQDA100_RANGE_RTD_1MAPT	Pt100:1mA	-200.0 to 600.0°C
DAQDA100_RANGE_RTD_2MAPT	Pt100:2mA	-200.0 to 250.0°C
DAQDA100_RANGE_RTD_1MAJPT	JPt100:1mA	-200.0 to 550.0°C
DAQDA100_RANGE_RTD_2MAJPT	JPt100:2mA	-200.0 to 250.0°C
DAQDA100_RANGE_RTD_2MAPT50	Pt50:2mA	-200.0 to 550.0°C
DAQDA100_RANGE_RTD_1MAPTH	Pt100:1mA-H	-140.00 to 150.00°C
DAQDA100_RANGE_RTD_2MAPTH	Pt100:2mA-H	-70.00 to 70.00°C
DAQDA100_RANGE_RTD_1MAJPTH	JPt100:1mA-H	-140.00 to 150.00°C
DAQDA100_RANGE_RTD_2MAJPTH	JPt100:2mA-H	-70.00 to 70.00°C
DAQDA100_RANGE_RTD_1MANS	Ni100:1mA-S	-200.0 to 250.0°C
DAQDA100_RANGE_RTD_1MANID	Ni100:1mA-D	-60.0 to 180.0°C
DAQDA100_RANGE_RTD_1MANI120	Ni120:1mA	-70.0 to 200.0°C
DAQDA100_RANGE_RTD_CU10GE	Cu10:GE	-200.0 to 300.0°C
DAQDA100_RANGE_RTD_CU10LN	Cu10:L&N	-200.0 to 300.0°C
DAQDA100_RANGE_RTD_CU10WEED	Cu10:WEED	-200.0 to 300.0°C
DAQDA100_RANGE_RTD_CU10BAILEY	Cu10:BAILEY	-200.0 to 300.0°C
DAQDA100_RANGE_RTD_J263B	J263*B	-0.0 to 300.0 K

Contact Input (DI) Ranges

Mnemonic	Description	Setting range
DAQDA100_RANGE_DI_LEVEL	Voltage input	Less than 2.4 V, Greater than or equal to 2.4 V
DAQDA100_RANGE_DI_CONTACT	Contact input	0:open,1:close

Strain Input Ranges

Mnemonic	Description	Setting range
DAQDA100_RANGE_STRAIN_2K	2 k	-2000 to 2000 mStrain *1 -1000 to 1000 mStrain *2 -500 to 500 mStrain *3
DAQDA100_RANGE_STRAIN_20K	20 k	-20000 to 20000 μ Strain *1 -10000 to 10000 mstrain *2 -5000 to 5000 mStrain *3
DAQDA100_RANGE_STRAIN_200K	200 k	-200000 to 200000 μ Strain *1 -100000 to 100000 mStrain *2 -50000 to 50000 mStrain *3

*1: 1-Gauge Method, *2: 2-Gauge Method, *3: 4-Gauge Method

Pulse Ranges

Mnemonic	Description	Setting range
DAQDA100_RANGE_PULSE_RATE	RATE	0 to 30000
DAQDA100_RANGE_PULSE_GATE	GATE	0 to 30000

Power Monitor Ranges

Mnemonic	Description	Setting range
DAQDA100_RANGE_POWER_25V05A	25 V 0.5 A	Voltage 25 V, current 0.5 A
DAQDA100_RANGE_POWER_25V5A	25 V 5 A	Voltage 25 V, current 5 A
DAQDA100_RANGE_POWER_250V05A	250 V 0.5 A	Voltage 250 V, current 0.5 A
DAQDA100_RANGE_POWER_250V5A	250 V 5 A	Voltage 250 V, current 5 A

DC Current Ranges

Mnemonic	Description	Setting range
DAQDA100_RANGE_MA_20MA	20 mA	-20.000-20.000 mA

SKIP Ranges

Mnemonic	Description
DAQDA100_RANGE_SKIP SKIP	(not used)

Power Connection Methods

Mnemonic	Description
DAQDA100_WIRE_1PH2W	Single-phase two-wire
DAQDA100_WIRE_1PH3W	Single phase, 3-wire (for 3-wire only)
DAQDA100_WIRE_3PH3W2I	3-phase 3-wire (for 2 voltage 2 current 3-wire only)
DAQDA100_WIRE_3PH3W3I	3-phase 3-wire (for 3 voltage 3 current 3-wire only)
DAQDA100_WIRE_3PH4W	3-phase, 4-wire (for 3-wire only)

Power Measurement Parameters

Mnemonic	Description
DAQDA100_POWERITEM_I0	$(I1+I2+I3)/3$
DAQDA100_POWERITEM_I1	Effective current 1
DAQDA100_POWERITEM_I2	Effective current 2
DAQDA100_POWERITEM_I3	Effective current 3
DAQDA100_POWERITEM_I13	$(I1+I3)/2$
DAQDA100_POWERITEM_P0	$P1+P2+P3$
DAQDA100_POWERITEM_P1	Active power 1
DAQDA100_POWERITEM_P2	Active power 2
DAQDA100_POWERITEM_P3	Active power 3
DAQDA100_POWERITEM_P13	$P1+P3$
DAQDA100_POWERITEM_PF0	$P0/(P0^2+VAR0^2)^{1/2}=P0/VA0$
DAQDA100_POWERITEM_PF1	Power factor 1
DAQDA100_POWERITEM_PF2	Power factor 2
DAQDA100_POWERITEM_PF3	Power factor 3
DAQDA100_POWERITEM_PF13	$P13/(P13^2+VAR13^2)^{1/2}=P13/VA13$
DAQDA100_POWERITEM_PH0	$\tan^{-1}(VAR0/P0)$
DAQDA100_POWERITEM_PH1	Phase 1
DAQDA100_POWERITEM_PH2	Phase 2
DAQDA100_POWERITEM_PH3	Phase 3
DAQDA100_POWERITEM_PH13	$\tan^{-1}(VAR13/P13)$
DAQDA100_POWERITEM_V0	$(V1+V2+V3)/3$
DAQDA100_POWERITEM_V1	Effective power 1
DAQDA100_POWERITEM_V2	Effective power 2
DAQDA100_POWERITEM_V3	Effective power 3
DAQDA100_POWERITEM_V13	$(V1+V3)/2$
DAQDA100_POWERITEM_VA0	$VA1+VA2+VA3$
DAQDA100_POWERITEM_VA1	Apparent power 1
DAQDA100_POWERITEM_VA2	Apparent power 2
DAQDA100_POWERITEM_VA3	Apparent power 3
DAQDA100_POWERITEM_VA13	$VA1+VA3$
DAQDA100_POWERITEM_VAR0	$VAR1+VAR2+VAR3$
DAQDA100_POWERITEM_VAR1	Reactive power 1
DAQDA100_POWERITEM_VAR2	Reactive power 2
DAQDA100_POWERITEM_VAR3	Reactive power 3
DAQDA100_POWERITEM_VAR13	$VAR1+VAR3$
DAQDA100_POWERITEM_FREQ	Frequency

25.3 DARWIN Types

DAQDA100

Type for storing the device descriptor for these functions.

Handled as a Long type in Visual Basic and as int in Visual C. Handled as an Integer type in Visual Basic.NET. Handled as the int type in C#.

Callback Type

Type	Description
Callback type	Add prefix "DLL" to the function name and write in uppercase. Example: callback type of the openDA100 function: DLLOPENDA100

The callback type is used to link the executable module (.dll) when using the Visual C.

25.4 Overview of DARWIN Constants for Loading Instantaneous Value Data

The types of constants provided are listed below.

Type	Description	Page
Constants	Channel numbers in the unit, etc.	25-2, 25-17
Numbers of items	Number of subunits, etc.	11-2, 25-17
Maximum values	Max length of the channel name string, etc.	11-2, 25-17
Boolean values	Valid (ON) setting or Invalid (OFF) setting	11-2, 25-17
Communication constants	The inst value data loading port number	25-16, 25-18
Data status values	Status of the measured data	11-3, 25-18
Alarm types	Upper-limit alarm, etc.	11-3, 25-18
Channel types/relay types	Channel or relay types	11-4, 25-19
Unit numbers	Expandable model, standalone model	11-5, 25-19

25.5 DARWIN Constants for Loading Instantaneous Value Data

Visual C/Visual C++ Constants

In Visual C/Visual C++, the constants from sections 11.2 and 25.2 are inherited. The following constants have been added.

Communication Constants

Mnemonic	Description
DAQDA100_DATAPORT	The instantaneous value data loading port number.

Constants for Visual Basic and Visual Basic.NET/C#

This section describes the mnemonics for and the meanings of the constants. For the details on the DARWIN functions, see the relevant user's manual.

In C#, it is the constant data for the DAQDA100Reader class. Prefix each constant with DAQDA100Reader (Ex.:

DAQDA100.Reader.DAQDA100READER_NUMCHANNEL).

Constants

Mnemonic	Description
DAQDA100_CHTYPE_MEASALL	Specifies all measurement channels (computation channels are not included)
DAQDA100_CHNO_ALL	Specification of all channel numbers.

Number of Items

Mnemonic	Description
DAQDA100READER_NUMCHANNEL	The number of channels.
DAQDA100READER_NUMALARM	The number of alarms.
DAQDA100READER_NUMUNIT	The number of subunits.
DAQDA100READER_NUMSLOT	The number of slots per subunit.
DAQDA100READER_NUMTERM	The number of terminals per slot (module).

Maximum Values

Mnemonic	Description
DAQDA100READER_MAXUNITLEN	Maximum length of the unit name string.

The maximum length of the string does not include the terminator (NULL).

Boolean values (Valid/Invalid Value)

Mnemonic	Description
DAQDA100READER_VALID_OFF	Invalid (OFF) value.
DAQDA100READER_VALID_ON	Valid (ON) value.

Communication Constants

Mnemonic	Description
DAQDA100READER_DATAPORT	The instantaneous value data loading port number.

Data Status Values

Mnemonic	Description
DAQDA100READER_UNKNWON	Unknown. The data status is not set.
DAQDA100READER_DATA_NORMAL	Normal status.
DAQDA100READER_DATA_DIFFINPUT	Differential input status.
DAQDA100READER_DATA_PLUSOVER	Positive overrange.
DAQDA100READER_DATA_MINUSOVER	Negative overrange.
DAQDA100READER_DATA_SKIP	SKIP status.
DAQDA100READER_DATA_ILLEGAL	Illegal data status.
DAQDA100READER_DATA_ABNORMAL	Abnormal data status.
DAQDA100READER_DATA_NODATA	No data status.
DAQDA100READER_DATA_READER	Loading instantaneous value data Status during communication.

When using the communication port for loading instantaneous value data, the status when loading and communicating instantaneous value data is the channel status in which channel information data is retrieved.

Alarm Types

◇ indicates a space.

Mnemonic	Description	String
DAQDA100READER_ALARM_NONE Alarm	OFF	◇◇ (Alarm OFF)
DAQDA100READER_ALARM_UPPER	Upper limit alarm	H◇
DAQDA100READER_ALARM_LOWER	Lower limit alarm	L◇
DAQDA100READER_ALARM_UPDIFF	Difference upper limit alarm	dH
DAQDA100READER_ALARM_LOWDIFF	Difference lower limit alarm	dL
DAQDA100READER_ALARM_INCRATE	High limit on rate-of-change alarm	RH
DAQDA100READER_ALARM_DECRATE	Low limit on rate-of-change alarm	RL

Channel/Relay types

Mnemonic	Description
DAQDA100READER_CHTYPE_MAINUNIT	Value expressing the main unit of the expandable model.
DAQDA100READER_CHTYPE_STANDALONE	Value expressing the main unit of the standalone model. Same as subunit number 0.
DAQDA100READER_CHTYPE_MATHTYPE	Value representing the computation.

Unit Number

Mnemonic	Description
DAQDA100READER_UNITNO_MAINUNIT	Main unit of the expandable model
DAQDA100READER_UNITNO_STANDALONE	Standalone model unit

The subunit number is numerical. See Channel/Relay types.

25.6 DARWIN Types for Loading Instantaneous Value Data

DAQDA100READER

Type for storing the device descriptor for these functions.

Handled as the Long type in Visual Basic and as int in Visual C. Handled as an Integer type in Visual Basic.NET. Handled as the int type in C#.

Callback Type

Type	Description
Callback type	Add prefix "DLL" to the function name and write in uppercase. Example: callback type of the openDA100Reader function: DLLOPENDA100READER

The callback type is used to link the executable module (.dll) when using the Visual C.

26.1 API Error Messages

When a function fails during execution, an error number is returned. Below is a list of error numbers, message strings, and corrective actions. They are common to the MX100 and DARWIN.

Error number	Message string	Description	Corrective Action/Remarks
0	Success	Completed normally.	-
1	Communication error	Communication error occurred.	Check the communication environment (address, cable, power to the device).
2	Timeout	Communication timeout occurred.	Check the communication environment (load condition).
3	Receive continue	Receive data continues.	Received data too long. Receive the rest of the data or check the communication procedure.
4	Creating connection is failure	Failed to create the communication descriptor.	Memory or resources may be low. Check the PC environment.
5	Creating descriptor is failure	Failed to create the device descriptor.	Memory or resources may be low. Check the PC environment.
6	Connection exists already	Communication already established.	Do not connect to a device that is already connected.
7	Not connected	Not connected.	May be executing a command without making a connection. Make a connection, then execute the command.
8	Not descriptor	No device descriptor.	The device descriptor designation may be incorrect. Specify the device descriptor.
9	Commands are not processed successfully	Failed to execute the command.	Error occurred on the measurement instrument. Check the transmitted command or the operation mode of the main unit.

26.1 API Error Messages

Error number	Message string
	Description
	Corrective Action/Remarks
10	Not acknowledge Received an unsupported response. The response from the measurement instrument is different from the expected response. Check the transmitted command or the procedure.
11	Not support Specified an unsupported function. May have specified a value outside the range. Check the values of the parameters passed to the function.
12	Not data There is no data. Invalid input to the function. The data specified in the parameters may be incorrect. Or, if the parameter is a string, the string may not be correct. Check the values of the parameters passed to the function.
13	Exception An exception occurred. A system exception may have occurred, or field may have failed to be established. Check the PC environment.

26.2 MX100 - Specific Error Messages

Below is a list of error values generated by the MX100, descriptions, and corrective actions. The function CDAQMX::getLastError or getLastErrorMX can be used to retrieve the value. With the extension API, a function (DAQMX100::getLastError or lastErrorMX100) can be used to retrieve the value.

Value	Description Corrective Action/Remarks
0	No error. -
1	Not supported because the version is different. Confirm the version number of MX100 and the API for the MX100/DARWIN.
2	The packet is too large. Confirm the packet size.
3	Unknown request. Confirm the request.
4	Inconsistent request. Confirm the packet format.
6	The session number is incorrect. Confirm the session number.
7	The FIFO number is incorrect. Confirm the FIFO number.
8	Channel number is incorrect. Confirm the channel number.
9	Data does not exist in the specified range. Confirm the data number.
10	Configuration failed. Confirm the module type and its status.
11	Failed due to the status. Confirm the mode of the MX100. Executes the request when idle.
12	Invalid request against the DO. Confirm the output setting for DO.
13	No CF card. Insert a CF card in the drive.
14	The CF card cannot be formatted (the CF card is present). The CF card may be out of order. Replace the CF card.
23	Initial balancing not correctly performed. Check the module's installation conditions, then perform initial balancing again. If the error occurs even after recalibrating, please contact your nearest Yokogawa representative.
255	Communication error. Communication error. Please contact your nearest Yokogawa representative.
256	Other error. Other error. Please contact your nearest Yokogawa representative.

Appendix 1 MX100 Terminology

This section describes the terminology related to this software and the MX100. The main terms are listed in alphabetical order.

For more details, see the MX100 user's manual.

7-Segment LED

Data indicating the display of the 7-segment LED.

See the MXSegment structure.

There are two 7-segment LEDs on the MX100. The two 7-segment LEDs must always be handled as a group.

Display Format

The display status of all segments are indicated using a display format value.

Display Pattern

The display pattern value for each 7-segment LED.

It is a hexadecimal integer value between 0 and F.

If the value is outside the range, nothing is displayed.

Display Time

Display time of the display pattern.

The unit is milliseconds.

Cannot exceed the maximum value.

Segment Number

A value used to identify the 7-segment LED position.

It is an integer value between 0 and 1.

Numbers are defined.

Alarm

Alarm function.

See the MXAlarm structure.

Alarm Level

Alarm function ID number within the channel.

It is an integer value between 1 and 2.

Numbers are defined.

Alarm Value

The threshold level for alarm activation (the ON value).

It is an integer value with the decimal point excluded.

Hysteresis

The difference between the alarm value and alarm OFF threshold value when assigning a width between the two.

It is a positive integer value with the decimal point excluded.

AO/PWM Data

Data that indicates the AO/PWM channel output.

See the MXAOPWM structure.

The MXAOPWMData structure includes all channels.

It consists of valid and invalid values and the data.

When data is retrieved, the command AO/PWM channels are shown as valid or invalid.

When data is sent, only channels specified as valid are sent.

AO/PWM Data Number (AO Data Number, PWM Data Number)

A value for identifying the AO or PWM channel position.

It is an integer value between 1 and 60.

Numbers are defined.

Output Data Value

Data values that indicate the output values that identify the instrument.

Differs depending on the range type.

As these values differ from the actual output values, they are determined using the conversion function for output values and output data values.

Note that the data values and span may differ.

Backup

Function used to record the measured data to the CF card when the communication is cut off.

Channel

The channel is represented by the channel type and channel number.

Channel numbers are determined by the slot position on the unit and the terminal position of the module installed in that slot.

Channel Name

The name is created from the unit number and the channel number within the unit.

Expressed with an integer.

In the case of strings, a five-digit decimal (ex. "0001" is used.

It is not used for this API.

Channel Number

Specified with an integer.

It is an integer value between 1 and 60.

Numbers are defined.

Example: The channel number for terminal number 2 of the module in slot number 3 is 32.

In this case the module number is "3."

Channel Range

Represents the consecutive channels of the same channel type.

It is specified using the channel type, start channel number, and end channel number.

If the end channel number is less than or equal to the start channel number, it is considered a single channel specified by the start channel number.

The channel type may be omitted.

Channel Type

A value used to identify the channel position and type.

Not required for the MX100.

However, it is set to 0 (measurement channel) for consistency with DARWIN.

Channel ID Information

A value for identifying channels using channel setup data and channel information data.

See the MXChID structure.

It includes the following information.

- Channel number
- Decimal point position
- Channel status (Boolean value)
- Channel kind
- Range type
- Scale type
- Unit name
- Tag
- Comment
- Alarm

Comment

An arbitrary string of up to 30 bytes.

The terminator is NULL.

Tag

An arbitrary string of up to 15 bytes.

The terminator is NULL.

Unit Name

An arbitrary string of up to 6 bytes.

The terminator is NULL.

Can be assigned regardless of the range configured.

Channel Information Data

Information for identifying the position within the FIFO when retrieving the measured data of the input channel.

See the MXChInfo structure.

It includes the channel identification information and the following information.

- FIFO number
- Channel sequence number in the FIFO
- Reference range (not used)
- Display range (valid range specified by the span and scale).
- Actual range (measurable range of the range type)

Channel Setup Data

Setup information for each channel.

See the MXChConfig structure.

The MXChConfigData structure includes all channels.

It includes the channel identification information and the by-type settings for each channel.

Data Identifier

The value that identifies the data values created by the user with data manipulation functions of the extension API.

It is an integer starting from 0. The above depends on the system.

It is assigned for each data type. The following types are available.

- DO data identifier
- AO/PWM data identifier
- Initial balance data identifier
- Transmission output data identifier

Device Descriptor

Value used to identify the measuring instrument.

The device descriptor is required when executing the functions.

For the API, the entity is a reference to the CDAQMX object. The DAQMX type is used for storage and designation.

For the extension API, the entity is a reference to the CDAQMX100 object.

The DAQMX100 type is used for storage and designation.

DO Data

Data that indicates the DO channel output.

See the MXDO structure.

The MXDOData structure includes all channels.

It consists of valid and invalid values and the ON/OFF values.

When data is retrieved, the command DO channels are shown as valid or invalid.

When data is sent, only channels specified as valid are sent.

DO Data Number

A value used to identify the DO channel position.

It is an integer value between 1 and 60.

Numbers are defined.

FIFO

The operation of writing measured data to the FIFO buffer.

On the MX100, the FIFO must be started to retrieve measured data.

The user can specify the starting and stopping of the FIFO. The FIFO can also be set to auto control.

In the case of the MX100, the FIFO is divided up by measurement interval type.

Each type is identified by the FIFO number.

The measurement interval and FIFO structure are as follows.

- Measured data from input modules of the same measurement interval are acquired in the order of the channel numbers of the same FIFO.
- Interval numbers (FIFO numbers) are assigned starting from the shortest measurement interval.
- Writing to the FIFO (data buffer) is handled according to the data number updated each measurement.
- Data readout is performed by specifying FIFO numbers and data numbers.
- The range of data that can be read out is provided using the data numbers.

Auto Control

The FIFO is stopped when a setup command is executed while the FIFO is running.

When auto control is enabled, the FIFO is automatically started after the execution of the setup command is complete.

Data Number

A sequential number applied to stored data in the order of measurement.

The number is different for each FIFO number.

FIFO Number

A number assigned to the FIFO.

A number is assigned to measurement interval types in order of speed.

It is an integer value between 0 and 2.

Numbers are defined.

Initial Balance Data

Data that shows the initial balance values of strain channels.

See the MXBalance structure.

The MXBalanceData structure includes all channels.

It consists of valid and invalid values, and the initial balance values.

When data is retrieved, it is displayed as the valid and invalid values of the strain channels.

When data is sent, only channels specified as valid are updated and sent.

Initial Balance Data Number

A value used to identify the strain channel position.

It is an integer value between 1 and 60.

Numbers are defined.

Measured Data

Data value information for the measurement points of each channel.

See the MXDateInfo structure.

The API allows the instantaneous values and FIFO values to be retrieved.

It includes the following information.

- Data value
- Data status values
- Alarm (presence/absence) Boolean

FIFO Value

The data values stored in the FIFO.

When retrieving measured data, specifies the range to be retrieved using the start and end data numbers.

You can retrieve data numbers from the status data.

Instantaneous Value

The newest data values stored in the FIFO.

When retrieving measured data, you can omit the data numbers indicating the range to be retrieved.

The constant for “Data number for instantaneous value specification” is also defined. Instantaneous values can be retrieved at 100 ms at the fastest.

Measured Value

The measured values are expressed using a data value and a decimal point position.

The value with an engineering unit is obtained by using the data value and the decimal point position.

Data Value

A value that expresses the mantissa of the measured value as an integer value.

Decimal Point Position

A value that expresses the exponent of the measured value.

It is an integer value between 0 and 4.

Is -1 only for the strain range of 200000 μ strain.

Module Information

Information on each module excluding the main module.

Identified by the slot position on the unit.

It consists of the following items.

- Product information
- Module type (at startup, actual)
- Number of channels
- Interval type
- AD integration time type
- Terminal type
- Valid/Invalid value
- Module version
- FIFO number

Module Number

A number used to identify the module position (slot position).

It is an integer value between 0 and 5.

Numbers are defined.

Network Information Data

Network setup information.

See the MXNetInfo structure.

Output Channel Data

Data that specifies the control method for output data using output channels (AO/PWM channels).

For details of the specified actions, see the “MX100 Data Acquisition Unit User’s Manual” (IM MX100-01E) and the “MX100 Standard Software User’s Manual” (IM-MX180-01E).

See MXOutput structure.

The MXOutputData structure includes all channels.

It is a portion of the setup data.

It consists of the following items.

- Output type
- Value selected (when idle or during error)
- User specified output value
- Pulse interval integer multiple (PWM channels only)

The user specified output values, are specified with integers in the same manner as the data value and span.

Output Channel Data Number

A value used to identify the output channel position.

It is an integer value between 1 and 60.

Numbers are defined.

Packet

In the case of the MX100, the communication is carried out using binary transmission of packets.

The packet is created inside the class and transmitted according to the request command. The response packet is received, analyzed, and the necessary data is stored in the structure and returned.

Pulse interval integer multiple

The value that indicates the pulse width using PWM channels.

Specifies the integer multiple of the resolution of the range type.

It is an integer value between 1 and 30000.

The maximum value is defined.

Reference Alarm

Shows to which alarms the DO channels correspond. The alarm is specified using the channel number and the alarm level.

Reference Channel Number

The channel used as a reference for difference computations with AI/DI channels or for remote RJC with AI channels.

It is the channel number of the transmission source for transmission output with AO/PWM channels.

Response

For every request command there is a corresponding reply. For every request command there is a corresponding reply. The response is either of the following.

- Processing was carried out normally.
- Processing was not carried out normally.

If processing is not carried out normally or execution fails, the function returns an error number. Detailed error numbers are shown in “MX-Specific Errors.” You can retrieve them separately.

RJC Voltage

Compensation voltage when using the external RJC function.

It is an integer with units mV.

Status Data

System (unit) status.

See the MXStatus structure.

It includes the following status information.

- Unit status value
- CF status information
- FIFO status information

Setup Item Numbers

These numbers are added to each item of the setup data structure to unify them.

When sending the setup data, the numbers show the detected location when an error occurs during a consistency check.

A definition file is available. See section 6.3, “MX100 Setup Item Numbers.”

Setup Data

Setup information of the MX100.

See the MXConfigData structure.

All setup data gets the basic settings, system configuration data, and status data and merges the information.

For information about the items retrieved for each data, see the data structures for each item under "Type."

It consists of the following data.

- System configuration data
- Network information data
- Status data
- Channel setup data
- Initial balance data
- Output channel data

Basic Settings

Basic settings and static information about the system from among the setup information.

System Configuration Data

The system consists of multiple modules.

See the MXSystemInfo structure.

It consists of the following items.

- Unit information
- Module information

Time Information Data

The date/time data indicating the measurement time.

Generally, this is the number of seconds from Jan. 1, 1970.

See the MXDateTime structure.

The milliseconds value is also included.

Transmission Output Data

Data that indicates the AO/PWM channel transmission status.

See the MXTransmit structure.

See AO/PWM data for more about the command AO/PWM channels.

When data is retrieved, it shows the status of the current transmission output.

When data is sent, it indicates the start and stop control of the transmission output.

Unit

Expresses the status when data is retrieved.

Used to indicate the data terminator for data retrieval.

Represents the status when the data is retrieved using the talker.

A value synthesized through logical OR computations of the flag status.

Unit Information

A unit is a single system centering on a main module.

It consists of the following items.

- Product information
- Unit type
- Style
- Temperature unit type
- Unit number
- Timeout value
- CF write mode
- Power supply frequency
- Part number

Timeout Value

Time until the saving of the measured data to the CF card is started when the connection is cut off.

It is an integer greater than or equal to 60. The unit is seconds.

See appendix 3.

Unit Number

Unit identification number. The user can specify the number.

It is an integer value between 0 and 98.

It is displayed on the 7-segment LED of the main module.

User Count

User-defined sequence information.

Sent to the main unit via communications after the settings.

The main unit is added to the data number of the acquired data.

In retrieval of measured data, you can retrieve this value at the same time as the time information data corresponding to the data number.

Appendix 2 DARWIN Terminology

This section describes the terminology related to the API and the DARWIN. The main terms are listed in alphabetical order.

For more details, see the DARWIN user's manual.

Alarm

Alarm function.

Alarm Level

Alarm function ID number within the channel.

It is an integer value between 1 and 4.

Numbers are defined.

Alarm Type

For DARWIN, it is the alarm type indicated in the alarm type list.

Alarm Value

The value at which the alarm turns ON.

It is an integer value with the decimal point excluded.

Channel

The channel is represented by the channel type and channel number.
DARWIN has measurement channels and computation channels.

Measurement Channels

Channels whose channel type represents the subunit number of the expandable model or the unit on the standalone model.

It is the input position identifying number that determines the module connection position.

The channel number is created from the slot number and the terminal number.
The number may not be consecutive depending on the system configuration.

Computation Channels

The channel when the channel type is a value that represents "computation."

The computation channels can be used on models with the computation option.

On the standalone model, the channel number is an integer from 1 to 30.

On the expandable model, the channel number is an integer from 1 to 60.

Channel Type

A value used to identify the channel position and type.

For DARWIN, it is a value used to identify the channel position and type.

See Channel/Relay types.

Either the subunit number of the expandable model, the unit of the standalone model, or computation.

Channel Number

Specified with an integer.

It is an integer value between 1 and 60.

Example: The channel number for terminal number 2 of the 10-CH, Medium-Speed Universal Input Module in slot number 3 is 32.

For computation channels, it is an integer value between 1 and 60 on the expandable model and 1 and 30 on the standalone model.

Channel Range

Represents the consecutive channels of the same channel type.

It is specified using the channel type, start channel number, and end channel number.

If the end channel number is something other than the start channel number, it is considered a single channel specified by the start channel number.

Channel Information Data

Static information such as the channel type, channel number, and decimal point position for each channel.

See the DarwinChInfo structure.

The data can be retrieved using the output of the channel information data by the talker.

Unit Name

An arbitrary string of up to 6 bytes.

Channel Status

Uses a value that is common with the data status.

See the data status value.

Device Descriptor

Value used to identify the measuring instrument.

The device descriptor is required when executing the functions.

For the API, the entity is a reference to the CDAQDARWIN object.

DAQDARWIN type is used for storage and designation.

For the extension API, the entity is a reference to the CDAQDA100 object.

CDAQDA100 type is used for storage and designation.

Measured Data

Data value information for the measurement points of each channel.

See the DarwinDataInfo structure.

The measured data can be retrieved using the output of the measured data by the talker.

The alarm presence/absence is expressed by the alarm type.

Data Status

Uses a value that is common with the channel status. See the data status value.

Measured Value

The measured value is expressed by the data value and the decimal point position.

The engineering unit is obtained from the data value and the decimal point position.

Data Value

A value that expresses the mantissa of the measured value as an integer value.

Decimal Point Position

A value that expresses the exponent of the measured value.

It is an integer value between 0 and 4.

Measurement Interval

The measurement interval.

The unit is seconds.

Relay

The relay is represented by the relay type and relay number.

Relay Type

A value used to identify the relay position and type. See Channel/Relay types.

Either the main unit and subunit number of the expandable model, or internal switch.

Relay Number

Noncontinuous between relay types.

It is an integer value between 1 and 60.

If the relay type is a unit number, the number represents the relay position ID number determined by the position where the module is connected. Consists of the slot number and the terminal number. The number may not be consecutive depending on the system configuration.

Response

After a command is sent, a response is received from the measurement instrument. Unless the data output request is made as a talker, any of the following is received.

- Processing was carried out normally.
- Processing was not carried out normally.

If processing is not carried out normally or execution fails, the function returns an error number.

Scale

Consists of the left value, right value, and decimal point position. The value is in the range from -30000 to 30000.

With the specification of this API, if the left and right values are the same, they are considered omitted.

Span

Consists of the left value and right value. The values vary depending on the type of measurement range.

With the specification of this API, if the left and right values are the same, they are considered omitted.

Status Byte

A value indicating the status of the instrument. A value synthesized through logical OR computations of the status byte.

System Configuration Data

Unit and module information.
See the DarwinSystemInfo structure.

Unit Number

A number identifying the unit in the system configuration.
See Channel/Relay types and Unit Number.
It is either the main unit or subunit number.
On the standalone model, the only valid unit number is 0.

Subunit Number

A number used to identify the subunit connected on the expandable model.
It is an integer value between 0 and 5.
On the standalone model, this number is the same as the unit number, and the only valid subunit number is 0.

Slot Number

Represents the position where the module is connected for each unit.
It is an integer value between 0 and 5.

Terminal Number

A number used to identify the position of the channel/relay in the module connected to the slot.
It is an integer value between 1 and 10.
Differs depending on the module type.

Talker

A function that performs data output. The types indicated by the talker function type are supported.
Since the data of multiple channels or lines are output, data retrieval commands are executed by channel or by line after the start command is executed.

Terminator

String indicating the end of the command.

Time Information Data

Date/Time data indicating the measurement time. Generally, this is the number of seconds from Jan. 1, 1970.

See the DarwinDateTime structure.

The milliseconds value is not used.

Milliseconds is used when using the instantaneous value data loading communication port.

Unit

Expresses the status when data is retrieved.

Used to indicate the data terminator for data retrieval.

A value synthesized through logical OR computations of the flag status.

Appendix 3 Calculation of the MX100 Timeout Value

The timeout value is the time until the saving of the data sampled at the measurement interval to the CF card is started when the connection is cut off. It is an integer greater than or equal to 60. The unit is seconds. The default value is 60 s. To save the sampled data to the CF card without dropouts when the communication is cut off, the data must be saved to the CF card before the unsaved data in the FIFO buffer is overwritten. If the value is not appropriate, an error will occur when saving to the CF card starts, and the sampled data will not be saved. Refer to the calculation method of the timeout value shown below.

Timeout value (s) < (FIFO buffer size/data size per second) - 20 s

FIFO buffer size = 2 MB (2097152 bytes)

FIFO data size per second = (the byte size of time + the byte size of the measured value × the number of channels) × the number of measurements in 1 second

The byte size of time = 16 bytes

The byte size of the measured value = 4 bytes

20 s: Time required to save the data to the CF card (estimate)

Calculation Example 1

FIFO: 60 channels and scan interval of 10 ms

Data size per second = $(16 + 4 \times 60 \text{ ch}) \times 1000/10 = 25600$ bytes

Setting = $(2 \text{ MB}/25600 \text{ bytes}) - 20 \text{ s} = 61.9 > 60 \text{ s}$ (estimated setting)

Calculation Example 2

When three FIFOs below are running

FIFO: 40 channels and scan interval of 10 ms, 4 channels and scan interval of 50 ms, and 10 channels and scan interval of 100 ms

Data size per second = $(16 + 4 \times 40 \text{ ch}) \times 1000/10 = 17600$ bytes

Data size per second = $(16 + 4 \times 4 \text{ ch}) \times 1000/50 = 640$ bytes

Data size per second = $(16 + 4 \times 10 \text{ ch}) \times 1000/100 = 560$ bytes

Setting = $(2 \text{ MB}/(17600 + 640 + 560 \text{ bytes})) - 20 \text{ s} = 91.5 > 90 \text{ s}$ (estimated setting)

Appendix 4 API Revision History

The following lists the changes and deletions that were made to functions when the style was changed from 1 to 2.

Visual C and Visual Basic Functions

New MX100 Functions

changeAOPWMDDataMX
changeBalanceMX
changeTransmitMX
getAlarmNameMX (Visual C only)
getAOPWMDDataMX
getBalanceMX
getItemErrorMX
getMaxLenAlarmNameMX
getOutputMX
isDataNoMX (Visual C only)
isDataNoVBMX
resetBalanceMX
runBalanceMX
setAOMX
setAOPWMDDataMX
setAOTypeMX
setBalanceMX
setChoiceMX
setOutputMX
setOutputTypeMX
setPulseTimeMX
setPWMMX
setPWMTTypeMX
setRESMX
setSTRAINMX
setTransmitMX
toAlarmNameMX
toAOPWMValueMX
toRealValueMX
toStyleVersionMX

New DARWIN Functions

computeDARWIN
establishDARWIN
getAlarmNameDARWIN (Visual C only)
getMaxLenAlarmNameDARWIN
getReportStatusDARWIN
receiveByteDARWIN
reportingDARWIN
setMADARWIN
setPOWERDARWIN
setPULSEDARWIN
setSTRAINDARWIN

Visual C++ Functions

New MX100 Classes

CDAQMXAOPWMDData Class
CDAQMXBalanceData Class
CDAQMXBalanceResult Class
CDAQMXOutputData Class
CDAQMXTransmit Class

New MX100 Members

CDAQMX::clearLastDataNoCh
CDAQMX::clearLastDataNoFIFO
CDAQMX::getAOPWMDData
CDAQMX::getBalance
CDAQMX::getItemError
CDAQMX::getOutput
CDAQMX::getPacketVersion
CDAQMX::isObject
CDAQMX::m_bTalkChInfo
CDAQMX::m_bTalkConfig
CDAQMX::m_bTalkData
CDAQMX::m_nItemError
CDAQMX::m_nTimeNum
CDAQMX::m_packetVer
CDAQMX::receiveBuffer
CDAQMX::resetBalance
CDAQMX::runBalance
CDAQMX::runPacket
CDAQMX::setAOPWMDData
CDAQMX::setBalance
CDAQMX::setOutput
CDAQMX::setTransmit
CDAQMXChConfig::getItemError
CDAQMXChConfig::getRangeMax
CDAQMXChConfig::getRangeMin
CDAQMXChConfig::getRangePoint
CDAQMXChConfig::initMXChConfig
CDAQMXChConfig::isObject
CDAQMXChConfig::m_nItemError

CDAQMXChConfig::setAO
CDAQMXChConfig::setPWM
CDAQMXChConfig::setRES
CDAQMXChConfig::setSTRAIN
CDAQMXChConfigData::getItemError
CDAQMXChConfigData::initMXChConfigData
CDAQMXChConfigData::isObject
CDAQMXChConfigData::m_cMXChConfig
CDAQMXChConfigData::m_nItemError
CDAQMXChID::getChName
CDAQMXChID::initMXChID
CDAQMXChID::isObject
CDAQMXChID::toChName
CDAQMXChID::toChNo
CDAQMXChID::toUnitNo
CDAQMXChInfo::initMXChInfo
CDAQMXChInfo::isObject
CDAQMXConfig::getChName
CDAQMXConfig::getClassMXBalanceData
CDAQMXConfig::getClassMXChConfig
CDAQMXConfig::getClassMXOutputData
CDAQMXConfig::getItemError
CDAQMXConfig::getRangePoint
CDAQMXConfig::getSpanPoint
CDAQMXConfig::initMXConfigData
CDAQMXConfig::isObject
CDAQMXConfig::m_cMXBalanceData
CDAQMXConfig::m_cMXOutputData
CDAQMXConfig::m_nItemError
CDAQMXConfig::setAO
CDAQMXConfig::setAOType
CDAQMXConfig::setChKind
CDAQMXConfig::setPWM
CDAQMXConfig::setPWMTType
CDAQMXConfig::setRES
CDAQMXConfig::setSTRAIN
CDAQMXDataInfo::initMXDataInfo
CDAQMXDataInfo::isObject
CDAQMXDateTime::initMXDateTime
CDAQMXDateTime::isObject

CDAQMXDODData::initMXDODData
CDAQMXDODData::isObject
CDAQMXDODData::setDOONOFF
CDAQMXNetInfo::getPart
CDAQMXNetInfo::initMXNetInfo
CDAQMXNetInfo::isObject
CDAQMXSegment::initMXSegment
CDAQMXSegment::isObject
CDAQMXStatus::getDateTime
CDAQMXStatus::getMilliSecond
CDAQMXStatus::getTime
CDAQMXStatus::initMXStatus
CDAQMXStatus::isBackup
CDAQMXStatus::isDataNo
CDAQMXStatus::isObject
CDAQMXSysInfo::getItemError
CDAQMXSysInfo::initMXSystemInfo
CDAQMXSysInfo::isObject
CDAQMXSysInfo::m_nItemError

Deleted MX100 Members

Deleted due to version incompatibility.
CDAQMXChConfig::setFromBlockChConfig
CDAQMXChConfigData::setFromAckGetConfigPacket
CDAQMXChInfo::setFromBlockChInfo
CDAQMXConfig::setFromAckGetConfigPacket
CDAQMXDataInfo::setFromBlockData
CDAQMXNetInfo::setFromAckGetConfigPacket
CDAQMXStatus::setFromAckGetConfigPacket
CDAQMXStatus::setFromAckGetStatusPacket
CDAQMXSysInfo::setFromAckGetConfigPacket
CDAQMXSysInfo::setFromAckGetUnitInfoPacket

New DARWIN Members

CDAQDARWIN::compute
CDAQDARWIN::establish
CDAQDARWIN::getReportStatus
CDAQDARWIN::isObject
CDAQDARWIN::receiveByte
CDAQDARWIN::reporting
CDAQDARWIN::setMA
CDAQDARWIN::setSTRAIN
CDAQDARWIN::setPULSE
CDAQDARWIN::setPOWER
CDAQDARWINChInfo::initDarwinChInfo
CDAQDARWINChInfo::isObject
CDAQDARWINDataInfo::getMaxLenAlarmName
CDAQDARWINDataInfo::initDarwinDataInfo
CDAQDARWINDataInfo::isObject
CDAQDARWINDateTime::getFullYear
CDAQDARWINDateTime::initDarwinDateTime
CDAQDARWINDateTime::isObject
CDAQDARWINSysInfo::getModuleCode
CDAQDARWINSysInfo::initDarwinSystemInfo
CDAQDARWINSysInfo::isObject

Index

Symbols

.bas	1-5
.cs	1-5
.dll	1-5
.h	1-5
.lib	1-5
.txt	1-5
.vb	1-5
7-Segment LED	App-1

A

A/D Integral Time Types	18-8
A/D Integration Time Type	6-7
Adding the Path of the Include File	2-9
Adding the Path to the Include File	3-6, 7-5, 8-4, 13-12, 19-7, 19-12, 20-6, 20-10
Alarm	App-2, App-13
Alarm Level	App-2, App-13
Alarm Type	11-3, App-13
Alarm Types	6-5, 18-6, 25-7, 25-18
Alarm Value	App-2, App-13
AO Ranges	6-15, 18-16
AO/PWM Data	App-2
AO/PWM Data Number	App-2
API Revision History	App-20
Auto Control	App-6

B

Backup	App-3
Basic Settings	6-24, App-11
Boolean Value	6-4
Boolean Value (valid/invalid)	11-2, 18-5
Burnout Types	6-7, 18-8

C

Callback	18-18
Callback Type	25-14
CDAQChInfo Class	2-15
CDAQChInfo::CDAQChInfo	2-16
CDAQChInfo::getChNo	2-16
CDAQChInfo::getChType	2-16
CDAQChInfo::getPoint	2-17
CDAQChInfo::initialize	2-17
CDAQChInfo::isObject	2-17
CDAQChInfo::operator=	2-17
CDAQChInfo::setChNo	2-18
CDAQChInfo::setChType	2-18
CDAQChInfo::setPoint	2-18
CDAQDA100::ackAlarm	19-17
CDAQDA100::CDAQDA100	19-18
CDAQDA100::chNumMax	19-18
CDAQDA100::chNumMaxReport	19-19
CDAQDA100::getByte	19-19
CDAQDA100::getChannel	19-19
CDAQDA100::getClassDataBuffer	19-20
CDAQDA100::getClassSysInfo	19-20
CDAQDA100::getCode	19-20
CDAQDA100::getData	19-21
CDAQDA100::getInfoCh	19-21

CDAQDA100::getInstChASCII	19-22
CDAQDA100::getInstChBINARY	19-22
CDAQDA100::getReport	19-23
CDAQDA100::getRevisionDA100DLL	19-23
CDAQDA100::getVersionDA100DLL	19-23
CDAQDA100::initSetValue	19-23
CDAQDA100::isObject	19-24
CDAQDA100::mathInfoCh	19-24
CDAQDA100::mathInstCh	19-25
CDAQDA100::measClear	19-25
CDAQDA100::measInfoCh	19-26
CDAQDA100::measInstCh	19-27
CDAQDA100::open	19-27
CDAQDA100::reconstruct	19-28
CDAQDA100::setChAlarm	19-28
CDAQDA100::setChDELTA	19-29
CDAQDA100::setChRRJC	19-30
CDAQDA100::setChUnit	19-31
CDAQDA100::setDateTime	19-31
CDAQDA100::setRange	19-32
CDAQDA100::switchCode	19-33
CDAQDA100::switchCompute	19-33
CDAQDA100::switchMode	19-33
CDAQDA100::switchReport	19-34
CDAQDA100::talkCalibrationChData	19-34
CDAQDA100::talkOperationChData	19-35
CDAQDA100::talkSetupChData	19-35
CDAQDA100::updateAll	19-36
CDAQDA100::updateChInfo	19-36
CDAQDA100::updateRenew	19-36
CDAQDA100::updateReportStatus	19-37
CDAQDA100::updateStatus	19-37
CDAQDA100::updateSystemConfig	19-37
CDAQDA100Reader::CDAQDA100Reader	19-40
CDAQDA100Reader::getInfoCh	19-41
CDAQDA100Reader::getInstCh	19-41
CDAQDA100Reader::isObject	19-42
CDAQDA100Reader::measInfoCh	19-42
CDAQDA100Reader::measInstCh	19-43
CDAQDA100Reader::open	19-43
CDAQDARWIN Class	7-13
CDAQDARWIN::CDAQDARWIN	7-15
CDAQDARWIN::checkAck	7-16
CDAQDARWIN::compute	7-16
CDAQDARWIN::establish	7-17
CDAQDARWIN::getChannel	7-17
CDAQDARWIN::getChDataByASCII	7-18
CDAQDARWIN::getChDataByBinary	7-18
CDAQDARWIN::getChInfo	7-19
CDAQDARWIN::getData	7-19
CDAQDARWIN::getRevisionDLL	7-20
CDAQDARWIN::getSetDataByLine	7-21
CDAQDARWIN::getStatusByte	7-21
CDAQDARWIN::getSystemConfig	7-22
CDAQDARWIN::getVersionDLL	7-22
CDAQDARWIN::initSystem	7-22
CDAQDARWIN::isObject	7-23
CDAQDARWIN::open	7-23
CDAQDARWIN::receiveByte	7-24
CDAQDARWIN::reporting	7-24
CDAQDARWIN::runCommand	7-25
CDAQDARWIN::sendTrigger	7-25
CDAQDARWIN::setAlarm	7-26
CDAQDARWIN::setDateTime	7-27

Index

CDAQDARWIN::setDELTA	7-27	CDAQDARWINDateTime Class	7-61
CDAQDARWIN::setDI	7-28	CDAQDARWINDateTime::CDAQDARWINDateTime	7-63
CDAQDARWIN::setMA	7-29	CDAQDARWINDateTime::getDarwinDateTime	7-63
CDAQDARWIN::setPOWER	7-30	CDAQDARWINDateTime::getDay	7-63
CDAQDARWIN::setPULSE	7-31	CDAQDARWINDateTime::getFullYear	7-64
CDAQDARWIN::setRRJC	7-32	CDAQDARWINDateTime::getHour	7-64
CDAQDARWIN::setRTD	7-33	CDAQDARWINDateTime::getMinute	7-64
CDAQDARWIN::setScallingUnit	7-34	CDAQDARWINDateTime::getMonth	7-64
CDAQDARWIN::setSKIP	7-34	CDAQDARWINDateTime::getSecond	7-65
CDAQDARWIN::setSTRAIN	7-35	CDAQDARWINDateTime::getYear	7-65
CDAQDARWIN::setTC	7-36	CDAQDARWINDateTime::initDarwinDateTime	7-65
CDAQDARWIN::setVOLT	7-37	CDAQDARWINDateTime::initialize	7-65
CDAQDARWIN::startTalker	7-37	CDAQDARWINDateTime::isObject	7-66
CDAQDARWIN::talkCalibrationData	7-38	CDAQDARWINDateTime::operator=	7-66
CDAQDARWIN::talkChInfo	7-38	CDAQDARWINDateTime::setByte	7-67
CDAQDARWIN::talkDataByASCII	7-39	CDAQDARWINDateTime::setDarwinDateTime	7-67
CDAQDARWIN::talkDataByBinary	7-40	CDAQDARWINDateTime::setLine	7-68
CDAQDARWIN::talkOperationData	7-41	CDAQDARWINDateTime::setNow	7-68
CDAQDARWIN::talkSetupData	7-41	CDAQDARWINDateTime::toDateTime	7-68
CDAQDARWIN::transMode	7-42	CDAQDARWINDateTime::toString	7-69
CDAQDARWINChInfo Class	7-43	CDAQDARWINSysInfo Class	7-70
CDAQDARWINChInfo::CDAQDARWINChInfo	7-45	CDAQDARWINSysInfo::CDAQDARWINSysInfo	7-71
CDAQDARWINChInfo::getChName	7-45	CDAQDARWINSysInfo::getDarwinModuleInfo	7-71
CDAQDARWINChInfo::getChStatus	7-46	CDAQDARWINSysInfo::getDarwinSystemInfo	7-72
CDAQDARWINChInfo::getDarwinChInfo	7-46	CDAQDARWINSysInfo::getDarwinUnitInfo	7-72
CDAQDARWINChInfo::getStatusName	7-46	CDAQDARWINSysInfo::getInterval	7-72
CDAQDARWINChInfo::getUnit	7-46	CDAQDARWINSysInfo::getModuleCode	7-73
CDAQDARWINChInfo::initDarwinChInfo	7-47	CDAQDARWINSysInfo::getModuleName	7-73
CDAQDARWINChInfo::initialize	7-47	CDAQDARWINSysInfo::initDarwinSystemInfo	7-73
CDAQDARWINChInfo::isObject	7-47	CDAQDARWINSysInfo::initialize	7-74
CDAQDARWINChInfo::operator=	7-48	CDAQDARWINSysInfo::isExist	7-74
CDAQDARWINChInfo::setChStatus	7-48	CDAQDARWINSysInfo::isObject	7-74
CDAQDARWINChInfo::setDarwinChInfo	7-48	CDAQDARWINSysInfo::operator=	7-75
CDAQDARWINChInfo::setLine	7-49	CDAQDARWINSysInfo::setDarwinSystemInfo	7-75
CDAQDARWINChInfo::setUnit	7-49	CDAQDARWINSysInfo::setLine	7-76
CDAQDARWINChInfo::toChName	7-50	CDAQDARWINSysInfo::toRelayName	7-76
CDAQDARWINChInfo::toChRange	7-50	CDAQDataInfo Class	2-19
CDAQDARWINChInfo::toChType	7-51	CDAQDataInfo::CDAQDataInfo	2-20
CDAQDARWINChInfo::toFlag	7-51	CDAQDataInfo::getClassChInfo	2-20
CDAQDARWINChInfo::toStatus	7-52	CDAQDataInfo::getDoubleValue	2-20
CDAQDARWINDataBuffer::CDAQDARWINDataBuffer ..	19-45	CDAQDataInfo::getStringValue	2-21
CDAQDARWINDataBuffer::getClassDARWINChInfo	19-45	CDAQDataInfo::getValue	2-21
CDAQDARWINDataBuffer::getClassDARWINDataInfo ..	19-45	CDAQDataInfo::initialize	2-21
CDAQDARWINDataBuffer::getClassDARWINDateTime ..	19-45	CDAQDataInfo::isObject	2-22
CDAQDARWINDataBuffer::initialize	19-46	CDAQDataInfo::operator=	2-22
CDAQDARWINDataBuffer::isAlarm	19-46	CDAQDataInfo::setClassChInfo	2-22
CDAQDARWINDataBuffer::setChInfo	19-46	CDAQDataInfo::setValue	2-23
CDAQDARWINDataBuffer::setDataInfo	19-47	CDAQDataInfo::toDoubleValue	2-23
CDAQDARWINDataBuffer::setDateTime	19-47	CDAQDataInfo::toStringValue	2-23
CDAQDARWINDataInfo Class	7-53	CDAQDateTime Class	2-24
CDAQDARWINDataInfo::CDAQDARWINDataInfo	7-55	CDAQDateTime::CDAQDateTime	2-25
CDAQDARWINDataInfo::getAlarm	7-55	CDAQDateTime::getMilliSecond	2-25
CDAQDARWINDataInfo::getAlarmName	7-56	CDAQDateTime::getTime	2-25
CDAQDARWINDataInfo::getClassDARWINChInfo	7-56	CDAQDateTime::initialize	2-25
CDAQDARWINDataInfo::getDarwinDataInfo	7-56	CDAQDateTime::isObject	2-26
CDAQDARWINDataInfo::getMaxLenAlarmName	7-57	CDAQDateTime::operator=	2-26
CDAQDARWINDataInfo::getStatus	7-57	CDAQDateTime::setMilliSecond	2-26
CDAQDARWINDataInfo::initDarwinDataInfo	7-57	CDAQDateTime::setNow	2-26
CDAQDARWINDataInfo::initialize	7-57	CDAQDateTime::setTime	2-27
CDAQDARWINDataInfo::isObject	7-58	CDAQDateTime::toLocalDateTime	2-27
CDAQDARWINDataInfo::operator=	7-58	CDAQHandler Class	2-28
CDAQDARWINDataInfo::setAlarm	7-58	CDAQHandler::CDAQHandler	2-29
CDAQDARWINDataInfo::setByte	7-59	CDAQHandler::close	2-30
CDAQDARWINDataInfo::setClassDARWINChInfo	7-59	CDAQHandler::getChannel	2-30
CDAQDARWINDataInfo::setDarwinDataInfo	7-59	CDAQHandler::getData	2-31
CDAQDARWINDataInfo::setLine	7-60	CDAQHandler::getErrorMessage	2-31
CDAQDARWINDataInfo::setStatus	7-60	CDAQHandler::getMaxLenErrorMessage	2-32
CDAQDARWINDataInfo::toAlarmType	7-60	CDAQHandler::getRevisionAPI	2-32

CDAQHandler::getRevisionDLL	2-32	CDAQMX::setUserTime	2-67
CDAQHandler::getVersionAPI	2-32	CDAQMX::startFIFO	2-67
CDAQHandler::getVersionDLL	2-33	CDAQMX::stopFIFO	2-67
CDAQHandler::isObject	2-33	CDAQMX::talkChData	2-68
CDAQHandler::open	2-33	CDAQMX::talkChInfo	2-68
CDAQHandler::receive	2-34	CDAQMX::talkConfig	2-69
CDAQHandler::receiveLine	2-35	CDAQMX::talkFIFOData	2-69
CDAQHandler::receiveRemain	2-35	CDAQMXAOPWMDData Class	2-70
CDAQHandler::send	2-36	CDAQMXAOPWMDData::CDAQMXAOPWMDData	2-71
CDAQHandler::sendLine	2-36	CDAQMXAOPWMDData::getAOPWMValid	2-71
CDAQMX Class	2-37	CDAQMXAOPWMDData::getAOPWMValue	2-72
CDAQMX::autoFIFO	2-41	CDAQMXAOPWMDData::getMXAOPWM	2-72
CDAQMX::CDAQMX	2-41	CDAQMXAOPWMDData::getMXAOPWMDData	2-72
CDAQMX::clearAttr	2-42	CDAQMXAOPWMDData::initialize	2-72
CDAQMX::clearData	2-42	CDAQMXAOPWMDData::initMXAOPWMDData	2-73
CDAQMX::clearLastDataNoCh	2-42	CDAQMXAOPWMDData::isObject	2-73
CDAQMX::clearLastDataNoFIFO	2-42	CDAQMXAOPWMDData::operator=	2-73
CDAQMX::formatCF	2-43	CDAQMXAOPWMDData::setAOPWM	2-74
CDAQMX::getAOPWMDData	2-43	CDAQMXAOPWMDData::setMXAOPWMDData	2-74
CDAQMX::getBalance	2-44	CDAQMXAOPWMDData::toAOPWMValue	2-74
CDAQMX::getChannel	2-44	CDAQMXAOPWMDData::toRealValue	2-75
CDAQMX::getChConfig	2-45	CDAQMXAOPWMDList Class	12-62
CDAQMX::getChData	2-46	CDAQMXAOPWMDList::add	12-63
CDAQMX::getChDataNo	2-46	CDAQMXAOPWMDList::CDAQMXAOPWMDList	12-63
CDAQMX::getChInfo	2-47	CDAQMXAOPWMDList::change	12-64
CDAQMX::getConfig	2-47	CDAQMXAOPWMDList::copy	12-64
CDAQMX::getData	2-48	CDAQMXAOPWMDList::copyData	12-65
CDAQMX::getDataNo	2-48	CDAQMXAOPWMDList::create	12-65
CDAQMX::getDODData	2-49	CDAQMXAOPWMDList::getClassMXAOPWMDData	12-65
CDAQMX::getFIFODataNo	2-49	CDAQMXAOPWMDList::getCurrent	12-66
CDAQMX::getItemError	2-49	CDAQMXAOPWMDList::initCurrent	12-66
CDAQMX::getLastError	2-50	CDAQMXBalanceData Class	2-76
CDAQMX::getMXConfig	2-50	CDAQMXBalanceData::CDAQMXBalanceData	2-77
CDAQMX::getNo	2-50	CDAQMXBalanceData::getBalanceValid	2-77
CDAQMX::getOutput	2-51	CDAQMXBalanceData::getBalanceValue	2-78
CDAQMX::getPacketVersion	2-51	CDAQMXBalanceData::getMXBalance	2-78
CDAQMX::getRevisionDLL	2-51	CDAQMXBalanceData::getMXBalanceData	2-78
CDAQMX::getStatusData	2-52	CDAQMXBalanceData::initialize	2-79
CDAQMX::getSystemConfig	2-52	CDAQMXBalanceData::isObject	2-79
CDAQMX::getTimeData	2-53	CDAQMXBalanceData::operator=	2-79
CDAQMX::getUserTime	2-53	CDAQMXBalanceData::setBalance	2-80
CDAQMX::getVersionDLL	2-53	CDAQMXBalanceData::setMXBalanceData	2-80
CDAQMX::incCurDataNo	2-54	CDAQMXBalanceList Class	12-67
CDAQMX::incCurFIFOIdx	2-54	CDAQMXBalanceList::add	12-68
CDAQMX::initSystem	2-54	CDAQMXBalanceList::CDAQMXBalanceList	12-68
CDAQMX::isObject	2-55	CDAQMXBalanceList::change	12-69
CDAQMX::nop	2-55	CDAQMXBalanceList::copy	12-69
CDAQMX::open	2-56	CDAQMXBalanceList::copyData	12-70
CDAQMX::receiveBlock	2-56	CDAQMXBalanceList::create	12-70
CDAQMX::receiveBuffer	2-57	CDAQMXBalanceList::getClassMXBalanceData	12-70
CDAQMX::receivePacket	2-57	CDAQMXBalanceList::getCurrent	12-71
CDAQMX::registry	2-58	CDAQMXBalanceList::initCurrent	12-71
CDAQMX::resetBalance	2-58	CDAQMXBalanceResult Class	2-81
CDAQMX::runBalance	2-59	CDAQMXBalanceResult::CDAQMXBalanceResult	2-82
CDAQMX::runCommand	2-59	CDAQMXBalanceResult::getMXBalanceResult	2-83
CDAQMX::runPacket	2-60	CDAQMXBalanceResult::getResult	2-83
CDAQMX::searchChNo	2-60	CDAQMXBalanceResult::initialize	2-83
CDAQMX::sendPacket	2-61	CDAQMXBalanceResult::initMXBalanceData	2-80
CDAQMX::setAOPWMDData	2-61	CDAQMXBalanceResult::initMXBalanceResult	2-83
CDAQMX::setBackup	2-62	CDAQMXBalanceResult::isObject	2-84
CDAQMX::setBalance	2-62, 2-63	CDAQMXBalanceResult::operator=	2-84
CDAQMX::setConfig	2-63	CDAQMXBalanceResult::setMXBalanceResult	2-85
CDAQMX::setDateTime	2-64	CDAQMXBalanceResult::setResult	2-85
CDAQMX::setDODData	2-64	CDAQMXChConfig Class	2-86
CDAQMX::setMXConfig	2-65	CDAQMXChConfig::CDAQMXChConfig	2-88
CDAQMX::setOutput	2-65	CDAQMXChConfig::changeRange	2-89
CDAQMX::setSegment	2-66	CDAQMXChConfig::getBurnout	2-89
CDAQMX::setTransmit	2-66	CDAQMXChConfig::getFilter	2-89

Index

CDAQMXChConfig::getItemError	2-89	CDAQMXChID::getTag	2-116
CDAQMXChConfig::getMXChConfig	2-90	CDAQMXChID::getUnit	2-116
CDAQMXChConfig::getRangeMax	2-90	CDAQMXChID::initialize	2-116
CDAQMXChConfig::getRangeMin	2-91	CDAQMXChID::initMXChID	2-117
CDAQMXChConfig::getRangePoint	2-91	CDAQMXChID::isObject	2-117
CDAQMXChConfig::getRefChNo	2-91	CDAQMXChID::isValid	2-117
CDAQMXChConfig::getRJCType	2-92	CDAQMXChID::operator=	2-118
CDAQMXChConfig::getRJCVolt	2-92	CDAQMXChID::setAlarmValue	2-118
CDAQMXChConfig::getScaleMax	2-92	CDAQMXChID::setChType	2-118
CDAQMXChConfig::getScaleMin	2-92	CDAQMXChID::setComment	2-119
CDAQMXChConfig::getSpanMax	2-93	CDAQMXChID::setMXChID	2-119
CDAQMXChConfig::getSpanMin	2-93	CDAQMXChID::setTag	2-119
CDAQMXChConfig::initialize	2-93	CDAQMXChID::setType	2-120
CDAQMXChConfig::initMXChConfig	2-93	CDAQMXChID::setUnit	2-120
CDAQMXChConfig::isCorrect	2-94	CDAQMXChID::setValid	2-120
CDAQMXChConfig::isDeenergize	2-94	CDAQMXChID::toChName	2-120
CDAQMXChConfig::isHold	2-94	CDAQMXChID::toChNo	2-121
CDAQMXChConfig::isObject	2-95	CDAQMXChID::toUnitNo	2-121
CDAQMXChConfig::isRefAlarm	2-95	CDAQMXChInfo Class	2-122
CDAQMXChConfig::operator=	2-96	CDAQMXChInfo::CDAQMXChInfo	2-124
CDAQMXChConfig::setAlarm	2-96	CDAQMXChInfo::getDisplayMax	2-124
CDAQMXChConfig::setAO	2-97	CDAQMXChInfo::getDisplayMin	2-124
CDAQMXChConfig::setBurnout	2-97	CDAQMXChInfo::getFIFOIndex	2-124
CDAQMXChConfig::setDeenergize	2-97	CDAQMXChInfo::getFIFONo	2-125
CDAQMXChConfig::setDELTA	2-98	CDAQMXChInfo::getMXChInfo	2-125
CDAQMXChConfig::setDI	2-98	CDAQMXChInfo::getOriginalMax	2-125
CDAQMXChConfig::setFilter	2-99	CDAQMXChInfo::getOriginalMin	2-125
CDAQMXChConfig::setHold	2-99	CDAQMXChInfo::getRealMax	2-126
CDAQMXChConfig::setMXChConfig	2-99	CDAQMXChInfo::getRealMin	2-126
CDAQMXChConfig::setPWM	2-100	CDAQMXChInfo::initialize	2-126
CDAQMXChConfig::setRefAlarm	2-100	CDAQMXChInfo::initMXChInfo	2-126
CDAQMXChConfig::setRefChNo	2-101	CDAQMXChInfo::isObject	2-127
CDAQMXChConfig::setRES	2-101	CDAQMXChInfo::operator=	2-127
CDAQMXChConfig::setRJCType	2-101	CDAQMXChInfo::setFIFOIndex	2-127
CDAQMXChConfig::setRTD	2-102	CDAQMXChInfo::setFIFONo	2-128
CDAQMXChConfig::setScalling	2-102	CDAQMXChInfo::setMXChInfo	2-128
CDAQMXChConfig::setSKIP	2-103	CDAQMXConfig Class	2-129
CDAQMXChConfig::setSpan	2-103	CDAQMXConfig::CDAQMXConfig	2-131
CDAQMXChConfig::setSTRAIN	2-103	CDAQMXConfig::getChName	2-131
CDAQMXChConfig::setTC	2-104	CDAQMXConfig::getClassMXBalanceData	2-131
CDAQMXChConfig::setVOLT	2-104	CDAQMXConfig::getClassMXChConfig	2-132
CDAQMXChConfigData class	2-105	CDAQMXConfig::getClassMXChConfigData	2-132
CDAQMXChConfigData::CDAQMXChConfigData	2-106	CDAQMXConfig::getClassMXNetInfo	2-132
CDAQMXChConfigData::changeRange	2-106	CDAQMXConfig::getClassMXOutputData	2-132
CDAQMXChConfigData::getClassMXChConfig	2-107	CDAQMXConfig::getClassMXStatus	2-133
CDAQMXChConfigData::getItemError	2-107	CDAQMXConfig::getClassMXSysInfo	2-133
CDAQMXChConfigData::getMXChConfigData	2-107	CDAQMXConfig::getItemError	2-133
CDAQMXChConfigData::initialize	2-107	CDAQMXConfig::getRangePoint	2-134
CDAQMXChConfigData::initMXChConfigData	2-108	CDAQMXConfig::getSpanPoint	2-134
CDAQMXChConfigData::isCorrect	2-108	CDAQMXConfig::initialize	2-134
CDAQMXChConfigData::isObject	2-109	CDAQMXConfig::initMXConfigData	2-135
CDAQMXChConfigData::operator=	2-109	CDAQMXConfig::isCorrect	2-135
CDAQMXChConfigData::setMXChConfig	2-109	CDAQMXConfig::isObject	2-136, 12-90
CDAQMXChConfigData::setMXChConfigData	2-110	CDAQMXConfig::operator=	2-136
CDAQMXChConfigData::setRRJC	2-110	CDAQMXConfig::reconstruct	2-137
CDAQMXChID Class	2-111	CDAQMXConfig::setAO	2-138
CDAQMXChID::CDAQMXChID	2-113	CDAQMXConfig::setAOType	2-139
CDAQMXChID::getAlarmType	2-113	CDAQMXConfig::setChKind	2-140
CDAQMXChID::getAlarmValueOFF	2-113	CDAQMXConfig::setDELTA	2-140
CDAQMXChID::getAlarmValueON	2-114	CDAQMXConfig::setDI	2-141
CDAQMXChID::getChName	2-114	CDAQMXConfig::setDOType	2-141
CDAQMXChID::getChType	2-114	CDAQMXConfig::setInterval	2-142
CDAQMXChID::getComment	2-115	CDAQMXConfig::setMXConfigData	2-142
CDAQMXChID::getKind	2-115	CDAQMXConfig::setPWM	2-143
CDAQMXChID::getMXAlarm	2-115	CDAQMXConfig::setPWMTType	2-144
CDAQMXChID::getMXChID	2-115	CDAQMXConfig::setRES	2-145
CDAQMXChID::getRange	2-116	CDAQMXConfig::setRRJC	2-145
CDAQMXChID::getScale	2-116	CDAQMXConfig::setRTD	2-146

CDAQMXConfig::setScaling	2-146	CDAQMXItemConfig::getDoubleHisterisys	12-86
CDAQMXConfig::setSKIP	2-147	CDAQMXItemConfig::getDoublePresetValue	12-87
CDAQMXConfig::setSTRAIN	2-147	CDAQMXItemConfig::getDoubleScaleMax	12-87
CDAQMXConfig::setTC	2-148	CDAQMXItemConfig::getDoubleScaleMin	12-88
CDAQMXConfig::setTempUnit	2-148	CDAQMXItemConfig::getDoubleSpanMax	12-88
CDAQMXConfig::setVOLT	2-149	CDAQMXItemConfig::getDoubleSpanMin	12-89
CDAQMXDataBuffer Class	12-72	CDAQMXItemConfig::getHisterisys	12-89
CDAQMXDataBuffer::CDAQMXDataBuffer	12-73	CDAQMXItemConfig::getMaxLenItemName	12-90
CDAQMXDataBuffer::create	12-73	CDAQMXItemConfig::readItem	12-91
CDAQMXDataBuffer::currentDataInfo	12-74	CDAQMXItemConfig::toltemName	12-91, 12-92
CDAQMXDataBuffer::currentDateTime	12-74	CDAQMXItemConfig::toltemNo	12-92
CDAQMXDataBuffer::getClassMXChInfo	12-74	CDAQMXItemConfig::writeItem	12-93
CDAQMXDataBuffer::getDataNum	12-75	CDAQMXList Class	12-94
CDAQMXDataBuffer::getDateTime	12-75	CDAQMXList::addData	12-95
CDAQMXDataBuffer::initialize	12-75	CDAQMXList::CDAQMXList	12-95
CDAQMXDataBuffer::isCurrent	12-75	CDAQMXList::copy	12-95
CDAQMXDataBuffer::next	12-76	CDAQMXList::create	12-96
CDAQMXDataBuffer::setChInfo	12-76	CDAQMXList::del	12-96
CDAQMXDataBuffer::setDataInfo	12-76	CDAQMXList::delData	12-96
CDAQMXDataBuffer::setDateTime	12-77	CDAQMXList::getData	12-97
CDAQMXDataInfo Class	2-150	CDAQMXList::getMaxNo	12-97
CDAQMXDataInfo::CDAQMXDataInfo	2-151	CDAQMXList::getNum	12-97
CDAQMXDataInfo::getAlarmName	2-152	CDAQMXList::initialize	12-97
CDAQMXDataInfo::getClassMXChInfo	2-152	CDAQMXList::isData	12-98
CDAQMXDataInfo::getMXDataInfo	2-152	CDAQMXNetInfo Class	2-166
CDAQMXDataInfo::getStatus	2-153	CDAQMXNetInfo::CDAQMXNetInfo	2-167
CDAQMXDataInfo::initialize	2-153	CDAQMXNetInfo::getAddress	2-167
CDAQMXDataInfo::initMXDataInfo	2-153	CDAQMXNetInfo::getGateway	2-167
CDAQMXDataInfo::isAlarm	2-153	CDAQMXNetInfo::getMXNetInfo	2-168
CDAQMXDataInfo::isObject	2-154	CDAQMXNetInfo::getPart	2-168
CDAQMXDataInfo::operator=	2-154	CDAQMXNetInfo::getPort	2-168
CDAQMXDataInfo::setAlarm	2-155	CDAQMXNetInfo::getSubMask	2-169
CDAQMXDataInfo::setClassMXChInfo	2-155	CDAQMXNetInfo::initialize	2-169
CDAQMXDataInfo::setMXDataInfo	2-155	CDAQMXNetInfo::initMXNetInfo	2-169
CDAQMXDataInfo::setStatus	2-155	CDAQMXNetInfo::isObject	2-170
CDAQMXDateTime Class	2-156	CDAQMXNetInfo::operator=	2-170
CDAQMXDateTime::CDAQMXDateTime	2-157	CDAQMXNetInfo::setMXNetInfo	2-170
CDAQMXDateTime::getMXDateTime	2-158	CDAQMXOutput Class	2-171
CDAQMXDateTime::initMXDateTime	2-158	CDAQMXOutputData::CDAQMXOutputData	2-172
CDAQMXDateTime::isObject	2-158	CDAQMXOutputData::getErrorChoice	2-172
CDAQMXDateTime::operator=	2-158	CDAQMXOutputData::getIdleChoice	2-173
CDAQMXDateTime::setMXDateTime	2-159	CDAQMXOutputData::getMXOutput	2-173
CDAQMXDODData Class	2-160	CDAQMXOutputData::getMXOutputData	2-173
CDAQMXDODData::CDAQMXDODData	2-161	CDAQMXOutputData::getOutputType	2-174
CDAQMXDODData::getDOONOFF	2-161	CDAQMXOutputData::getPresetValue	2-174
CDAQMXDODData::getDOValid	2-162	CDAQMXOutputData::getPulseTime	2-174
CDAQMXDODData::getMXDO	2-162	CDAQMXOutputData::initialize	2-175
CDAQMXDODData::getMXDODData	2-162	CDAQMXOutputData::initMXOutputData	2-175
CDAQMXDODData::initialize	2-163	CDAQMXOutputData::isObject	2-175
CDAQMXDODData::initMXDODData	2-163	CDAQMXOutputData::operator=	2-176
CDAQMXDODData::isObject	2-163	CDAQMXOutputData::setChoice	2-176
CDAQMXDODData::operator=	2-164	CDAQMXOutputData::setMXOutputData	2-176
CDAQMXDODData::setDO	2-164	CDAQMXOutputData::setOutputType	2-177
CDAQMXDODData::setDOONOFF	2-164	CDAQMXOutputData::setPulseTime	2-177
CDAQMXDODData::setMXDODData	2-165	CDAQMXSegment Class	2-178
CDAQMXDOList Class	12-78	CDAQMXSegment::CDAQMXSegment	2-179
CDAQMXDOList::add	12-79	CDAQMXSegment::getMXSegment	2-179
CDAQMXDOList::CDAQMXDOList	12-79	CDAQMXSegment::getPattern	2-179
CDAQMXDOList::change	12-80	CDAQMXSegment::initialize	2-180
CDAQMXDOList::copyData	12-81	CDAQMXSegment::initMXSegment	2-180
CDAQMXDOList::create	12-81	CDAQMXSegment::isObject	2-180
CDAQMXDOList::getClassMXDODData	12-81	CDAQMXSegment::operator=	2-181
CDAQMXDOList::getCurrent	12-82	CDAQMXSegment::setMXSegment	2-181
CDAQMXDOList::initCurrent	12-82	CDAQMXSegment::setPattern	2-181
CDAQMXItemConfig Class	12-83	CDAQMXStatus Class	2-182
CDAQMXItemConfig::CDAQMXItemConfig	12-85	CDAQMXStatus::CDAQMXStatus	2-183
CDAQMXItemConfig::getDoubleAlarmOFF	12-85	CDAQMXStatus::getCFRemain	2-183
CDAQMXItemConfig::getDoubleAlarmON	12-86	CDAQMXStatus::getCFSize	2-184

Index

- CDAQMXStatus::getCFStatus 2-184
CDAQMXStatus::getConfigCnt 2-184
CDAQMXStatus::getDateTime 2-184
CDAQMXStatus::getFIFONum 2-185
CDAQMXStatus::getFIFOStatus 2-185
CDAQMXStatus::getInterval 2-185
CDAQMXStatus::getMilliSecond 2-186
CDAQMXStatus::getMXFIFOInfo 2-186
CDAQMXStatus::getMXStatus 2-186
CDAQMXStatus::getNewDataNo 2-187
CDAQMXStatus::getOldDataNo 2-187
CDAQMXStatus::getTime 2-187
CDAQMXStatus::getTimeCnt 2-188
CDAQMXStatus::getUnitStatus 2-188
CDAQMXStatus::initialize 2-188
CDAQMXStatus::initMXStatus 2-188
CDAQMXStatus::isBackup 2-189
CDAQMXStatus::isDataNo 2-189
CDAQMXStatus::isObject 2-189
CDAQMXStatus::operator= 2-190
CDAQMXStatus::setMXStatus 2-190
CDAQMXSysInfo Class 2-191
CDAQMXSysInfo::CDAQMXSysInfo 2-192
CDAQMXSysInfo::getCFTimeout 2-193
CDAQMXSysInfo::getCFWriteMode 2-193
CDAQMXSysInfo::getChNum 2-193
CDAQMXSysInfo::getFIFONo 2-194
CDAQMXSysInfo::getFrequency 2-194
CDAQMXSysInfo::getIntegral 2-194
CDAQMXSysInfo::getInterval 2-195
CDAQMXSysInfo::getItemError 2-195
CDAQMXSysInfo::getMAC 2-195
CDAQMXSysInfo::getModuleSerial 2-196
CDAQMXSysInfo::getModuleType 2-196
CDAQMXSysInfo::getModuleVersion 2-197
CDAQMXSysInfo::getMXModuleData 2-197
CDAQMXSysInfo::getMXSystemInfo 2-197
CDAQMXSysInfo::getOption 2-198
CDAQMXSysInfo::getPartNo 2-198
CDAQMXSysInfo::getRealType 2-198
CDAQMXSysInfo::getStandbyType 2-199
CDAQMXSysInfo::getStyle 2-199
CDAQMXSysInfo::getTempUnit 2-199
CDAQMXSysInfo::getTerminalType 2-200
CDAQMXSysInfo::getUnitNo 2-200
CDAQMXSysInfo::getUnitSerial 2-200
CDAQMXSysInfo::getUnitType 2-200
CDAQMXSysInfo::initialize 2-201
CDAQMXSysInfo::initMXSystemInfo 2-201
CDAQMXSysInfo::isCorrect 2-201
CDAQMXSysInfo::isModuleValid 2-202
CDAQMXSysInfo::isObject 2-202
CDAQMXSysInfo::operator= 2-202
CDAQMXSysInfo::setCFTimeout 2-203
CDAQMXSysInfo::setCFWriteMode 2-203
CDAQMXSysInfo::setModule 2-203
CDAQMXSysInfo::setMXSystemInfo 2-204
CDAQMXSysInfo::setRealModule 2-204
CDAQMXSysInfo::setTempUnit 2-204
CDAQMXSysInfo::setUnitNo 2-205
CDAQMXTransmit Class 2-206
CDAQMXTransmit::CDAQMXTransmit 2-207
CDAQMXTransmit::getMXTransmit 2-207
CDAQMXTransmit::getTransmit 2-207
CDAQMXTransmit::initialize 2-208
CDAQMXTransmit::initMXTransmit 2-208
CDAQMXTransmit::isObject 2-208
CDAQMXTransmit::operator= 2-209
CDAQMXTransmit::setMXTransmit 2-209
CDAQMXTransmit::setTransmit 2-209
CDAQMXTransmitList Class 12-99
CDAQMXTransmitList::add 12-100
CDAQMXTransmitList::CDAQMXTransmitList 12-100
CDAQMXTransmitList::change 12-101
CDAQMXTransmitList::copy 12-101
CDAQMXTransmitList::copyData 12-102
CDAQMXTransmitList::create 12-102
CDAQMXTransmitList::getClassMXTransmit 12-102
CDAQMXTransmitList::getCurrent 12-103
CDAQMXTransmitList::initCurrent 12-103
CF Status Types 6-8, 18-9
CF Write Modes 6-8, 18-9
Channel App-3, App-14
Channel ID Information App-4
Channel Information 19-5, 20-4, 21-4, 22-4
Channel Information Data 12-9, 13-7, 14-7, 15-7, 16-7, 19-10, 20-9, 21-9, 22-9, 23-4, 23-9, App-4, App-15
Channel Kinds 18-6
Channel Name App-3
Channel Number App-3, App-14
Channel Numbers 6-6
Channel Range App-3, App-14
Channel Setup Data 12-10, 13-8, 14-8, 15-8, 16-8, App-5
Channel Status App-15
Channel Type App-3, App-14
Channel Types 6-5
Channel/Relay Types 25-7
Channel/Relay types 11-4, 25-19
Comment App-4
Communication Constant 6-3
Communication Constants 11-2, 25-16, 25-18
Communication Functions 2-4, 3-1, 4-1, 7-2, 8-1, 9-1, 12-3, 13-1, 14-1, 15-1, 16-1, 19-9, 20-1, 20-8, 21-1, 21-8, 22-1, 22-8, 23-1, 23-8
Computation 11-5, 25-9
Computation Channels App-14
Constants 6-4, 18-5, 25-2, 25-5, 25-17
constants 18-1
Constants (DARWIN) 11-1
Constants (MX100) 6-1
Contact Input (DI) Range 11-7
Contact Input (DI) Ranges 25-11
Contact Input Ranges 25-4
Control Functions 2-4, 3-1, 4-1, 7-3, 8-2, 9-2, 12-4, 13-2, 14-2, 15-2, 16-2, 19-3, 20-2, 21-2, 22-2, 23-2
Current Data 12-12, 13-10, 14-10, 15-10, 16-10
- ## D
- DAQDA100 25-14
DAQDA100READER 25-20
DAQDARWIN 11-12
DAQINT64 6-25
DAQMX 6-25
DAQMX100 18-18
DARWIN 1-6
 ackAlarmDA100 24-2
 alarmMaxLengthDA100 24-40
 alarmMaxLengthDA100Reader 24-82
 alarmTypeDA100 24-41
 alarmTypeDA100Reader 24-83
 channelPointDA100 24-42
 channelPointDA100Reader 24-84
 channelStatusDA100 24-43

channelStatusDA100Reader	24-85	receiveByteDA100	24-12
closeDA100	24-3	receiveByteDARWIN	10-20
closeDA100Reader	24-75	receiveLineDA100	24-13
closeDARWIN	10-2	receiveLineDARWIN	10-21
computeDARWIN	10-3	reconstructDA100	24-14
dataAlarmDA100	24-44	reportingDARWIN	10-22
dataAlarmDA100Reader	24-86	revisionAPIDA100	24-61
dataDayDA100	24-45	revisionAPIDA100Reader	24-102
dataDayDA100Reader	24-87	runCommandDA100	24-15
dataDoubleValueDA100	24-46	runCommandDARWIN	10-23
dataDoubleValueDA100Reader	24-88	sendLineDA100	24-16
dataHourDA100	24-47	sendLineDARWIN	10-24
dataHourDA100Reader	24-89	sendTriggerDA100	24-17
dataMilliSecDA100Reader	24-90	sendTriggerDARWIN	10-25
dataMinuteDA100	24-48	setAlarmDARWIN	10-26
dataMinuteDA100Reader	24-91	setChAlarmDA100	24-18
dataMonthDA100	24-49	setChDELTA100	24-20
dataMonthDA100Reader	24-92	setChRRJCDA100	24-21
dataSecondDA100	24-50	setChUnitDA100	24-22
dataSecondDA100Reader	24-93	setDateTimeDARWIN	10-27
dataStatusDA100	24-51	setDateTimeNowDA100	24-23
dataStatusDA100Reader	24-94	setDateTimeNowDARWIN	10-28
dataStringValueDA100	24-52	setDELTADARWIN	10-29
dataStringValueDA100Reader	24-95	setDIDARWIN	10-30
dataValueDA100	24-53	setMADARWIN	10-31
dataValueDA100Reader	24-96	setPOWERDARWIN	10-32
dataYearDA100	24-54	setPULSEDARWIN	10-33
dataYearDA100Reader	24-97	setRangeDA100	24-24
errorMaxLengthDA100	24-55	setRRJCDARWIN	10-34
errorMaxLengthDA100Reader	24-98	setRTDDARWIN	10-35
establishDA100	24-4	setScallingUnitDARWIN	10-36
establishDARWIN	10-4	setSKIPDARWIN	10-37
getAlarmNameDA100 [Visual C only]	24-56	setSTRAINDARWIN	10-38
getAlarmNameDA100Reader [Visual C only]	24-99	setTCDARWIN	10-39
getAlarmNameDARWIN	10-5	setTimeOutDARWIN	10-40
getChannelUnitDA100 [Visual C only]	24-57	setVOLTDARWIN	10-41
getChannelUnitDA100Reader [Visual C only]	24-100	statusByteDA100	24-62
getChDataByASCIIDARWIN	10-6	statusCodeDA100	24-63
getChDataByBinaryDARWIN	10-7	statusReportDA100	24-64
getChInfoDARWIN	10-8	switchCodeDA100	24-26
getErrorMessageDA100 [Visual C only]	24-58	switchComputeDA100	24-27
getErrorMessageDA100Reader [Visual C only]	24-101	switchModeDA100	24-28
getErrorMessageDARWIN	10-9	talkCalibrationChDataDA100	24-30
getMaxLenAlarmNameDARWIN	10-10	talkCalibrationDataDA100	24-31
getMaxLenErrorMessageDARWIN	10-11	talkCalibrationDataDARWIN	10-42
getModuleNameDA100 [Visual C only]	24-59	talkChInfoDARWIN	10-43
getReportStatusDARWIN	10-12	talkDataByASCIIDARWIN	10-44
getRevisionAPIDARWIN	10-13	talkDataByBinaryDARWIN	10-45
getSetDataByLineDA100	24-5	talkOperationChDataDA100	24-32
getSetDataByLineDARWIN	10-14	talkOperationDataDA100	24-33
getStatusByteDARWIN	10-15	talkOperationDataDARWIN	10-46
getSystemConfigDARWIN	10-16	talkSetupChDataDA100	24-34
getVersionAPIDARWIN	10-17	talkSetupDataDA100	24-35
initSetValueDA100	24-6	talkSetupDataDARWIN	10-47
initSystemDARWIN	10-18	toAlarmNameDA100	24-65
mathInfoChDA100	24-7	toAlarmNameDA100Reader	24-103
mathInfoChDA100Reader	24-76	toAlarmNameDARWIN	10-48
mathInstChDA100	24-8	toChannelUnitDA100	24-66
mathInstChDA100Reader	24-77	toChannelUnitDA100Reader	24-104
measInfoChDA100	24-9	toDoubleValueDA100	24-67
measInfoChDA100Reader	24-78	toDoubleValueDA100Reader	24-105
measInstChDA100	24-10	toDoubleValueDARWIN	10-49
measInstChDA100Reader	24-79	toErrorMessageDA100	24-68
moduleCodeDA100	24-60	toErrorMessageDA100Reader	24-106
openDA100	24-11	toErrorMessageDARWIN	10-50
openDA100Reader	24-80	toModuleNameDA100	24-69
openDARWIN	10-19	toStringValueDA100	24-70

Index

toStringValueDA100Reader 24-107
toStringValueDARWIN 10-51
transModeDARWIN 10-52
unitIntervalDA100 24-71
unitValidDA100 24-72
updateReportStatusDA100 24-36
updateStatusDA100 24-37
updateSystemConfigDA100 24-38
versionAPIIDA100 24-73
versionAPIIDA100Reader 24-108
DarwinChInfo 11-12
DarwinDataInfo 11-13
DarwinDateTime 11-12
DarwinModuleInfo 11-13
DarwinSystemInfo 11-14
DarwinUnitInfo 11-14
Data Identifier App-5
Data Manipulation Functions 12-7, 13-5, 14-5, 15-5, 16-5
Data Number App-6
Data Retrieval 12-16, 19-8
Data Retrieval Functions 2-7, 3-3, 4-3, 7-3, 8-3, 9-3, 19-4, 19-9, 20-3, 20-8, 21-3, 21-8, 22-3, 22-8, 23-8
Data Status Values 6-4, 11-3, 18-5, 25-6, 25-18
Data Value App-8
DC current range 11-8
DC Current Ranges 25-4, 25-12
DC Voltage Range Types 6-10, 11-6, 18-11, 25-3, 25-10
Decimal Point Position App-8
Declaration (Visual Basic) 4-6, 9-4
Declaration in the Source File 2-9, 3-6, 7-5, 8-4, 12-15, 13-12, 19-7, 19-12, 20-6, 20-10
Declaration of Functions and Constants 15-12, 16-12
Declaration of Types, Functions, and Constants 14-12, 21-6
Deleted MX100 Members App-24
Device Descriptor App-5, App-15
Digital Input (DI) Detailed Range Types 6-14, 18-15
Digital Input (DI) Range Types 6-14, 18-11
Display Format App-1
Display Format Values 6-8, 18-9
Display Pattern App-1
Display Time App-1
DO Data App-5
DO Data Number App-5

E

Enumeration Constants 6-3, 11-2
error number 26-1
Error Processing 2-14, 3-12, 4-11, 7-12, 8-14, 9-10
Establish Setup Mode 25-8
Establish setup mode 11-5
Example App-3

F

FIFO App-6
FIFO Number App-6
FIFO Status Values 6-8, 18-9
FIFO Value App-7
Files Included 1-5
Filter Coefficient 6-7
Filter Time Constants 18-8
Flag Status 11-3, 25-6
Flag Statuses 6-4
Functionality 4-2
Functions of the API for DARWIN 1-2
Functions of the API for the MX100 1-1

H

Hysteresis App-2

I

Implementing Function Commands 8-3
Initial Balance Data App-7
Initial Balance Data Number App-7
Initial Balance Results 18-10
Initial balance Results 6-9
Installation 1-7
Instantaneous Value App-7
Interval Types 6-6, 18-7

L

Library Designation 2-9, 7-5, 12-15, 19-7, 19-12
List of Data Retrieval Functions 23-3
List of Setting Items 6-16
Load Library Statement 3-6, 8-4, 13-12, 20-6, 20-10

M

Maximum Value 25-6
Maximum Values 6-3, 11-2, 18-4, 25-17
Measured Data 12-9, 13-7, 14-7, 16-7, 19-5, 19-10, 20-4, 20-9, 21-4, 21-9, 22-4, 22-9, 23-4, 23-9, App-7, App-15
Measured Value App-8, App-15
Measurement Channels App-14
Measurement Interval App-16
Module Information App-8
Module Number App-8
Module Settings 16-4
Module Types 6-6, 18-7
MX100
 ackAlarmMX100 17-2
 addressPartMX100 17-91
 alarmDoubleHisterisysMX100 17-92
 alarmDoubleValueOFFMX100 17-93
 alarmDoubleValueONMX100 17-94
 alarmHisterisysMX100 17-95
 alarmMaxLengthMX100 17-96
 alarmTypeMX100 17-97
 alarmValueOFFMX100 17-98
 alarmValueONMX100 17-99
 autoFIFOMX 5-2
 CDAQMX100::ackAlarm 12-23
 CDAQMX100::CDAQMX100 12-23
 CDAQMX100::CDAQMXDataBuffer 12-28
 CDAQMX100::CDAQMXItemConfig 12-27, 12-29
 CDAQMX100::changeAOPWMValue 12-24
 CDAQMX100::clearBalance 12-24
 CDAQMX100::commandAOPWM 12-25
 CDAQMX100::commandDO 12-25
 CDAQMX100::commandTransmit 12-26
 CDAQMX100::currentDoubleAOPWMValue 12-26
 CDAQMX100::displaySegment 12-27
 CDAQMX100::formatCF 12-27
 CDAQMX100::getClassMXAOPWMList 12-28
 CDAQMX100::getClassMXBalanceList 12-28
 CDAQMX100::getClassMXDOList 12-28
 CDAQMX100::getClassMXTransmitList 12-29
 CDAQMX100::getDataCh 12-29
 CDAQMX100::getDataFIFO 12-30
 CDAQMX100::getDataNum 12-30

CDAQMX100::getInstCh	12-31	changeBalanceMX	5-4
CDAQMX100::getInstFIFO	12-31	changeBalanceMX100	17-5
CDAQMX100::getItemAll	12-32	changeDODDataMX	5-5
CDAQMX100::getRevisionMX100DLL	12-32	changeDOMX100	17-6
CDAQMX100::getVersionMX100DLL	12-32	changeTransmitMX	5-6
CDAQMX100::initBalance	12-32	changeTransmitMX100	17-7
CDAQMX100::initDataCh	12-33	channelBalanceValidMX100	17-100
CDAQMX100::initDataFIFO	12-33	channelBalanceValueMX100	17-101
CDAQMX100::initSetValue	12-33	channelBurnoutMX100	17-102
CDAQMX100::isObject	12-34	channelDeenergizeMX100	17-103
CDAQMX100::measClear	12-34	channelDisplayMaxMX100	17-104
CDAQMX100::measDataCh	12-35	channelDisplayMinMX100	17-105
CDAQMX100::measDataFIFO	12-35	channelDoublePresetValueMX100	17-106
CDAQMX100::measInstCh	12-36	channelDoubleScaleMaxMX100	17-107
CDAQMX100::measInstFIFO	12-36	channelDoubleScaleMinMX100	17-108
CDAQMX100::measStart	12-37	channelDoubleSpanMaxMX100	17-109
CDAQMX100::measStop	12-37	channelDoubleSpanMinMX100	17-110
CDAQMX100::nextFIFO	12-37	channelErrorChoiceMX100	17-111
CDAQMX100::open	12-38	channelFIFOIndexMX100	17-112
CDAQMX100::reconstruct	12-38	channelFIFONoMX100	17-113
CDAQMX100::reloadBalance	12-39	channelFilterMX100	17-114
CDAQMX100::sendConfig	12-39	channelHoldMX100	17-115
CDAQMX100::setAlarm	12-40	channelIdleChoiceMX100	17-116
CDAQMX100::setBurnout	12-40	channelKindMX100	17-117
CDAQMX100::setCFWriteMode	12-41	channelNumberMX100	17-118
CDAQMX100::setChComment	12-41	channelOutputTypeMX100	17-119
CDAQMX100::setChDELTA	12-42	channelPointMX100	17-120
CDAQMX100::setChKind	12-43	channelPresetValueMX100	17-121
CDAQMX100::setChoice	12-44	channelPulseTimeMX100	17-122
CDAQMX100::setChRRJC	12-44	channelRangeMX100	17-123
CDAQMX100::setChTag	12-45	channelRealMaxMX100	17-124
CDAQMX100::setChUnit	12-45	channelRealMinMX100	17-125
CDAQMX100::setDateTime	12-46	channelRefAlarmMX100	17-126
CDAQMX100::setDeenergize	12-46	channelRefChNoMX100	17-127
CDAQMX100::setFilter	12-47	channelRJCTypeMX100	17-128
CDAQMX100::setHisterisys	12-47	channelRJCVoltMX100	17-129
CDAQMX100::setHold	12-48	channelScaleMaxMX100	17-130
CDAQMX100::setIntegral	12-48	channelScaleMinMX100	17-131
CDAQMX100::setInterval	12-49	channelScaleTypeMX100	17-132
CDAQMX100::setItemAll	12-49	channelSpanMaxMX100	17-133
CDAQMX100::setOutputType	12-50	channelSpanMinMX100	17-134
CDAQMX100::setPulseTime	12-50	channelValidMX100	17-135
CDAQMX100::setRange	12-51	clearBalanceMX100	17-8
CDAQMX100::setRefAlarm	12-52	closeMX	5-7
CDAQMX100::setRJCType	12-52	closeMX100	17-9
CDAQMX100::setScale	12-53	commandAOPWMMX100	17-10
CDAQMX100::setSpan	12-53	commandBalanceMX100	17-11
CDAQMX100::setUnitNo	12-54	commandDOMX100	17-12
CDAQMX100::setUnitTemp	12-54	commandTransmitMX100	17-13
CDAQMX100::switchBackup	12-55	compareDataNoMX	5-8
CDAQMX100::switchDO	12-55	copyAOPWMMX100	17-14
CDAQMX100::switchTransmit	12-56	copyBalanceMX100	17-15
CDAQMX100::toChNo	12-56	copyDOMX100	17-16
CDAQMX100::updateAll	12-57	copyTransmitMX100	17-17
CDAQMX100::updateAOPWMData	12-57	createAOPWMMX100	17-18
CDAQMX100::updateBalance	12-58	createBalanceMX100	17-19
CDAQMX100::updateConfig	12-58	createDOMX100	17-20
CDAQMX100::updateDODData	12-58	createTransmitMX100	17-21
CDAQMX100::updateInfoCh	12-59	currentAOPWMValidMX100	17-136
CDAQMX100::updateOutput	12-59	currentAOPWMValueMX100	17-137
CDAQMX100::updateRenew	12-59	currentBalanceResultMX100	17-138
CDAQMX100::updateStatus	12-60	currentBalanceValidMX100	17-139
CDAQMX100::updateSystem	12-60	currentBalanceValueMX100	17-140
CDAQMX100::userClear	12-60	currentDoubleAOPWMValueMX100	17-141
CDAQMX100::userDoubleAOPWMValue	12-61	currentDOValidMX100	17-142
changeAOPWMDataMX	5-3	currentDOValueMX100	17-143
changeAOPWMMX100	17-3	currentTransmitMX100	17-144
changeAOPWMValueMX100	17-4	dataAlarmMX100	17-145

Index

dataDayMX100	17-146	initItemMX100	17-32
dataDoubleValueMX100	17-147	initSetValueMX100	17-33
dataHourMX100	17-148	initSystemMX	5-33
dataMilliSecMX100	17-149	isDataNoMX	5-34
dataMinuteMX100	17-150	isDataNoVBMX	5-35
dataMonthMX100	17-151	itemErrorMX100	17-171
dataNumChMX100	17-152	itemMaxLengthMX100	17-172
dataNumFIFOMX100	17-153	lastErrorMX100	17-173
dataSecondMX100	17-154	measDataChMX100	17-34
dataStatusMX100	17-155	measDataFIFOMX100	17-35
dataStringValueMX100	17-156	measInstChMX100	17-36
dataTimeMX100	17-157	measInstFIFOMX100	17-37
dataValidMX100	17-158	measStartMX100	17-38
dataValueMX100	17-159	measStopMX100	17-39
dataYearMX100	17-160	moduleChNumMX100	17-174
decrementDataNoMX	5-9	moduleFIFONoMX100	17-175
deleteAOPWMMX100	17-22	moduleIntegralMX100	17-176
deleteBalanceMX100	17-23	moduleIntervalMX100	17-177
deleteDOMX100	17-24	moduleRealTypeMX100	17-178
deleteTransmitMX100	17-25	moduleStandbyTypeMX100	17-179
displaySegmentMX100	17-26	moduleTerminalMX100	17-180
errorMaxLengthMX100	17-161	moduleTypeMX100	17-181
formatCFMX	5-10	moduleValidMX100	17-182
formatCFMX100	17-27	moduleVersionMX100	17-183
getAlarmNameMX	5-11	netAddressMX100	17-184
getAlarmNameMX100		netGatewayMX100	17-185
[Visual C only]	17-162	netPortMX100	17-186
getAOPWMDDataMX	5-12	netSubmaskMX100	17-187
getBalanceMX	5-13	openMX	5-36
getChannelCommentMX100		openMX100	17-40
[Visual C only]	17-163	rangePointMX100	17-188
getChannelTagMX100		readItemMX100	17-41
[Visual C only]	17-164	reconstructMX100	17-42
getChannelUnitMX100		resetBalanceMX	5-37
[Visual C only]	17-165	revisionAPIMX100	17-189
getChConfigMX	5-14	runBalanceMX	5-38
getChDataMX	5-15	sendConfigMX100	17-43
getChDataNoMX	5-16	setAlarmMX	5-39
getChInfoMX	5-17	setAlarmMX100	17-44
getConfigDataMX	5-18	setAlarmValueMX100	17-45
getDODataMX	5-19	setAOMX	5-40
getErrorMessageMX	5-20	setAOPWMDDataMX	5-41
getErrorMessageMX100		setAOTypeMX	5-42
[Visual C only]	17-166	setBackupMX	5-43
getFIFODataNoMX	5-21	setBalanceMX	5-44
getItemAllMX100	17-28	setBurnoutMX	5-45
getItemErrorMX	5-22	setBurnoutMX100	17-46
getLastErrorMX	5-23	setCFWriteModeMX100	17-47
getMaxLenAlarmNameMX	5-24	setChCommentMX100	17-48
getMaxLenErrorMessageMX	5-25	setChConfigMX	5-46
getModuleSerialMX100		setChDELTAMX100	17-49
[Visual C only]	17-167	setChKindMX100	17-50
getNetHostMX100		setChoiceMX	5-47
[Visual C only]	17-168	setChoiceMX100	17-51
getOutputMX	5-26	setChRRJCMX100	17-52
getRevisionAPIMX	5-27	setChTagMX100	17-53
getStatusDataMX	5-28	setChUnitMX100	17-54
getSystemConfigMX	5-29	setCommentMX	5-48
getTimeDataMX	5-30	setConfigDataMX	5-49
getUnitPartNoMX100		setDateTimeMX	5-50
[Visual C only]	17-169	setDateTimeNowMX	5-51
getUnitSerialMX100		setDateTimeNowMX100	17-55
[Visual C only]	17-170	setDeenergizeMX100	17-56
getVersionAPIMX	5-31	setDELTAMX	5-52
incrementDataNoMX	5-32	setDIMX	5-53
initBalanceMX100	17-29	setDODataMX	5-54
initDataChMX100	17-30	setDOTypeMX	5-55
initDataFIFOMX100	17-31	setDoubleAlarmMX100	17-57

setDoubleAlarmValueMX100	17-58	talkChDataInstMX	5-85
setDoubleChoiceMX100	17-59	talkChDataMX	5-86
setDoubleHisterisysMX100	17-60	talkChDataVBMX	5-87
setDoubleScaleMX100	17-61	talkChInfoMX	5-88
setDoubleSpanMX100	17-62	talkConfigMX	5-89
setFilterMX	5-56	talkFIFODataInstMX	5-90
setFilterMX100	17-63	talkFIFODataMX	5-91
setHisterisysMX100	17-64	talkFIFODataVBMX	5-92
setHoldMX100	17-65	toAlarmNameMX	5-93
setIntegralMX100	17-66	toAlarmNameMX100	17-206
setIntervalMX	5-57	toAOPWMValueMX	5-94
setIntervalMX100	17-67	toAOPWMValueMX100	17-207
setItemAllMX100	17-68	toChannelCommentMX100	17-208
setOutputMX	5-58	toChannelTagMX100	17-209
setOutputTypeMX	5-59	toChannelUnitMX100	17-210
setOutputTypeMX100	17-69	toDateTimeMX	5-95
setPulseTimeMX	5-60	toDoubleValueMX	5-96
setPulseTimeMX100	17-70	toDoubleValueMX100	17-211
setPWMMX	5-61	toErrorMessageMX	5-97
setPWMTypeMX	5-62	toErrorMessageMX100	17-212
setRangeMX100	17-71	toItemNameMX100	17-213
setRefAlarmMX	5-63	toItemNoMX100	17-214
setRefAlarmMX100	17-72	toModuleSerialMX100	17-215
setRESMX	5-64	toNetHostMX100	17-216
setRJCTypeMX	5-65	toRealValueMX	5-98
setRJCTypeMX100	17-73	toRealValueMX100	17-217
setRRJCMX	5-66	toStringValueMX	5-99
setRTDMX	5-67	toStringValueMX100	17-218
setScaleMX100	17-74	toStyleVersionMX	5-100
setScalingUnitMX	5-68	toStyleVersionMX100	17-219
setSegmentMX	5-69	toUnitPartNoMX100	17-220
setSKIPMX	5-70	toUnitSerialMX100	17-221
setSpanMX100	17-75	unitCFWriteModeMX100	17-222
setSTRAINMX	5-71	unitFrequencyMX100	17-223
setSystemConfigMX	5-72	unitMACMX100	17-224
setSystemTimeoutMX	5-73	unitNoMX100	17-225
setTagMX	5-74	unitOptionMX100	17-226
setTCMX	5-75	unitStyleMX100	17-227
setTempUnitMX	5-76	unitTempMX100	17-228
setTimeOutMX	5-77	unitTypeMX100	17-229
setTransmitMX	5-78	updateAOPWMDataMX100	17-81
setUnitNoMX	5-79	updateBalanceMX100	17-82
setUnitNoMX100	17-76	updateConfigMX100	17-83
setUnitTempMX100	17-77	updateDODDataMX100	17-84
setUserTimeMX	5-80	updateInfoChMX100	17-85
setUserTimeVBMX	5-81	updateOutputMX100	17-86
setVOLTMX	5-82	updateStatusMX100	17-87
startFIFOMX	5-83	updateSystemMX100	17-88
statusBackupMX100	17-190	userAOPWMValidMX100	17-230
statusCFMX100	17-191	userAOPWMValueMX100	17-231
statusCFRemainMX100	17-192	userBalanceValidMX100	17-232
statusCFSizeMX100	17-193	userBalanceValueMX100	17-233
statusDayMX100	17-194	userDoubleAOPWMValueMX100	17-234
statusFIFOIntervalMX100	17-195	userDOValidMX100	17-235
statusFIFOMX100	17-196	userDOValueMX100	17-236
statusFIFONumMX100	17-197	userTransmitMX100	17-237
statusHourMX100	17-198	versionAPIMX100	17-238
statusMilliSecMX100	17-199	writeltemMX100	17-89
statusMinuteMX100	17-200	MX100 - Specific Error Messages	26-3
statusMonthMX100	17-201	MX100 Class	2-1
statusSecondMX100	17-202	MX100 Dedicated Class	2-1
statusTimeMX100	17-203	MX100/DARWIN Common Class	2-1, 2-15
statusUnitMX100	17-204	The MX100/DARWIN common classes	12-18
statusYearMX100	17-205	MXAlarm	6-26
stopFIFOMX	5-84	MXAOPWM	6-36
switchBackupMX100	17-78	MXAOPWMData	6-36
switchDOMX100	17-79	MXBalance	6-34
switchTransmitMX100	17-80	MXBalanceData	6-34

Index

MXBalanceResult 6-34
MXCFInfo 6-32
MXCFInfo structure 6-32
MXChConfig 6-28
MXChConfigAI 6-27
MXChConfigAIDI 6-26
MXChConfigData 6-29
MXChConfigDO 6-27
MXChID 6-28
MXChInfo 6-29
MXConfigData 6-35
MXDataInfo 6-26
MXDataNo 6-25
MXDateTime 6-25
MXDO 6-36
MXDOData 6-36
MXFIFOInfo 6-32
MXINT64 6-25
MXModuleData 6-31
MXNetInfo 6-34
MXOutput 6-35
MXOutputData 6-35
MXProductInfo 6-30
MXSegment 6-36
MXStatus 6-33
MXSystemInfo 6-32
MXTransmit 6-36
MXUnitData 6-30
MXUserTime 6-25

N

Network Information Dat 14-9
Network Information Data 12-11, 13-9, 15-9, 16-9, App-8
New DARWIN Functions App-21
New DARWIN Members App-25
New MX100 Classes App-22
New MX100 Functions App-20
New MX100 Members App-22
Number of Channels 18-7
Number of Items 25-5, 25-17
Numbers 18-4

O

Operation Modes 11-4, 25-7
Options 6-9, 18-10
Output Channel Data App-9
Output Channel Data Number App-9
Output Data Value App-2
Output Types 6-9, 18-10

P

Packet App-9
PC (Personal Computer) 1-6
Power connection method 11-8
Power Connection Methods 25-12
Power Measurement Parameters 25-13
Power measurement parameters 11-9
Power monitor range 11-8
Power Monitor Ranges 25-4, 25-12
Program Example
 Implementing Function Commands 7-10, 8-10, 9-8
 Implementing the Talker Function 7-11, 8-12, 9-9
 Retrieval of Setup Data and Configuration .. 3-11, 4-10, 7-8, 8-8, 9-7

 Retrieval of the Measured Data 2-10, 3-7, 4-7, 7-6, 8-5, 9-5
 Retrieval of Setup Data and Configuration 2-13
Pulse interval integer multiple App-9
Pulse range 11-7
Pulse Ranges 25-4, 25-11
PWM Ranges 6-15, 18-16

R

Range Type 6-9
Range Types 18-3, 18-11, 25-2, 25-10
Reference Alarm App-9
Reference Channel Number App-10
Reference range 6-9
Reference Ranges 18-11
Relay App-16
Relay Number App-16
Relay Type App-16
Report execution type 11-5
Report Execution Types 25-9
Report status 11-6
Report Statuses 25-9
Report type 11-6
Report Types 25-9
Resistance Ranges 6-14
Response App-10, App-15
Retrieval Functions 12-8, 13-7, 14-7, 15-6, 15-7, 16-6, 16-7, 19-5, 19-10, 20-4, 20-9, 21-4, 21-9, 22-4, 22-9, 23-4, 23-9
Retrieval of Initial Balance Data 2-8
Retrieval of Measured Data 2-8, 15-13
Retrieval of Setup Data 16-13
Retrieval of the Measured Data 14-13, 19-13, 20-7, 20-11, 21-7, 21-11, 22-7, 22-11, 23-7, 23-11
Retrieve Code Types 25-2, 25-5
Retrieves output channel data 2-8
RJC Types 6-7, 18-8
RJC Voltage App-10
RTD (1 mA) Range Types 18-12
RTD (2 mA) Range Types 6-13
RTD (2 mA) Ranges 18-14
RTD (Other) Ranges 6-14
RTD (other) Ranges 18-15
RTD Range 11-7
RTD Range Types 25-3
RTD Ranges 25-11

S

Scale App-16
Scale Types 6-6, 18-7
Segment Number App-1
Selected Values 6-9, 18-10
setHoldMX100 17-65
Setting (Operation Mode) Functions 19-3, 20-2, 21-2, 22-2, 23-2
Setup Change Functions 12-5, 13-3, 14-3, 15-3, 16-3
Setup Data App-11
Setup Functions 2-5, 7-3, 8-2, 9-2, 12-4, 13-2, 14-2, 15-2, 16-2
Setup Item Numbers App-10
Setup Items 15-6, 16-6
Skip 18-11
SKIP Range 25-4
SKIP Ranges 25-12
Slot Number App-17
Software Components 1-4
Span App-16
Starting/Stopping the FIFO 14-1, 15-1, 16-1

Status Byte App-17
 Status Byte Value 11-5
 Status Byte Values 25-8
 Status Data 12-12, 13-10, 14-10, 15-10, 16-10, 19-6, 20-4, 23-4, App-10
 Status data 21-4, 22-4
 Status Retrieval Function 21-8, 22-8
 Status Retrieval Functions 13-1, 14-1, 15-1, 16-1
 Status Transition Functions .. 19-2, 19-9, 20-1, 20-8, 21-1, 22-1, 23-1, 23-8
 Strain Input Ranges 25-4, 25-11
 Strain Ranges 6-15, 18-16
 String 11-2
 Manual vi
 Subunit Number App-17
 Supported Languages 1-1
 Supported Models 1-1
 System Configuration Data . 12-11, 13-9, 14-9, 15-9, 16-9, 19-5, 20-4, 23-4, App-11, App-17
 System configuration data 21-4, 22-4
 System Control Types 6-5, 11-3

T

Tag App-4
 Talker App-17
 Talker Function Types 11-4, 25-8
 TC Range 11-6
 TC Range Types 6-10, 25-3
 TC Ranges 18-12, 25-10
 Temperature Unit Types 6-8, 18-9
 Terminal Number App-17
 Terminal Types 6-7, 18-8
 Terminator App-17
 Terms and Conditions of the Software License ii
 Time Information Data App-11, App-18
 Timeout Value App-12
 timeout value App-19
 Transmission Output Data App-11
 Transmission Statuses 6-9, 18-10
 Types (DARWIN) 11-10

U

Unit App-12, App-18
 Unit Information App-12
 Unit Name App-4, App-15
 Unit Number 25-19, App-12, App-17
 Unit Numbers 25-8
 Unit Status Values 6-8, 18-9
 Unit Type Logic 6-7, 18-8
 User Count App-12
 User Dat 14-11
 User Data 12-13, 13-11, 15-11, 16-11
 Using Setup Items 12-17, 13-14, 14-14, 15-14, 16-14
 Utilities 2-8, 3-5, 4-5, 7-4, 8-3, 9-3, 12-14, 13-11, 14-11, 15-11, 16-11, 19-6, 19-11, 20-5, 20-9, 21-5, 21-9, 22-5, 22-9, 23-5, 23-9

V

Valid/Invalid Value 25-17
 Visual C/Visual C++ Constants 25-2