User's
Manual

Model MX190
MX100/DARWIN 用 API
ユーザーズマニュアル

vigilantplant.®

YOKOGAWA
横河電機株式会社

IM MX190-01
3版

はじめに

このたびは，MX100/DARWIN用APIをお買い上げいただきましてありがとうございます。
このユーザーズマニュアルは，MX100/DARWIN用APIの使用方法について説明したものです。ご使用前にこのマニュアルをよくお読みいただき，正しくお使いください。
お読みになったあとは，ご使用時にすぐにご覧になれるところに，大切に保存してください。ご使用中に取り扱いがわからなくなったときなどにきっとお役に立ちます。

ご注意

- 本書の内容は，性能・機能の向上などにより，将来予告なしに変更することがあります。
- 本書の内容に関しては万全を期していますが，万一ご不審の点や誤りなどお気づきのことがありましたら，お手数ですが，当社支社・支店・営業所までご連絡ください。
- 本書の内容の全部または一部を無断で転載，複製することは禁止されています。
- ソフトウエアを同時に複数のコンピュータで使用することを禁止します。また，複数の使用者によって使用することも禁止します。
- ソフトウエアを第三者に譲渡することおよび貸与することを禁止します。
- 当社は，ソフトウエアのパッケージを開封した時点で，オリジナルディスクに物理的な欠陥がある場合を除き，いかなる保証もいたしません。
- 当社は，ソフトウエアの使用に関して直接または間接に生じるいっさいの損害について責任を負いません。
- **シリアル番号の再発行はしません。シリアル番号は大切に保管してください。**

商標

- vigilantplantおよびDAQMASTERは，当社の登録商標です。
- Microsoft およびWindows は，米国Microsoft Corporation の米国およびその他の国における登録商標または商標です。
- Adobe およびAcrobat は，Adobe Systems Incorporated(アドビシステムズ社) の登録商標または商標です。
- 本書に記載している製品名および会社名は，各社の登録商標または商標です。
- 本書では各社の登録商標または商標に，®および™ マークを表示していません。

履歴

- 2003年　5月　　初版発行
- 2004年　11月　2版発行
- 2008年　3月　　3版発行

3rd Edition : March 2008 (YG)

ソフトウエア使用許諾契約書

ご使用前に必ずお読みください。

このたびは横河電機株式会社のソフトウエアをご購入いただきまして誠にありがとうございます。お客様がこのパッケージを開封された場合には，下記の「ソフトウエア使用許諾契約書」に同意したものとみなします。横河ソフトウエアは当社の著作物であり，同ソフトウエアを開封のうえ，インストールし，ご使用されるにあたっては，下記の「ソフトウエア使用許諾契約書」を必ずお読みのうえ，ご承諾いただくようお願いします。ご承諾いただけない場合には，パッケージを開封しないでください。

ソフトウエア使用許諾契約書

お客様が本ソフトウエア使用許諾契約書(以下，「本契約」といいます)に合意することを条件として，横河電機株式会社(以下，「当社」といいます)は，包装されたソフトウエア製品(以下，「横河ソフトウエア」といいます)の使用権をお客様に許諾します。なお，当社は，横河ソフトウエアの使用権をお客様に許諾するものであり，横河ソフトウエアを販売するものではありません。

製品　　：MX100/DARWIN用API
ライセンス数　：1

第1条(適用範囲)
1. 本契約は，当社がお客様に提供する横河ソフトウエア製品に適用するものとします。
2. 横河ソフトウエアは，それに含まれる一切の技術，アルゴリズム，およびプロセスを包含するものとします。

第2条(使用権の許諾)
1. お客様は，横河ソフトウエアについて，別途合意した使用料を対価として，前文に定めるライセンス数に対応する台数のコンピュータに限りインストールできるものとし，当社は，お客様の自己使用を目的とした，非独占的かつ譲渡不能の使用権(以下「使用権」といいます)を許諾します。
2. お客様は，当社の事前の書面による承諾なしに，横河ソフトウエアを第三者に頒布，転貸，複製，譲渡，質入，伝送もしくは再使用権を許諾しないものとします。
3. お客様は，バックアップ目的として一組のみ横河ソフトウエアを複製する以外は，横河ソフトウエアの全部または一部を複製しないものとします。また当該複製物の保管および管理については厳重な注意を払うものとします。
4. お客様は，いかなる理由においても横河ソフトウエアをダンプ，逆アッセンブル，逆コンパイル，リバースエンジニアリングなどによるソースプログラムその他人間が読み取り可能な形式への変換もしくは複製または横河ソフトウエアの修正もしくは他の言語への翻訳など，提供された形式以外に改変しないものとします。また，当社は，別に同意しない限り，お客様にソースプログラムを提供しないものとします。
5. 横河ソフトウエアおよびそれらに含まれる一切の技術，アルゴリズム，およびプロセスなどのノウハウは，当社または当社に対し再使用許諾を含む使用許諾権を付与している第三者の固有財産であり，当社または当社に対し再使用許諾権を付与している第三者が権利を有しているものであり，お客様に権利の移転や譲渡を一切行うものではありません。
6. 当社は，横河ソフトウエアに保護の機構(コピープロテクト)を使用または付加することがあります。当該コピープロテクトを除去したり，除去を試みることは認められないものとします。
7. 横河ソフトウエアには，当社が第三者から再使用許諾を含む使用許諾権を付与されているソフトウエアプログラム(以下「第三者プログラム」といい，当社の関連会社が独自に製作・販売しているソフトウエアプログラムもこれに含みます)を含む場合があります。かかる第三者プログラムに関し，当社が当該第三者より本契約と異なる再使用許諾条件を受け入れている場合には，別途書面により通知される当該条件を遵守していただきます。

第3条(特定用途に関する制限)
1. 横河ソフトウエアは，下記の各号を目的として，製作または頒布されるものではありません。
   (a)航空機の運航または船舶の航行や，これらを地上でサポートする機器の立案，設計，開発，保守，運用および使用されること。(b)原子力施設の立案，設計，開発，建設，保守，運用および使用されること。(c)核兵器，化学兵器または生物兵器の立案，設計，開発，保守，運用および使用されること。(d)医療機器などの人身に直接関わるような状況下で使用されることを目的に立案，設計，開発，保守，運用および使用されること。
2. お客様が前項の目的で横河ソフトウエアを使用する場合には，当社は当該使用により発生するいかなる請求および損害に対しても責任を負わないものとし，お客様は，お客様の責任においてこれを解決するものとし，当社を免責するものとします。

第4条(保証)
1. 横河ソフトウエアは，当該製品完成時または出荷時の現状のままでお客様に提供されるものとし，お客様は，これに合意するものとします。横河ソフトウエアの記録媒体に破損，損傷が発見された場合は，開封後7日間に限り無償で交換をいたします(お客様の費用で当社の指定するサービス拠点に当該ソフトウエア製品の記憶媒体を送付していただくものとします)が，いかなる場合であっても横河ソフトウエアに瑕疵のないこと，的確性，正確性，信頼性もしくは最新性などの品質上または性能上の明示または黙示の保証をするものではありません。また，横河ソフトウエアが他のソフトウエアとの間で不整合，相互干渉などの影響のないことを保証するものでもありません。
2. 前項の規定に関わらず，横河ソフトウエアに第三者プログラムが存在する場合の保証期間，保証条件については，かかるプログラムの供給者の定めるところによるものとします。
3. 当社は，自己の判断により必要と認めた場合，横河ソフトウエアに関するレビジョンアップおよびバージョンアップ(以下，アップデートサービスといいます)を実施することがあります。
4. 前項の定めにも拘らず，当社は，いかなる場合であってもお客様により改変または修正された横河ソフトウエアに関するアップデートサービスについては，第三者により改変・修正された場合を含め，一切対応しないものとします。

第5条(特許権，著作権の侵害に関する損害賠償責任)
1. お客様は，横河ソフトウエアについて，第三者から特許権，商標権，著作権その他の権利に基づき使用の差し止め，損害賠償請求などが行われた場合は，書面にて速やかに請求の内容を当社に通知するものとします。
2. 前項の請求などが当社の責に帰すべき事由による場合は，その防御および和解交渉について，お客様から当社に防御，交渉に必要なすべての権限を与えていただき，かつ必要な情報および援助をいただくことを条件に，当社は自己の費用負担で当該請求などの防御および交渉を行い，前項記載の第三者に対して最終的に認められた責任を負うものとします。
3. 当社は第1項における請求またはその恐れがあると判断した場合は，当社の選択により，当社の費用で下記のいずれかの処置を取るものとします。
   (a)正当な権利を有する者からかかる横河ソフトウエアの使用を継続する権利を取得する。(b)第三者の権利の侵害を回避できるようなソフトウエア製品と交換する。(c)第三者の権利を侵害しないようにかかる横河ソフトウエアを改造する。
4. 前項各号の処置がとれない場合，当社は，お客様から当社にお支払い頂いた第2条第1項に定める使用料の対価を限度として損害を賠償するものとします。

第6条(責任の制限)
本契約に基づいて当社がお客様に提供した横河ソフトウエアによって，当社の責に帰すべき事由によりお客様が損害を被った場合は，当社は，本契約の規定に従って対応するものとしますが，いかなる場合においても，派生損害，結果損害，その他の間接損害(営業上の利益の損失，業務の中断，営業情報の喪失などによる損害その他)については一切責任を負わないものとし，かつ当社の損害賠償責任は，かかる横河ソフトウエアについてお客様からお支払いを受けた第2条第1項に定める使用料の対価を限度とします。なお，当社が納入した製品をお客様が当社の書面による事前の承諾なく改造，改変，他のソフトウエアとの結合を行い，またはその他基本仕様書または機能仕様書との相違を生ぜしめた場合は，当社は一部または全ての責任を免れることができるものとします。

第7条(輸出規制)
お客様は，事前に当社の同意を得た場合を除き，横河ソフトウエアを，直接，間接を問わず輸出または他国に伝送しないものとします。

第8条(本契約の期間)
本契約は，お客様が横河ソフトウエアを受領した日から，契約解除されない限り，お客様または当社が相手方に対し，1ヶ月前に書面による通知によって当該ソフトウエア製品の使用を終了させるまで，またはお客様の横河ソフトウエアの使用終了時まで，有効とします。

第9条(使用の差止め)
横河ソフトウエアの使用許諾後といえども，使用環境の変化または許諾時には見出せなかった不適切な環境条件が見られる場合，その他横河ソフトウエアを使用するに著しく不適切であると当社が判断した場合には，当社はお客様に対して当該使用を差止めることができるものとします。

第10条(解除)
当社は，お客様が本契約に違反した場合には，何ら催告を要することなく通知をもって本契約を解除できます。ただし，本契約終了または解除後といえども第5条，第6条ならびに第11条は効力を有するものとします。

第11条(管轄裁判所)
本契約に関して生じた紛争，疑義については，両者誠意を持って協議解決するものとします。ただし，一方当事者が他方当事者に協議解決をしたい旨の通知後90日以内に両当事者間で協議が整わない場合は東京地方裁判所(本庁)を第一審の専属的管轄裁判所とします。

以上

梱包内容の確認

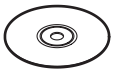
梱包箱を開けたら，ご使用前に以下のことを確認してください。万一，お届けした品の間違いや品不足，または外観に異常が認められる場合は，お買い求め先にご連絡ください。

形名

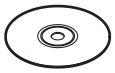
形名	名称
MX190	MX100/DARWIN用API

お買いあげいただいたパッケージの構成

CD-ROM　1枚
MX100/DARWIN用API



CD-ROM　1枚
Manuals for the API for the MX100/DARWIN
(ユーザーズマニュアル)



CD-ROMの取り扱い上の注意

次の注意事項をお守りください。

警　告

- **ゴミやほこりの多いところで使用，保管しないでください。**
- **文字などが印刷されていない面には，触れないでください。**
  指先の汚れ，汗などが付着すると故障の原因になります。また，文字などを書き込まないでください。
- **鉛筆の芯や消しゴムのカスが付着すると，故障の原因になります。**
- **折り曲げないでください。また，傷を付けないでください。**
  データの読み出しができなくなります。
- **上にものを置かないでください。**
  変形して使用不可能になることがあります。
- **高い所から，落とさないでください。**
  CD-ROMを落すと，破損，変形により，使用不可能になることがあります。
- **直射日光の当たる場所や暖房機器の近くに置かないでください。**
- **溶剤は使用しないでください。**
  アルコール，シンナー，フレオンなどの溶剤は，絶対に使用しないでください。
- **CD-ROMドライブ装置への装着は，ていねいにしてください。**
- **CD-ROMにアクセス中は，CD-ROMをドライブから取り出す操作をしたり，コンピュータの電源を落としたり，コンピュータをリセットしないでください。**
- **専用のケースに入れて保管してください。**
  使用後は，コンピュータに装着したままにしないでください。ケースに入れないで放置すると，変形やほこりが付着する原因になります。

このマニュアルの利用方法

このマニュアルの構成

このユーザーズマニュアルは，以下に示す第1章から第26章，付録，および索引で構成されています。

章	タイトル 内容
1	**ご使用になる前に** MX100/DARWIN用APIの概要を説明しています。 また，本ソフトウエアを使用するときに必要なPC環境，インストールの方法などについて説明しています。
2	**API用MX100－Visual C++－** 本APIを，Visual C++を使用してMX100に適用する場合について説明しています。機能，プログラム例，クラスの詳細を説明しています。
3	**API用MX100－Visual C－** 本APIを，Visual Cを使用してMX100に適用する場合について説明しています。機能やプログラム例を説明しています。
4	**API用MX100－Visual Basic－** 本APIを，Visual Basicを使用してMX100に適用する場合について説明しています。機能やプログラム例を説明しています。
5	**API用MX100用関数－Visual C/Visual Basic－** Visual C/Visual Basicで使用できるMX100用関数について詳細を説明しています。
6	**API用MX100の定数と型** Visual C++/Visual C/Visual Basicで使用できるMX100用定数と型について説明しています。
7	**API用DARWIN－Visual C++－** 本APIを，Visual C++を使用してDARWINに適用する場合について説明しています。機能，プログラム例，クラスの詳細を説明しています。
8	**API用DARWIN－Visual C－** 本APIを，Visual Cを使用してDARWINに適用する場合について説明しています。機能やプログラム例を説明しています。
9	**API用DARWIN－Visual Basic－** 本APIを，Visual Basicを使用してDARWINに適用する場合について説明しています。機能やプログラム例を説明しています。
10	**API用DARWIN用関数－Visual C/Visual Basic－** Visual C/Visual Basicで使用できるDARWIN用関数について詳細を説明しています。
11	**DARWINの定数と型** Visual C++/Visual C/Visual Basicで使用できるDARWIN用定数と型について説明しています。
12	**拡張API用MX100－Visual C++－** 本拡張APIを，Visual C++を使用してMX100に適用する場合について説明しています。機能，プログラム例，クラスの詳細を説明しています。
13	**拡張API用MX100－Visual C－** 本拡張APIを，Visual Cを使用してMX100に適用する場合について説明しています。機能やプログラム例を説明しています。
14	**拡張API用MX100－Visual Basic－** 本拡張APIを，Visual Basicを使用してMX100に適用する場合について説明しています。機能やプログラム例を説明しています。
15	**拡張API用MX100－Visual Basic.NET－** 本拡張APIを，Visual Basic.NETを使用してMX100に適用する場合について説明しています。機能やプログラム例を説明しています。
16	**拡張API用MX100－C#－** 本拡張APIを，C#を使用してMX100に適用する場合について説明しています。機能やプログラム例を説明しています。
17	**拡張API用MX100用関数－Visual C/Visual Basic/Visual Basic.NET/C#－** Visual C/Visual Basic/Visual Basic.NET/C#で使用できるMX100用関数について詳細を説明しています。

章	タイトル 内容
18	**拡張API用MX100の定数と型** Visual C++/Visual C/Visual Basic/Visual Basic.NET/C#で使用できるMX100用定数と型について説明しています。
19	**拡張API用DARWIN－Visual C++－** 本拡張APIを，Visual C++を使用してDARWINに適用する場合について説明しています。機能やプログラム例を説明しています。
20	**拡張API用DARWIN－Visual C－** 本拡張APIを，Visual Cを使用してDARWINに適用する場合について説明しています。機能やプログラム例を説明しています。
21	**拡張API用DARWIN－Visual Basic－** 本拡張APIを，Visual Basicを使用してDARWINに適用する場合について説明しています。機能やプログラム例を説明しています。
22	**拡張API用DARWIN用関数－Visual Basic.NET－** 本拡張APIを，Visual Basic.NETを使用してDARWINに適用する場合について説明しています。機能やプログラム例を説明しています。
23	**拡張API用DARWIN－C#－** 本拡張APIを，C#を使用してDARWINに適用する場合について説明しています。機能やプログラム例を説明しています。
24	**拡張API用DARWIN用関数(Visual C/Visual Basic/Visual Basic.NET/Visual C#)** Visual C/Visual Basic/Visual Basic.NET/Visual C#で使用できるDARWIN用関数について詳細を説明しています。
25	**拡張API用DARWINの定数と型** Visual C++/Visual C/Visual Basic/Visual Basic.NET/Visual C#で使用できるDARWIN用定数と型について説明しています。
26	**エラーメッセージ** エラーメッセージについて説明しています。
付録	本ソフトウエアで使用している用語と，MX100/DARWINの用語について説明しています。
索引	アルファベット順，五十音順の索引を記載しています。

このマニュアルにおける説明の範囲

OS

このマニュアルでは，OSの基本的な操作については説明していません。OSの基本的な操作については，それぞれのユーザーズガイドなどをお読みください。
また，開発環境であるVisual Studioや対応言語についての説明はしていません。それぞれの説明書をお読みください。

MX100/DARWIN

本APIで使用する用語の説明のみを掲載しています。MX100，DARWINの詳細については，それぞれのユーザーズマニュアルをご覧ください。

マニュアルの改版履歴

版数	説明
2版	MXAPI R2.01に対応。
3版	MXAPI R3.01に対応。

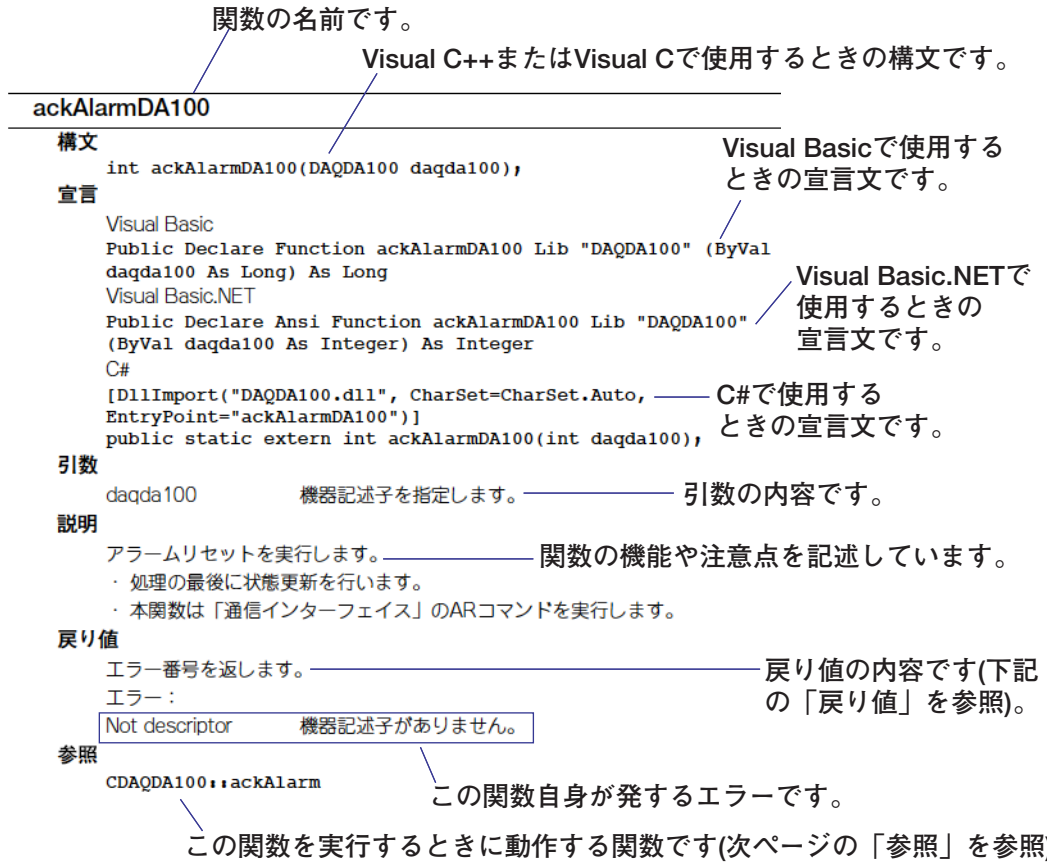
サンプルプログラム

・本書には，プログラム言語ごとのサンプルプログラムが記載されています。
・下記のホームページからサンプルプログラムをダウンロードできます。
取扱説明書CDの起動画面内の「最新情報へアクセス」ボタンをクリック＞「ダウンロード」メニュー＞「ソフトウエア一覧」と進むことによりアクセスできます。
＊ URLは変更されることがあります。
http://www.yokogawa.co.jp/ns/download/ns-index-free-ja.htm

このマニュアルで使用している記述方法

関数の記述方法

関数の説明では，次の記述方法を使用しています。クラスの関数メンバの記述についても下記を参照してください。



戻り値

エラーには，本関数自身が発するエラー，本関数を実行するときに動作する他の関数が発するエラー，および通信エラーがあります。
本関数自身が発するエラーを「エラー：」の部分に記述しています。
本関数を実行するときに動作する他の関数が発するエラーについては，「参照」項に記述されている関数または関数メンバの説明をご覧ください。

通信エラー

通信記述子を用いて通信を実行したときに発生した通信実行エラーです。エラー番号0から3のいずれかです。
エラー番号と対処方法については，26.1節，「APIによるエラーメッセージ」をご覧ください。

参照

ひとつの関数は，より単純な複数の機能(関数)で構成されています。「参照」欄には，関数の機能を構成する関数を記述しています。その関数の動作の詳細を理解したり，エラーが発生した場合の対応などに利用できます。

Visual C/Visual Basic/Visual Basic.NET/C#用の関数郡は，Visual C++クラスのインスタンスを呼び出して実行します。したがって「参照」欄には，クラス名と関数メンバ名が載ることがあります。
関数，クラス名と関数メンバは，アルファベット順で並んでいます。

プログラムの記述

プログラムは等幅フォントで記述されています。次の点にご注意ください。

・ 1行の記述が長い文は，ページ幅の制限により，複数行に渡って記述されています。

用語の参照

本APIの用語，MX100やDARWINの用語については，付録で説明しています。

その他

単位

K　「1024」の意味です。　使用例：100KB
M　「1024K」の意味です。　使用例：10MB

シンボル

Note　操作をするうえで，知っていると便利な情報が記載されています。

目次

はじめに i

ソフトウエア使用許諾契約書 ii
梱包内容の確認 iii
CD-ROMの取り扱い上の注意 iv
このマニュアルの利用方法 v
サンプルプログラム vii
このマニュアルで使用している記述方法 viii

第1章 ご使用になる前に

1.1 機能紹介 1-1
1.2 ソフトウエアの構成と特長 1-4
1.3 使用環境 1-6
1.4 インストール 1-8

API (2章～11章)

第2章 API用MX100－Visual C++－

2.1 MX100のクラス 2-1
2.2 機能とクラス/関数メンバの対応－MX100－ 2-4
2.3 プログラム－MX100/Visual C++－ 2-9
2.4 MX100/DARWIN共通クラス詳細 2-15
2.5 MX100用クラス詳細 2-37

第3章 API用MX100－Visual C－

3.1 機能と関数の対応 - MX100/Visual C 3-1
3.2 プログラム－MX100/Visual C－ 3-6

第4章 API用MX100－Visual Basic－

4.1 機能と関数の対応－MX100/Visual Basic－ 4-1
4.2 プログラム－MX100/Visual Basic－ 4-6

第5章 API用MX100用関数－Visual C/Visual Basic－

5.1 関数の詳細－MX100(Visual C/Visual Basic)－ 5-1

第6章 API用MX100の定数と型

6.1 MX100の定数の概要 6-1
6.2 MX100の定数 6-3
6.3 MX100の設定項目番号 6-17
6.4 MX100の型の概要 6-24
6.5 MX100の型 6-26

第7章 API用DARWIN－Visual C++－

7.1 DARWINのクラス 7-1
7.2 機能とクラス/関数メンバの対応－DARWIN－ 7-2
7.3 プログラム－DARWIN/Visual C++－ 7-5
7.4 DARWIN用クラス詳細 7-13

第8章 API用DARWIN－Visual C－
8.1 機能と関数の対応－DARWIN/Visual C－ 8-1
8.2 プログラム－DARWIN/Visual C－ 8-4

第9章 API用DARWIN－Visual Basic－
9.1 機能と関数の対応－DARWIN/Visual Basic－ 9-1
9.2 プログラム－DARWIN/Visual Basic－ 9-4

第10章 API用DARWIN用関数－Visual C/Visual Basic－
10.1 関数の詳細－DARWIN(Visual C/Visual Basic)－ 10-1

第11章 DARWINの定数と型
11.1 DARWINの定数の概要 11-1
11.2 DARWINの定数 11-2
11.3 DARWINの型の概要 11-10
11.4 DARWINの型 11-11

拡張API (12章～25章)

第12章 拡張API用MX100－Visual C++－
12.1 MX100のクラス 12-1
12.2 機能とクラス/関数メンバの対応－MX100－ 12-3
12.3 プログラム－MX100/Visual C++－ 12-15
12.4 MX100用クラス詳細 12-20

第13章 拡張API用MX100－Visual C－
13.1 機能と関数の対応－MX100/Visual C－ 13-1
13.2 プログラム－MX100/Visual C－ 13-12

第14章 拡張API用MX100－Visual Basic－
14.1 機能と関数の対応－MX100/Visual Basic－ 14-1
14.2 プログラム－MX100/Visual Basic－ 14-12

第15章 拡張API用MX100－Visual Basic.NET－
15.1 機能と関数の対応－MX100/Visual Basic.NET－ 15-1
15.2 プログラム－MX100/Visual Basic.NET－ 15-12

第16章 拡張API用MX100－C#－
16.1 機能と関数の対応－MX100/C#－ 16-1
16.2 プログラム－MX100/C#－ 16-12

第17章 拡張API用MX100用関数－Visual C/Visual Basic/Visual Basic.NET/C#－
17.1 関数の詳細－MX100(Visual C/Visual Basic/Visual Basic.NET/C#)－
状態遷移関数 17-1
17.2 関数の詳細－MX100(Visual C/Visual Basic/Visual Basic.NET/C#)－
取得関数 17-91

第18章 拡張API用MX100の定数と型

18.1 MX100の定数の概要 18-1
18.2 MX100の定数 18-3
18.3 M100の設定項目番号 18-19
18.4 MX100の型 18-20

第19章 拡張API用DARWIN－Visual C++－

19.1 DARWINのクラス 19-1
19.2 機能とクラス/関数メンバの対応－DARWIN－ 19-2
19.3 プログラム－DARWIN/Visual C++－ 19-7
19.4 瞬時値データ読み込み用機能と関数/クラスメンバの対応 19-10
19.5 瞬時値データ読み込み用プログラム－DARWIN/Visual C++－ 19-13
19.6 DARWIN用クラス詳細 19-16

第20章 拡張API用DARWIN－Visual C－

20.1 機能と関数の対応－DARWIN/Visual C－ 20-1
20.2 プログラム－DARWIN/Visual C－ 20-6
20.3 瞬時値データ読み込み用機能と関数の対応－DARWIN/Visual C－ 20-9
20.4 瞬時値データ読み込み用プログラム－DARWIN/Visual C－ 20-11

第21章 拡張API用DARWIN－Visual Basic－

21.1 機能と関数の対応－DARWIN/Visual Basic－ 21-1
21.2 プログラム－DARWIN/Visual Basic－ 21-6
21.3 瞬時値データ読み込み用機能と関数メンバの対応－DARWIN/Visual Basic－ 21-8
21.4 瞬時値データ読み込み用プログラム－DARWIN/Visual Basic－ 21-10

第22章 拡張API用DARWIN用関数－Visual Basic.NET－

22.1 機能と関数の対応－DARWIN/Visual Basic.NET－ 22-1
22.2 プログラム－DARWIN/Visual Basic.NET－ 22-6
22.3 瞬時値データ読み込み用機能と関数の対応－DARWIN/Visual Basic.NET－ 22-8
22.4 瞬時値データ読み込み用プログラム－DARWIN/Visual Basic.NET－ 22-10

第23章 拡張API用DARWIN－C#－

23.1 機能と関数の対応－DARWIN/C#－ 23-1
23.2 プログラム－DARWIN/C#－ 23-6
23.3 瞬時値データ読み込み用機能と関数の対応－DARWIN/C#－ 23-9
23.4 瞬時値データ読み込み用プログラム－DARWIN/C#－ 23-11

第24章 拡張API用DARWIN用関数(Visual C/Visual Basic/Visual Basic.NET/Visual C#)

24.1 関数の詳細－DARWIN(Visual C/Visual Basic/Visual Basic.NET/C#)－状態遷移関数 24-1
24.2 関数の詳細－DARWIN(Visual C/Visual Basic/Visual Basic.NET/C#)－取得関数 24-39
24.3 瞬時値データ読み込み用関数の詳細－DARWIN(Visual C/Visual Basic/Visual Basic.NET/C#)－状態遷移関数 24-74
24.4 瞬時値データ読み込み用関数の詳細－DARWIN(Visual C/Visual Basic/Visual Basic.NET/C#)－取得関数 24-81

第25章　拡張API用DARWINの定数と型
25.1　DARWINの定数の概要 25-1
25.2　DARWINの定数 25-2
25.3　DARWINの型 25-14
25.4　瞬時値データ読み込み用DARWINの定数の概要 25-15
25.5　瞬時値データ読み込み用DARWINの定数 25-16
25.6　瞬時値データ読み込み用DARWINの型 25-20

API，拡張API

第26章　エラーメッセージ
26.1　APIによるエラーメッセージ 26-1
26.2　MX100固有エラーメッセージ 26-3

付録
付録1 MX100に関する用語 付-1
付録2 DARWINに関する用語 付-14
付録3 MX100のタイムアウト値の算出 付-19
付録4 API改訂履歴(R2.01) 付-20
付録5 API改訂履歴(R3.01) 付-26

索引

1.1　機能紹介

対応機種

このソフトウエアは，MX100とDARWINに対応したAPI(Application Programming Interface)と拡張API(APIを容易に使用できるソフトウエア)を提供しています。

MX100　：すべての機種

DARWIN　：すべての機種(DA，DC，DR)。ただし，イーサネット通信機能(イーサネットモジュール)が必要です。

対応言語

APIの対応言語：　Visual C++，Visual C，およびVisual Basic

拡張APIの対応言語：　Visual C++，Visual C，Visual Basic，Visual Basic.NET，およびC#

MX100用の機能

通信機能

イーサネット接続されたMX100と通信する機能です。

設定機能

MX100のユニットやモジュールに関する設定や，チャネルに関する設定ができます。ただし，CAN Busモジュールのハードウエア設定はサポートしていません。ハードウエア設定には，モジュールに付属のソフトウエアをご使用ください。

データ取得機能

- FIFOごとの測定データの取得
  FIFO(First-In First-Out)バッファを介して，最新の測定データを取得することができます。また，瞬時値を取得することもできます。瞬時値の取得周期は，100ms以上です。
- チャネルごとの測定データの取得
  指定したチャネルの最新の測定データを取得することができます。また，瞬時値を取得することもできます。瞬時値の取得周期は，100ms以上です。
- 設定データの取得
  MX100のシステム構成情報，IPアドレスなどネットワークに関する情報，チャネルに関する設定情報などを取得できます。
- DO(Digital Output)データの一括取得
  DOチャネルのON/OFF状態を，一括取得します。
- チャネル情報データ(チャネル番号など)の取得
- AO/PWMデータの取得
  AO/PWMチャネルの出力を表すデータを取得します。
- 伝送出力データの取得
  AO/PWMチャネルの伝送状態を表すデータを取得します。

制御機能

- 日付/時刻の設定
- CFカードに測定データを保存する機能(バックアップ)の設定，CFカードのフォーマット
- システム(ユニット)の初期化，再構築
- アラームのリセット(アラームACK)
- 7セグメントLED表示内容の指定

ユーティリティ機能

測定値を文字列に変換，エラーメッセージ文字列の取得，その他のユーティリティ機能を提供しています。

DARWIN用の機能

本APIでは，DARWIN通信機能と同等の機能を提供できます。
DARWIN通信機能のうちいくつかの基本的な機能は，本APIのクラス(Visual C++の場合)または関数(Visual C，Visual Basic，Visual Basic.NET，C#の場合)として用意されています。
そのほかの機能は，DARWIN通信機能のコマンドを使用して実装することができます。

本ソフトウエアで用意されている機能

・通信機能

イーサネット接続されたDARWINと通信する機能です。

・設定機能

チャネルの測定レンジ設定やアラーム設定ができます。

・データ取得機能

- 測定データの取得
  測定データを，ASCIIまたはバイナリの形式で取得できます。瞬時値です。
- 設定データの取得
  運転モード，基本設定モード，A/D校正モードの設定データを取得できます。
- チャネル情報データ(チャネル番号など)の取得

- **制御機能**
  - 日付/時刻の設定
  - システムの再構築
  - 運転モードパラメータの初期化
  - アラームのリセット
  - 操作モードの切り替え
- **ユーティリティ機能**

  測定値を文字列に変換，エラーメッセージ文字列の取得，その他のユーティリティ機能を提供しています。

DARWIN通信機能のコマンドに相当した機能の実装

DARWIN通信機能のコマンドに相当した機能を，下記の方法で実装できます。

- Visual C++の場合：CDAQDARWINの継承クラスを作成し，runCommand関数メンバを使用してDARWIN通信機能のコマンドを実行する関数メンバを追加
- Visual C，Visual Basicの場合：runCommandDARWIN関数を使用してDARWIN通信機能のコマンドを実行する関数を作成
- Visual Basic.NET，C#の場合：拡張APIのrunCommandDA100関数を使用してDARWIN通信機能のコマンドを実行する関数を作成

注意事項

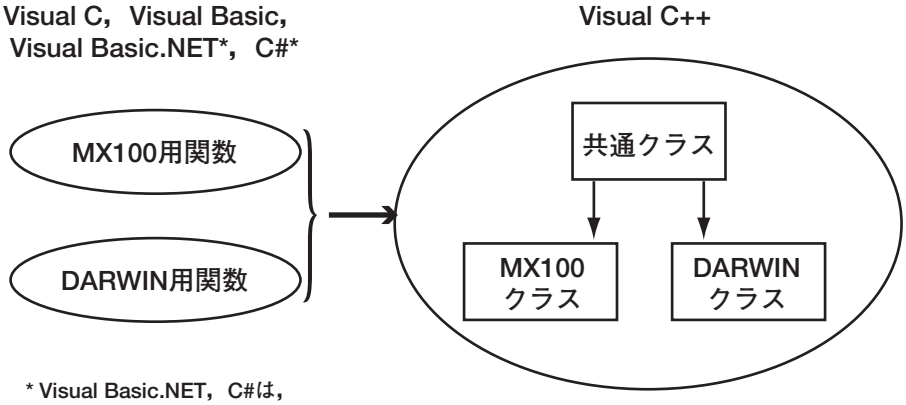
- MXAPI R3.01では構造体などが変更されているため，ユーザープログラムは再コンパイルと再リンクを行う必要があります。

1.2　ソフトウエアの構成と特長

ソフトウエアの構成

本ソフトウエアは，APIと拡張APIの2つのソフトウエアで構成されています。
本ソフトウエアでは，Visual C++のクラスを提供しています。
クラスは，共通クラスと，MX100用クラス，およびDARWIN用クラスで構成されています。MX100用クラスおよびDARWIN用クラスは，共通クラスを継承しています。

Visual C，Visual Basic，Visual Basic.NET，およびC#で使用する場合の関数群が用意されています。MX100用の関数群とDARWIN用の関数群があります。これらの関数は，Visual C++のクラスインスタンスを呼び出して実行します。



* Visual Basic.NET，C#は，
拡張APIで対応しています。

特長

- コントロール対象のユニットとの接続を意味するハンドル(「機器記述子」と呼んでいます)を取得することにより，コントロールを行います。このため，MX100やDARWINのハードウエア環境に左右されない記述ができます。
- 本APIでは，機能に対応したデータのまとまりを構造体として提供しています。たとえば，MX100のモジュール情報の構造体には，「モジュールの種類」，「チャネル数」，「測定周期」そのほかのモジュールに関する情報がまとめられています。同じデータを異なるコマンドにより異なる形式で取得した場合でも，同じ構造体で記述することができます。
- 拡張APIは，本APIの上位APIに位置付けられています。APIを呼び出して動作します。拡張APIは，内部に現在の状態のデータを保持します。拡張APIは，構造体を使用しなくてもプログラムを記述することができます。

ファイル構成

本ソフトウエアは，以下のファイルで構成されています。

種類	拡張子	記事
実行可能モジュール	.dll	実行ファイルです。
インクルードファイル	.h	Visual C，Visual C++のインクルードファイルです。
ライブラリファイル	.lib	Visual C，Visual C++でリンクするファイルです。
Visual Basic用標準モジュールファイル	.bas	Visual Basic用の定義をまとめたファイルです。
APIビューアテキストファイル	.txt	Visual Basic用関数，型，定数を，テキスト形式で記述したファイルです。Visual StudioのAPIビューアで利用できます。
Visual Basic.NET用標準モジュール	.vb	Visual Basic.NET用の定義をまとめたファイルです。
C#用標準モジュールファイル	.cs	C#用の定義をまとめたファイルです。

1.3　使用環境

PC(パーソナルコンピュータ)

本ソフトウエアは，下記の条件を満たすPCにインストールして使用できます。

Note

- CPU，メモリ，ハードディスクには制限を設けませんが，性能は，それらの環境に左右されます。
- 本APIをインストールするには，ハードディスクに10MB以上の空き領域が必要です。

オペレーティングシステム

次のいずれかのオペレーティングシステムが動作していること。

- Windows NT 4.0 SP3以降
- Windows 2000
- Windows XP
- Windows Vista Home Premium
- Windows Vista Business

CD-ROMドライブ

インストール時に使用します。

マウス

OSに対応したマウス。

ディスプレイ

OSに対応したディスプレイ。

通信ポート

OSがサポートするイーサネットポート。また，TCP/IPプロトコルがインストールされていること。

対応言語

- Visual C++
- Visual C
- Visual Basic
- Visual Basic.NET(拡張APIだけ)
- C#(拡張APIだけ)

ユーザー開発環境

対応言語を使用できる環境であること。

Visual Studio 6.0 SP5以降を使用してください。他の環境での本ソフトウエアの動作を保証しません。

Visual Studio 2005以降のVisual StudioでVisual C/Visual C++を使用する場合，time_t型に互換性がありません。

Visual Studio 2005の場合，「構成プロパティ」-「C/C++」-「プリプロセッサ」の「プリプロセッサの定義」の項に「USE_32BIT_TIME_T」を指定してください。

MX100

特に制限はありません。

DARWIN

DA100，DC100，DR130，DR230，およびDR240
イーサネット通信機能(イーサネットモジュール)が必要です。

1.4　インストール

操作手順

1. パーソナルコンピュータ本体の電源を入れ，OSを立ち上げた状態にします。

Note

インストールする前に，ウイルス防止用などの常駐ソフトウエアを終了してください。

2. 本ソフトウエアのCDをCD-ROMドライブにセットします。
3. しばらくすると，下記のDAQMASTER　DARWIN　MXAPIウインドウとInstallShield Wizardのウインドウが表示されたあと，その次の画面に切り替わります。あとは画面の指示に従って操作します。

DAQMASTER DARWIN MXAPIウインドウが表示されない場合は，下記のように操作します。

「マイコンピュータ」のアイコンをダブルクリックして開き，CD-ROMアイコンをダブルクリックします。「Disk1」のアイコンをダブルクリックし，フォルダ内の「Setup.exe」をダブルクリックします。DAQMASTER DARWIN MXAPIウインドウとInstall Shield Wizardのウインドウが表示されたあと，その次の画面に切り替わります。あとは，画面の指示に従って操作します。

Note

・本ソフトウエアの再インストール
　本ソフトウエアがインストールされているPCで，操作手順3の操作を行うと，「ファイル削除の確認」ウインドウが表示されます。OKを選択して本ソフトウエアを削除します。そのあと，操作手順3の操作でインストールします。
・Windowsのコントロールパネルにある「アプリケーションの追加と削除」で，本APIをアンインストールできます。

2.1　MX100のクラス

本APIは，下図のように，MX100/DARWIN共通のクラスとMX100専用のクラスで構成されています。

- ■ CDAQChInfo
  - ・ CDAQMXChID
    - ・ CDAQMXChInfo
    - ・ CDAQMXChConfig
- ■ CDAQDataInfo
  - ・ CDAQMXDataInfo
- ■ CDAQDateTime
  - ・ CDAQMXDateTime
- ■ CDAQHandler
  - ・ CDAQMX
- ・ CDAQMXChConfigData
- ・ CDAQMXConfig
- ・ CDAQMXDOData
- ・ CDAQMXNetInfo
- ・ CDAQMXSegment
- ・ CDAQMXStatus
- ・ CDAQMXSysInfo
- ・ CDAQMXBalanceData
  - ・ CDAQMXBalanceResult
- ・ CDAQMXOutputData
- ・ CDAQMXAOPWMData
- ・ CDAQMXTransmit

■ ：MX100とDARWINに共通のクラスです。
・ ：MX100専用のクラスです。

CDAQChInfoクラス

チャネル情報データを格納する基底クラスです。

CDAQDataInfoクラス

測定データを格納する基底クラスです。

CDAQDateTimeクラス

時刻情報データを格納する基底クラスです。

CDAQHandlerクラス

機器(MX100/DARWIN)本体と通信を行うハンドラの基底クラスです。

CDAQMXクラス

CDAQHandlerクラスの派生クラスです。MX100用の機能を提供します。

CDAQMXAOPWMData

MX100でのAO/PWMデータを格納するクラスです。MXAOPWMData構造体のラッパクラスです。

CDAQMXBalanceData

MX100での初期バランスデータを格納するクラスです。MXBalanceData構造体のラッパクラスです。

CDAQMXBalanceResult

MX100での初期バランス結果を格納するクラスです。MXBalanceResult構造体のラッパクラスです。

CDAQMXChConfigクラス

CDAQMXChIDクラスの派生クラスです。チャネル設定データを格納するクラスです。MXChConfig構造体のラッパクラスです。

CDAQMXChConfigDataクラス

全チャネル分のチャネル設定データを格納するクラスです。MXChConfigData構造体のラッパクラスです。

CDAQMXChIDクラス

CDAQChInfoクラスの派生クラスです。チャネル識別情報を格納するクラスです。MXChID構造体のラッパクラスです。

CDAQMXChInfoクラス

CDAQMXChIDクラスの派生クラスです。チャネル情報データを格納するクラスです。MXChInfo構造体のラッパクラスです。

CDAQMXConfigクラス

設定データを格納するクラスです。MXConfigData構造体のラッパクラスです。

CDAQMXDataInfoクラス

CDAQDataInfoクラスの派生クラスです。測定データを格納するクラスです。MXDataInfo構造体のラッパクラスです。

CDAQMXDateTimeクラス

CDAQDateTimeクラスの派生クラスです。時刻情報データを格納するクラスです。MXDateTime構造体のラッパクラスです。

CDAQMXDODataクラス

DOデータを格納するクラスです。MXDODData構造体のラッパクラスです。

CDAQMXNetInfoクラス

ネットワーク情報データを格納するクラスです。MXNetInfo構造体のラッパクラスです。

CDAQMXOutputData

MX100での出力チャネルのデータを格納するクラスです。MXOutputData構造体のラッパクラスです。

CDAQMXSegmentクラス

7セグメントLEDの表示パターンを格納するクラスです。MXSegment構造体のラッパクラスです。

CDAQMXStatusクラス

MX100のステータスデータを格納するクラスです。MXStatus構造体のラッパクラスです。

CDAQMXSysInfoクラス

MX100のシステム構成データを格納するクラスです。MXSystemInfo構造体のラッパクラスです。

CDAQMXTransmit

MX100での伝送出力データを格納するクラスです。MXTransmit構造体のラッパクラスです。

2.2　機能とクラス/関数メンバの対応ーMX100ー

本ソフトウエアでサポートする機能と，クラスの対応を示します。

通信機能

機能	クラスと関数メンバ
MX100と通信接続	CDAQMX::open
MX100との通信を切断	CDAQMX::close
通信タイムアウトを設定	CDAQMX::setTimeOut

Note

通信タイムアウトの設定を推奨しません。**理由**：データ取得時にタイムアウト時間に抵触して予期しない通信切断が発生する場合があります。

制御機能

FIFOの開始/停止

機能	クラスと関数メンバ
FIFOを開始	CDAQMX::startFIFO
FIFOを停止	CDAQMX::stopFIFO
FIFOの自動制御を設定	CDAQMX::autoFIFO

そのほかの制御

機能	FIFO	クラスと関数メンバ
MX100に時刻情報(基準日時(1970年1月1日)からの時間)を秒数で設定	停止	CDAQMX::setDateTime
CFへのデータ保存(バックアップ)のON/OFFを設定	継続	CDAQMX::setBackup
CF書き込み種類	停止	CDAQMXSysInfo::setCFWriteMode
CFをフォーマット	停止	CDAQMX::formatCF
・ユニットのシステム再構築	停止	CDAQMX::initSystem
・ユニットのシステム初期化	停止	
・ユニットのアラームリセット(アラームACK)	継続	

表の「FIFO」欄は，FIFO中に関数メンバを実行したときの，FIFOの動作を示します。
停止：関数メンバを実行するとFIFOを停止します。
継続：関数メンバを実行してもFIFOを継続します。
バックアップの設定でCF書き込み種類は，一括取得したデータの部分変更です。変更後，一括送信が必要です。

Note

FIFOの自動制御を設定しておくと，関数メンバの実行によりFIFOが停止したあと，FIFOを自動的に再開します。

設定機能

一括設定

機能		FIFO	クラスと関数メンバ
設定データを一括設定		停止	CDAQMX::setConfig
DO(Digital Output)データを一括設定		継続	CDAQMX::setDOData
基本設定を一括設定		停止	CDAQMX::setMXConfig
7セグメントLEDの表示を設定		継続	CDAQMX::setSegment
AO/PWMデータ送信		継続	CDAQMX::setAOPWMData
伝送出力データ送信		継続	CDAQMX::setTransmit
初期バランスデータの設定		停止	CDAQMX::setBalance
出力チャネルデータの設定		停止	CDAQMX::setOutput
初期バランスデータ	実行	停止	CDAQMX::runBalance
	リセット	停止	CDAQMX::resetBalance

表の「FIFO」欄については，前ページの「そのほかの制御」の説明をご覧ください。

設定変更

機能		クラスと関数メンバ
レンジ設定	スキップ(未使用)	CDAQMXConfig::setSKIP
	直流電圧入力	CDAQMXConfig::setVOLT
	熱電対入力	CDAQMXConfig::setTC
	測温抵抗体入力	CDAQMXConfig::setRTD
	ディジタル入力(DI)	CDAQMXConfig::setDI
	チャネル間差演算	CDAQMXConfig::setDELTA
	リモートRJC	CDAQMXConfig::setRRJC
	抵抗	CDAQMXConfig::setRES
	ひずみ	CDAQMXConfig::setSTRAIN
	AO	CDAQMXConfig::setAO
	PWM	CDAQMXConfig::setPWM
	パルス	CDAQMXConfig::setPULSE
	通信	CDAQMXConfig::setCOM
チャネルに単位名を設定		CDAQMXChID::setUnit
チャネルのタグを設定		CDAQMXChID::setTag
チャネルにコメントを設定		CDAQMXChID::setComment
チャネルにアラームを設定		CDAQMXChConfig::setAlarm
チャネルで使用する基準接点補償(RJC)を設定		CDAQMXChConfig::setRJCType
チャネルにフィルタを設定		CDAQMXChConfig::setFilter
チャネルにバーンアウト検出時動作を設定		CDAQMXChConfig::setBurnout
アラーム出力を指定したDOチャネルに，関連づけるアラームを設定 (DOチャネルをアラーム出力に指定するときは，setDOType関数メンバを用います。)		CDAQMXChConfig::setRefAlarm
チャネルにチャタリングフィルタを設定		CDAQMXChConfig::setChatFilter
測定周期を設定		CDAQMXConfig::setInterval
温度単位を設定		CDAQMXConfig::setTempUnit
ユニットの識別番号を設定		CDAQMXSysInfo::setUnitNo
タイムアウト値(通信切断時に，CFへのデータ保存を開始するまでの時間)を設定 タイムアウト値の算出については，付録3を参照してください。		CDAQMXSysInfo::setCFTimeout
DOチャネルに関連づける信号の種類を設定		CDAQMXConfig::setDOType
AOチャネルに関連づける信号の種類を選択		CDAQMXConfig::setAOType
PWMチャネルに関連づける信号の種類を選択		CDAQMXConfig::setPWMType
出力チャネルデータの種類を設定		CDAQMXOutputData::setOutputType
出力データの電源ON時，エラー発生時の出力値の設定		CDAQMXOutputData::setChoice
PWM出力チャネルのパルス周期の倍率の設定		CDAQMXOutputData::setPulstime
DOデータを部分変更		CDAQMXDOData::setDO
AO/PWMデータの部分変更		CDAQMXAOPWMData::setAOPWM
初期バランスデータの部分変更		CDAQMXBalanceData::setBalance
伝送出力データの部分変更		CDAQMXTransmit::setTransmit

Note

- 設定変更するときは，CDAQMX::getConfigで取得した設定データを，設定変更関数メンバで変更後，CDAQMX::setConfigでMX100に一括設定します。
- DOデータの場合は，CDAQMXDODataの内容を，設定変更関数メンバで変更後，CDAQMX::setDODataでMX100に一括設定します。
- AO/PWMデータの場合は，CDAQMXAOPWMDataの内容を，設定変更関数メンバで変更後，CDAQMX::setAOPWMDataでMX100に一括設定します。
- 伝送出力データの場合は，CDAQMXTransmitの内容を，設定変更関数メンバで変更後，CDAQMX::setTransmitでMX100に一括設定します。

データ取得機能

システムステータスデータ/システム構成データの取得

機能	クラスと関数メンバ
システムステータスデータを取得	CDAQMX::getStatusData
システム構成データを取得	CDAQMX::getSystemConfig

設定データの取得

機能	クラスと関数メンバ
設定データを一括取得	CDAQMX::getConfig
基本設定を一括取得	CDAQMX::getMXConfig
設定データの取得を宣言 チャネル設定データ以外の設定データを取得します。	CDAQMX::talkConfig
チャネル設定データを取得 talkConfig関数メンバで設定データの取得を宣言した後にチャネル設定データを取得する関数です。	CDAQMX::getChConfig

DOデータの取得

機能	クラスと関数メンバ
DOデータを一括取得	CDAQMX::getDOData

チャネル情報データの取得

機能	クラスと関数メンバ
チャネル情報データの取得を宣言	CDAQMX::talkChInfo
チャネル情報データの取得	CDAQMX::getChInfo

測定データの取得(チャネル指定)

機能	クラスと関数メンバ
指定したチャネルの最新のデータ範囲を取得	CDAQMX::getChDataNo
指定したチャネルの測定データ取得を宣言 指定したチャネルの瞬時値取得を宣言	CDAQMX::talkChData
指定したチャネルの時刻情報をデータ番号ごとに取得	CDAQMX::getTimeData
指定したチャネルの測定データを取得	CDAQMX::getChData

測定データの取得(FIFO指定)

機能	クラスと関数メンバ
指定したFIFO番号の最新のデータ範囲を取得	CDAQMX::getFIFODataNo
指定したFIFO番号の測定データ取得を宣言 指定したFIFO番号の瞬時値取得を宣言	CDAQMX::talkFIFOData
指定したFIFO番号の時刻情報をデータ番号ごとに取得	CDAQMX::getTimeData
指定したFIFO番号の測定データを取得	CDAQMX::getChData

初期バランスデータの取得

機能	クラスと関数メンバ
初期バランスデータの取得	CDAQMX::getBalance

出力チャネルデータの取得

機能	クラスと関数メンバ
出力チャネルデータの取得	CDAQMX::getOutput
AO/PWMデータと伝送出力データの取得	CDAQMX::getAOPWMData

ユーティリティ

機能	クラスと関数メンバ
指定したユーザカウント(ユーザーが定義した順序情報)を，次に発行するパケットに挿入	CDAQMX::setUserTime
通信で最後に受信したMX100固有エラーを取得	CDAQMX::getLastError
測定値を倍精度浮動小数に変換	CDAQMXDataInfo::toDoubleValue
測定値を文字列に変換	CDAQMXDataInfo::toStringValue
アラーム種類の文字列を取得	CDAQMXDataInfo::getAlarmName
アラーム文字列の最大長を取得	CDAQMXDataInfo:: getMaxLenAlarmName
本APIのバージョン番号を取得	CDAQMX::getVersionAPI
本APIのリビジョン番号を取得	CDAQMX::getRevisionAPI
エラーメッセージ文字列を取得	CDAQMX::getErrorMessage
エラーメッセージ文字列の最大長を取得	CDAQMX::getMaxLenErrorMessage
エラー検出した設定項目番号を取得	CDAQMX::getItemError
AO/PWM出力値を出力データ値に変換	CDAQMXAOPWMData::toAOPWMValue
AO/PWM出力データ値を出力値に変換	CDAQMXAOPWMData::toRealValue
データ番号の有効性チェック	CDAQMXStatus::isDataNo
スタイルバージョンに変換	CDAQMXSysInfo::toStyleVersion

2.3　プログラムーMX100/Visual C++ー

インクルードファイルのパスを追加

プロジェクトに，インクルードファイル(DAQMX.h)のパスを追加します。追加方法は，ご使用の環境により異なります。

ソースファイルでの宣言

ソースファイルに宣言を記述します。

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

Note

共通部のインクルードファイル(DAQHandler.h)は，上記インクルードファイルから参照されているので，宣言を記述する必要はありません。

ライブラリの指定

プロジェクトにライブラリ(DAQMX.lib，DAQHandler.lib)を追加します。追加方法は，ご使用の環境により異なります。
すべてのクラスが使用可能になります。Visual C用の関数郡も使用できます。

測定データの取得

プログラム例1

測定データを取得するプログラムです。

```
////////////////////////////////////////////////////////////
// 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;
}
////////////////////////////////////////////////////////////
```

説明

全般

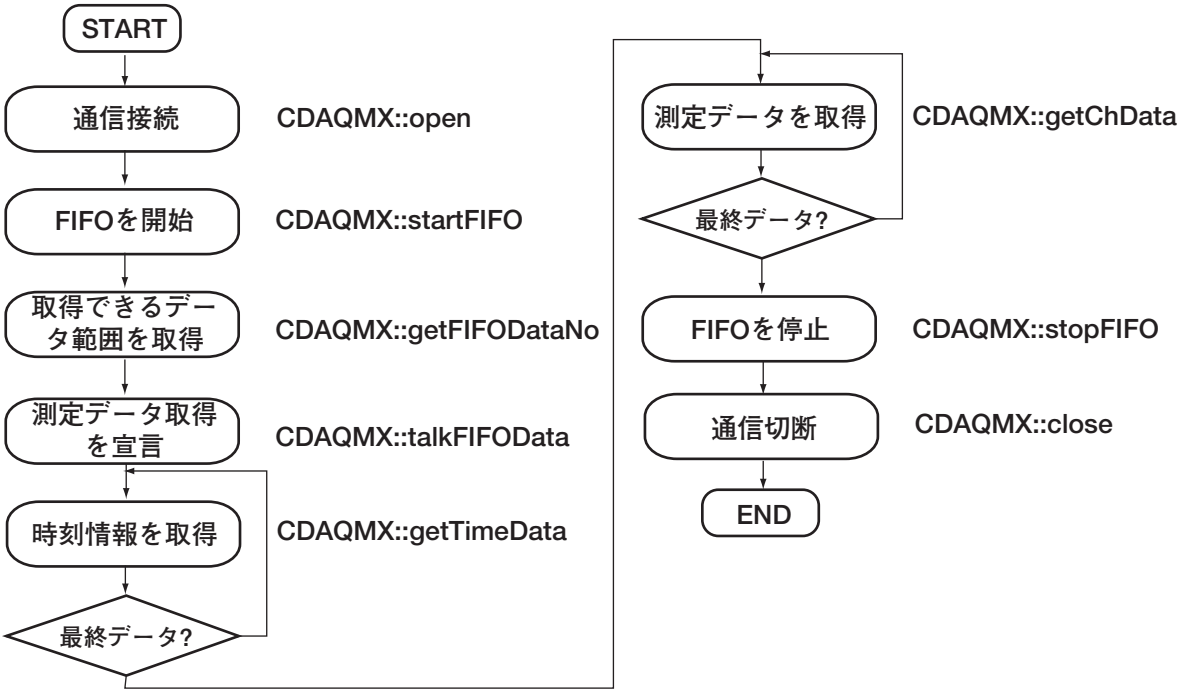
データ取得は，FIFOを開始することで可能になります。取得範囲はFIFO番号とデータ番号で指定します。データ番号に対応する時刻と測定データを個別に取得します。終了はフラグにより判断します。

インクルードファイルの記述

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

処理の流れ

下記のフローチャートでは，宣言部分を省略しています。



通信処理

最初に通信接続を行います。通信接続後，各関数メンバが利用可能です。最後に終了処理として，通信切断を行います。

Note

約3分間アクセスがない場合，MX100が通信を切断します。長時間アクセスをしない場合には通信を切断し，必要なときに通信接続してください。

通信接続

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

MX100のIPアドレスを指定しています。
通信用ポートは，通信用定数DAQMX_COMMPORT(MX100の通信ポート番号)を指定したことになります。

Note

クラスの構築時に通信接続をすることも可能です。 消滅時には，通信切断を行います。

FIFO開始

```
startFIFO()
```

FIFOを開始します。

データ範囲の取得

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

指定したFIFO番号の，最後に取得したデータの次のデータから最新データまでの範囲を，データ番号で取得します。

トーカ

```
talkFIFOData(0, startNo, endNo)
```

データ範囲を指定して，FIFOデータの取得を宣言します(測定データ取得宣言)。

FIFOデータ時刻情報の取得

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

指定範囲の時刻情報を，データ番号単位で取得します。
終了はフラグ(DAQMX_FLAG_ENDDATA定数)により判断します。

Note

usertimeは，ユーザーによる順序情報(ユーザーカウント)です。あらかじめ，setUserTime関数メンバで設定した値が格納されます。

FIFOデータの取得

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

指定範囲の測定データを，チャネル単位で取得します。
終了はフラグ(DAQMX_FLAG_ENDDATA定数)により判断します。

FIFO停止

```
stopFIFO()
```

FIFOを停止します。

通信切断

```
closeMX(comm)
```

通信を切断します。

設定データの取得/設定

プログラム例2

下記の3つを実行するプログラムです。このプログラムではまとめて記述していますが，それぞれ個別に記述して実行できます。

・ 設定データを一括取得
・ MX100に設定データを一括設定
・ チャネルに直流電圧レンジを設定

```
////////////////////////////////////////////////////////////
// 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;
}
////////////////////////////////////////////////////////////
```

説明

設定データの一括取得

`getConfig(configdata)`

設定データの一括取得により，下記の設定データを取得できます。

・ システム構成データ：2.5節のCDAQMXSysInfoクラスを参照。
・ ステータスデータ：2.5節のCDAQMXStatusクラスを参照。
・ 基本設定：2.5節のCDAQMXConfigクラスを参照。

Note

設定データの取得方法には，talkConfig関数メンバとgetChConfig関数メンバを使用する方法もあります。talkConfigで設定データ取得宣言をして，システム構成データ，ステータス，ネットワーク情報データを取得し，getChConfigでチャネル設定データをチャネル単位で取得します。

設定データの一括設定

```
setConfig(configdata)
```

設定データの一括設定では，下記のデータを設定できます。

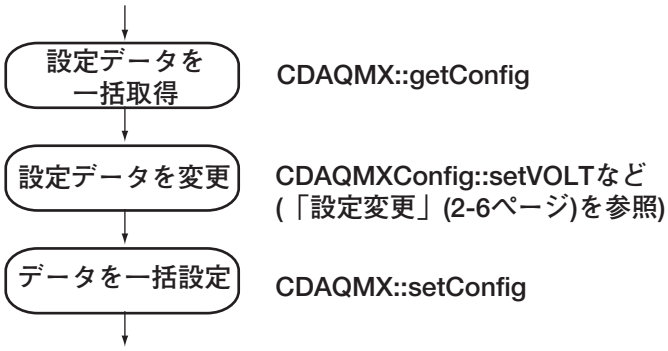
- システム構成データ：2.5節のCDAQMXSysInfoクラスを参照。
- 基本設定：2.5節のCDAQMXConfigクラスを参照。

チャネルに直流電圧レンジを設定

```
setVOLT(1, DAQMX_RANGE_VOLT_20MV)
```

チャネル番号1に，直流電圧レンジ「20mV」を設定します。スケーリングは使用しません。

最初に設定データを一括取得し，上記の変更をした後，MX100にデータを一括設定しています。設定変更を行う場合には，このように処理します。



エラー処理

- ほとんどの関数メンバは，戻り値として，関数の処理結果の状態をエラー番号で返します(正常終了の場合はエラー番号「0」)。
- エラー番号に対応するエラーメッセージ文字列を得ることができる関数メンバ(CDAQMX::getErrorMessage)があります。 また，エラーメッセージ文字列の最大長を得る関数メンバ(CDAQMX::getMaxLenErrorMessage)もあります。
- MX100からのMX100固有エラーは，関数メンバ(CDAQMX::getLastError)で取得できます。
- 設定データ不正のエラーになった場合，エラー検出した設定項目番号を関数で取得できます。4.2節を参照してください。

2.4 MX100/DARWIN共通クラス詳細

クラスは，クラス名のアルファベット順で並んでいます。

CDAQChInfoクラス

本クラスは，チャネル情報データの基底クラスです。
データメンバ操作は，継承クラスで必要に応じてオーバーライドすることができます。
インスタンスが本クラスを継承しているかをチェックする機能を提供します。継承クラスでオーバーライドすることができます。

パブリックメンバ

構築・消滅

CDAQChInfo	オブジェクトを構築します。
~CDAQChInfo	オブジェクトを消滅します。

データメンバ操作

initialize	データメンバを初期化します。
getChType	チャネルタイプを取得します。
getChNo	チャネル番号を取得します。
getPoint	小数点位置を取得します。
setChType	チャネルタイプを設定します。
setChNo	チャネル番号を設定します。
setPoint	小数点位置を設定します。

ユーティリティ

isObject	オブジェクトをチェックします。

演算子

operator=	代入を実行します。

プロテクトメンバ

データメンバ

m_chType	チャネルタイプの格納領域です。
m_chNo	チャネル番号の格納領域です。
m_point	小数点位置の格納領域です。

プライベートメンバ

なし。

関数メンバ(アルファベット順)

CDAQChInfo::CDAQChInfo

構文

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

引数

chType	チャネルタイプを指定します。
chNo	チャネル番号を指定します。
point	小数点位置を指定します。

説明

オブジェクトを構築，消滅します。
構築時，データメンバに指定された値を設定します。

参照

`setChNo setChType setPoint`

CDAQChInfo::getChNo

構文

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

説明

データメンバのチャネル番号領域の値を取得します。

戻り値

チャネル番号を返します。

CDAQChInfo::getChType

構文

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

説明

データメンバのチャネルタイプ領域の値を取得します。

戻り値

チャネルタイプを返します。

CDAQChInfo::getPoint

構文

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

説明

データメンバの小数点位置領域の値を取得します。

戻り値

小数点位置を返します。

CDAQChInfo::initialize

構文

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

説明

データメンバを初期化します。初期値は，0です。

参照

```
setChType setChNo setPoint
```

CDAQChInfo::isObject

構文

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

引数

classname　　クラス名を文字列で指定します。

説明

指定されたクラス名を継承しているかをチェックします。
引数が省略された場合，本クラスであるかをチェックします。
本クラスを継承したクラスが，自分自身のクラスをチェックするためには，オーバーライドする必要があります。
クラスを継承している場合，真(有効)を返します。それ以外は，偽(無効)を返します。

戻り値

真偽を1(真)，または，0(偽)で返します。

CDAQChInfo::operator=

構文

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

引数

cChInfo　　代入するオブジェクトを指定します。

説明

指定されたオブジェクトのデータメンバを複写します。

戻り値

本オブジェクトへの参照を返します。

CDAQChInfo::setChNo

構文

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

引数

chNo　　チャネル番号を指定します。

説明

データメンバのチャネル番号領域に指定された値を格納します。

CDAQChInfo::setChType

構文

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

引数

chType　　チャネルタイプを指定します。

説明

データメンバのチャネルタイプ領域に指定された値を格納します。

CDAQChInfo::setPoint

構文

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

引数

point　　小数点位置を指定します。

説明

データメンバの小数点位置領域に指定された値を格納します。

CDAQDataInfoクラス

本クラスは，測定データの基底クラスです。
チャネル情報データと関連付けることで，測定値を取得できます。
データメンバ操作は，継承クラスで必要に応じてオーバーライドすることができます。
インスタンスが本クラスを継承しているかをチェックする機能を提供します。
継承クラスでオーバーライドすることができます。

パブリックメンバ

構築・消滅

CDAQDataInfo	オブジェクトを構築します。
~CDAQDataInfo	オブジェクトを消滅します。

データメンバ操作

initialize	データメンバを初期化します。
getValue	データ値を取得します。
setValue	データ値を設定します。

関連付け

getClassChInfo	チャネル情報データとの関連を取得します。
setClassChInfo	チャネル情報データとの関連を設定します。

測定値書式

getDoubleValue	測定値を取得します。
getStringValue	測定値を文字列で取得します。
toDoubleValue	測定値を生成します。
toStringValue	測定値を文字列で生成します。

ユーティリティ

isObject	オブジェクトをチェックします。

演算子

operator=	代入を実行します。

プロテクトメンバ

データメンバ

m_value	データ値の格納領域です。
m_pChInfo	チャネル情報データとの関連です。CDAQChInfoへのポインタの格納領域です。

プライベートメンバ

なし。

関数メンバ(アルファベット順)

CDAQDataInfo::CDAQDataInfo

構文

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

引数

value	データ値を指定します。
pcChInfo	チャネル情報データとの関連を指定します。

説明

オブジェクトを構築，消滅します。
構築時，データメンバに指定された値を設定します。
消滅時，関連付けされたチャネル情報データは削除されません。

参照

`setClassChInfo setValue`

CDAQDataInfo::getClassChInfo

構文

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

説明

データメンバのチャネル情報データとの関連領域の値を取得します。
設定されていない場合，NULLを返します。

戻り値

チャネル情報データとの関連を返します。

CDAQDataInfo::getDoubleValue

構文

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

説明

データメンバのデータ値とチャネル情報データとの関連の小数点位置から，測定値を生成します。
チャネル情報データとの関連が存在しない場合，小数点位置は0です。

戻り値

測定値を倍精度浮動小数で返します。

参照

`getClassChInfo getValue toDoubleValue`
`CDAQChInfo::getPoint`

CDAQDataInfo::getStringValue

構文

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

引数

strValue　　文字列を格納する領域を指定します。
lenValue　　文字列を格納する領域のバイト数を指定します。

説明

データメンバのデータ値とチャネル情報データとの関連の小数点位置から，測定値を生成します。
生成された測定値を文字列に変換して，指定された領域に格納します。
チャネル情報データとの関連が存在しない場合，小数点位置は0です。
領域に格納する文字列には，終端も含まれます。
実際の文字列の長さが戻り値になります。戻り値に，終端は含まれません。

戻り値

文字列の長さを返します。

参照

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

CDAQDataInfo::getValue

構文

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

説明

データメンバのデータ値領域の値を取得します。

戻り値

データ値を返します。

CDAQDataInfo::initialize

構文

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

説明

データメンバを初期化します。
初期値は，0です。
チャネル情報データとの関連は初期化しません。

参照

```
setValue
```

CDAQDataInfo::isObject

構文

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

引数

classname　　クラス名を文字列で指定します。

説明

指定されたクラス名を継承しているかをチェックします。
引数が省略された場合，本クラスであるかをチェックします。
本クラスを継承したクラスが，自分自身のクラスをチェックするためには，オーバーライドする必要があります。
クラスを継承している場合，真(有効)を返します。それ以外は，偽(無効)を返します。

戻り値

真偽を1(真)，または，0(偽)で返します。

CDAQDataInfo::operator=

構文

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

引数

cDataInfo　　代入するオブジェクトを指定します。

説明

指定されたオブジェクトのデータメンバを複写します。
チャネル情報データとの関連も複写されます。

戻り値

本オブジェクトへの参照を返します。

CDAQDataInfo::setClassChInfo

構文

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

引数

pcChInfo　　チャネル情報データとの関連を指定します。

説明

データメンバのチャネル情報データとの関連領域に指定された値を格納します。

CDAQDataInfo::setValue

構文

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

引数

value	データ値を指定します。

説明

データメンバのデータ値領域に指定された値を格納します。

CDAQDataInfo::toDoubleValue

構文

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

引数

value	データ値を指定します。
point	小数点位置を指定します。

説明

指定されたデータ値と小数点位置から測定値を生成します。

戻り値

測定値を倍精度浮動小数で返します。

CDAQDataInfo::toStringValue

構文

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

引数

value	データ値を指定します。
point	小数点位置を指定します。
strValue	文字列を格納する領域を指定します。
lenValue	文字列を格納する領域のバイト数を指定します。

説明

指定されたデータ値と小数点位置から測定値を生成します。
生成された測定値を文字列に変換して，指定された領域に格納します。
領域に格納する文字列には，終端も含まれます。
実際の文字列の長さが戻り値になります。戻り値に，終端は含まれません。

戻り値

文字列の長さを返します。

参照

`toDoubleValue`

CDAQDateTimeクラス

本クラスは，時刻情報データの基底クラスです。
秒数とミリ秒をデータメンバに持ちます。
データメンバ操作は，継承クラスで必要に応じてオーバーライドすることができます。
インスタンスが本クラスを継承しているかをチェックする機能を提供します。継承クラスでオーバーライドすることができます。

パブリックメンバ

構築・消滅

CDAQDateTime	オブジェクトを構築します。
~CDAQDateTime	オブジェクトを消滅します。

データメンバ操作

initialize	データメンバを初期化します。
getTime	秒数を取得します。
getMilliSecond	ミリ秒を取得します。
setTime	秒数を設定します。
setMilliSecond	ミリ秒を設定します。
setNow	現在の日付時刻を設定します。

ユーティリティ

isObject	オブジェクトをチェックします。
toLocalDateTime	タイムゾーンに従った年月日時分秒に変換します。

演算子

operator=	代入を実行します。

プロテクトメンバ

データメンバ

m_time	1970年01月01日からの秒数の格納領域です。
m_milliSecond	ミリ秒の格納領域です。

プライベートメンバ

なし。

関数メンバ(アルファベット順)

CDAQDateTime::CDAQDateTime

構文

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

引数

time	秒数を指定します。
milliSecond	ミリ秒を指定します。

説明

オブジェクトを構築，消滅します。
構築時，データメンバに指定された値を設定します。

参照

`setMilliSecond setTime`

CDAQDateTime::getMilliSecond

構文

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

説明

データメンバのミリ秒領域の値を取得します。

戻り値

ミリ秒を返します。

CDAQDateTime::getTime

構文

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

説明

データメンバの秒数領域の値を取得します。

戻り値

秒数を返します。

CDAQDateTime::initialize

構文

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

説明

データメンバを初期化します。初期値は，0です。

参照

`setMillioSecond setTime`

CDAQDateTime::isObject

構文

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

引数

classname クラス名を文字列で指定します。

説明

指定されたクラス名を継承しているかをチェックします。
省略された場合，本クラスであるかをチェックします。
本クラスを継承したクラスが，自分自身のクラスをチェックするためには，オーバーライドする必要があります。
クラスを継承している場合，真(有効)を返します。それ以外は，偽(無効)を返します。

戻り値

真偽を1(真)，または，0(偽)で返します。

CDAQDateTime::operator=

構文

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

引数

cDateTime 代入するオブジェクトを指定します。

説明

指定されたオブジェクトのデータメンバを複写します。

戻り値

本オブジェクトへの参照を返します。

CDAQDateTime::setMilliSecond

構文

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

引数

milliSecond ミリ秒を指定します。

説明

データメンバのミリ秒領域に指定された値を格納します。

CDAQDateTime::setNow

構文

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

説明

現在の日付時刻を取得して，データメンバに格納します。
ミリ秒は，0になります。

参照

```
setMilliSecond setTime
```

CDAQDateTime::setTime

構文

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

引数

time　　秒数を指定します。

説明

データメンバの秒数領域に指定された値を格納します。

CDAQDateTime::toLocalDateTime

構文

```
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);
```

引数

sectime	秒数を指定します。
pYear	年の値の返却先を指定します。
pMonth	月の値の返却先を指定します。
pDay	日の値の返却先を指定します。
pHour	時の値の返却先を指定します。
pMinute	分の値の返却先を指定します。
pSecond	秒の値の返却先を指定します。

説明

指定された秒数をタイムゾーンに従った年月日時分秒の値に変換します。
指定する秒数は，1970年01月01日からの秒数です。 秒数の指定がない構文は，データメンバから秒数を取得します。
年には4桁の数値を返します。
月には1から12を返します。
日には1から31を返します。
時には0から23を返します。
分には0から59を返します。
秒には0から59を返します。
変換に失敗した場合は，0を返します。

参照

`getTime`

CDAQHandlerクラス

本クラスは，ハンドラの基底クラスです。通信機能を提供します。
通信方式はTCP/IPです。通信記述子を使い通信を制御しています。通信記述子は，汎用型へのポインタでデータメンバの通信記述子領域に格納しています。通信記述子は，通信接続で構築し，通信切断で消滅させています。
通信方式を変更する場合，継承クラスを作成し，通信機能の関数メンバを全てオーバーライドします。
データ取得機能は，機種を意識せずに呼び出せるように定義されています。ただし，本クラスでは定義のみで，実装されていません。継承クラスで，オーバーライドして実装する必要があります。
インスタンスが本クラスを継承しているかをチェックする機能を提供します。継承クラスでオーバーライドすることができます。

パブリックメンバ

構築・消滅

CDAQHandler	オブジェクトを構築します。
~CDAQHandler	オブジェクトを消滅します。

通信機能

open	通信接続をします。
close	通信切断をします。
sendLine	文字列データを送信します。
receiveLine	文字列データを行単位で受信します。
setTimeOut	通信タイムアウトを設定します(通信タイムアウトの設定を推奨しません(2.2節を参照))。

データ取得機能

getData	測定データを取得します。
getChannel	チャネル情報データを取得します。

ユーティリティ

getVersionAPI	本APIのバージョンを取得します。
getRevisionAPI	本APIのリビジョンを取得します。
getErrorMessage	エラーメッセージ文字列を取得します。
getMaxLenErrorMessage	エラーメッセージ文字列の最大長を取得します。
isObject	オブジェクトをチェックします。

プロテクトメンバ

データメンバ

m_comm	通信記述子の格納領域です。
m_nRemainSize	受信データの残りサイズの格納領域です。

通信機能

send	データを送信します。
receive	データを受信します。

ユーティリティ

receiveRemain	残りバイトを受信して破棄します。
getVersionDLL	本DLLのバージョンを取得します。
getRevisionDLL	本DLLのリビジョンを取得します。

プライベートメンバ

なし

関数メンバ(アルファベット順)

CDAQHandler::CDAQHandler

構文

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

引数

strAddress	IPアドレスを文字列で指定します。
uiPort	ポート番号を指定します。
errCode	エラー番号の返却先を指定します。

説明

オブジェクトを構築，消滅します。

構築時，データメンバを初期化します。初期値は，原則0(NULL)です。引数が指定されている場合，構築時に通信接続(open)を行います。返却先が指定されていれば，通信接続時のエラー番号を返します。

消滅時，データメンバの領域を開放します。通信記述子が存在する場合，通信切断(close)を行います。エラー番号は返却されません。

参照

close open

CDAQHandler::close

構文

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

説明

通信切断します。通信記述子を消滅します。

戻り値

エラー番号を返します。

エラー：

Not connected	通信接続されていません。
通信エラー	通信中にエラーを検出しました。

CDAQHandler::getChannel

構文

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

引数

chType	チャネルタイプを指定します。
chNo	チャネル番号を指定します。
cChInfo	チャネル情報データの返却先を指定します。

説明

チャネル単位で，チャネル情報データを取得するための関数です。

機種別の継承クラスで，オーバライドする必要があります。オーバライドしないと，エラー番号を返します。

戻り値

エラー番号を返します。

エラー：

Not Support	サポートしていません。

CDAQHandler::getData

構文

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

引数

chType	チャネルタイプを指定します。
chNo	チャネル番号を指定します。
cDateTime	時刻情報データの返却先を指定します。
cDataInfo	測定データの返却先を指定します。

説明

チャネル単位で，瞬時値を取得するための関数です。
機種別の継承クラスで，オーバライドする必要があります。オーバライドしないと，エラー番号を返します。

戻り値

エラー番号を返します。
エラー：

Not Support	サポートしていません。

CDAQHandler::getErrorMessage

構文

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

引数

errCode	エラー番号を指定します。

説明

引数で指定されたエラー番号に対応するエラーメッセージ文字列を取得します。
値が範囲外の場合，メッセージ文字列は，「Unknown」になります。

戻り値

エラーメッセージ文字列へのポインタを返します。

CDAQHandler::getMaxLenErrorMessage

構文

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

説明

エラーメッセージ文字列の最大長を取得します。
値はバイト数です。
戻り値に，終端は含まれません。

戻り値

文字列の長さを返します。

CDAQHandler::getRevisionAPI

構文

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

説明

本APIのリビジョン番号を取得します。

戻り値

リビジョン番号を返します。

CDAQHandler::getRevisionDLL

構文

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

説明

本DLLのリビジョン番号を取得します。

戻り値

リビジョン番号を返します。

CDAQHandler::getVersionAPI

構文

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

説明

本APIのバージョン番号を取得します。

戻り値

バージョン番号を返します。

CDAQHandler::getVersionDLL

構文

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

説明

本DLLのバージョン番号を取得します。

戻り値

バージョン番号を返します。

CDAQHandler::isObject

構文

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

引数

classname　　クラス名を文字列で指定します。

説明

指定されたクラス名を継承しているかをチェックします。
省略された場合，本クラスであるかをチェックします。
本クラスを継承したクラスが，自分自身のクラスをチェックするためには，オーバーライドする必要があります。
クラスを継承している場合，真(有効)を返します。それ以外は，偽(無効)を返します。

戻り値

真偽を1(真)，または，0(偽)で返します。

CDAQHandler::open

構文

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

引数

strAddress　　IPアドレスを文字列で指定します。
uiPort　　ポート番号を指定します。

説明

引数で指定されたIPアドレスとポート番号の機器と通信接続をします。
通信記述子を構築し，データメンバの通信記述子領域に格納します。
既に通信記述子が存在する場合，通信記述子を変更しません。

戻り値

エラー番号を返します。
エラー：

Creating connection is failure	通信記述子の構築に失敗しました。
Connection exists already	既に通信記述子が存在します。
通信エラー	通信中にエラーを検出しました。

CDAQHandler::receive

構文

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

引数

bufData	受信データを格納する領域をバイト配列で指定します。
maxData	受信データのバイト数を指定します。
lenData	実際に受信したデータのバイト数の返却先を指定します。

説明

引数で指定された領域に，バイト数分になるまで受信データを格納します。
返却先が指定されていれば，実際に受信したデータのバイト数を返します。

戻り値

エラー番号を返します。
エラー：

Not connected	通信接続されていません。
通信エラー	通信中にエラーを検出しました。

CDAQHandler::receiveLine

構文

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

引数

strLine	受信文字列を格納する領域を指定します。
maxLine	受信文字列を格納する領域のバイト数を指定します。
lenLine	実際に受信した文字列のバイト数の返却先を指定します。

説明

引数で指定された領域に，改行を検出するまで，または，バイト数分になるまで受信文字列を格納します。改行を除いた受信文字列を格納します。
返却先が指定されていれば，実際に受信し格納した文字列のバイト数を返します。

戻り値

エラー番号を返します。
エラー：

Not connected	通信接続されていません。
通信エラー	通信中にエラーを検出しました。

CDAQHandler::receiveRemain

構文

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

説明

残りサイズ領域が0より大きい場合，残りサイズ分のデータを受信して破棄します。
残りサイズ領域を0にします。
残りサイズ領域が0以下の場合，何もせず，正常終了します。

戻り値

エラー番号を返します。

参照

receive

CDAQHandler::send

構文

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

引数

bufData	送信データをバイト配列で指定します。
lenData	送信データのバイト数を指定します。

説明

引数で指定された送信データをバイト数分だけ送信します。

戻り値

エラー番号を返します。
エラー：

Not connected	通信接続されていません。
通信エラー	通信中にエラーを検出しました。

CDAQHandler::sendLine

構文

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

引数

strLine　　送信文字列を指定します。

説明

指定された文字列を送信します。
改行を付加して送信します。改行を除いた文字列を指定します。

戻り値

エラー番号を返します。
エラー：
Not connected　通信接続されていません。
通信エラー　　通信中にエラーを検出しました。

CDAQHandler::setTimeOut

構文

```
virtual int setTimeOut(int seconds);
```

引数

seconds　　通信のタイムアウト値を秒単位で指定します。

説明

引数で指定された値を送信，受信，両方のタイムアウトに設定します。
指定された値が負の場合，タイムアウトを無効にします。
使用を推奨しません。

戻り値

エラー番号を返します。
エラー：
Not connected　通信接続されていません。
通信エラー　　通信中にエラーを検出しました。

2.5　MX100用クラス詳細

クラスは，クラス名のアルファベット順で並んでいます。

CDAQMXクラス

■ CDAQHandler
　・ CDAQMX

本クラスは，CDAQHandlerクラスの派生クラスです。
本クラスは，MX100用の機能を提供します。
通信はバイナリですので，基底クラスの直接的な通信機能は使用しないでください。
通信に必要な情報を内部に格納しています。コマンド番号，ユーザカウント，通信パケットバージョンから送信パケットを内部で生成しています。
データを取得する場合，取得する宣言のメンバを実行してから，データごとにデータ個数分を取得するメンバを実行します。ここで，宣言の有無をチェックするようになりました。
インスタンスが本クラスを継承しているかをチェックする機能を提供します。継承クラスでオーバーライドすることができます。

パブリックメンバ

構築・消滅

CDAQMX	オブジェクトを構築します。
~CDAQMX	オブジェクトを消滅します。

制御機能

FIFOの開始/停止

startFIFO	FIFOを開始します。
stopFIFO	FIFOを停止します。
autoFIFO	FIFOの自動制御を設定します。

そのほかの制御

setDateTime	時刻情報データを設定します。
setBackup	CFカードへのバックアップを設定します。
formatCF	CFカードをフォーマットします。
initSystem	システムを初期化します。
setSegment	7セグメントLEDを設定します。
setDOData	DOデータを設定します。
setAOPWMData	AO/PWMデータを設定します。
setTransmit	伝送出力データを設定します。

設定機能

一括設定

setConfig	設定データを設定します。
setMXConfig	基本設定を設定します。
setOutput	基本設定出力チャネルデータを設定します。
setBalance	初期バランスデータを設定します。
runBalance	初期バランスを実行します。
resetBalance	初期バランス値を初期化します。

データ取得機能

システムステータスデータ/システム構成データの取得

getStatusData	ステータスを取得します。
getSystemConfig	システム構成データを取得します。

設定データの取得

getConfig	設定データを取得します。
talkConfig	設定データを取得する宣言をします。
getChConfig	チャネル設定データを取得します。
getMXConfig	基本設定を取得します。
getOutput	出力チャネルデータを取得します。
getBalance	初期バランスデータを取得します。

出力データの取得

getDOData	DOデータを取得します。
getAOPWMData	AO/PWMデータと伝送出力データを取得します。

チャネル情報データの取得

talkChInfo	チャネル情報データを取得する宣言をします。
getChInfo	チャネル情報データを取得します。

測定データの取得

getChDataNo	チャネルのデータ番号を取得します。
talkChData	測定データをチャネル指定で取得する宣言をします。
getTimeData	測定データの時刻情報データを取得します。
getFIFODataNo	FIFOのデータ番号を取得します。
talkFIFOData	測定データをFIFO指定で取得する宣言をします。
getChData	測定データを取得します。
getTimeData	測定データの時刻情報データを取得します。

ユーティリティ

getUserTime	ユーザーカウントを取得します。
setUserTime	ユーザーカウントを設定します。
getLastError	MX100固有エラーを取得します。
getItemError	設定項目番号を取得します。

●オーバライドしたメンバ

通信機能

open	通信接続をします。

データ取得機能

getData	測定データを取得します。
getChannel	チャネル情報データを取得します。

ユーティリティ

isObject	オブジェクトをチェックします。

●継承するメンバ

CDAQHandler参照

```
close getErrorMessage getMaxLenErrorMessage getRevisionAPI
getVersionAPI receiveLine sendLine setTimeOut
```

プロテクトメンバ

データメンバ

m_nNo	コマンド番号の格納領域です。
m_nLastError	MX100固有エラーの格納領域です。
m_bAutoFIFO	FIFO自動制御の格納領域です。
m_llUserTime	ユーザーカウントの格納領域です。
m_nSessionNo	セッション番号の格納領域です。
m_chFIFONo	チャネル毎FIFO番号の格納領域です。
m_chFIFOIndex	チャネル毎FIFO内チャネル順序番号の格納領域です。
m_chDataType	チャネル毎データ種類の格納領域です。
m_chDeciPos	チャネル毎小数点位置の格納領域です。
m_lastFIFODataNo	FIFO毎最終データ番号の格納領域です。
m_lastChDataNo	チャネル毎最終データ番号の格納領域です。
m_startChNo	開始チャネル番号の格納領域です。
m_endChNo	終了チャネル番号の格納領域です。
m_curChNo	現在チャネル番号の格納領域です。
m_startFIFOIdx	FIFO内開始チャネル順序番号の格納領域です。
m_endFIFOIdx	FIFO内終了チャネル順序番号の格納領域です。
m_curFIFOIdx	FIFO内現在チャネル順序番号の格納領域です。
m_startDataNo	開始データ番号の格納領域です。
m_endDataNo	終了データ番号の格納領域です。
m_curDataNo	現在データ番号の格納領域です。
m_nFIFONo	FIFO番号の格納領域です。
m_nDataNum	データ数の格納領域です。
m_nChNum	チャネル数の格納領域です。
m_nTimeNum	測定データの時刻情報データの残り個数の格納領域です。

m_packetVer	通信パケットバージョンの格納領域です。
m_nItemError	設定項目番号の格納領域です。
m_bTalkConfig	設定データを取得する宣言のフラグです。
m_bTalkChInfo	チャネル情報データを取得する宣言のフラグです。
m_bTalkData	測定データを取得する宣言のフラグです。

通信機能

runCommand	コマンドを実行します。
sendPacket	パケットを送信します。
receivePacket	パケットを受信します。
receiveBlock	ブロックを受信します。
runPacket	パケットを送受信します。
receiveBuffer	サイズ情報を含むデータをサイズ分受信します。

内部コマンド

nop	NOPコマンドを実行します。
registry	レジストリコマンドを実行します。

データメンバ操作

getNo	コマンド番号を取得します。
incCurDataNo	現在データ番号をインクリメントします。
incCurFIFOIdx	現在FIFO内チャネル順序番号をインクリメントします。
getDataNo	データ番号を取得します。
searchChNo	FIFO内チャネル順序番号からチャネル番号を取得します。
clearAttr	データメンバを初期化します。
clearData	測定データの取得に関するデータメンバを初期化します。
getPacketVersion	通信パケットバージョンを取得します。
clearLastDataNoCh	チャネル毎最終データ番号を初期化します。
clearLastDataNoFIFO	FIFO毎最終データ番号を初期化します。

ユーティリティ

getVersionDLL	本DLLのバージョンを取得します。
getRevisionDLL	本DLLのリビジョンを取得します。

●継承するメンバ

CDAQHandler参照

```
m_comm m_nRemainSize
receive receiveRemain send
```

プライベートメンバ

なし。

関数メンバ(アルファベット順)

CDAQMX::autoFIFO

構文

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

引数

bAuto　　自動制御を有効無効値で指定します。

説明

自動制御を設定します。
データメンバの自動制御領域に指定された値を格納します。
「有効」を指定した場合，FIFOを開始します。

戻り値

エラー番号を返します。

参照

`startFIFO`

CDAQMX::CDAQMX

構文

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

引数

strAddress　　IPアドレスを文字列で指定します。
uiPort　　ポート番号を指定します。
errCode　　エラー番号の返却先を指定します。

説明

オブジェクトを構築，消滅します。
構築時，データメンバを初期化します。初期値は，原則0(NULL)です。引数が指定されている場合，構築時に通信接続(open)を行います。返却先が指定されていれば，通信接続時のエラー番号を返します。
消滅時，データメンバの領域を開放します。通信記述子が存在する場合，通信切断(close)を行います。エラー番号は返却されません。

参照

`clearAttr close open`
`CDAQHandler::CDAQHandler`

CDAQMX::clearAttr

構文

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

説明

データメンバを全て初期化します。初期値は，原則0です。

参照

```
clearData
```

CDAQMX::clearData

構文

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

引数

sessionNo　　セッション番号を指定します。

説明

測定データの取得開始のためのデータメンバを初期化します。
データメンバのセッション番号領域に指定された値を格納します。

参照

```
clearLastDataNoCh clearLastDataNoFIFO
```

CDAQMX::clearLastDataNoCh

構文

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

引数

chNo　　チャネル番号を指定します。

説明

データメンバの指定されたチャネル番号のチャネル毎最終データ番号領域を初期化します。
チャネル番号に，定数値の「全チャネル番号指定」をすると，全チャネルを処理します。

CDAQMX::clearLastDataNoFIFO

構文

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

引数

fifoNo　　FIFO番号を指定します。

説明

データメンバの指定されたFIFO番号のFIFO毎最終データ番号領域を初期化します。
FIFO番号に，定数値の「全FIFO番号指定」をすると，全FIFOを処理します。

CDAQMX::formatCF

構文

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

説明

CFカードをフォーマットします。
FIFOは停止します。 自動制御が有効なら，正常終了時，開始します。

戻り値

エラー番号を返します。

参照

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

CDAQMX::getAOPWMData

構文

```
int getAOPWMData(CDAQMXAOPWMData & cMXAOPWMData,
CDAQMXTransmit & cMXTransmit);
```

引数

cMXAOPWMData	AO/PWMデータの返却先を指定します。
cMXTransmit	伝送出力データの返却先を指定します。

説明

AO/PWMデータと伝送出力データを取得します。

戻り値

エラー番号を返します。

参照

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

CDAQMX::getBalance

構文

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

引数

cMXBalanceData　　初期バランスデータの返却先を指定します。

説明

初期バランスデータを取得します。
設定データを取得して，初期バランスデータの部分を指定された返却先に格納します。

戻り値

エラー番号を返します。

参照

`getMXConfig CDAQMXBalanceData::initialize CDAQMXConfig::getClassMXBalanceData`

CDAQMX::getChannel

構文

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

引数

chType	チャネルタイプを指定します。
chNo	チャネル番号を指定します。
cChInfo	チャネル情報データの返却先を指定します。

説明

チャネル単位で，チャネル情報データを取得するための関数です。
指定されたチャネルのチャネル情報データを取得します。
チャネルタイプは無視されます。

戻り値

エラー番号を返します。

参照

`getChInfo talkChInfo`

CDAQMX::getChConfig

構文

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

引数

cMXChConfig　チャネル設定データの返却先を指定します。
pFlag　フラグの返却先を指定します。

説明

設定データを取得する宣言(talkConfig)による出力をチャネル単位で取得します。
情報を解析して，返却先に格納します。
最終データを取得した場合， フラグにフラグステータスがセットされます。 また，エラーで終了した場合もセットします。
データ取得を終了するまでは，他関数で通信を行わないでください。
本体のスタイルナンバーでパケットが異なります。

戻り値

エラー番号を返します。
エラー：
Not support　サポートしていないバージョンです。 または，実行順序が間違っています。

参照

`getPacketVersion receiveBlock CDAQMXChConfig::setChNo`

CDAQMX::getChData

構文

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

引数

dataNo	データ番号の返却先を指定します。
cMXDataInfo	測定データの返却先を指定します。
pFlag	フラグの返却先を指定します。

説明

測定データを取得する宣言(talkChData，talkFIFOData)による出力の測定データをチャネル単位で取得します。
情報を解析して，返却先に格納します。
測定データの返却先にチャネル情報データへの関連が存在する場合，測定データを識別するチャネル情報データを格納します。
最終データを取得した場合，フラグにフラグステータスがセットされます。また，エラーで終了した場合もセットします。
データ取得を終了するまでは，他関数で通信を行わないでください。

戻り値

エラー番号を返します。

参照

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

CDAQMX::getChDataNo

構文

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

引数

chNo	チャネル番号を指定します。
startDataNo	開始データ番号の返却先を指定します。
endDataNo	終了データ番号の返却先を指定します。

説明

取得可能な測定データのデータ番号を取得します。
チャネル指定で最後に取得した測定データの次のデータから開始して，取得できるデータ範囲を取得します。

戻り値

エラー番号を返します。
エラー：

Not support	チャネル番号が範囲外です。

参照

`getDataNo`

CDAQMX::getChInfo

構文

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

引数

cMXChInfo　　チャネル情報データの返却先を指定します。
pFlag　　フラグの返却先を指定します。

説明

チャネル情報データを取得する宣言(talkChInfo)による出力をチャネル単位で取得します。
情報を解析して，返却先に格納します。
最終データを取得した場合，フラグにフラグステータスがセットされます。また，エラーで終了した場合もセットします。
データ取得を終了するまでは，他関数で通信を行わないでください。
データメンバの各種チャネル情報の格納領域に必要な情報を格納します。

戻り値

エラー番号を返します。
エラー：
Not support　　実行順序が間違っています。

参照

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

CDAQMX::getConfig

構文

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

引数

cMXConfig　　設定データの返却先を指定します。

説明

設定データを取得します。
システム構成データ，ステータスデータ，基本設定の取得を実行して，情報をマージします。

戻り値

エラー番号を返します。

参照

getMXConfig getStatusData getSystemConfig

CDAQMX::getData

構文

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

引数

chType	チャネルタイプを指定します。
chNo	チャネル番号を指定します。
cDateTime	時刻情報データの返却先を指定します。
cDataInfo	測定データの返却先を指定します。

説明

チャネル単位で，瞬時値を取得するための関数です。
指定されたチャネルの測定データを取得します。
チャネルタイプは無視されます。

戻り値

エラー番号を返します。

参照

`getChData getTimeData talkChData`

CDAQMX::getDataNo

構文

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

引数

fifoNo	FIFO番号を指定します。
prevLast	最後に取得したデータ番号を指定します。
startDataNo	開始データ番号の返却先を指定します。
endDataNo	終了データ番号の返却先を指定します。

説明

ステータスデータを取得し，測定データの取得できる範囲を算出します。
取得できる測定データが存在しない場合，負の数を返却先に返します。

戻り値

エラー番号を返します。
エラー：

Notsupport	FIFO番号が範囲外です。

参照

`getStatusData`
`CDAQMXStatus::getNewDataNo CDAQMXStatus::getOldDataNo`

CDAQMX::getDOData

構文

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

引数

cMXDoData　DOデータの返却先を指定します。

説明

DOデータを取得します。
全チャネル分一括取得します。

戻り値

エラー番号を返します。

参照

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

CDAQMX::getFIFODataNo

構文

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

引数

fifoNo　FIFO番号を指定します。
startDataNo　開始データ番号の返却先を指定します。
endDataNo　終了データ番号の返却先を指定します。

説明

取得可能な測定データのデータ番号を取得します。
FIFO指定で最後に取得した測定データの次のデータから開始して，取得できるデータ範囲を取得します。

戻り値

エラー番号を返します。
エラー：
Not support　FIFO番号が範囲外です。

参照

```
getDataNo
```

CDAQMX::getItemError

構文

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

説明

データメンバから， 設定項目番号領域の値を取得します。

戻り値

設定項目番号を返します。

CDAQMX::getLastError

構文

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

説明

データメンバからMX100固有エラー領域の値を取得します。

戻り値

MX100固有エラーを返します。

CDAQMX::getMXConfig

構文

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

引数

cMXConfig　　設定データの返却先を指定します。

説明

基本設定を取得します。
MX100本体のスタイルナンバーでパケットが異なります。

戻り値

エラー番号を返します。
エラー：
Not support　サポートしていないバージョンです。

参照

```
getNo getPacketVersion getUserTime runCommand
```

CDAQMX::getNo

構文

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

説明

データメンバからコマンド番号領域の値を取得します。
コマンド番号をインクリメントします。

戻り値

コマンド番号を返します。

CDAQMX::getOutput

構文

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

引数

cMXOutputData　　出力チャネルデータの返却先を指定します。

説明

出力チャネルデータを取得します。
設定データを取得して， 出力チャネルデータの部分を指定された返却先に格納します。

戻り値

エラー番号を返します。

参照

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

CDAQMX::getPacketVersion

構文

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

説明

データメンバから， 通信パケットバージョン領域の値を取得します。

戻り値

通信パケットバージョンを返します。

CDAQMX::getRevisionDLL

構文

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

説明

本DLLのリビジョン番号を取得します。

戻り値

本DLLのリビジョン番号を返します。

CDAQMX::getStatusData

構文

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

引数

cMXStatus　　ステータスデータの返却先を指定します。

説明

ステータスデータを取得します。
MX100本体のスタイルナンバーでパケットが異なります。

戻り値

エラー番号を返します。
エラー：
Not support　　サポートしていないバージョンです。

参照

```
getNo getPacketVersion getUserTime runCommand
```

CDAQMX::getSystemConfig

構文

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

引数

cMXSysInfo　　システム構成データの返却先を指定します。

説明

システム構成データを取得します。

戻り値

エラー番号を返します。

参照

```
getNo getPacketVersion getUserTime runCommand
```

CDAQMX::getTimeData

構文

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

引数

dataNo	データ番号の返却先を指定します。
cMXDateTime	時刻情報データの返却先を指定します。
pUserTime	ユーザーカウントの返却先を指定します。
pFlag	フラグの返却先を指定します。

説明

測定データを取得する宣言(talkChData，talkFIFOData)による出力の時刻情報データをデータ番号単位で取得します。情報を解析して，返却先に格納します。
最終データを取得した場合，フラグにフラグステータスがセットされます。また，エラーで終了した場合もセットします。
データ取得を終了するまでは，他関数で通信を行わないでください。
本関数メンバでデータ取得を終了後，データ番号内のチャネル毎にgetChDataを使用して測定データを取得します。

戻り値

エラー番号を返します。
エラー：

Not support	実行順序が間違っています。

参照

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

CDAQMX::getUserTime

構文

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

説明

データメンバからユーザーカウント領域の値を取得します。

戻り値

ユーザーカウントを返します。

CDAQMX::getVersionDLL

構文

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

説明

本DLLのバージョン番号を取得します。

戻り値

本DLLのバージョン番号を返します。

CDAQMX::incCurDataNo

構文

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

説明

データメンバから現在データ番号領域の値を取得します。
現在データ番号をインクリメントします。終了データ番号を超えたら開始データ番号にリセットします。

戻り値

現在データ番号を返します。

CDAQMX::incCurFIFOIdx

構文

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

説明

データメンバから現在FIFO内チャネル順序番号領域の値を取得します。
現在FIFO内チャネル順序番号をインクリメントします。終了FIFO内チャネル順序番号を越えたら開始FIFO内チャネル順序番号にリセットします。

戻り値

現在FIFO内チャネル順序番号を返します。

CDAQMX::initSystem

構文

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

引数

iCtrl　　システム制御種類を指定します。

説明

指定されたシステム制御種類の動作を実行します。
FIFOは停止します。 自動制御が有効なら，正常終了時，開始します。
ただし，アラームリセット時は除きます。

戻り値

エラー番号を返します。

参照

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

CDAQMX::isObject

構文

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

引数

classname　　クラス名を文字列で指定します。

説明

指定されたクラス名を継承しているかをチェックします。
省略された場合，本クラスであるかをチェックします。
本クラスを継承したクラスが，自分自身のクラスをチェックするためには，オーバーライドする必要があります。
クラスを継承している場合，「有効値」(真)を返します。それ以外は，「無効値」(偽)を返します。
本クラスと異なる場合，親クラスをチェックします。

戻り値

有効無効値を返します。

参照

`CDAQHandler::isObject`

CDAQMX::nop

構文

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

説明

コマンドを実行します。
MX100が応答を返すだけのコマンドです。

戻り値

エラー番号を返します。

参照

`getNo getPacketVersion getUserTime runCommand`

CDAQMX::open

構文

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

引数

strAddress　IPアドレスを文字列で指定します。
uiPort　ポート番号を指定します。

説明

引数で指定されたIPアドレスとポート番号の機器と通信接続をします。
ポート番号は省略可能で，省略時は，「MX100の通信ポート番号」になります。
データメンバを初期化します。
レジストリコマンドを実行します。実行に失敗した場合，接続切断します。

戻り値

エラー番号を返します。

参照

`clearAttr close registry`
`CDAQHandler::open`

CDAQMX::receiveBlock

構文

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

引数

pBlock　ブロックを格納する領域をバイト配列で指定します。
lenBlock　ブロックのバイト数を指定します。

説明

指定されたブロックを受信します。
残りサイズ領域を更新します。

戻り値

エラー番号を返します。
エラー：
Not data　領域サイズが合致しません。

参照

`receive`

CDAQMX::receiveBuffer

構文

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

引数

pBuf	データを格納する領域をバイト配列で指定します。
lenBuf	データ領域のバイト数を指定します。
realLen	実際に受信したデータのバイト数の返却先を指定します。
sizeBuf	サイズ情報の返却先を指定します。

説明

サイズ情報を含むデータを受信します。
引数で指定された領域に， 受信したデータのサイズ情報分， または，指定されたバイト数分のデータを受信し格納します。
返却先が指定されていれば， 実際に受信したデータのバイト数やサイズ情報を返します。

戻り値

エラー番号を返します。
エラー：

Not acknowledge	応答パケットのサイズがおかしい。

参照

`receiveRemain receive`

CDAQMX::receivePacket

構文

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

引数

ackBuf	応答パケットを格納する領域をバイト配列で指定します。
lenAck	応答パケットのバイト数を指定します。
realLen	実際に受信したバイト数の返却先を指定します。

説明

指定されたパケットを受信してデコードします。
受信したパケットが，エラーパケットの場合，MX100固有エラー領域にMX100固有エラーを格納します。

戻り値

エラー番号を返します。
エラー：

Commands are not processed successfully	エラーパケットを受信しました。

参照

`receiveBuffer`

CDAQMX::registry

構文

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

説明

コマンドを実行します。
MX100本体にPCの情報(ホスト名，アドレス)を伝えるコマンドです。
通信パケットバージョンをデータメンバの通信パケットバージョン領域に格納します。

戻り値

エラー番号を返します。

参照

```
getNo getPacketVersion getUserTime runCommand
```

CDAQMX::resetBalance

構文

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

引数

cMXBalanceResult　初期バランス結果の返却先を指定します。

説明

初期バランスをリセットします。
リセットした結果， 初期バランス結果と初期バランス値を指定された返却先に格納します。
有効／無効が「有効」に指定されたチャネルのみをリセットします。
FIFOは停止します。 自動制御が有効なら，正常終了時，開始します。
本関数は，応答に5秒以上の時間がかかることがあります。

戻り値

エラー番号を返します。

参照

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

CDAQMX::runBalance

構文

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

引数

cMXBalanceResult　初期バランス結果の返却先を指定します。

説明

初期バランスを実行します。
実行した結果，初期バランス結果と初期バランス値を指定された返却先に格納します。
有効／無効が「有効」に指定されたチャネルのみを実行します。
FIFOは停止します。自動制御が有効なら，正常終了時，開始します。
本関数は，応答に5秒以上の時間がかかることがあります。

戻り値

エラー番号を返します。

参照

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

CDAQMX::runCommand

構文

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

引数

reqBuf	要求パケットをバイト配列で指定します。
lenReq	要求パケットのバイト数を指定します。
ackBuf	応答パケットを格納する領域をバイト配列で指定します。
lenAck	応答パケットのバイト数を指定します。

説明

指定された要求パケットを送信し，応答パケットを受信します。応答パケットを指定された領域に格納します。

戻り値

エラー番号を返します。
エラー：

Not acknowledge	要求に対応する応答ではありません。
Not support	要求パケットが間違っています。
Not data	応答パケットの領域が不足しています。

参照

```
receivePacket sendPacket
```

CDAQMX::runPacket

構文

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

引数

reqBuf	要求パケットをバイト配列で指定します。
lenReq	要求パケットのバイト数を指定します。
ackBuf	応答パケットを格納する領域をバイト配列で指定します。
lenAck	応答パケットのバイト数を指定します。

説明

指定された要求パケットを送信し，応答パケットを受信します。
応答パケットは指定された領域に格納します。
非標準のパケットのために用意されたメンバです。 パケットの処理をユーザ側で行う必要があります。
使用を推奨しません。

戻り値

エラー番号を返します。

参照

`receiveBuffer send`

CDAQMX::searchChNo

構文

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

引数

fifoNo	FIFO番号を指定します。
fifoIndex	FIFO内チャネル順序番号を指定します。

説明

データメンバに格納されている情報から指定された値に対応するチャネル番号を探します。
存在しない場合，0を返します。

戻り値

チャネル番号を返します。

CDAQMX::sendPacket

構文

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

引数

reqBuf	要求パケットをバイト配列で指定します。
lenReq	要求パケットのバイト数を指定します。

説明

指定されたパケットをエンコードして送信します。

戻り値

エラー番号を返します。

参照

`send`

CDAQMX::setAOPWMData

構文

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

引数

cMXAOPWMData AO/PWMデータを指定します。

説明

AO/PWMデータを設定します。

戻り値

エラー番号を返します。

参照

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

CDAQMX::setBalance

構文

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

引数

cMXBalanceData　　初期バランスデータを指定します。

説明

初期バランスデータを設定します。
設定データを取得して， 指定された初期バランスデータで更新して， 設定データを送信します。
有効／無効が「有効」に指定されたチャネルのみを更新します。

戻り値

エラー番号を返します。

参照

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

CDAQMX::setBackup

構文

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

引数

bBackup　　バックアップを有効無効値で指定します。

説明

機器本体にバックアップを設定します。

戻り値

エラー番号を返します。

参照

```
getNo getPacketVersion getUserTime runCommand
```

CDAQMX::setBalance

構文

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

引数

cMXBalanceData 初期バランスデータを指定します。

説明

初期バランスデータを設定します。
設定データを取得して， 指定された初期バランスデータで更新して， 設定データを送信します。
有効/無効が「有効」に指定されたチャネルのみを更新します。

戻り値

エラー番号を返します。

参照

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

CDAQMX::setConfig

構文

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

引数

cMXConfig 設定データを指定します。

説明

設定データを設定します。
基本設定を設定します。

戻り値

エラー番号を返します。

参照

```
setMXConfig
```

CDAQMX::setDateTime

構文

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

引数

pcMXDateTime 時刻情報データを指定します。

説明

機器本体に時刻情報データを設定します。
引数の時刻情報データを省略すると，PCの現在の日付時刻を設定します。
ミリ秒は無視されます。
FIFOは停止します。自動制御が有効なら，正常終了時，開始します。
本関数は，応答に1秒以上の時間がかかることがあります。
時刻に負の値を指定するとエラーになります。

戻り値

エラー番号を返します。
エラー：
Not data 指定された値が不正です。

参照

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

CDAQMX::setDOData

構文

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

引数

cMXDoData DOデータを指定します。

説明

DOデータを設定します。
全チャネル分一括設定します。

戻り値

エラー番号を返します。

参照

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

CDAQMX::setMXConfig

構文

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

引数

cMXConfig　　設定データを指定します。

説明

機器本体に基本設定を設定します。
送信前に入力データの妥当性検証をします。
FIFOは停止します。 自動制御が有効なら，正常終了時，開始します。
本体のスタイルナンバーでパケットが異なります。
妥当性検証による結果の設定項目番号をデータメンバの設定項目番号領域に格納します。

戻り値

エラー番号を返します。
エラー：
Not data　　入力データが妥当ではありません。
Not support　　サポートしていないバージョンです。

参照

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

CDAQMX::setOutput

構文

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

引数

cMXOutputData　　出力チャネルデータを指定します。

説明

出力チャネルデータを設定します。
設定データを取得して， 指定された出力チャネルデータで更新して， 設定データを送信します。
出力種類を変更すると， チャネル設定データと一致しなくなります。

戻り値

エラー番号を返します。

参照

`getMXConfig setMXConfig CDAQMXConfig::getClassMXOutputData`

CDAQMX::setSegment

構文

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

引数

dispType	表示形式を指定します。
dispTime	表示時間を指定します。
cNewMXSegment	表示パターンを指定します。
cOldMXSegment	以前の表示パターンの返却先を指定します。

説明

7セグメントLEDの表示を設定します。
返却先が指定されていれば，変更前の7セグメントLEDの表示パターンを格納します。

戻り値

エラー番号を返します。

参照

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

CDAQMX::setTransmit

構文

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

引数

cMXTransmit	伝送出力データを指定します。

説明

伝送出力データを設定します。

戻り値

エラー番号を返します。
エラー：

Not support	指定された値が範囲外です。

参照

`getNo getPacketVersion getUserTime runCommand CDAQMXTransmit::getTransmit`

CDAQMX::setUserTime

構文

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

引数

userTime　　ユーザーカウントを指定します。

説明

データメンバのユーザーカウント領域に指定された値を格納します。

CDAQMX::startFIFO

構文

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

説明

FIFO を開始します。
チャネル情報データを取得して，測定データの取得に関するデータメンバに必要な情報を設定します。

戻り値

エラー番号を返します。

参照

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

CDAQMX::stopFIFO

構文

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

説明

FIFOを停止します。
自動制御が有効でも停止します。

戻り値

エラー番号を返します。

参照

```
getNo getPacketVersion getUserTime runCommand
```

CDAQMX::talkChData

構文

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

引数

chNo	チャネル番号を指定します。
startDataNo	開始データ番号を指定します。
endDataNo	終了データ番号を指定します。

説明

測定データを取得する宣言を実行します。
指定されたデータ番号の範囲の測定データを取得します。実際に取得される範囲は指定と等しいとは限りません。
データ番号を省略した場合，瞬時値を取得します。
本関数メンバの実行後，データ番号毎にgetTimeDataを使用して時刻情報データを取得します。
次に，データ番号内のチャネル毎にgetChDataを使用して測定データを取得します。

戻り値

エラー番号を返します。

参照

```
getNo getPacketVersion getUserTime receivePacket sendPacket
```

CDAQMX::talkChInfo

構文

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

引数

startChNo	開始チャネル番号を指定します。
endChNo	終了チャネル番号を指定します。

説明

開始チャネル番号から終了チャネル番号までのチャネル情報データを取得する宣言を実行します。
本関数メンバの実行後，チャネル毎のデータ取得には，getChInfoを使用します。

戻り値

エラー番号を返します。

参照

```
getNo getPacketVersion getUserTime receivePacket sendPacket
```

CDAQMX::talkConfig

構文

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

引数

cMXSysInfo	システム構成データの返却先を指定します。
cMXStatus	ステータスデータの返却先を指定します。
cMXNetInfo	ネットワーク情報データの返却先を指定します。

説明

設定データを取得する宣言をします。
チャネル設定データを除く設定データを取得します。
本関数メンバの実行後， チャネルごとのデータ取得には， getChConfigを使用します。
設定データの内， 初期バランスデータと出力チャネルデータは， 別途各取得関数で取得します。

戻り値

エラー番号を返します。

参照

```
getNo getPacketVersion getStatusData getSystemConfig
getUserTime receivePacket sendPacket
```

CDAQMX::talkFIFOData

構文

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

引数

fifoNo	FIFO番号を指定します。
startDataNo	開始データ番号を指定します。
endDataNo	終了データ番号を指定します。

説明

測定データを取得する宣言を実行します。
指定されたデータ番号の範囲の測定データを取得します。実際に取得される範囲は指定と等しいとは限りません。
データ番号を省略した場合，瞬時値を取得します。
本関数メンバの実行後，データ番号毎にgetTimeDataを使用して時刻情報データを取得します。
次に，データ番号内のチャネル毎にgetChDataを使用して測定データを取得します。

戻り値

エラー番号を返します。

参照

```
getNo getPacketVersion getUserTime receivePacket sendPacket
```

CDAQMXAOPWMDataクラス

本クラスは，MX100でのAO/PWMデータを格納するクラスです。
MXAOPWMData構造体のラッパクラスになります。
全チャネル分のAO/PWMデータをまとめたものです。
PWMデータ番号，または， AOデータ番号で各データにアクセスできます。
AO/PWMデータの取得と設定のインタフェースとして仕様するクラスです。
指定する出力データ値と実際の出力値とを変換するユーティリティをサポートします。

パブリックメンバ

構築・消滅

CDAQMXAOPWMData	オブジェクトを構築します。
~CDAQMXAOPWMData	オブジェクトを消滅します。

構造体操作

getMXAOPWMData	構造体でデータを取得します。
setMXAOPWMData	構造体でデータを設定します。
initMXAOPWMData	構造体のデータを初期化します。

データメンバ操作

initialize	データメンバを初期化します。
getAOPWMValid	有効無効値を取得します。
getAOPWMValue	出力データ値を取得します。
setAOPWM	AO/PWMデータを設定します。

演算子

operator=	代入を実行します。

ユーティリティ

toAOPWMValue	出力値を出力データ値に変換します。
toRealValue	出力データ値を出力値に変換します。
isObject	オブジェクトをチェックします。

プロテクトメンバ

データメンバ

m_MXAOPWMData	AO/PWMデータの格納領域です。

メンバアクセス

getMXAOPWM 各チャネルごとのAO/PWMデータの構造体を取得します。

プライベートメンバ

なし。

関数メンバ

CDAQMXAOPWMData::CDAQMXAOPWMData

構文

```
CDAQMXAOPWMData(MXAOPWMData * pMXAOPWMData = NULL);
virtual ~CDAQMXAOPWMData(void);
```

引数

pMXAOPWMData AO/PWMデータを指定します。

説明

オブジェクトを構築，消滅します。
構築時， データメンバに指定された値を設定します。 指定がない場合， データメンバを初期化します。

参照

`setMXAOPWMData`

CDAQMXAOPWMData::getAOPWMValid

構文

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

引数

aopwmNo AO/PWMデータ番号を指定します。

説明

データメンバから指定されたデータ番号の示す有効無効値を取得します。
存在しない場合，「無効値」を返します。

戻り値

有効無効値を返します。

参照

`getMXAOPWM`

CDAQMXAOPWMData::getAOPWMValue

構文

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

引数

aopwmNo　　AO/PWMデータ番号を指定します。

説明

データメンバから指定されたデータ番号の示す出力データ値を取得します。
存在しない場合，0を返します。

戻り値

出力データ値を返します。

参照

`getMXAOPWM`

CDAQMXAOPWMData::getMXAOPWM

構文

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

引数

aopwmNo　　AO/PWMデータ番号を指定します。

説明

データメンバから指定されたデータ番号の示す構造体を取得します。
存在しない場合，NULLを返します。

戻り値

構造体へのポインタを返します。

CDAQMXAOPWMData::getMXAOPWMData

構文

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

引数

pMXAOPWMData　AO/PWMデータの返却先を指定します。

説明

構造体でデータを取得します。
データメンバの内容を指定された構造体に格納します。

CDAQMXAOPWMData::initialize

構文

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

説明

データメンバを初期化します。初期値は，原則0です。

参照

`initMXAOPWMData`

CDAQMXAOPWMData::initMXAOPWMData

構文

```
static void initMXAOPWMData(MXAOPWMData * pMXAOPWMData);
```

引数

pMXAOPWMData　AO/PWMデータの領域を指定します。

説明

指定された領域を初期化します。
初期値は，原則0です。

CDAQMXAOPWMData::isObject

構文

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

引数

classname　　クラス名を文字列で指定します。

説明

指定されたクラス名を継承しているかをチェックします。
引数が省略された場合，本クラスであるかをチェックします。
本クラスを継承したクラスが，自分自身のクラスをチェックするためには，オーバーライドする必要があります。
クラスを継承している場合，「有効値」(真)を返します。それ以外は，「無効値」(偽)を返します。

戻り値

有効無効値を返します。

CDAQMXAOPWMData::operator=

構文

```
CDAQMXAOPWMData & operator=(CDAQMXAOPWMData & cMXAOPWMData);
```

引数

cMXAOPWMData　代入するオブジェクトを指定します。

説明

指定されたオブジェクトのデータメンバを複写します。

戻り値

本オブジェクトへの参照を返します。

CDAQMXAOPWMData::setAOPWM

構文

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

引数

aopwmNo　　AO/PWMデータ番号を指定します。
bValid　　有効無効値を指定します。
iAOPWMValue　出力データ値を指定します。

説明

データメンバの指定されたデータ番号の示す領域に，指定された値を格納します。
データ番号に，定数値の「全AO/PWMデータ番号指定」をした場合， 全データに同じ値を格納します。

参照

`getMXAOPWM`

CDAQMXAOPWMData::setMXAOPWMData

構文

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

引数

pMXAOPWMData　　AO/PWMデータを指定します。

説明

構造体でデータを設定します。
データメンバに指定された構造体の内容を格納します。
指定がない場合， データメンバは初期化されます。

参照

`initMXAOPWMData`

CDAQMXAOPWMData::toAOPWMValue

構文

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

引数

realValue　　出力値を指定します。
iRangeAOPWM　AOレンジ，または，PWMレンジのレンジ種類を指定します。

説明

指定されたレンジ種類に従って， 出力値を出力データ値に変換します。
不正なレンジ種類の場合，0を返します。

戻り値

出力データ値を返します。

CDAQMXAOPWMData::toRealValue

構文

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

引数

iAOPWMValue　出力データ値を指定します。

iRangeAOPWM AOレンジ，または，PWMレンジのレンジ種類を指定します。

説明

指定されたレンジ種類に従って， 出力データ値を出力値に変換します。

不正なレンジ種類の場合，0を返します。

戻り値

出力値を返します

CDAQMXBalanceDataクラス

本クラスは，MX100での初期バランスデータを格納するクラスです。
MXBalanceData構造体のラッパクラスになります。
全チャネル分の初期バランスデータをまとめたものです。
初期バランスデータ番号で各データにアクセスできます。初期バランスデータ番号は，ひずみチャネルのチャネル番号です。
初期バランスの取得と設定のインタフェースとして使用するクラスです。

パブリックメンバ

構築・消滅

CDAQMXBalanceData	オブジェクトを構築します。
~CDAQMXBalanceData	オブジェクトを消滅します。

構造体操作

getMXBalanceData	構造体でデータを取得します。
setMXBalanceData	構造体でデータを設定します。
initMXBalanceData	構造体のデータを初期化します。

データメンバ操作

initialize	データメンバを初期化します。
getBalanceValid	有効無効値を取得します。
getBalanceValue	初期バランス値を取得します。
setBalance	初期バランスデータを設定します。

演算子

operator=	代入を実行します。

ユーティリティ

isObject	オブジェクトをチェックします。

プロテクトメンバ

データメンバ

m_MXBalanceData	初期バランスデータの格納領域です。

メンバアクセス

getMXBalance	各チャネルごとの初期バランスデータの構造体を取得します。

プライベートメンバ

なし。

関数メンバ

CDAQMXBalanceData::CDAQMXBalanceData

構文

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

引数

pMXBalanceData 初期バランスデータを指定します。

説明

オブジェクトを構築，消滅します。
構築時， データメンバに指定された値を設定します。指定がない場合， データメンバを初期化します。

参照

`setMXBalanceData`

CDAQMXBalanceData::getBalanceValid

構文

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

引数

balanceNo 初期バランスデータ番号を指定します。

説明

データメンバから指定されたデータ番号の示す有効無効値を取得します。
存在しない場合，「無効値」を返します。

戻り値

有効無効値を返します。

参照

`getMXBalance`

CDAQMXBalanceData::getBalanceValue

構文

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

引数

balanceNo 初期バランスデータ番号を指定します。

説明

データメンバから指定されたデータ番号の示す初期バランス値を取得します。
存在しない場合，0を返します。

戻り値

初期バランス値を返します。

参照

`getMXBalance`

CDAQMXBalanceData::getMXBalance

構文

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

引数

balanceNo 初期バランスデータ番号を指定します。

説明

データメンバから指定されたデータ番号の示す構造体を取得します。
存在しない場合，NULLを返します。

戻り値

構造体へのポインタを返します。

CDAQMXBalanceData::getMXBalanceData

構文

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

引数

pMXBalanceData 初期バランスデータの返却先を指定します。

説明

構造体でデータを取得します。
データメンバの内容を指定された構造体に格納します。

CDAQMXBalanceData::initialize

構文

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

説明

データメンバを初期化します。初期値は，原則0です。

参照

```
initMXBalanceData
```

CDAQMXBalanceData::isObject

構文

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

引数

classname　　クラス名を文字列で指定します。

説明

指定されたクラス名を継承しているかをチェックします。
引数が省略された場合，本クラスであるかをチェックします。
本クラスを継承したクラスが，自分自身のクラスをチェックするためには，オーバーライドする必要があります。
クラスを継承している場合，「有効値」(真)を返します。それ以外は，「無効値」(偽)を返します。

戻り値

有効無効値を返します。

CDAQMXBalanceData::operator=

構文

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

引数

cMXBalanceData　代入するオブジェクトを指定します。

説明

指定されたオブジェクトのデータメンバを複写します。

戻り値

本オブジェクトへの参照を返します。

CDAQMXBalanceData::setBalance

構文

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

引数

balanceNo　　初期バランスデータ番号を指定します。
bValid　　有効無効値を指定します。
iBalanceValue　初期バランス値を指定します。

説明

データメンバの指定されたデータ番号の示す領域に，指定された値を格納します。
データ番号に定数値の「全初期バランスデータ番号指定」をした場合， 全データに同じ値を格納します。

参照

`getMXBalance`

CDAQMXBalanceData::setMXBalanceData

構文

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

引数

pMXBalanceData　初期バランスデータを指定します。

説明

構造体でデータを設定します。
データメンバに指定された構造体の内容を格納します。
指定がない場合， データメンバは初期化されます。

参照

`initMXBalanceData`

CDAQMXBalanceResult::initMXBalanceData

構文

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

引数

pMXBalanceData　初期バランスデータの領域を指定します。

説明

指定された領域を初期化します。
初期値は，原則0です。

CDAQMXBalanceResultクラス

- **CDAQMXBalanceData**
  - **CDAQMXBalanceResult**

本クラスは，MX100での初期バランス結果を格納するクラスです。
MXBalanceResult構造体のラッパクラスになります。
CDAQMXBalanceDataクラスを継承していて，初期バランスデータを包含しています。
全チャネル分の初期バランス結果を集約したものです。
初期バランスデータ番号で各データにアクセスできます。初期バランスデータ番号は，ひずみチャネルのチャネル番号です。
初期バランスの実行とリセットで，指定と結果のインターフェイスとして使用するクラスです。

パブリックメンバ

構築・消滅

CDAQMXBalanceResult	オブジェクトを構築します。
~CDAQMXBalanceResult	オブジェクトを消滅します。

構造体操作

getMXBalanceResult	構造体でデータを取得します。
setMXBalanceResult	構造体でデータを設定します。
initMXBalanceResult	構造体のデータを初期化します。

データメンバ操作

getResult	初期バランス結果を取得します。
setResult	初期バランス結果を設定します。

演算子

operator=	代入を実行します。

●オーバライドしたメンバ

データメンバ操作

initialize	データメンバを初期化します。

ユーティリティ

isObject	オブジェクトをチェックします。

●継承するメンバ

CDAQMXBalanceData参照

```
getMXBalanceData setMXBalanceData initMXBalanceData
getBalanceValid getBalanceValue setBalance
```

プロテクトメンバ

データメンバ

継承するメンバも参照。

m_MXBalanceResult　初期バランス結果の格納領域です。

●継承するメンバ

CDAQMXBalanceData参照

```
getMXBalance m_MXBalanceData
```

プライベートメンバ

なし。

関数メンバ

CDAQMXBalanceResult::CDAQMXBalanceResult

構文

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

引数

pMXBalanceData　　初期バランスデータを指定します。

pMXBalanceResult　初期バランス結果を指定します。

説明

オブジェクトを構築，消滅します。

構築時，データメンバに指定された値を設定します。指定がない場合，データメンバを初期化します。

参照

```
setMXBalanceResult
CDAQMXBalanceData::CDAQMXBalanceData
```

CDAQMXBalanceResult::getMXBalanceResult

構文

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

引数

pMXBalanceResult　初期バランス結果の返却先を指定します。

説明

構造体でデータを取得します。
データメンバの内容を指定された構造体に格納します。

CDAQMXBalanceResult::getResult

構文

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

引数

balanceNo　　初期バランスデータ番号を指定します。

説明

データメンバから指定されたデータ番号の示す初期バランス結果を取得します。
存在しない場合，「指定なし」を返します。

戻り値

初期バランス結果を返します。

CDAQMXBalanceResult::initialize

構文

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

説明

データメンバを初期化します。
初期値は，原則0です。

参照

```
initMXBalanceResult
CDAQMXBalanceData::initialize
```

CDAQMXBalanceResult::initMXBalanceResult

構文

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

引数

pMXBalanceResult　初期バランス結果の領域を指定します。

説明

指定された領域を初期化します。
初期値は，原則0です。

CDAQMXBalanceResult::isObject

構文

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

引数

classname　　クラス名を文字列で指定します。

説明

指定されたクラス名を継承しているかをチェックします。
引数が省略された場合，本クラスであるかをチェックします。
本クラスを継承したクラスが，自分自身のクラスをチェックするためには，オーバーライドする必要があります。
クラスを継承している場合，「有効値」(真)を返します。それ以外は，「無効値」(偽)を返します。
本クラスと異なる場合，親クラスでチェックします

戻り値

有効無効値を返します。

参照

`CDAQMXBalanceData::isObject`

CDAQMXBalanceResult::operator=

構文

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

引数

cMXBalanceResult　代入するオブジェクトを指定します。

説明

指定されたオブジェクトのデータメンバを複写します。

戻り値

本オブジェクトへの参照を返します。

CDAQMXBalanceResult::setMXBalanceResult

構文

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

引数

pMXBalanceResult　初期バランス結果を指定します。

説明

構造体でデータを設定します。
データメンバに指定された構造体の内容を格納します。
指定がない場合，データメンバは初期化されます。

参照

`initMXBalanceResult`

CDAQMXBalanceResult::setResult

構文

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

引数

balanceNo　　初期バランスデータ番号を指定します。
iResult　　初期バランス結果を指定します。

説明

データメンバの指定されたデータ番号の示す領域に，指定された値を格納します。
データ番号に，「全初期バランス番号指定」をした場合，全データに同じ値を格納します。

CDAQMXChConfigクラス

- **CDAQChInfo**
  - **CDAQMXChID**
    - **CDAQMXChConfig**

本クラスは，MX100でのチャネル設定データを格納するクラスです。
MXChConfig構造体のラッパクラスになります。
設定データ取得において，チャネル単位の設定データを格納するインターフェイスとして使用するクラスです。

パブリックメンバ

構築・消滅

CDAQMXChConfig	オブジェクトを構築します。
~CDAQMXChConfig	オブジェクトを消滅します。

構造体操作

getMXChConfig	構造体でデータを取得します。
setMXChConfig	構造体でデータを設定します。
initMXChConfig	構造体のデータを初期化します。

データメンバ操作

getSpanMin	スパン最小値を取得します。
getSpanMax	スパン最大値を取得します。
getScaleMin	スケール最小値を取得します。
getScaleMax	スケール最大値を取得します。
getRefChNo	基準チャネル番号を取得します。
getFilter	フィルタ係数を取得します。
getRJCType	RJC種類を取得します。
getRJCVolt	RJC電圧値を取得します。
getBurnout	バーンアウト種類を取得します。
isDeenergize	リレーの非励磁動作の有効/無効を取得します。
isHold	リレーの保持動作の有効/無効を取得します。
isRefAlarm	参照アラームを取得します。
isChatFilter	チャタリングフィルタの値を取得します。
setRefChNo	基準チャネル番号を設定します。
setFilter	フィルタ係数を設定します。
setBurnout	バーンアウト種類を設定します。
setRJCType	RJC種類を設定します。
setAlarm	アラームを設定します。
setDeenergize	リレーの非励磁動作の有効/無効を設定します。
setHold	リレーの保持動作の有効/無効を設定します。
setRefAlarm	参照アラームを設定します。
setChatFilter	チャタリングフィルタを設定します。

レンジ設定

setSKIP	スキップ(未使用)を設定します。
setVOLT	直流電圧レンジを設定します。
setTC	熱電対レンジを設定します。
setRTD	測温抵抗体レンジを設定します。
setDI	ディジタル入力(DI)レンジを設定します。
setDELTA	チャネル間差演算を設定します。
setSpan	スパンを設定します。
setScalling	スケールを設定します。
changeRange	温度単位種類でレンジを変更します。
setRES	抵抗レンジを設定します。
setSTRAIN	ひずみレンジを設定します。
setAO	AOレンジを設定します。
setPWM	PWMレンジを設定します。
setCOM	通信レンジを設定します。
setPULSE	パルスレンジを設定します。

検証

isCorrect	妥当性を検証します。

ユーティリティ

getItemError	エラー検出した設定項目番号を取得します。
getRangePoint	レンジ種類の小数点位置を取得します。
getRangeMin	レンジ種類の設定範囲最小値を取得します。
getRangeMax	レンジ種類の設定範囲最大値を取得します。

演算子

operator=	代入を実行します。

●オーバライドしたメンバ

データメンバ操作

initialize	データメンバを初期化します。

ユーティリティ

isObject	オブジェクトをチェックします。

●継承するメンバ

CDAQChInfo参照

```
getChNo getPoint setChNo setPoint
```

CDAQMXChID参照

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

プロテクトメンバ

データメンバ

m_MXChConfigAIDI	AI，DIの設定情報の格納領域です。
m_MXChConfigAI	AIの設定情報の格納領域です。
m_MXChConfigDO	DOの設定情報の格納領域です。
m_nItemError	設定項目番号の格納領域です。

●継承するメンバ

CDAQChInfo参照

```
m_chNo m_chType m_point
```

CDAQMXChID参照

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

プライベートメンバ

なし。

関数メンバ(アルファベット順)

CDAQMXChConfig::CDAQMXChConfig

構文

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

引数

pMXChConfig　チャネル設定データを指定します。

説明

オブジェクトを構築，消滅します。

構築時，指定されたデータをデータメンバに格納します。指定がない場合，データメンバを初期化します。

参照

```
setMXChConfig
```

CDAQMXChConfig::changeRange

構文

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

引数

iTempUnit　　温度単位種類を指定します。

説明

温度単位種類でレンジを変更します。
熱電対レンジ，測温抵抗体レンジの設定値がレンジの既定値になります。
スパン，スケール，小数点位置，単位名，基準チャネル番号が規定値になります。アラーム設定は初期化(クリア)されます。

参照

```
getRange setTC setRTD
```

CDAQMXChConfig::getBurnout

構文

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

説明

データメンバのAIの設定情報領域からバーンアウトの値を取得します。

戻り値

バーンアウトを返します。

CDAQMXChConfig::getFilter

構文

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

説明

データメンバのAIの設定情報領域からフィルタ係数の値を取得します。

戻り値

フィルタ時定数を返します。

CDAQMXChConfig::getItemError

構文

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

説明

データメンバの設定項目番号領域の値を取得します。

戻り値

設定項目番号を返します。

CDAQMXChConfig::getMXChConfig

構文

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

引数

pMXChConfig　チャネル設定データの返却先を指定します。

説明

構造体でデータを取得します。
データメンバの内容を指定された構造体に格納します。

参照

`getMXChID`

CDAQMXChConfig::getRangeMax

構文

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

引数

iRange	レンジ種類を指定します。
iTempUnit	温度単位種類を指定します。

説明

指定されたレンジ種類の設定範囲最大値を取得します。
デジタル入力の場合，詳細レンジを指定します。
指定が不明の場合，0を返します。
返却される値は，小数点位置を取り除いた値です。

戻り値

レンジ設定範囲の最大値を返します。

CDAQMXChConfig::getRangeMin

構文

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

引数

iRange	レンジ種類を指定します。
iTempUnit	温度単位種類を指定します。

説明

指定されたレンジ種類の設定範囲最小値を取得します。
デジタル入力の場合，詳細レンジを指定します。
指定が不明の場合，0を返します。
返却される値は，小数点位置を取り除いた値です。

戻り値

レンジ設定範囲の最小値を返します。

CDAQMXChConfig::getRangePoint

構文

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

引数

iRange	レンジ種類を指定します。
iTempUnit	温度単位種類を指定します。

説明

指定されたレンジ種類の小数点位置を取得します。
デジタル入力の場合，詳細レンジを指定します。
指定が不明の場合，0を返します。

戻り値

小数点位置を返します。

CDAQMXChConfig::getRefChNo

構文

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

説明

データメンバのAI，DIの設定情報領域から基準チャネル番号の値を取得します。

戻り値

基準チャネル番号を返します。

CDAQMXChConfig::getRJCType

構文

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

説明

データメンバのAIの設定情報領域からRJC種類の値を取得します。

戻り値

RJC種類を返します。

CDAQMXChConfig::getRJCVolt

構文

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

説明

データメンバのAIの設定情報領域からRJC電圧値の値を取得します。

戻り値

RJC電圧値を返します。

CDAQMXChConfig::getScaleMax

構文

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

説明

データメンバのAI，DIの設定情報領域からスケール最大値の値を取得します。

戻り値

スケール最大値を返します。

CDAQMXChConfig::getScaleMin

構文

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

説明

データメンバのAI，DIの設定情報領域からスケール最小値の値を取得します。

戻り値

スケール最小値を返します。

CDAQMXChConfig::getSpanMax

構文

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

説明

データメンバのAI，DIの設定情報領域からスパン最大値の値を取得します。

戻り値

スパン最大値を返します。

CDAQMXChConfig::getSpanMin

構文

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

説明

データメンバのAI，DIの設定情報領域からスパン最小値の値を取得します。

戻り値

スパン最小値を返します。

CDAQMXChConfig::initialize

構文

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

説明

データメンバを初期化します。初期値は，原則0です。

参照

```
CDAQMXChID::initialize
```

CDAQMXChConfig::initMXChConfig

構文

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

引数

pMXChConfig　チャネル設定データの領域を指定します。

説明

指定された領域を初期化します。
初期値は，原則0です。

CDAQMXChConfig::isChatFilter

構文

```
int isChatFilter(void);
```

説明

データメンバのAI，DIの設定情報の格納領域からチャタリングフィルタの値を取得します。

0なら無効，それ以外は有効です。

戻り値

有効無効値を返します。

CDAQMXChConfig::isCorrect

構文

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

引数

iTempUnit　　温度単位種類を指定します。

説明

妥当性の検証をします。

チャネル種類に従った各設定項目をチェックします。

不正な値を検出した場合，「無効値」を返します。

不正な値を検出した場合，データメンバの設定項目番号領域に検出した場所を示す設定項目番号を格納します。

戻り値

有効無効値を返します。

参照

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

CDAQMXChConfig::isDeenergize

構文

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

説明

データメンバのDOの設定情報領域から非励磁動作の値を取得します。

0なら無効，それ以外は有効です。

戻り値

有効無効値を返します。

CDAQMXChConfig::isHold

構文

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

説明

データメンバのDOの設定情報領域から保持動作の値を取得します。
0なら無効，それ以外は有効です。

戻り値

有効無効値を返します。

CDAQMXChConfig::isObject

構文

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

引数

classname　　クラス名を文字列で指定します。

説明

指定されたクラス名を継承しているかをチェックします。
引数が省略された場合，本クラスであるかをチェックします。
本クラスを継承したクラスが，自分自身のクラスをチェックするためには，オーバーライドする必要があります。
クラスを継承している場合，「有効値」(真)を返します。それ以外は，「無効値」(偽)を返します。
本クラスと異なる場合，親クラスをチェックします。

戻り値

有効無効値を返します。

参照

`CDAQMXChID::isObject`

CDAQMXChConfig::isRefAlarm

構文

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

引数

refChNo　　参照するチャネル番号を指定します。
levelNo　　アラームレベルを指定します。

説明

データメンバのDOの設定情報領域から指定された参照アラームの値を取得します。
0なら無効，それ以外は有効です。
指定が範囲外の場合，「無効値」を返します。

戻り値

有効無効値を返します。

CDAQMXChConfig::operator=

構文

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

引数

cMXChConfig　代入するオブジェクトを指定します。

説明

指定されたオブジェクトのデータメンバを複写します。

戻り値

本オブジェクトへの参照を返します。

参照

```
getMXChConfig setMXChConfig
```

CDAQMXChConfig::setAlarm

構文

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

引数

levelNo	アラームレベルを指定します。
iAlarmType	アラーム種類を指定します。
value	アラーム値を指定します。
histerisys	ヒステリシスを指定します。

説明

データメンバのアラーム領域に指定された値を格納します。
アラーム値，ヒステリシスからアラーム発生のしきい値(On値)とアラーム停止のしきい値(Off値)を生成します。
アラーム種類が，すでに設定されているチャネル種類で許されていない場合，無視されます。

参照

```
getkind setAlarmValue
```

CDAQMXChConfig::setAO

構文

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

引数

iRangeAO　　AOレンジのレンジ種類を指定します。

説明

指定されたレンジを設定します。
チャネルステータスを有効にします。
各設定値は指定されたレンジの既定値になります。
既に設定されているチャネル種類に従って，基準チャネル番号を保持します。コマンドAOチャネルの場合， 基準チャネル番号は「未定義参照チャネル番号」です。
AOレンジ以外の指定の場合，無視します。AOまたはコマンドAOチャネルに設定されていない場合，無視します。
スパン，スケール，小数点位置，単位名が既定値になります。
アラーム設定は初期化(クリア)されます。

参照

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

CDAQMXChConfig::setBurnout

構文

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

引数

iBurnout　　バーンアウトを指定します。

説明

データメンバのAIの設定情報領域に指定された値を格納します。

CDAQMXChConfig::setChatFilter

構文

```
void setChatFilter(int bChatFilter);
```

引数

bChatFilterチャタリングフィルタを有効無効値で指定します。

説明

データメンバのAI，DIの設定情報に指定された値を格納します。

CDAQMXChConfig::setCOM

構文

```
void setCOM(int iRangeCOM);
```

引数

iRangeCOM　　レンジ種類から通信レンジを指定します。

説明

指定されたレンジを設定します。
チャネルステータスを有効にします。
各設定値は指定されたレンジの既定値になります。
チャネル種類は，「CAN Bus入力」になります。
通信レンジ以外の指定の場合，無視します。
スパン，スケール，小数点位置，単位名，基準チャネル番号が既定値になります。
アラーム設定は初期化(クリア)されます。

参照

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

CDAQMXChConfig::setDeenergize

構文

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

引数

bDeenergize　　非励磁動作を有効無効値で指定します。

説明

データメンバのDOの設定情報領域に指定された値を格納します。

CDAQMXChConfig::setDELTA

構文

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

引数

refChNo	参照するチャネル番号を指定します。
iRange	レンジ種類を指定します。
iTempUnit	温度単位種類を指定します。

説明

指定されたレンジを設定します。
チャネルステータスを有効にします。各設定値は指定されたレンジの既定値になります。
チャネル種類は，R3.01より前のAPIでは「AI(チャネル間差)」または「DI(チャネル間差)」になります。R3.01からは「パルス入力(チャネル間差)」，「CAN Bus入力(チャネル間差)」にも対応しています。
レンジ種類の指定には参照レンジを使用せず，個別に各レンジを指定します。ディジタル入力(DI)レンジの場合，ディジタル入力(DI)詳細レンジで指定してください。
入力レンジ以外の指定の場合，無視します。
スパン，スケール，小数点位置，単位名，基準チャネル番号が既定値になります。
アラーム設定は初期化(クリア)されます。

参照

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

CDAQMXChConfig::setDI

構文

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

引数

iRangeDI	ディジタル入力(DI)詳細レンジのレンジ種類を指定します。

説明

指定されたレンジを設定します。
チャネルステータスを有効にします。各設定値は指定されたレンジの既定値になります。
チャネル種類は，「DI」になります。
デジタル入力の詳細レンジ以外の指定の場合，無視します。
スパン，スケール，小数点位置，単位名，基準チャネル番号が既定値になります。
アラーム設定は初期化(クリア)されます。

参照

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

CDAQMXChConfig::setFilter

構文

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

引数

iFilter　　フィルタ係数を指定します。

説明

データメンバのAIの設定情報領域に指定された値を格納します。

CDAQMXChConfig::setHold

構文

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

引数

bHold　　保持動作を有効無効値で指定します。

説明

データメンバのDOの設定情報領域に指定された値を格納します。

CDAQMXChConfig::setMXChConfig

構文

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

引数

pMXChConfig　チャネル設定データを指定します。

説明

構造体でデータを設定します。データメンバに指定された構造体の内容を格納します。

指定がない場合，データメンバは初期化されます。

参照

`initialize setMXChID`

CDAQMXChConfig::setPULSE

構文

```
void setPULSE(int iRangePULSE);
```

引数

iRangePULSE　レンジ種類からパルスレンジを指定します。

説明

指定されたレンジを設定します。
チャネルステータスを有効にします。
各設定値は指定されたレンジの既定値になります。
チャネル種類は，「パルス入力」になります。
パルスレンジ以外の指定の場合，無視します。
スパン，スケール，小数点位置，単位名，基準チャネル番号が既定値になります。
アラーム設定は初期化(クリア)されます。

参照

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

CDAQMXChConfig::setPWM

構文

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

引数

iRangePWM　　PWMレンジのレンジ種類を指定します。

説明

指定されたレンジを設定します。
チャネルステータスを有効にします。
各設定値は指定されたレンジの既定値になります。
既に設定されているチャネル種類に従って，基準チャネル番号を保持します。コマンドPWMチャネルの場合， 基準チャネル番号は「未定義参照チャネル番号」です。
PWMレンジ以外の指定の場合，無視します。PWMまたはコマンドPWMチャネルに設定されていない場合，無視します。
スパン，スケール，小数点位置，単位名が既定値になります。
アラーム設定は初期化(クリア)されます。

参照

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

CDAQMXChConfig::setRefAlarm

構文

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

引数

refChNo	参照するチャネル番号を指定します。
levelNo	アラームレベルを指定します。
bValid	有効無効値を指定します。

説明

データメンバのDOの設定情報領域に指定された値を格納します。
参照するチャネル番号に定数値の「全参照チャネル番号指定」をした場合，全チャネル分に値を格納します。
アラームレベルに定数値の「全アラームレベル番号指定」を指定した場合，全アラームレベル分に値を格納します。

CDAQMXChConfig::setRefChNo

構文

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

引数

refChNo	参照するチャネル番号を指定します。

説明

データメンバのAI，DIの設定情報領域に指定された値を格納します。
自分自身と範囲外のチャネル番号は無視します。
参照チャネルが存在しない指定をする場合，定数値の「未定義参照チャネル番号」を指定します。

CDAQMXChConfig::setRES

構文

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

引数

iRangeRES　　抵抗レンジのレンジ種類を指定します。

説明

指定されたレンジを設定します。
チャネルステータスを有効にします。
各設定値は指定されたレンジの既定値になります。
チャネル種類は，「AI」になります。
抵抗レンジ以外の指定の場合，無視します。
スパン，スケール，小数点位置，単位名，基準チャネル番号が既定値になります。
アラーム設定は初期化(クリア)されます。

参照

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

CDAQMXChConfig::setRJCType

構文

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

引数

iRJCType　　RJC種類を指定します。
volt　　RJC電圧値を指定します。

説明

データメンバのAIの設定情報領域に指定された値を格納します。

CDAQMXChConfig::setRTD

構文

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

引数

iRangeRTD	測温抵抗体レンジのレンジ種類を指定します。
iTempUnit	温度単位種類を指定します。

説明

指定されたレンジを設定します。
チャネルステータスを有効にします。各設定値は指定されたレンジの既定値になります。
チャネル種類は，「AI」の値になります。
測温抵抗体レンジ以外の指定の場合，無視します。
スパン，スケール，小数点位置，単位名，基準チャネル番号が既定値になります。アラーム設定は初期化(クリア)されます。

参照

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

CDAQMXChConfig::setScalling

構文

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

引数

scaleMin	スケール最小値を指定します。
scaleMax	スケール最大値を指定します。
scalePoint	小数点位置を指定します。
iTempUnit	温度単位種類を指定します。

説明

データメンバのAI，DIの設定情報領域に指定された値を格納します。
スケール種類を「線形」に設定します。
既に設定されているレンジ種類とチャネル種類に従って値をチェックします。
最大，最小値が同じ場合，スケール種類を「なし」に設定します。また，既定値の小数点位置を設定します。
AI(リモートRJC)，DO，AO，PWMなどスケールを設定できないチャネル種類の場合は無視します。

参照

```
getKind setPoint getRange getRangePoint setType
```

CDAQMXChConfig::setSKIP

構文

```
void setSKIP(void);
```

説明

スキップ(未使用)を設定します。
チャネルステータスを無効にします。

参照

```
setValid
```

CDAQMXChConfig::setSpan

構文

```
void setSpan(int spanMin, int spanMax, int iTempUnit =
DAQMX_TEMPUNIT_C);
```

引数

spanMin	スパン最小値を指定します。
spanMax	スパン最大値を指定します。
iTempUnit	温度単位種類を指定します。

説明

データメンバのAI，DIの設定情報領域に指定された値を格納します。
既に設定されているレンジ種類とチャネル種類に従って値をチェックします。
最大，最小値が同じ場合，格納しません。値が範囲外の場合，可能な値に丸め込みます。
AO/PWMチャネルで，最大，最小値が逆転している場合，無視します。

参照

```
getKind getRange
```

CDAQMXChConfig::setSTRAIN

構文

```
void setSTRAIN(int iRangeSTR);
```

引数

iRangeSTR	ひずみレンジのレンジ種類を指定します。

説明

指定されたレンジを設定します。
チャネルステータスを有効にします。
各設定値は指定されたレンジの既定値になります。
チャネル種類は，「AI」になります。
ひずみレンジ以外の指定の場合，無視します。
スパン，スケール，小数点位置，単位名，基準チャネル番号が既定値になります。
アラーム設定は初期化(クリア)されます。

参照

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

CDAQMXChConfig::setTC

構文

```
void setTC(int iRangeTC, int iTempUnit = DAQMX_TEMPUNIT_C);
```

引数

iRangeTC　　熱電対レンジのレンジ種類を指定します。
iTempUnit　　温度単位種類を指定します。

説明

指定されたレンジを設定します。
チャネルステータスを有効にします。各設定値は指定されたレンジの既定値になります。
チャネル種類は，「AI」の値になります。
熱電対レンジ以外の指定の場合，無視します。
スパン，スケール，小数点位置，単位名，基準チャネル番号が既定値になります。
アラーム設定は初期化(クリア)されます。

参照

`setAlarm setPoint setRefChNo setScalling setSpan setType setUnit setValid`

CDAQMXChConfig::setVOLT

構文

```
void setVOLT(int iRangeVOLT);
```

引数

iRangeVOLT　　直流電圧レンジのレンジ種類を指定します。

説明

指定されたレンジを設定します。
チャネルステータスを有効にします。各設定値は指定されたレンジの既定値になります。
チャネル種類は，「AI」の値になります。
直流電圧レンジ以外の指定の場合，無視します。
スパン，スケール，小数点位置，単位名，基準チャネル番号が既定値になります。
アラーム設定は初期化(クリア)されます。

参照

`setAlarm setPoint setRefChNo setScalling setSpan setType setUnit setValid`

CDAQMXChConfigDataクラス

本クラスは，MX100での全チャネル分のチャネル設定データを格納するクラスです。
MXChConfigData構造体のラッパクラスになります。
設定データ取得において，全チャネルの設定データを格納するインターフェイスとして使用するクラスです。
CDAQMXChConfigクラスを全チャネル個数分集約したクラスになります。
チャネル間の関連情報を必要とする処理を実装しています。

パブリックメンバ

構築・消滅

CDAQMXChConfigData	オブジェクトを構築します。
~CDAQMXChConfigData	オブジェクトを消滅します。

構造体操作

getMXChConfigData	構造体でデータを取得します。
setMXChConfigData	構造体でデータを設定します。
initMXChConfigData	構造体のデータを初期化します。
setMXChConfig	チャネル毎の構造体でデータを設定します。

データメンバ操作

initialize	データメンバを初期化します。
getClassMXChConfig	チャネル毎のデータを取得します。

レンジ設定

setRRJC	リモートRJCを設定します。
changeRange	温度単位種類でレンジを変更します。

検証

isCorrect	妥当性を検証します。

ユーティリティ

getItemError	エラー検出した設定項目番号を取得します。
isObject	オブジェクトをチェックします。

演算子

operator=	代入を実行します。

プロテクトメンバ

データメンバ

m_cMXChConfig	全チャネル分のチャネル設定データの格納領域です。
m_pcMXChConfig	全チャネル分のチャネル設定データの格納領域の先頭ポインタです。
m_nItemError	設定項目番号の格納領域です。

プライベートメンバ

なし。

関数メンバ(アルファベット順)

CDAQMXChConfigData::CDAQMXChConfigData

構文

```
CDAQMXChConfigData(MXChConfigData * pMXChConfigData = NULL);
virtual ~CDAQMXChConfigData(void);
```

引数

pMXChConfigData　全チャネル分のチャネル設定データを指定します。

説明

オブジェクトを構築，消滅します。
構築時，データメンバに指定された値を設定します。指定がない場合，データメンバを初期化します。

参照

`setMXChConfigData`

CDAQMXChConfigData::changeRange

構文

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

引数

iTempUnit　　温度単位種類を指定します。

説明

温度単位種類でレンジを変更します。
熱電対レンジ，測温抵抗体レンジの設定値がレンジの既定値になります。
全チャネルを一括変更します。

参照

`CDAQMXChConfig::changeRange`

CDAQMXChConfigData::getClassMXChConfig

構文

```
CDAQMXChConfig * getClassMXChConfig(int chNo);
```

引数

chNo　　チャネル番号を指定します。

説明

データメンバから，指定されたチャネル番号に対応するチャネル設定データ領域をオブジェクトで取得します。
存在しない場合，NULLを返します。

戻り値

オブジェクトへの参照を返します。

CDAQMXChConfigData::getItemError

構文

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

説明

データメンバの設定項目番号領域の値を取得します。

戻り値

設定項目番号を返します。

CDAQMXChConfigData::getMXChConfigData

構文

```
void getMXChConfigData(MXChConfigData * pMXChConfigData);
```

引数

pMXChConfigData　全チャネル分のチャネル設定データの返却先を指定します。

説明

構造体でデータを取得します。データメンバの内容を指定された構造体に格納します。

参照

`CDAQMXChConfig::getMXChConfig`

CDAQMXChConfigData::initialize

構文

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

説明

データメンバを初期化します。初期値は，原則0です。

参照

`CDAQMXChConfig::initialize`

CDAQMXChConfigData::initMXChConfigData

構文

```
static void initMXChConfigData(MXChConfigData *
pMXChConfigData);
```

引数

pMXChConfig 全チャネル分のチャネル設定データの領域を指定します。

説明

指定された領域を初期化します。
初期値は，原則0です。

CDAQMXChConfigData::isCorrect

構文

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

引数

iTempUnit 温度単位種類を指定します。

説明

妥当性の検証をします。
一括で全チャネルを検証します。
チャネル間の関連をチェックします。
不正な値を検出した場合，「無効値」を返します。
不正な値を検出した場合，データメンバの設定項目番号領域に検出した場所を示す設定項目番号を格納します。

戻り値

有効無効値を返します。

参照

CDAQMXChConfig::getItemError CDAQMXChConfig::getKind
CDAQMXChConfig::getRange CDAQMXChConfig::getRefChNo
CDAQMXChConfig::isCorrect CDAQMXChConfig::isValid

CDAQMXChConfigData::isObject

構文

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

引数

classname　クラス名を文字列で指定します。

説明

指定されたクラス名を継承しているかをチェックします。
引数が省略された場合，本クラスであるかをチェックします。
本クラスを継承したクラスが，自分自身のクラスをチェックするためには，オーバーライドする必要があります。
クラスを継承している場合，「有効値」(真)を返します。それ以外は，「無効値」(偽)を返します。

戻り値

有効無効値を返します。

CDAQMXChConfigData::operator=

構文

```
CDAQMXChConfigData & operator=(CDAQMXChConfigData &
cMXChConfigData);
```

引数

cMXChConfigData　代入するオブジェクトを指定します。

説明

指定されたオブジェクトのデータメンバを複写します。

戻り値

本オブジェクトへの参照を返します。

CDAQMXChConfigData::setMXChConfig

構文

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

引数

pMXChConfig　チャネル設定データを指定します。

説明

構造体でデータを設定します。
指定された構造体の中のチャネル番号に対応するデータメンバ領域に指定された構造体の内容を格納します。対応するデータメンバ領域が存在しない場合，何もしません。

参照

```
getClassMXChConfig
CDAQMXChConfig::setMXChConfig
```

CDAQMXChConfigData::setMXChConfigData

構文

```
void setMXChConfigData(MXChConfigData * pMXChConfigData);
```

引数

pMXChConfigData　全チャネル分のチャネル設定データを指定します。

説明

構造体でデータを設定します。データメンバに指定された構造体の内容を格納します。

指定がない場合，データメンバは初期化されます。

参照

```
initialize
CDAQMXChConfig::setMXChConfig
```

CDAQMXChConfigData::setRRJC

構文

```
void setRRJC(int chNo, int refChNo);
```

引数

chNo　　チャネル番号を指定します。

refChNo　　参照するチャネル番号を指定します。

説明

リモートRJCを設定します。

指定された参照チャネルと同じ測定レンジが設定されます。

参照チャネルの内容を複写し，チャネル番号，チャネル種類，参照チャネル番号を上書きします。

チャネル種類は，「AI(リモートRJC)」になります。

スケール種類は，「なし」になります。

熱電対レンジ以外の場合，何もしません。

アラーム設定は初期化されます。

参照

```
getClassMXChConfig
CDAQMXChConfig::getKind CDAQMXChConfig::getRange
CDAQMXChConfig::isValid CDAQMXChConfig::setAlarm
CDAQMXChConfig::setChNo CDAQMXChConfig::setRefChNo
CDAQMXChConfig::setType
```

CDAQMXChIDクラス

■ CDAQChInfo
・ CDAQMXChID

本クラスは，MX100でのチャネル識別情報を格納するクラスです。
MXChID構造体のラッパクラスになります。
チャネル情報データ，チャネル設定データの共通部です。

パブリックメンバ

構築・消滅

CDAQMXChID	オブジェクトを構築します。
~CDAQMXChID	オブジェクトを消滅します。
initMXChID	構造体のデータを初期化します。

構造体操作

getMXChID	構造体でデータを取得します。
setMXChID	構造体でデータを設定します。
initMXChID	構造体のデータを初期化します。

データメンバ操作

isValid	チャネルステータスを取得します。
getKind	チャネル種類を取得します。
getRange	レンジ種類を取得します。
getScale	スケール種類を取得します。
getUnit	単位名を取得します。
getTag	タグを取得します。
getComment	コメントを取得します。
getAlarmType	アラーム種類を取得します。
getAlarmValueON	On値を取得します。
getAlarmValueOFF	Off値を取得します。
setValid	チャネルステータスを設定します。
setType	チャネル種類，レンジ種類，スケール種類を設定します。
setUnit	単位名を設定します。
setTag	タグを設定します。
setComment	コメントを設定します。
setAlarmValue	アラームを設定します。

ユーティリティ

getChName	チャネル名を取得します。
toChName	チャネル名を生成します。
toChNo	チャネル名からチャネル番号を抽出します。
toUnitNo	チャネル名からユニット番号を抽出します。

演算子

operator=	代入を実行します。

●オーバライドしたメンバ

データメンバ操作

initialize	データメンバを初期化します。
getChType	チャネルタイプを取得します。
setChType	チャネルタイプを設定します。

ユーティリティ

isObject	オブジェクトをチェックします。

●継承するメンバ

CDAQChInfo参照

```
getChNo getPoint setChNo setPoint
```

プロテクトメンバ

データメンバ

m_valid	チャネルステータスの格納領域です。
m_kind	チャネル種類の格納領域です。
m_range	レンジ種類の格納領域です。
m_scaleType	スケール種類の格納領域です。
m_unit	単位名の格納領域です。
m_tag	タグの格納領域です。
m_comment	コメントの格納領域です。
m_alarm	アラームの格納領域です。

メンバアクセス

getMXAlarm	各アラームレベルごとのアラーム情報の構造体を取得します。

●継承するメンバ

CDAQChInfo参照

```
m_chNo m_chType m_point
```

プライベートメンバ

なし。

関数メンバ

CDAQMXChID::CDAQMXChID

構文

```
CDAQMXChID(MXChID * pMXChID = NULL);
virtual ~CDAQMXChID(void);
```

引数

pMXChID　　チャネル識別情報を指定します。

説明

オブジェクトを構築，消滅します。
構築時，指定されたデータをデータメンバに格納します。指定がない場合，データメンバを初期化します。

参照

`setMXChID`

CDAQMXChID::getAlarmType

構文

```
int getAlarmType(int levelNo);
```

引数

levelNo　　アラームレベルを指定します。

説明

データメンバのアラーム領域から指定されたアラームレベルのアラーム種類を取得します。
存在しない場合，「アラームなし」を返します。

戻り値

アラーム種類を返します。

参照

`getMXAlarm`

CDAQMXChID::getAlarmValueOFF

構文

```
int getAlarmValueOFF(int levelNo);
```

引数

levelNo　　アラームレベルを指定します。

説明

データメンバのアラーム領域から指定されたアラームレベルのアラーム停止のしきい値(Off値)を取得します。アラームレベルが範囲外の場合，0を返します。

戻り値

アラーム停止のしきい値(Off値)を返します。

参照

`getMXAlarm`

CDAQMXChID::getAlarmValueON

構文

```
int getAlarmValueON(int levelNo);
```

引数

levelNo　　アラームレベルを指定します。

説明

データメンバのアラーム領域から指定されたアラームレベルのアラーム発生のしきい値(On値)を取得します。
アラームレベルが範囲外の場合，0を返します。

戻り値

アラーム発生のしきい値(On値)を返します。

参照

`getMXAlarm`

CDAQMXChID::getChName

構文

```
int getChName(int unitno = 0);
```

引数

unitno　　ユニット番号を指定します。

説明

データメンバのチャネル番号と指定されたユニット番号からチャネル名を生成します。

戻り値

チャネル名を返します。

参照

`getChNo toChName`

CDAQMXChID::getChType

構文

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

説明

データメンバからチャネルタイプ領域の値を取得します。
チャネルタイプは0なので，必ず0を返します。

戻り値

チャネルタイプを返します。

CDAQMXChID::getComment

構文

```
const char * getComment(void);
```

説明

データメンバからコメント領域のコメントを取得します。

戻り値

文字列へのポインタを返します。

CDAQMXChID::getKind

構文

```
int getKind(void);
```

説明

データメンバからチャネル種類領域の値を取得します。

戻り値

チャネル種類を返します。

CDAQMXChID::getMXAlarm

構文

```
MXAlarm * getMXAlarm(int levelNo);
```

引数

levelNo　　アラームレベルを指定します。

説明

データメンバのアラーム領域から指定されたアラームレベルの構造体を取得します。存在しない場合，NULLを返します。

戻り値

構造体へのポインタを返します。

CDAQMXChID::getMXChID

構文

```
void getMXChID(MXChID * pMXChID);
```

引数

pMXChID　　チャネル識別情報の返却先を指定します。

説明

構造体でデータを取得します。データメンバの内容を指定された構造体に格納します。

参照

```
getChNo getComment getKind getPoint getRange getScale getTag
getUnit isValid
```

CDAQMXChID::getRange

構文

```
int getRange(void);
```

説明

データメンバからレンジ種類領域の値を取得します。

戻り値

レンジ種類を返します。

CDAQMXChID::getScale

構文

```
int getScale(void);
```

説明

データメンバからスケール種類領域の値を取得します。

戻り値

スケール種類を返します。

CDAQMXChID::getTag

構文

```
const char * getTag(void);
```

説明

データメンバからタグ領域のタグを取得します。

戻り値

文字列へのポインタを返します。

CDAQMXChID::getUnit

構文

```
const char * getUnit(void);
```

説明

データメンバから単位名領域の単位名を取得します。

戻り値

文字列へのポインタを返します。

CDAQMXChID::initialize

構文

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

説明

データメンバを初期化します。初期値は，原則0です。

参照

```
CDAQChInfo::initialize
```

CDAQMXChID::initMXChID

構文

```
static void initMXChID(MXChID * pMXChID);
```

引数

pMXChID チャネル識別情報の領域を指定します。

説明

指定された領域を初期化します。
初期値は，原則0です。

CDAQMXChID::isObject

構文

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

引数

classname　　クラス名を文字列で指定します。

説明

指定されたクラス名を継承しているかをチェックします。
引数が省略された場合，本クラスであるかをチェックします。
本クラスを継承したクラスが，自分自身のクラスをチェックするためには，オーバーライドする必要があります。
クラスを継承している場合，「有効値」(真)を返します。それ以外は，「無効値」(偽)を返します。
本クラスと異なる場合，親クラスをチェックします。

戻り値

有効無効値を返します。

参照

CDAQChInfo::isObject

CDAQMXChID::isValid

構文

```
int isValid(void);
```

説明

データメンバからチャネルステータス領域の値を取得します。
0なら「無効値」，それ以外は「有効値」です。

戻り値

有効無効値を返します。

CDAQMXChID::operator=

構文

```
CDAQMXChID & operator=(CDAQMXChID & cMXChID);
```

引数

cMXChID　　代入するオブジェクトを指定します。

説明

指定されたオブジェクトのデータメンバを複写します。

戻り値

本オブジェクトへの参照を返します。

参照

```
getMXChID setMXChID
```

CDAQMXChID::setAlarmValue

構文

```
void setAlarmValue(int levelNo, int iAlarmType =
DAQMX_ALARM_NONE, int valueON = 0, int valueOFF = 0);
```

引数

levelNo　　アラームレベルを指定します。
iAlarmType　　アラーム種類を指定します。
valueON　　アラーム発生のしきい値(On値)を指定します。
valueOFF　　アラーム停止のしきい値(Off値)を指定します。

説明

データメンバのアラーム領域に指定された値を格納します。
アラームレベルに，定数値の「全アラームレベル指定」をした場合，全アラームレベルに同じ値を格納します。

CDAQMXChID::setChType

構文

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

引数

chType　　チャネルタイプを指定します。

説明

データメンバのチャネルタイプ領域に指定された値を格納します。
チャネルタイプは0なので，何もしません。

CDAQMXChID::setComment

構文

```
void setComment(const char * strComment);
```

引数

strComment　　コメントを指定します。

説明

データメンバのコメント領域に指定された値を格納します。

CDAQMXChID::setMXChID

構文

```
void setMXChID(MXChID * pMXChID);
```

引数

pMXChID　　チャネル識別情報を指定します。

説明

構造体でデータを設定します。データメンバに指定された構造体の内容を格納します。

指定がない場合，データメンバは初期化されます。

参照

`initialize setChNo setComment setPoint setTag setType setUnit setValid`

CDAQMXChID::setTag

構文

```
void setTag(const char * strTag);
```

引数

strTag　　タグを指定します。

説明

データメンバのタグ領域に指定された値を格納します。

CDAQMXChID::setType

構文

```
void setType(int iKind, int iRange, int iScale =
DAQMX_SCALE_NONE);
```

引数

iKind	チャネル種類を指定します。
iRange	レンジ種類を指定します。
iScale	スケール種類を指定します。

説明

データメンバのチャネル種類領域，レンジ種類領域，スケール種類領域に指定された値を格納します。

CDAQMXChID::setUnit

構文

```
void setUnit(const char * strUnit);
```

引数

strUnit	単位名を指定します。

説明

データメンバの単位名領域に指定された値を格納します。

CDAQMXChID::setValid

構文

```
void setValid(int bValid);
```

引数

bValid	有効無効値を指定します。

説明

データメンバのチャネルステータス領域に指定された値を格納します。

CDAQMXChID::toChName

構文

```
static int toChName(int chno, int unitno = 0);
```

引数

chno	チャネル番号を指定します。
unitno	ユニット番号を指定します。

説明

指定されたチャネル番号とユニット番号からチャネル名を生成します。

戻り値

チャネル名を返します。

CDAQMXChID::toChNo

構文

```
static int toChNo(int chname);
```

引数

chname　　チャネル名を指定します。

説明

指定されたチャネル名からチャネル番号を分離します。

戻り値

チャネル番号を返します。

CDAQMXChID::toUnitNo

構文

```
static int toUnitNo(int chname);
```

引数

chname　　チャネル名を指定します。

説明

指定されたチャネル名からユニット番号を分離します。

戻り値

ユニット番号を返します。

CDAQMXChInfoクラス

- **CDAQChInfo**
  - **CDAQMXChID**
    - **CDAQMXChInfo**

本クラスは，MX100でのチャネル情報データを格納するクラスです。
MXChInfo構造体のラッパクラスになります。
基準最小値と基準最大値は，未使用です。
チャネル情報データ取得において， チャネル情報データを格納するインターフェイスとして使用するクラスです。
測定データのクラスと関連させることで， 測定値を扱いやすくできます。

パブリックメンバ

構築・消滅

CDAQMXChInfo	オブジェクトを構築します。
~CDAQMXChInfo	オブジェクトを消滅します。

構造体操作

getMXChInfo	構造体でデータを取得します。
setMXChInfo	構造体でデータを設定します。
initMXChInfo	構造体のデータを初期化します。

データメンバ操作

getFIFONo	FIFO番号を取得します。
getFIFOIndex	FIFO内チャネル順序番号を取得します。
getOriginalMin	基準最小値*を取得します。
getOriginalMax	基準最大値*を取得します。

＊ 基準最小値/最大値は，現在本APIでは使用していません。

getDisplayMin	表示最小値を取得します。
getDisplayMax	表示最大値を取得します。
getRealMin	レンジの測定可能範囲の最小値を取得します。
getRealMax	レンジの測定可能範囲の最大値を取得します。
setFIFONo	FIFO番号を設定します。
setFIFOIndex	FIFO内チャネル順序番号を設定します。

演算子

operator=	代入を実行します。

●オーバライドしたメンバ

データメンバ操作

initialize　　データメンバを初期化します。

ユーティリティ

isObject　　オブジェクトをチェックします。

●継承するメンバ

CDAQChInfo参照

```
getChNo getPoint setChNo setPoint
```

CDAQMXChID参照

```
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
```

プロテクトメンバ

データメンバ

m_FIFONo　　FIFO番号の格納領域です。
m_FIFOIndex　　FIFO内チャネル順序番号の格納領域です。
m_origMin　　基準最小値*の格納領域です。
m_origMax　　基準最大値*の格納領域です。
　* 基準最小値/最大値は，現在本APIでは使用していません。
m_dispMin　　表示最小値の格納領域です。
m_dispMax　　表示最大値の格納領域です。
m_realMin　　レンジの測定可能範囲の最小値の格納領域です。
m_realMax　　レンジの測定可能範囲の最大値の格納領域です。

●継承するメンバ

CDAQChInfo参照

```
m_chType m_chNo m_point
```

CDAQMXChID参照

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

プライベートメンバ

なし。

関数メンバ(アルファベット順)

CDAQMXChInfo::CDAQMXChInfo

構文

```
CDAQMXChInfo(MXChInfo * pMXChInfo = NULL);
virtual ~CDAQMXChInfo(void);
```

引数

pMXChInfo　　チャネル情報データを指定します。

説明

オブジェクトを構築，消滅します。
構築時，指定されたデータをデータメンバに格納します。指定がない場合，データメンバを初期化します。

参照

`setMXChInfo`

CDAQMXChInfo::getDisplayMax

構文

```
double getDisplayMax(void);
```

説明

データメンバから表示最大値領域の値を取得します。

戻り値

表示最大値を返します。

CDAQMXChInfo::getDisplayMin

構文

```
double getDisplayMin(void);
```

説明

データメンバから表示最小値領域の値を取得します。

戻り値

表示最小値を返します。

CDAQMXChInfo::getFIFOIndex

構文

```
int getFIFOIndex(void);
```

説明

データメンバからFIFO内チャネル順序番号領域の値を取得します。

戻り値

FIFO内チャネル順序番号を返します。

CDAQMXChInfo::getFIFONo

構文

```
int getFIFONo(void);
```

説明

データメンバからFIFO番号領域の値を取得します。

戻り値

FIFO番号を返します。

CDAQMXChInfo::getMXChInfo

構文

```
void getMXChInfo(MXChInfo * pMXChInfo);
```

引数

pMXChInfo　　チャネル情報データの返却先を指定します。

説明

構造体でデータを取得します。データメンバの内容を指定された構造体に格納します。

参照

```
getDisplayMax getDisplayMin getFIFOIndex getFIFONo getMXChID
getOriginalMax getOriginalMin getRealMax getRealMin
```

CDAQMXChInfo::getOriginalMax

構文

```
double getOriginalMax(void);
```

説明

データメンバから基準最大値領域の値を取得します。

* 基準最大値は，現在本APIでは使用していません。

戻り値

基準最大値を返します。

CDAQMXChInfo::getOriginalMin

構文

```
double getOriginalMin(void);
```

説明

データメンバから基準最小値領域の値を取得します。

* 基準最小値は，現在本APIでは使用していません。

戻り値

基準最小値を返します。

CDAQMXChInfo::getRealMax

構文

```
double getRealMax(void);
```

説明

データメンバからレンジの測定可能範囲最大値領域の値を取得します。

戻り値

レンジの測定可能範囲の最大値を返します。

CDAQMXChInfo::getRealMin

構文

```
double getRealMin(void);
```

説明

データメンバからレンジの測定可能範囲最小値領域の値を取得します。

戻り値

レンジの測定可能範囲の最小値を返します。

CDAQMXChInfo::initMXChInfo

構文

```
static void initMXChInfo(MXChInfo * pMXChInfo);
```

引数

pMXChInfo チャネル情報データの領域を指定します。

説明

指定された領域を初期化します。
初期値は，原則0です。

CDAQMXChInfo::initialize

構文

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

説明

データメンバを初期化します。初期値は，原則0です。

参照

`CDAQMXChID::initialize`

CDAQMXChInfo::isObject

構文

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

引数

classname クラス名を文字列で指定します。

説明

指定されたクラス名を継承しているかをチェックします。
引数が省略された場合，本クラスであるかをチェックします。
本クラスを継承したクラスが，自分自身のクラスをチェックするためには，オーバーライドする必要があります。
クラスを継承している場合，「有効値」(真)を返します。それ以外は，「無効値」(偽)を返します。
本クラスと異なる場合，親クラスをチェックします。

戻り値

有効無効値を返します。

参照

`CDAQMXChID::isObject`

CDAQMXChInfo::operator=

構文

```
CDAQMXChInfo & operator=(CDAQMXChInfo & cMXChInfo);
```

引数

cMXChInfo 代入するオブジェクトを指定します。

説明

指定されたオブジェクトのデータメンバを複写します。

戻り値

本オブジェクトへの参照を返します。

参照

`getMXChInfo setMXChInfo`

CDAQMXChInfo::setFIFOIndex

構文

```
void setFIFOIndex(int fifoIndex);
```

引数

fifoIndex FIFO内チャネル順序番号を指定します。

説明

データメンバのFIFO内チャネル順序番号領域に指定された値を格納します。

CDAQMXChInfo::setFIFONo

構文

```
void setFIFONo(int fifoNo);
```

引数

fifoNo　　FIFO番号を指定します。

説明

データメンバのFIFO番号領域に指定された値を格納します。

CDAQMXChInfo::setMXChInfo

構文

```
void setMXChInfo(MXChInfo * pMXChInfo);
```

引数

pMXChInfo　　チャネル情報データを指定します。

説明

構造体でデータを設定します。データメンバに指定された構造体の内容を格納します。

指定がない場合，データメンバは初期化されます。

参照

```
initialize setFIFOIndex setFIFONo setMXChID
```

CDAQMXConfigクラス

本クラスは，MX100での設定データを格納するクラスです。
MXConfigData構造体のラッパクラスになります。
設定データ取得において， 設定データを格納するインターフェイスとして使用するクラスです。

パブリックメンバ

構築・消滅

CDAQMXConfig	オブジェクトを構築します。
~CDAQMXConfig	オブジェクトを消滅します。

構造体操作

getMXConfigData	構造体でデータを取得します。
setMXConfigData	構造体でデータを設定します。
initMXConfigData	構造体のデータを初期化します。

データメンバ操作

initialize	データメンバを初期化します。
getClassMXSysInfo	システム構成データを取得します。
getClassMXStatus	ステータスを取得します。
getClassMXNetInfo	ネットワーク情報データを取得します。
getClassMXChConfigData	チャネル設定データを取得します。
getClassMXBalanceData	初期バランスデータを取得します。
getClassMXOutputData	出力チャネルデータを取得します。
getClassMXChConfig	個別のチャネル設定データを取得します。
reconstruct	再構築します。
setTempUnit	温度単位種類を設定します。
setDOType	DOのチャネル種類を設定します。
setInterval	周期種類を設定します。
setAOType	AOのチャネル種類を設定します。
setPWMType	PWMのチャネル種類を設定します。

レンジ設定

setSKIP	スキップ(未使用)を設定します。
setVOLT	直流電圧レンジを設定します。
setTC	熱電対レンジを設定します。
setRTD	測温抵抗体レンジを設定します。
setDI	ディジタル入力(DI)レンジを設定します。
setDELTA	チャネル間差演算を設定します。
setRRJC	リモートRJCを設定します。

setScalling	スケールを設定します。
setRES	抵抗レンジを設定します。
setSTRAIN	ひずみレンジを設定します。
setAO	AOレンジを設定します。
setPWM	PWMレンジを設定します。
setCOM	通信レンジを設定します。
setPULSE	パルスレンジを設定します。

検証

isCorrect	妥当性を検証します。

演算子

operator=	代入を実行します。

ユーティリティ

getItemError	エラー検出した設定項目番号を取得します。
isObject	オブジェクトをチェックします。
getSpanPoint	チャネルの小数点位置を取得します。
getRangePoint	レンジ種類の小数点位置を取得します。
getChName	チャネル名を取得します。
setChKind	チャネル種類を設定します。

プロテクトメンバ

データメンバ

m_cMXSysInfo	システム構成データの格納領域です。
m_cMXStatus	ステータスデータの格納領域です。
m_cMXNetInfo	ネットワーク情報データの格納領域です。
m_cMXChConfigData	チャネル設定データの格納領域です。
m_cMXBalanceData	初期バランスデータの格納領域です。
m_cMXOutputData	出力チャネルデータの格納領域です。
m_nItemError	設定項目番号の格納領域です。

プライベートメンバ

なし。

関数メンバ(アルファベット順)

CDAQMXConfig::CDAQMXConfig

構文

```
CDAQMXConfig(MXConfigData * pMXConfigData);
virtual ~CDAQMXConfig(void);
```

引数

pMXConfigData　　設定データを指定します。

説明

オブジェクトを構築，消滅します。
構築時，指定されたデータをデータメンバに格納します。指定がない場合，データメンバを初期化します。

参照

`setMXConfigData`

CDAQMXConfig::getChName

構文

```
int getChName(int chNo);
```

引数

chNo　　チャネル番号を指定します。

説明

指定されたチャネル番号とデータメンバのシステム構成データ領域のユニット番号からチャネル名を生成します。

戻り値

チャネル名を返します。

参照

`getClassMXChConfig`
`getClassMXSysInfo`
`CDAQMXChConfig::getChName`
`CDAQMXSysInfo::getUnitNo`

CDAQMXConfig::getClassMXBalanceData

構文

```
CDAQMXBalanceData & getClassMXBalanceData(void);
```

説明

データメンバから初期バランスデータ領域を取得します。

戻り値

オブジェクトへの参照を返します。

CDAQMXConfig::getClassMXChConfig

構文

```
CDAQMXChConfig * getClassMXChConfig(int chNo);
```

引数

chNo　　チャネル番号を指定します。

説明

データメンバから指定されたチャネル番号のチャネル設定データを取得します。
存在しない場合，NULLを返します。

戻り値

オブジェクトへのポインタを返します。

参照

```
getClassMXChConfigData
CDAQMXChConfigData::getClassMXChConfig
```

CDAQMXConfig::getClassMXChConfigData

構文

```
CDAQMXChConfigData & getClassMXChConfigData(void);
```

説明

データメンバからチャネル設定データ領域を取得します。

戻り値

オブジェクトへの参照を返します。

CDAQMXConfig::getClassMXNetInfo

構文

```
CDAQMXNetInfo & getClassMXNetInfo(void);
```

説明

データメンバからネットワーク情報データ領域を取得します。

戻り値

オブジェクトへの参照を返します。

CDAQMXConfig::getClassMXOutputData

構文

```
CDAQMXOutputData & getClassMXOutputData(void);
```

説明

データメンバから出力チャネルデータ領域を取得します。

戻り値

オブジェクトへの参照を返します。

CDAQMXConfig::getClassMXStatus

構文

```
CDAQMXStatus & getClassMXStatus(void);
```

説明

データメンバからステータスデータ領域を取得します。

戻り値

オブジェクトへの参照を返します。

CDAQMXConfig::getClassMXSysInfo

構文

```
CDAQMXSysInfo & getClassMXSysInfo(void);
```

説明

データメンバからシステム構成データ領域を取得します。

戻り値

オブジェクトへの参照を返します。

CDAQMXConfig::getItemError

構文

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

説明

データメンバから設定項目番号領域の値を取得します。

戻り値

設定項目番号を返します。

CDAQMXConfig::getMXConfigData

構文

```
void getMXConfigData(MXConfigData * pMXConfigData);
```

引数

pMXConfigData　　設定データの返却先を指定します。

説明

構造体でデータを取得します。

データメンバの内容を指定された構造体に格納します。

参照

```
DAQMXBalanceData::getMXBalanceData
CDAQMXOutputData::getMXOutputData
CDAQMXChConfigData::getMXChConfigData
CDAQMXNetInfo::getMXNetInfo
CDAQMXStatus::getMXStatus
CDAQMXSysInfo::getMXSystemInfo
```

CDAQMXConfig::getRangePoint

構文

```
int getRangePoint(int iRange);
```

引数

iRange　　レンジ種類を指定します。

説明

指定されたレンジ種類の小数点位置を取得します。
接点レンジの場合，接点詳細レンジを指定します。
存在しない場合，0を返します。

戻り値

小数点位置を返します。

参照

```
getClassMXSysInfo
CDAQMXChConfig::getRangePoint
CDAQMXSysInfo::getTempUnit
```

CDAQMXConfig::getSpanPoint

構文

```
int getSpanPoint(int chNo);
```

引数

chNo　　チャネル番号を指定します。

説明

指定されたチャネル番号のレンジ種類の小数点位置を取得します。
存在しない場合，0を返します。

戻り値

小数点位置を返します。

CDAQMXConfig::initialize

構文

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

説明

データメンバを初期化します。初期値は，原則0です。

参照

```
CDAQMXBalanceData::initialize
CDAQMXChConfigData::initialize
CDAQMXNetInfo::initialize
CDAQMXOutputData::initialize
CDAQMXStatus::initialize
CDAQMXSysInfo::initialize
```

CDAQMXConfig::initMXConfigData

構文

static void initMXConfigData(MXConfigData * pMXConfigData);

引数

pMXConfigData　　設定データの領域を指定します。

説明

指定された領域を初期化します。
初期値は，原則0です。

参照

CDAQMXBalanceData::initMXBalanceData
CDAQMXChConfigData::initMXChConfigData
CDAQMXNetInfo::initMXNetInfo
CDAQMXOutputData::initMXOutputData
CDAQMXStatus::initMXStatus
CDAQMXSysInfo::initMXSystemInfo

CDAQMXConfig::isCorrect

構文

```
int isCorrect(void);
```

説明

妥当性の検証をします。
システム構成データに従った各設定項目をチェックします。
不正な値を検出した場合，「無効値」を返します。
不正な値を検出した場合，データメンバの設定項目番号領域に検出した場所を示す設定項目番号を格納します。

戻り値

有効無効値を返します。

参照

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

構文

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

引数

classname　　クラス名を文字列で指定します。

説明

指定されたクラス名を継承しているかをチェックします。
引数が省略された場合，本クラスであるかをチェックします。
本クラスを継承したクラスが，自分自身のクラスをチェックするためには，オーバーライドする必要があります。
クラスを継承している場合，「有効値」(真)を返します。それ以外は，「無効値」(偽)を返します。

戻り値

有効無効値を返します。

CDAQMXConfig::operator=

構文

```
CDAQMXConfig & operator=(CDAQMXConfig & cMXConfig);
```

引数

cMXConfig　　代入するオブジェクトを指定します。

説明

指定されたオブジェクトのデータメンバを複写します。

戻り値

本オブジェクトへの参照を返します。

CDAQMXConfig::reconstruct

構文

```
void reconstruct(int bRealType);
```

引数

bRealType　実際のモジュール種類を設定するか否かを，有効無効値で指定します。

説明

再構築します。
指定された値が有効の場合，実際のモジュール種類に従って設定データを作成します。
指定された値が無効の場合，現在のモジュール種類に従って設定データを作成します。
各設定値は既定値になります。

参照

```
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

構文

```
void setAO(int chNo, int iRangeAO, int spanMin = 0, int
spanMax = 0);
```

引数

chNo	チャネル番号を指定します。
iRangeAO	AOレンジのレンジ種類を指定します。
spanMin	スパン最小値を指定します。
spanMax	スパン最大値を指定します。

説明

指定されたチャネル番号のチャネルに指定されたレンジを設定します。
最大，最小値が同じ場合，スパンは省略とみなします。
存在しない，または，対応モジュール上のチャネルでない場合，設定しません。
チャネル番号に定数値の「全チャネル番号指定」を指定すると，全チャネルを処理します。
出力チャネルデータの出力種類を指定されたレンジに合わせてリセットします。

参照

```
getClassMXChConfig getClassMXOutputData getClassMXSysInfo
CDAQMXChConfig::setAO
CDAQMXChConfig::setSpan
CDAQMXOutputData::setOutputType
CDAQMXSysInfo::getModuleType
CDAQMXSysInfo::getTempUnit
```

CDAQMXConfig::setAOType

構文

```
void setAOType(int aoNo, int iKind, int refChNo =
DAQMX_REFCHNO_NONE);
```

引数

aoNo	AOデータ番号を指定します。
iKind	AOの種類をチャネル種類で指定します。
refChNo	参照するチャネル番号を指定します。

説明

AOモジュールのチャネルの種類を設定します。
チャネルの種類に「未使用」を指定した場合，スキップ(未使用)に設定されます。
チャネルの種類に「AO(伝送出力)」を指定した場合，参照チャネルに入力チャネルのチャネル番号を指定してください。
存在しない，AOモジュール上のチャネルでない，または，AOの種類でないものを指定した場合，設定しません。
AOデータ番号に定数値の「全AO/PWM番号指定」を指定すると，全AOチャネルを処理します。
チャネルの各設定項目は，出力チャネルデータの出力種類に合わせてリセットされます。

参照

```
getClassMXChConfig getClassMXOutputData getClassMXSysInfo
setSKIP
CDAQMXChConfig::setAO
CDAQMXChConfig::setRefChNo
CDAQMXChConfig::setType
CDAQMXChConfig::setValid
CDAQMXOutputData::getOutputType
CDAQMXSysInfo::getModuleType
```

CDAQMXConfig::setChKind

構文

```
void setChKind(int chNo, int iKind, int refChNo =
DAQMX_REFCHNO_NONE);
```

引数

chNo	チャネル番号を指定します。
iKind	チャネル種類を指定します。
refChNo	参照するチャネル番号を指定します。

説明

指定されたチャネル番号のチャネルにチャネルの種類を設定します。
チャネルの種類が「AI(チャネル間差)」，「DI(チャネル間差)」，「AI(リモートRJC)」，「AO(伝送出力)」，「PWM(伝送出力)」の場合，参照チャネルの指定が有効になります。
各チャネルの設定項目は既定値になります。

参照

`setAOType setDELTA setDI setDOType setPWMType setRRJC setSKIP setVOLT`

CDAQMXConfig::setCOM

構文

```
void setCOM(int chNo, int iRangeCOM, int spanMin = 0, int
spanMax = 0);
```

引数

chNo	チャネル番号を指定します。
iRangeCOM	レンジ種類から通信レンジを指定します。
spanMin	スパン最小値を指定します。
spanMax	スパン最大値を指定します。

説明

指定されたチャネル番号のチャネルに指定されたレンジを設定します。
最大，最小値が同じ場合，スパンは省略とみなします。
存在しない，または，対応モジュール上のチャネルでない場合，設定しません。
チャネル番号に定数値の「全チャネル番号指定」を指定すると，全チャネルを処理します。

参照

`getClassMXChConfig getClassMXSysInfo CDAQMXChConfig::setCOM CDAQMXChConfig::setSpan CDAQMXSysInfo::getModuleType CDAQMXSysInfo::getTempUnit`

CDAQMXConfig::setDELTA

構文

```
void setDELTA(int chNo, int refChNo, int spanMin = 0, int
spanMax = 0, int iRange = DAQMX_RANGE_REFERENCE);
```

引数

chNo	チャネル番号を指定します。
refChNo	参照するチャネル番号を指定します。
spanMin	スパン最小値を指定します。
spanMax	スパン最大値を指定します。
iRange	レンジ種類を指定します。

説明

指定されたチャネル番号のチャネルに指定されたレンジを設定します。
最大，最小値が同じ場合，スパンは省略とみなします。
存在しない，または，対応モジュール上のチャネルでない場合，設定しません。
レンジ種類に「参照チャネル」を指定した場合，自チャネルに参照するチャネル番号のレンジを適用します。
チャネル番号に定数値の「全チャネル番号指定」を指定すると， 全チャネルを処理します。
レンジ種類がひずみの場合， 初期バランスデータをリセットします。
参照チャネル番号と同じチャネル番号のチャネルは，処理されません(R3.01以降)。

参照

```
getClassMXBalanceData getClassMXChConfig getClassMXSysInfo
CDAQMXBalanceData::setBalance CDAQMXChConfig::getRange
CDAQMXChConfig::isValid CDAQMXChConfig::setDELTA
CDAQMXChConfig::setSpan CDAQMXSysInfo::getModuleType
CDAQMXSysInfo::getTempUnit
```

CDAQMXConfig::setDI

構文

```
void setDI(int chNo, int iRangeDI, int spanMin = 0, int
spanMax = 0);
```

引数

chNo	チャネル番号を指定します。
iRangeDI	ディジタル入力(DI)のレンジ種類を指定します。
spanMin	スパン最小値を指定します。
spanMax	スパン最大値を指定します。

説明

指定されたチャネル番号のチャネルに指定されたレンジを設定します。
最大，最小値が同じ場合，スパンは省略とみなします。
存在しない，または，対応モジュール上のチャネルでない場合，設定しません。
ディジタル入力(DI)のレンジ種類を，ディジタル入力(DI)詳細レンジにて設定します。
チャネル番号に定数値の「全チャネル番号指定」を指定すると，全チャネルを処理します。

参照

```
getClassMXChConfig getClassMXSysInfo
CDAQMXChConfig::setDI CDAQMXChConfig::setSpan
CDAQMXSysInfo::getModuleType CDAQMXSysInfo::getTempUnit
```

CDAQMXConfig::setDOType

構文

```
void setDOType(int doNo, int iKind, int bDeenergize =
DAQMX_VALID_OFF, int bHold = DAQMX_VALID_OFF);
```

引数

doNo	DOデータ番号を指定します。
iKind	DOの種類をチャネル種類で指定します。
bDeenergize	非励磁動作を有効無効値で指定します。
bHold	保持動作を有効無効値で指定します。

説明

DOモジュールのチャネルの種類を設定します。
チャネル種類に「未使用」を指定した場合，スキップ(未使用)に設定されます。
存在しない，DOモジュール上のチャネルでない，または，DOの種類でないものを指定した場合，設定しません。
DOデータの番号に定数値の「全DO番号指定」とすると全チャネルを処理します。

参照

```
getClassMXChConfig setSKIP
CDAQMXChConfig::setDeenergize CDAQMXChConfig::setHold
CDAQMXChConfig::setType CDAQMXChConfig::setValid
CDAQMXSysInfo::getModuleType
```

CDAQMXConfig::setInterval

構文

```
void setInterval(int moduleNo, int iInterval, int iHz =
DAQMX_INTEGRAL_AUTO);
```

引数

moduleNo	モジュール番号を指定します。
iInterval	周期種類を指定します。
iHz	A/D積分時間種類を指定します。

説明

指定されたモジュール番号のモジュールに指定された値を設定します。
存在しない場合，設定しません。
モジュール番号に定数値の「全モジュール番号指定」を指定すると，全モジュールを処理します。

参照

```
CDAQMXSysInfo::getChNum CDAQMXSysInfo::getModuleType
CDAQMXSysInfo::setModule
```

CDAQMXConfig::setMXConfigData

構文

```
void setMXConfigData(MXConfigData * pMXConfigData);
```

引数

pMXConfigData 設定データを指定します。

説明

構造体でデータを設定します。データメンバに指定された構造体の内容を格納します。
指定がない場合，データメンバは初期化されます。

参照

```
initialize
CDAQMXBalanceData::setMXBalanceData
CDAQMXChConfigData::setMXChConfigData
CDAQMXNetInfo::setMXNetInfo
CDAQMXOutputData::setMXOutputData
CDAQMXStatus::setMXStatus
CDAQMXSysInfo::setMXSystemInfo
```

CDAQMXConfig::setPULSE

構文

```
void setPULSE(int chNo, int iRangePULSE, int spanMin = 0, int
spanMax = 0);
```

引数

chNo	チャネル番号を指定します。
iRangePULSE	レンジ種類からパルスレンジを指定します。
spanMin	スパン最小値を指定します。
spanMax	スパン最大値を指定します。

説明

指定されたチャネル番号のチャネルに指定されたレンジを設定します。
最大，最小値が同じ場合，スパンは省略とみなします。
存在しない，または，対応モジュール上のチャネルでない場合，設定しません。
チャネル番号に定数値の「全チャネル番号指定」を指定すると，全チャネルを処理します。

参照

```
getClassMXChConfig getClassMXSysInfo
CDAQMXChConfig::setPULSE CDAQMXChConfig::setSpan
CDAQMXSysInfo::getModuleType CDAQMXSysInfo::getTempUnit
```

CDAQMXConfig::setPWM

構文

```
void setPWM(int chNoÅCint iRangePWMÅCint spanMin = 0ÅCint
spanMax = 0);
```

引数

chNo	チャネル番号を指定します。
iRangePWM	PWMレンジのレンジ種類を指定します。
spanMin	スパン最小値を指定します。
spanMax	スパン最大値を指定します。

説明

指定されたチャネル番号のチャネルに指定されたレンジを設定します。
最大，最小値が同じ場合，スパンは省略とみなします。
存在しない，または，対応モジュール上のチャネルでない場合，設定しません。
チャネル番号に定数値の「全チャネル番号指定」を指定すると，全チャネルを処理します。
出力チャネルデータの出力種類を指定されたレンジに合わせてリセットします。

参照

```
getClassMXChConfig getClassMXOutputData getClassMXSysInfo
CDAQMXChConfig::setSpan CDAQMXChConfig::setPWM
CDAQMXOutputData::setOutputType CDAQMXSysInfo::getModuleType
CDAQMXSysInfo::getTempUnit
```

CDAQMXConfig::setPWMType

構文

```
void setPWMType(int pwmNo, int iKind, int refChNo =
DAQMX_REFCHNO_NONE);
```

引数

pwmNo	PWMデータ番号を指定します。
iKind	PWMの種類をチャネル種類で指定します。
refChNo	参照するチャネル番号を指定します。

説明

PWMモジュールのチャネルの種類を設定します。
チャネルの種類に「未使用」を指定した場合，スキップ(未使用)に設定されます。
チャネルの種類に「PWM(伝送出力)」を指定した場合，参照チャネルに入力チャネルのチャネル番号を指定してください。
存在しない，PWMモジュール上のチャネルでない，または，PWMの種類でないものを指定した場合，設定しません。
PWMデータ番号に定数値の「全AO/PWM番号指定」を指定すると，全PWMチャネルを処理します。
チャネルの各設定項目は，出力チャネルデータの出力種類に合わせてリセットされます。

参照

```
getClassMXChConfig getClassMXOutputData getClassMXSysInfo
setSKIP
CDAQMXChConfig::setPWM
CDAQMXChConfig::setRefChNo
CDAQMXChConfig::setType
CDAQMXChConfig::setValid
CDAQMXOutputData::getOutputType
CDAQMXSysInfo::getModuleType
```

CDAQMXConfig::setRES

構文

```
void setRES(int chNo, int iRangeRES, int spanMin = 0, int
spanMax = 0);
```

引数

chNo	チャネル番号を指定します。
iRangeRES	抵抗レンジのレンジ種類を指定します。
spanMin	スパン最小値を指定します。
spanMax	スパン最大値を指定します。

説明

指定されたチャネル番号のチャネルに指定されたレンジを設定します。
最大，最小値が同じ場合，スパンは省略とみなします。
存在しない，または，対応モジュール上のチャネルでない場合，設定しません。
チャネル番号に定数値の「全チャネル番号指定」を指定すると，全チャネルを処理します。

参照

```
getClassMXChConfig getClassMXSysInfo
CDAQMXChConfig::setRES CDAQMXChConfig::setSpan
CDAQMXSysInfo::getModuleType CDAQMXSysInfo::getTempUnit
```

CDAQMXConfig::setRRJC

構文

```
void setRRJC(int chNo, int refChNo, int spanMin = 0, int
spanMax = 0);
```

引数

chNo	チャネル番号を指定します。
refChNo	参照するチャネル番号を指定します。
spanMin	スパン最小値を指定します。
spanMax	スパン最大値を指定します。

説明

指定されたチャネル番号のチャネルに指定されたレンジを設定します。
最大，最小値が同じ場合，スパンは省略とみなします。
存在しない，レンジが熱電対レンジでない，または，対応モジュール上のチャネルでない場合，設定しません。
チャネル番号に定数値の「全チャネル番号指定」を指定すると，全チャネルを処理します。

参照

```
getClassMXChConfig getClassMXChConfigData getClassMXSysInfo
CDAQMXChConfig::getRange CDAQMXChConfig::isValid
CDAQMXChConfig::setSpan CDAQMXChConfigData::setRRJC
CDAQMXSysInfo::getModuleType CDAQMXSysInfo::getTempUnit
```

CDAQMXConfig::setRTD

構文

```
void setRTD(int chNo, int iRangeRTD, int spanMin = 0, int
spanMax = 0);
```

引数

chNo	チャネル番号を指定します。
iRangeRTD	測温抵抗体レンジのレンジ種類を指定します。
spanMin	スパン最小値を指定します。
spanMax	スパン最大値を指定します。

説明

指定されたチャネル番号のチャネルに指定されたレンジを設定します。
最大，最小値が同じ場合，スパンは省略とみなします。
存在しない，または，対応モジュール上のチャネルでない場合，設定しません。
チャネル番号に定数値の「全チャネル番号指定」を指定すると， 全チャネルを処理します。

参照

`getClassMXChConfig getClassMXSysInfo CDAQMXChConfig::setRTD CDAQMXChConfig::setSpan CDAQMXSysInfo::getModuleType CDAQMXSysInfo::getTempUnit`

CDAQMXConfig::setScalling

構文

```
void setScalling(int chNo, int scaleMin = 0, int scaleMax = 0,
int scalePoint = 0);
```

引数

chNo	チャネル番号を指定します。
scaleMin	スケール最小値を指定します。
scaleMax	スケール最大値を指定します。
scalePoint	小数点位置を指定します。

説明

指定されたチャネル番号のチャネルにスケールを設定します。
スケール種類を「線形」に設定します。 最大，最小値が同じ場合，「なし」に設定します。
チャネル番号に定数値の「全チャネル番号指定」を指定すると， 全チャネルを処理します。

参照

`getClassMXChConfig getClassMXSysInfo CDAQMXChConfig::setScalling CDAQMXSysInfo::getTempUnit`

CDAQMXConfig::setSKIP

構文

```
void setSKIP(int chNo);
```

引数

chNo　　チャネル番号を指定します。

説明

指定されたチャネル番号のチャネルにスキップ(未使用)を設定します。
存在しない場合，設定しません。
チャネル番号に定数値の「全チャネル番号指定」を指定すると， 全チャネルを処理します。

参照

```
getClassMXChConfig CDAQMXChConfig::setSKIP
```

CDAQMXConfig::setSTRAIN

構文

```
void setSTRAIN(int chNo, int iRangeSTRAIN, int spanMin = 0,
int spanMax = 0);
```

引数

chNo　　チャネル番号を指定します。
iRangeSTRAIN　ひずみレンジのレンジ種類を指定します。
spanMin　　スパン最小値を指定します。
spanMax　　スパン最大値を指定します。

説明

指定されたチャネル番号のチャネルに指定されたレンジを設定します。
最大，最小値が同じ場合，スパンは省略とみなします。
存在しない，または，対応モジュール上のチャネルでない場合，設定しません。
チャネル番号に定数値の「全チャネル番号指定」を指定すると，全チャネルを処理します。
初期バランスデータをリセットします。

参照

```
getClassMXBalanceData
getClassMXChConfig
getClassMXSysInfo
CDAQMXBalanceData::setBalance
CDAQMXChConfig::setSpan
CDAQMXChConfig::setSTRAIN
CDAQMXSysInfo::getModuleType
CDAQMXSysInfo::getTempUnit
```

CDAQMXConfig::setTC

構文

```
void setTC(int chNo, int iRangeTC, int spanMin = 0, int
spanMax = 0);
```

引数

chNo	チャネル番号を指定します。
iRangeTC	熱電対レンジのレンジ種類を指定します。
spanMin	スパン最小値を指定します。
spanMax	スパン最大値を指定します。

説明

指定されたチャネル番号のチャネルに指定されたレンジを設定します。
最大，最小値が同じ場合，スパンは省略とみなします。
存在しない，または，対応モジュール上のチャネルでない場合，設定しません。
チャネル番号に定数値の「全チャネル番号指定」を指定すると， 全チャネルを処理します。

参照

`getClassMXChConfig getClassMXSysInfo CDAQMXChConfig::setSpan CDAQMXChConfig::setTC CDAQMXSysInfo::getModuleType CDAQMXSysInfo::getTempUnit`

CDAQMXConfig::setTempUnit

構文

```
void setTempUnit(int iTempUnit);
```

引数

iTempUnit	温度単位種類を指定します。

説明

設定データの温度単位種類を変更します。
システム構成データ領域に指定された値を格納します。
影響を受けるチャネルのレンジを変更します。

参照

`CDAQMXChConfigData::changeRange CDAQMXSysInfo::setTempUnit`

CDAQMXConfig::setVOLT

構文

void setVOLT(int chNo，int iRangeVOLT，int spanMin = 0，int spanMax = 0);

引数

chNo	チャネル番号を指定します。
iRangeVOLT	直流電圧レンジのレンジ種類を指定します。
spanMin	スパン最小値を指定します。
spanMax	スパン最大値を指定します。

説明

指定されたチャネル番号のチャネルに指定されたレンジを設定します。
最大，最小値が同じ場合，スパンは省略とみなします。
存在しない，または，対応モジュール上のチャネルでない場合，設定しません。
チャネル番号に定数値の「全チャネル番号指定」を指定すると，全チャネルを処理します。

参照

`getClassMXChConfig getClassMXSysInfo`
`CDAQMXChConfig::setSpan`
`CDAQMXChConfig::setVOLT`
`CDAQMXSysInfo::getModuleType`
`CDAQMXSysInfo::getTempUnit`

CDAQMXDataInfoクラス

■ CDAQDataInfo
・ CDAQMXDataInfo

本クラスは，MX100での測定データを格納するクラスです。
MXDataInfo構造体のラッパクラスになります。
測定データ取得において，測定値を格納するインターフェイスとして使用するクラスです。
チャネル情報データのクラスと関連させることで，実際の測定値を算出することができます。

パブリックメンバ

構築・消滅

CDAQMXDataInfo	オブジェクトを構築します。
~CDAQMXDataInfo	オブジェクトを消滅します。

構造体操作

getMXDataInfo	構造体でデータを取得する。
setMXDataInfo	構造体でデータを設定する。
initMXDataInfo	構造体のデータを初期化します

データメンバ操作

getStatus	データステータスを取得します。
isAlarm	アラームを取得します。
setStatus	データステータスを設定します。
setAlarm	アラームを設定します。

関連付け

getClassMXChInfo	チャネル情報データとの関連を取得します。
setClassMXChInfo	チャネル情報データとの関連を設定します。

ユーティリティ

getAlarmName	アラーム種類の名称を取得します。

演算子

operator=	代入を実行します。

●オーバライドしたメンバ

データメンバ操作

initialize	データメンバを初期化します。

ユーティリティ

isObject	オブジェクトをチェックします。

●継承するメンバ

CDAQDataInfo参照

```
getClassChInfo getDoubleValue getStringValue getValue
setClassChInfo setValue toDoubleValue toStringValue
```

プロテクトメンバ

データメンバ

m_dataStatus	データステータスの格納領域です。
m_alarm	アラーム有無の格納領域です。

●継承するメンバ

CDAQDataInfo参照

```
m_pChInfo m_value
```

プライベートメンバ

なし。

関数メンバ(アルファベット順)

CDAQMXDataInfo::CDAQMXDataInfo

構文

```
CDAQMXDataInfo(MXDataInfo * pMXDataInfo = NULL, CDAQMXChInfo *
pcMXChInfo = NULL);
virtual ~CDAQMXDataInfo(void);
```

引数

pMXDataInfo	測定データを指定します。
pcMXChInfo	チャネル情報データとの関連を指定します。

説明

オブジェクトを構築，消滅します。

構築時，指定されたデータをデータメンバに格納します。指定がない場合，データメンバを初期化します。

参照

```
setClassMXChInfo setMXDataInfo
```

CDAQMXDataInfo::getAlarmName

構文

```
static const char * getAlarmName(int iAlarmType);
```

引数

iAlarmType　　アラーム種類を指定します。

説明

指定されたアラーム種類に対応する文字列を取得します。
範囲外の場合，「アラームなし」と同じ文字列になります。

戻り値

文字列へのポインタを返します。

CDAQMXDataInfo::getClassMXChInfo

構文

```
CDAQMXChInfo * getClassMXChInfo(void);
```

説明

データメンバのチャネル情報データとの関連を取得します。
設定されていない場合，NULLを返します。

戻り値

チャネル情報データとの関連を返します。

参照

```
getClassChInfo
```

CDAQMXDataInfo::getMXDataInfo

構文

```
void getMXDataInfo(MXDataInfo * pMXDataInfo);
```

引数

pMXDataInfo　測定データの返却先を指定します。

説明

構造体でデータを取得します。
データメンバの内容を指定された構造体に格納します。

参照

```
getStatus getValue
```

CDAQMXDataInfo::getStatus

構文

```
int getStatus(void);
```

説明

データメンバのデータステータス領域の値を取得します。

戻り値

データステータスを返します。

CDAQMXDataInfo::initialize

構文

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

説明

データメンバを初期化します。
初期値は，原則0です。
チャネル情報データとの関連は初期化しません。

参照

```
CDAQDataInfo::initialize
```

CDAQMXDataInfo::initMXDataInfo

構文

```
static void initMXDataInfo(MXDataInfo * pMXDataInfo);
```

引数

pMXDataInfo 測定データの領域を指定します。

説明

指定された領域を初期化します。
初期値は，原則0です。

CDAQMXDataInfo::isAlarm

構文

```
int isAlarm(int levelNo);
```

引数

levelNo アラームレベルを指定します。

説明

データメンバのアラーム有無領域の値を取得します。
指定されたアラームレベルに対応する値を返します。アラームレベルが範囲外の場合，「無効値」(OFF)を返します。

戻り値

有効無効値を返します。

CDAQMXDataInfo::isObject

構文

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

引数

classname　クラス名を文字列で指定します。

説明

指定されたクラス名を継承しているかをチェックします。
引数が省略された場合，本クラスであるかをチェックします。
本クラスを継承したクラスが，自分自身のクラスをチェックするためには，オーバーライドする必要があります。
クラスを継承している場合，「有効値」(真)を返します。それ以外は，「無効値」(偽)を返します。
本クラスと異なる場合，親クラスをチェックします。

戻り値

有効無効値を返します。

参照

CDAQDataInfo::isObject

CDAQMXDataInfo::operator=

構文

```
CDAQMXDataInfo & operator=(CDAQMXDataInfo & cMXDataInfo);
```

引数

cMXDataInfo　代入するオブジェクトを指定します。

説明

指定されたオブジェクトのデータメンバを複写します。
チャネル情報データとの関連も複写されます。

戻り値

本オブジェクトへの参照を返します。

参照

getClassMXChInfo getMXDataInfo setClassMXChInfo setMXDataInfo

CDAQMXDataInfo::setAlarm

構文

```
void setAlarm(int levelNo, int bValid);
```

引数

levelNo　　アラームレベルを指定します。
bValid　　有効無効値を指定します。

説明

データメンバのアラーム有無領域に指定された値を設定します。
アラームレベルが範囲外の場合，設定されません。

CDAQMXDataInfo::setClassMXChInfo

構文

```
void setClassMXChInfo(CDAQMXChInfo * pcMXChInfo);
```

引数

pcMXChInfo　　チャネル情報データとの関連を指定します。

説明

データメンバのチャネル情報データとの関連領域に指定された値を格納します。

参照

```
setClassChInfo
```

CDAQMXDataInfo::setMXDataInfo

構文

```
void setMXDataInfo(MXDataInfo * pMXDataInfo);
```

引数

pMXDataInfo　　測定データを指定します。

説明

構造体でデータを設定します。
データメンバに指定された構造体の内容を格納します。
指定がない場合，データメンバは初期化されます。

参照

```
initialize setStatus setValue
```

CDAQMXDataInfo::setStatus

構文

```
void setStatus(int iDataStatus);
```

引数

iDataStatus　　データステータスを指定します。

説明

データメンバのデータステータス領域に指定された値を格納します。

CDAQMXDateTimeクラス

■ CDAQDateTime
・ CDAQMXDateTime

本クラスは，MX100での時刻情報データを格納するクラスです。
MXDateTime構造体のラッパクラスになります。
測定データの取得において， 時刻情報データを取得する場合に， 時刻情報データを格納するインターフェイスとして使用するクラスです。

パブリックメンバ

構築・消滅

CDAQMXDateTime	オブジェクトを構築します。
~CDAQMXDateTime	オブジェクトを消滅します。

構造体操作

getMXDateTime	構造体でデータを取得します。
setMXDateTime	構造体でデータを設定します。
initMXDateTime	構造体のデータを初期化します。

演算子

operator=	代入を実行します。

●オーバライドしたメンバ

ユーティリティ

isObject	オブジェクトをチェックします。

●継承するメンバ

CDAQDateTime参照

```
getMilliSecond getTime initialize setMilliSecond setNow
setTime toLocalDateTime
```

プロテクトメンバ

●継承するメンバ

CDAQDateTime参照

```
m_milliSecond m_time
```

プライベートメンバ

なし。

関数メンバ(アルファベット順)

CDAQMXDateTime::CDAQMXDateTime

構文

```
CDAQMXDateTime(time_t time = 0, int milliSecond = 0);
CDAQMXDateTime(MXDateTime * pMXDateTime);
virtual ~CDAQMXDateTime(void);
```

引数

time	秒数を指定します。
milliSecond	ミリ秒を指定します。
pMXDateTime	時刻情報データを指定します。

説明

オブジェクトを構築，消滅します。
構築時，データメンバに指定された値を設定します。指定がない場合，データメンバを初期化します。

参照

`setMXDateTime`

CDAQMXDateTime::getMXDateTime

構文

```
void getMXDateTime(MXDateTime * pMXDateTime);
```

引数

pMXDateTime	時刻情報データの返却先を指定します。

説明

構造体でデータを取得します。
データメンバの内容を指定された構造体に格納します。

CDAQMXDateTime::initMXDateTime

構文

```
static void initMXDateTime(MXDateTime * pMXDateTime);
```

引数

pMXDateTime　時刻情報データの領域を指定します。

説明

指定された領域を初期化します。
初期値は，原則0です。

CDAQMXDateTime::isObject

構文

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

引数

classname　　クラス名を文字列で指定します。

説明

指定されたクラス名を継承しているかをチェックします。
引数が省略された場合，本クラスであるかをチェックします。
本クラスを継承したクラスが，自分自身のクラスをチェックするためには，オーバーライドする必要があります。
クラスを継承している場合，「有効値」(真)を返します。それ以外は，「無効値」(偽)を返します。
本クラスと異なる場合，親クラスをチェックします。

戻り値

有効無効値を返します。

参照

CDAQDateTime::isObject

CDAQMXDateTime::operator=

構文

```
CDAQMXDateTime & operator=(CDAQMXDateTime & cMXDateTime);
```

引数

cMXDateTime　代入するオブジェクトを指定します。

説明

指定されたオブジェクトのデータメンバを複写します。

戻り値

本オブジェクトへの参照を返します。

CDAQMXDateTime::setMXDateTime

構文

```
void setMXDateTime(MXDateTime * pMXDateTime);
```

引数

pMXDateTime　時刻情報データを指定します。

説明

構造体でデータを設定します。
データメンバに指定された構造体の内容を格納します。
指定がない場合，データメンバは初期化されます。

参照

`initialize`

CDAQMXDODataクラス

本クラスは，MX100でのDOデータを格納するクラスです。
MXDOData構造体のラッパクラスになります。
全チャネル分のDOデータをまとめたものです。
DOデータの取得，設定において， DOデータを格納するインターフェイスとして使用するクラスです。

パブリックメンバ

構築・消滅

CDAQMXDOData	オブジェクトを構築します。
~CDAQMXDOData	オブジェクトを消滅します。

構造体操作

getMXDOData	構造体でデータを取得します。
setMXDOData	構造体でデータを設定します。
initMXDOData	構造体のデータを初期化します。

データメンバ操作

initialize	データメンバを初期化します。
getDOValid	有効／無効を取得します。
getDOONOFF	ON/OFFを取得します。
setDO	DOデータを設定します。
setDOONOFF	ON/OFFを設定します。

演算子

operator=	代入を実行します。

ユーティリティ

isObject	オブジェクトをチェックします。

プロテクトメンバ

データメンバ

m_MXDOData	DOデータの格納領域です。

メンバアクセス

getMXDO	チャネルごとのDOデータの構造体を取得します。

プライベートメンバ

なし。

関数メンバ(アルファベット順)

CDAQMXDOData::CDAQMXDOData

構文

```
CDAQMXDOData(MXDOData * pMXDOData = NULL);
virtual ~CDAQMXDOData(void);
```

引数

pMXDOData　DOデータを指定します。

説明

オブジェクトを構築，消滅します。
構築時，データメンバに指定された値を設定します。指定がない場合，データメンバを初期化します。

参照

setMXDOData

CDAQMXDOData::getDOONOFF

構文

```
int getDOONOFF(int doNo);
```

引数

doNo　DOデータ番号を指定します。

説明

データメンバのDOデータ領域から指定されたDOデータ番号の示すON/OFFの値を取得します。
存在しない場合，「無効」を返します。

戻り値

有効無効値を返します。

参照

getMXDO

CDAQMXDOData::getDOValid

構文

```
int getDOValid(int doNo);
```

引数

doNo　　DOデータ番号を指定します。

説明

データメンバのDOデータ領域から指定されたDOデータ番号の示す有効/無効の値を取得します。
存在しない場合，「無効」を返します。

戻り値

有効無効値を返します。

参照

`getMXDO`

CDAQMXDOData::getMXDO

構文

```
MXDO * getMXDO(int doNo);
```

引数

doNo　　DOデータ番号を指定します。

説明

データメンバのDOデータ領域から指定されたDOデータ番号の示す構造体を取得します。
存在しない場合，NULLを返します。

戻り値

構造体へのポインタを返します。

CDAQMXDOData::getMXDOData

構文

```
void getMXDOData(MXDOData * pMXDOData);
```

引数

pMXDOData　DOデータの返却先を指定します。

説明

構造体でデータを取得します。
データメンバの内容を指定された構造体に格納します。

CDAQMXDOData::initialize

構文

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

説明

データメンバを初期化します。
初期値は，原則0です。

参照

```
initMXDOData
```

CDAQMXDOData::initMXDOData

構文

```
static void initMXDOData(MXDOData * pMXDOData);
```

引数

pMXDOData　DOデータの領域を指定します。

説明

指定された領域を初期化します。
初期値は，原則0です。

CDAQMXDOData::isObject

構文

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

引数

classname　クラス名を文字列で指定します。

説明

指定されたクラス名を継承しているかをチェックします。
省略された場合，本クラスであるかをチェックします。
本クラスを継承したクラスが，自分自身のクラスをチェックするためには，オーバーライドする必要があります。
クラスを継承している場合，「有効値」(真)を返します。それ以外は，「無効値」(偽)を返します。

戻り値

有効無効値を返します。

CDAQMXDOData::operator=

構文

```
CDAQMXDOData & operator=(CDAQMXDOData & cMXDOData);
```

引数

cMXDOData　代入するオブジェクトを指定します。

説明

指定されたオブジェクトのデータメンバを複写します。

戻り値

本オブジェクトへの参照を返します。

CDAQMXDOData::setDO

構文

```
void setDO(int doNo, int bValid, int bONOFF =
DAQMX_VALID_OFF);
```

引数

doNo	DOデータ番号を指定します。
bValid	有効／無効を有効無効値で指定します。
bONOFF	ON/OFFを有効無効値で指定します。

説明

データメンバのDOデータ領域の指定されたDOデータ番号の示す領域に指定された値を格納します。

DOデータ番号に定数値の「全DO番号指定」を指定した場合，全DOデータに値を格納します。

参照

```
getMXDO
```

CDAQMXDOData::setDOONOFF

構文

```
void setDOONOFF(int bONOFF);
```

引数

bONOFF　ON/OFFを有効無効値で指定します。

説明

データメンバのDOデータ領域で，「有効」になっているDOデータ番号の全てのDOデータのON/OFFを指定された値に変更します。

参照

```
getMXDO
```

CDAQMXDOData::setMXDOData

構文

```
void setMXDOData(MXDOData * pMXDOData);
```

引数

pMXDOData　DOデータを指定します。

説明

構造体でデータを設定します。データメンバに指定された構造体の内容を格納します。

指定がない場合，データメンバは初期化されます。

参照

`initialize`

CDAQMXNetInfoクラス

本クラスは，MX100でのネットワーク情報データを格納するクラスです。
MXNetInfo構造体のラッパクラスになります。
設定データ取得において， ネットワーク情報データを格納するインターフェイスとして使用するクラスです。

パブリックメンバ

構築・消滅

CDAQMXNetInfo	オブジェクトを構築します。
~CDAQMXNetInfo	オブジェクトを消滅します。

構造体操作

getMXNetInfo	構造体でデータを取得します。
setMXNetInfo	構造体でデータを設定します。
initMXNetInfo	構造体のデータを初期化します。

データメンバ操作

initialize	データメンバを初期化します。
getAddress	IPアドレスを取得します。
getPort	ポート番号を取得します。
getSubMask	サブネットマスクを取得します。
getGateway	GATEWAYアドレスを取得します。
getHost	ホスト名を取得します。

演算子

operator=	代入を実行します。

ユーティリティ

getPart	IPアドレスのパート分割を取得します。
isObject	オブジェクトをチェックします。

プロテクトメンバ

データメンバ

m_MXNetInfo	ネットワーク情報データの格納領域です。

プライベートメンバ

なし。

関数メンバ(アルファベット順)

CDAQMXNetInfo::CDAQMXNetInfo

構文

```
CDAQMXNetInfo(MXNetInfo * pMXNetInfo);
virtual ~CDAQMXNetInfo(void);
```

引数

pMXNetInfo　　ネットワーク情報データを指定します。

説明

オブジェクトを構築，消滅します。
構築時，データメンバに指定された値を設定します。指定がない場合，データメンバを初期化します。

参照

`setMXNetInfo`

CDAQMXNetInfo::getAddress

構文

```
unsigned int getAddress(void);
```

説明

データメンバのネットワーク情報データ領域からIPアドレスの値を取得します。

戻り値

IPアドレスを返します。

CDAQMXNetInfo::getGateway

構文

```
unsigned int getGateway(void);
```

説明

データメンバのネットワーク情報データ領域からGATEWAYアドレスの値を取得します。

戻り値

GATEWAYアドレスを返します。

CDAQMXNetInfo::getHost

構文

```
const char * getHost(void);
```

説明

データメンバのネットワーク情報データ領域からホスト名を取得します。

戻り値

文字列へのポインタを返します。

CDAQMXNetInfo::getMXNetInfo

構文

```
void getMXNetInfo(MXNetInfo * pMXNetInfo);
```

引数

pMXNetInfo　　ネットワーク情報データの返却先を指定します。

説明

構造体でデータを取得します。
データメンバの内容を指定された構造体に格納します。

CDAQMXNetInfo::getPart

構文

```
static int getPart(unsigned int address, int index);
```

引数

address　　IPアドレスを指定します。
index　　パート位置を指定します。

説明

指定されたIPアドレスをパート位置で分割したバイト値を取得します。
指定されたパート位置のバイト値を返します。
パート位置は，バイト単位のインデックス値(0から)で指定します。範囲は，0から3です。
存在しない場合，0を返します。

戻り値

バイト値を返します。

CDAQMXNetInfo::getPort

構文

```
unsigned int getPort(void);
```

説明

データメンバのネットワーク情報データ領域からポート番号の値を取得します。

戻り値

ポート番号を返します。

CDAQMXNetInfo::getSubMask

構文

```
unsigned int getSubMask(void);
```

説明

データメンバのネットワーク情報データ領域からサブネットマスクの値を取得します。

戻り値

サブネットマスクを返します。

CDAQMXNetInfo::initialize

構文

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

説明

データメンバを初期化します。
初期値は，原則0です。

参照

`initMXNetInfo`

CDAQMXNetInfo::initMXNetInfo

構文

```
static void initMXNetInfo(MXNetInfo * pMXNetInfo);
```

引数

pMXNetInfo　　ネットワーク情報データの領域を指定します。

説明

指定された領域を初期化します。
初期値は，原則0です。

CDAQMXNetInfo::isObject

構文

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

引数

classname　　クラス名を文字列で指定します。

説明

指定されたクラス名を継承しているかをチェックします。
省略された場合，本クラスであるかをチェックします。
本クラスを継承したクラスが，自分自身のクラスをチェックするためには，オーバーライドする必要があります。
クラスを継承している場合，「有効値」(真)を返します。それ以外は，「無効値」(偽)を返します。

戻り値

有効無効値を返します。

CDAQMXNetInfo::operator=

構文

```
CDAQMXNetInfo & operator=(CDAQMXNetInfo & cMXNetInfo);
```

引数

cMXNetInfo　　代入するオブジェクトを指定します。

説明

指定されたオブジェクトのデータメンバを複写します。

戻り値

本オブジェクトへの参照を返します。

CDAQMXNetInfo::setMXNetInfo

構文

```
void setMXNetInfo(MXNetInfo * pMXNetInfo);
```

引数

pMXNetInfo　　ネットワーク情報データを指定します。

説明

構造体でデータを設定します。
データメンバに指定された構造体の内容を格納します。
指定がない場合，データメンバは初期化されます。

参照

```
initialize
```

CDAQMXOutputクラス

本クラスは，MX100での出力チャネルデータを格納するクラスです。
MXOutputData構造体のラッパクラスになります。
全チャネル分の出力チャネルデータを集約したものです。
出力チャネルデータ番号で各データにアクセスできます。
出力チャネルデータ番号は， AO/PWMデータ番号です。
出力チャネルデータの取得と設定のインターフェイスとして使用するクラスです。

パブリックメンバ

構築・消滅

CDAQMXOutputData	オブジェクトを構築します。
~CDAQMXOutputData	オブジェクトを消滅します。

構造体操作

getMXOutputData	構造体でデータを取得します。
setMXOutputData	構造体でデータを設定します。
initMXOutputData	構造体のデータを初期化します。

データメンバ操作

initialize	データメンバを初期化します。
getOutputType	出力種類を取得します。
getIdleChoice	アイドル時の選択値を取得します。
getErrorChoice	エラー時の選択値を取得します。
getPresetValue	選択値が「指定値」の場合の値を取得します。
getPulseTime	パルス周期倍率を取得します。
setOutputType	出力種類を設定します。
setChoice	選択値を設定します。
setPulseTime	パルス周期倍率を設定します。

ユーティリティ

isObject	オブジェクトをチェックします。

演算子

operator=	代入を実行します。

プロテクトメンバ

データメンバ

m_MXOutputData	出力チャネルデータの格納領域です。

メンバアクセス

getMXOutput	各チャネルごとの出力チャネルデータの構造体を取得します。

プライベートメンバ

なし。

関数メンバ(アルファベット順)

CDAQMXOutputData::CDAQMXOutputData

構文

```
CDAQMXOutputData(MXOutputData * pMXOutputData = NULL);
virtual ~CDAQMXOutputData(void);
```

引数

pMXOutputData　　出力チャネルデータを指定します。

説明

オブジェクトを構築，消滅します。
構築時， データメンバに指定された値を設定します。 指定が省略された場合， データメンバを初期化します。

参照

`setMXOutputData`

CDAQMXOutputData::getErrorChoice

構文

```
int getErrorChoice(int outputNo);
```

引数

outputNo　　出力チャネルデータ番号を指定します。

説明

データメンバから指定されたデータ番号の示すエラー時の選択値を取得します。
存在しない場合，「前回値」を返します。

戻り値

選択値を返します。

参照

`getMXOutput`

CDAQMXOutputData::getIdleChoice

構文

```
int getIdleChoice(int outputNo);
```

引数

outputNo　　出力チャネルデータ番号を指定します。

説明

データメンバから指定されたデータ番号の示すアイドル時の選択値を取得します。
存在しない場合，「前回値」を返します。

戻り値

選択値を返します。

参照

`getMXOutput`

CDAQMXOutputData::getMXOutput

構文

```
MXOutput * getMXOutput(int outputNo);
```

引数

outputNo　　出力チャネルデータ番号を指定します。

説明

データメンバから指定されたデータ番号の示す構造体を取得します。
存在しない場合，NULLを返します。

戻り値

構造体へのポインタを返します。

CDAQMXOutputData::getMXOutputData

構文

```
void getMXOutputData(MXOutputData * pMXOutputData);
```

引数

pMXOutputData　　出力チャネルデータの返却先を指定します。

説明

構造体でデータを取得します。
データメンバの内容を指定された構造体に格納します。

CDAQMXOutputData::getOutputType

構文

```
int getOutputType(int outputNo);
```

引数

outputNo　　出力チャネルデータ番号を指定します。

説明

データメンバから指定されたデータ番号の示す出力種類を取得します。
存在しない場合，「出力なし」を返します。

戻り値

出力種類を返します。

参照

getMXOutput

CDAQMXOutputData::getPresetValue

構文

```
int getPresetValue(int outputNo);
```

引数

outputNo　　出力チャネルデータ番号を指定します。

説明

データメンバから指定されたデータ番号の示す選択値が「指定値」の場合の値を取得します。
存在しない場合，0を返します。

戻り値

選択値が「指定値」の場合の値を返します。

参照

getMXOutput

CDAQMXOutputData::getPulseTime

構文

```
int getPulseTime(int outputNo);
```

引数

outputNo　　出力チャネルデータ番号を指定します。

説明

データメンバから指定されたデータ番号の示すパルス周期倍率を取得します。
存在しない場合，1を返します。

戻り値

パルス周期倍率を返します。

参照

getMXOutput

CDAQMXOutputData::initialize

構文

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

説明

データメンバを初期化します。
初期値は，原則0です。

参照

initMXOutputData

CDAQMXOutputData::initMXOutputData

構文

```
static void initMXOutputData(MXOutputData * pMXOutputData);
```

引数

pMXOutputData　　出力チャネルデータの領域を指定します。

説明

指定された領域を初期化します。
初期値は，原則0です。

CDAQMXOutputData::isObject

構文

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

引数

classname　　クラス名を文字列で指定します。

説明

指定されたクラス名を継承しているかをチェックします。
引数が省略された場合，本クラスであるかをチェックします。
本クラスを継承したクラスが，自分自身のクラスをチェックするためには，オーバーライドする必要があります。
クラスを継承している場合，「有効値」(真)を返します。それ以外は，「無効値」(偽)を返します。

戻り値

有効無効値を返します。

CDAQMXOutputData::operator=

構文

```
CDAQMXOutputData & operator=(CDAQMXOutputData &
cMXOutputData);
```

引数

cMXOutputData　　代入するオブジェクトを指定します。

説明

指定されたオブジェクトのデータメンバを複写します。

戻り値

本オブジェクトへの参照を返します。

CDAQMXOutputData::setChoice

構文

```
void setChoice(int outputNo, int idleChoice, int errorChoice,
int presetValue = 0);
```

引数

outputNo　　出力チャネルデータ番号を指定します。
idleChoice　　アイドル時の選択値を指定します。
errorChoice　　エラー時の選択値を指定します。
presetValue　　選択値が「指定値」の場合の値を指定します。

説明

データメンバの指定されたデータ番号の示す領域に，指定された値を格納します。
データ番号に，「全出力データ番号指定」をした場合， 全データに同じ値を格納します。

参照

`getMXOutput`

CDAQMXOutputData::setMXOutputData

構文

```
void setMXOutputData(MXOutputData * pMXOutputData);
```

引数

pMXOutputData　　出力チャネルデータを指定します。

説明

構造体でデータを設定します。
データメンバに指定された構造体の内容を格納します。
指定がない場合， データメンバは初期化されます。

参照

`initMXOutputData`

CDAQMXOutputData::setOutputType

構文

```
void setOutputType(int outputNo, int iOutput);
```

引数

outputNo　　出力チャネルデータ番号を指定します。
iOutput　　出力種類を指定します。

説明

データメンバの指定されたデータ番号の示す領域に，指定された値を格納します。
データ番号に，「全出力データ番号指定」をした場合， 全データに同じ値を格納します。
他の項目領域は既定値になります。

参照

```
getMXOutput setChoice setPulseTime
```

CDAQMXOutputData::setPulseTime

構文

```
void setPulseTime(int outputNo, int pulseTime);
```

引数

outputNo　　出力チャネルデータ番号を指定します。
pulseTime　　パルス周期倍率を指定します。

説明

データメンバの指定されたデータ番号の示す領域に，指定された値を格納します。
データ番号に，「全出力データ番号指定」をした場合， 全データに同じ値を格納します。

参照

```
getMXOutput
```

CDAQMXSegmentクラス

本クラスは，MX100での7セグメントLEDの表示パターンを格納するクラスです。
MXSegment構造体のラッパクラスになります。
7セグメントLEDの表示において，表示パターンを格納するインターフェイスとして使用するクラスです。

パブリックメンバ

構築・消滅

CDAQMXSegment	オブジェクトを構築します。
~CDAQMXSegment	オブジェクトを消滅します。

構造体操作

getMXSegment	構造体でデータを取得します。
setMXSegment	構造体でデータを設定します。
initMXSegment	構造体のデータを初期化します。

データメンバ操作

initialize	データメンバを初期化します。
getPattern	表示パターンを取得します。
setPattern	表示パターンを設定します。

演算子

operator=	代入を実行します。

ユーティリティ

isObject	オブジェクトをチェックします。

プロテクトメンバ

データメンバ

m_MXSegment	7セグメントLEDの格納領域です。

プライベートメンバ

なし。

関数メンバ(アルファベット順)

CDAQMXSegment::CDAQMXSegment

構文

```
CDAQMXSegment(MXSegment * pMXSegment = NULL);
CDAQMXSegment(int pattern0, int pattern1);
virtual ~CDAQMXSegment(void);
```

引数

pMXSegment　7セグメントLEDを指定します。
pattern0　　セグメント番号0の表示パターンを指定します。
pattern1　　セグメント番号1の表示パターンを指定します。

説明

オブジェクトを構築，消滅します。
構築時， データメンバに指定された値を設定します。 指定がない場合， データメンバを初期化します。

参照

```
initialize setMXSegment setPattern
```

CDAQMXSegment::getMXSegment

構文

```
void getMXSegment(MXSegment * pMXSegment);
```

引数

pMXSegment　7セグメントLEDの返却先を指定します。

説明

構造体でデータを取得します。データメンバの内容を指定された構造体に格納します。

CDAQMXSegment::getPattern

構文

```
int getPattern(int segmentNo);
```

引数

segmentNo　　セグメント番号を指定します。

説明

データメンバの7セグメントLED領域から指定されたセグメント番号の表示パターンの値を取得します。
存在しない場合，0を返します。

戻り値

表示パターンを返します。

CDAQMXSegment::initialize

構文

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

説明

データメンバを初期化します。
初期値は，原則0です。

参照

```
initMXSegment
```

CDAQMXSegment::initMXSegment

構文

```
static void initMXSegment(MXSegment * pMXSegment);
```

引数

pMXSegment　7セグメントLEDの領域を指定します。

説明

指定された領域を初期化します。
初期値は，原則0です。

CDAQMXSegment::isObject

構文

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

引数

classname　　クラス名を文字列で指定します。

説明

指定されたクラス名を継承しているかをチェックします。
省略された場合，本クラスであるかをチェックします。
本クラスを継承したクラスが，自分自身のクラスをチェックするためには，オーバーライドする必要があります。
クラスを継承している場合，「有効値」(真)を返します。それ以外は，「無効値」(偽)を返します。

戻り値

有効無効値を返します。

CDAQMXSegment::operator=

構文

```
CDAQMXSegment & operator=(CDAQMXSegment & cMXSegment);
```

引数

cMXSegment　代入するオブジェクトを指定します。

説明

指定されたオブジェクトのデータメンバを複写します。

戻り値

本オブジェクトへの参照を返します。

CDAQMXSegment::setMXSegment

構文

```
void setMXSegment(MXSegment * pMXSegment);
```

引数

pMXSegment　7セグメントLEDを指定します。

説明

構造体でデータを設定します。データメンバに指定された構造体の内容を格納します。

指定がない場合，データメンバは初期化されます。

参照

`initialize`

CDAQMXSegment::setPattern

構文

```
void setPattern(int segmentNo, int pattern);
```

引数

segmentNo　セグメント番号を指定します。

pattern　表示パターンを指定します。

説明

データメンバの7セグメントLED領域の指定されたセグメント番号の示す領域に指定された値を格納します。

セグメント番号に定数値の「全セグメント番号指定」を指定した場合，全7セグメントLEDに値を格納します。

CDAQMXStatusクラス

本クラスは，MX100でのステータスデータを格納するクラスです。
MXStatus構造体のラッパクラスになります。
ステータスデータ取得と設定データ取得において，ステータスデータを格納するインターフェイスとして使用するクラスです。

パブリックメンバ

構築・消滅

CDAQMXStatus	オブジェクトを構築します。
~CDAQMXStatus	オブジェクトを消滅します。

構造体操作

getMXStatus	構造体でデータを取得します。
setMXStatus	構造体でデータを設定します。
initMXStatus	構造体のデータを初期化します。

データメンバ操作

initialize	データメンバを初期化します。
getFIFONum	FIFOの有効個数を取得します。
getFIFOStatus	FIFOステータス値を取得します。
getInterval	周期種類を取得します。
getOldDataNo	最古のデータ番号を取得します。
getNewDataNo	最新のデータ番号を取得します。
getCFStatus	CFステータス種類を取得します。
getCFSize	CFの容量を取得します。
getCFRemain	CFの残容量を取得します。
getUnitStatus	ユニットステータス値を取得します。
getConfigCnt	設定番号(設定の実行ごとに増加する順序番号)を取得します。
getTimeCnt	時刻番号(時刻設定の実行ごとに増加する順序番号)を取得します。
isBackup	バックアップの有無を取得します。
getTime	時刻を取得します。
getMilliSecond	ミリ秒を取得します。

演算子

operator=	代入を実行します。

ユーティリティ

isDataNo	データ番号の有効性をチェックします。
isObject	オブジェクトをチェックします。

プロテクトメンバ

データメンバ

m_MXStatus　　ステータスデータの格納領域です。

メンバアクセス

getMXFIFOInfo　　各FIFOごとのFIFO情報の構造体を取得します。

プライベートメンバ

なし。

関数メンバ(アルファベット順)

CDAQMXStatus::CDAQMXStatus

構文

```
CDAQMXStatus(MXStatus * pMXStatus = NULL);
virtual ~CDAQMXStatus(void);
```

引数

pMXStatus　　ステータスデータを指定します。

説明

オブジェクトを構築，消滅します。
構築時，データメンバに指定された値を設定します。指定がない場合，データメンバを初期化します。

参照

`setMXStatus`

CDAQMXStatus::getCFRemain

構文

```
int getCFRemain(void);
```

説明

データメンバのステータスデータ領域からCFの残容量の値を取得します。
単位はKBです。

戻り値

残容量を返します。

CDAQMXStatus::getCFSize

構文

```
int getCFSize(void);
```

説明

データメンバのステータスデータ領域からCFの容量の値を取得します。
単位はKBです。

戻り値

容量を返します。

CDAQMXStatus::getCFStatus

構文

```
int getCFStatus(void);
```

説明

データメンバのステータスデータ領域からCFステータス種類の値を取得します。

戻り値

CFステータス種類を返します。

CDAQMXStatus::getConfigCnt

構文

```
int getConfigCnt(void);
```

説明

データメンバのステータスデータ領域から設定番号の値を取得します。

戻り値

設定番号を返します。

CDAQMXStatus::getDateTime

構文

```
void getDateTime(CDAQDateTime & cDateTime);
```

引数

cDateTime　　時刻情報データの返却先を指定します。

説明

データメンバのステータスデータ領域から時刻情報データを指定された領域に格納します。

参照

```
getMilliSecond getTime
CDAQDateTime::setMilliSecond
CDAQDateTime::setTime
```

CDAQMXStatus::getFIFONum

構文

```
int getFIFONum(void);
```

説明

データメンバのステータスデータ領域からFIFOの有効個数の値を取得します。

戻り値

FIFOの有効個数を返します。

CDAQMXStatus::getFIFOStatus

構文

```
int getFIFOStatus(int fifoNo);
```

引数

fifoNo　　FIFO番号を指定します。

説明

データメンバのステータスデータ領域から指定されたFIFO番号に対応するFIFO情報のFIFOステータス値の値を取得します。
存在しない場合，「不明」を返します。

戻り値

FIFOステータス値を返します。

参照

`getMXFIFOInfo`

CDAQMXStatus::getInterval

構文

```
int getInterval(int fifoNo);
```

引数

fifoNo　　FIFO番号を指定します。

説明

データメンバのステータスデータ領域から指定されたFIFO番号に対応するFIFO情報の周期種類の値を取得します。
存在しない場合，0を返します。

戻り値

周期種類を返します。

参照

`getMXFIFOInfo`

CDAQMXStatus::getMilliSecond

構文

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

説明

データメンバのステータスデータ領域からミリ秒を取得します。

戻り値

ミリ秒を返します。

CDAQMXStatus::getMXFIFOInfo

構文

```
MXFIFOInfo * getMXFIFOInfo(int fifoNo);
```

引数

fifoNo　　FIFO番号を指定します。

説明

データメンバのステータスデータ領域から指定されたFIFO番号の構造体を取得します。

存在しない場合，NULLを返します。

戻り値

構造体へのポインタを返します。

CDAQMXStatus::getMXStatus

構文

```
void getMXStatus(MXStatus * pMXStatus);
```

引数

pMXStatus　　ステータスデータの返却先を指定します。

説明

構造体でデータを取得します。

データメンバの内容を指定された構造体に格納します。

CDAQMXStatus::getNewDataNo

構文

```
MXDataNo getNewDataNo(int fifoNo);
```

引数

fifoNo　　FIFO番号を指定します。

説明

データメンバのステータスデータ領域から指定されたFIFO番号に対応するFIFO情報の最新のデータ番号の値を取得します。
存在しない場合，定数値の「瞬時値指定用データ番号」を返します。

戻り値

データ番号を返します。

参照

`getMXFIFOInfo`

CDAQMXStatus::getOldDataNo

構文

```
MXDataNo getOldDataNo(int fifoNo);
```

引数

fifoNo　　FIFO番号を指定します。

説明

データメンバのステータスデータ領域から指定されたFIFO番号に対応するFIFO情報の最古のデータ番号の値を取得します。
存在しない場合，定数値の「瞬時値指定用データ番号」を返します。

戻り値

データ番号を返します。

参照

`getMXFIFOInfo`

CDAQMXStatus::getTime

構文

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

説明

データメンバのステータスデータ領域から秒数を取得します。

戻り値

秒数を返します。

CDAQMXStatus::getTimeCnt

構文

```
int getTimeCnt(void);
```

説明

データメンバのステータスデータ領域から時刻番号の値を取得します。

戻り値

時刻番号を返します。

CDAQMXStatus::getUnitStatus

構文

```
int getUnitStatus(void);
```

説明

データメンバのステータスデータ領域からユニットステータス値の値を取得します。

戻り値

ユニットステータス値を返します。

CDAQMXStatus::initialize

構文

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

説明

データメンバを初期化します。
初期値は，原則0です。

参照

```
initMXStatus
```

CDAQMXStatus::initMXStatus

構文

```
static void initMXStatus(MXStatus * pMXStatus);
```

引数

pMXStatus　　ステータスデータの領域を指定します。

説明

指定された領域を初期化します。
初期値は，原則0です。

CDAQMXStatus::isBackup

構文

```
int isBackup(void);
```

説明

データメンバのステータスデータ領域からバックアップの有無を取得します。

戻り値

有効無効値を返します。

CDAQMXStatus::isDataNo

構文

```
static int isDataNo(MXDataNo dataNo);
```

引数

dataNo　　データ番号を指定します。

説明

指定されたデータ番号が有効な番号かどうかをチェックします。
データ番号が0以上の場合，「有効値」を返します。

戻り値

有効無効値を返します。

CDAQMXStatus::isObject

構文

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

引数

classname　　クラス名を文字列で指定します。

説明

指定されたクラス名を継承しているかをチェックします。
省略された場合，本クラスであるかをチェックします。
本クラスを継承したクラスが，自分自身のクラスをチェックするためには，オーバーライドする必要があります。
クラスを継承している場合，「有効値」（真）を返します。それ以外は，「無効値」（偽）を返します。

戻り値

有効無効値を返します。

CDAQMXStatus::operator=

構文

```
CDAQMXStatus & operator=(CDAQMXStatus & cMXStatus);
```

引数

cMXStatus　　代入するオブジェクトを指定します。

説明

指定されたオブジェクトのデータメンバを複写します。

戻り値

本オブジェクトへの参照を返します。

CDAQMXStatus::setMXStatus

構文

```
void setMXStatus(MXStatus * pMXStatus);
```

引数

pMXStatus　　ステータスデータを指定します。

説明

構造体でデータを設定します。
データメンバに指定された構造体の内容を格納します。
指定がない場合，データメンバは初期化されます。

参照

`initialize`

CDAQMXSysInfoクラス

本クラスは，MX100でのシステム構成データを格納するクラスです。
MXSystemInfo構造体のラッパクラスになります。
システム構成データ取得と設定データ取得において，システム構成データを格納するインターフェイスとして使用するクラスです。

パブリックメンバ

構築・消滅

CDAQMXSysInfo	オブジェクトを構築します。
~CDAQMXSysInfo	オブジェクトを消滅します。

構造体操作

getMXSystemInfo	構造体でデータを取得します。
setMXSystemInfo	構造体でデータを設定します。
initMXSystemInfo	構造体のデータを初期化します。

データメンバ操作

initialize	データメンバを初期化します。
getUnitType	ユニット種類を取得します。
getStyle	スタイルを取得します。
getUnitNo	ユニット番号を取得します。
getTempUnit	温度単位種類を取得します。
getCFTimeout	タイムアウト値を取得します。
getCFWriteMode	CF書き込み種類を取得します。
getFrequency	電源周波数を取得します。
getPartNo	パート番号を取得します。
getOption	オプションを取得します。
getUnitSerial	ユニットのシリアル番号を取得します。
getMAC	MACアドレスを取得します。
getModuleType	モジュール種類を取得します。
getChNum	チャネル数を取得します。
getInterval	周期種類を取得します。
getIntegral	A/D積分時間種類を取得します。
getStandbyType	起動時モジュール種類を取得します。
getRealType	実際のモジュール種類を取得します。
isModuleValid	モジュールの有効無効値を取得します。
getModuleVersion	モジュールバージョンを取得します。
getTerminalType	端子種類を取得します。
getFIFONo	FIFO番号を取得します。
getModuleSerial	モジュールのシリアル番号を取得します。
setUnitNo	ユニット番号を設定します。

setTempUnit	温度単位種類を設定します。
setCFTimeout	タイムアウト値を設定します。
setCFWriteMode	CF書き込み種類を設定します。
setModule	モジュールを設定します。
setRealModule	モジュールを実際のモジュールに変更します。

検証

isCorrect	妥当性を検証します。

ユーティリティ

getItemError	エラー検出した設定項目番号を取得します。
isObject	オブジェクトをチェックします。

演算子

operator=	代入を実行します。

プロテクトメンバ

データメンバ

m_MXSystemInfo	システム構成データの格納領域です。
m_nItemError	設定項目番号の格納領域です。

メンバアクセス

getMXModuleData	各モジュールごとのモジュール情報の構造体を取得します。

プライベートメンバ

なし。

関数メンバ(アルファベット順)

CDAQMXSysInfo::CDAQMXSysInfo

構文

```
CDAQMXSysInfo(MXSystemInfo * pMXSystemInfo = NULL);
virtual ~CDAQMXSysInfo(void);
```

引数

pMXSystemInfo	システム構成データを指定します。

説明

オブジェクトを構築，消滅します。
構築時，データメンバに指定された値を設定します。指定がない場合，データメンバを初期化します。

参照

`setMXSystemInfo`

CDAQMXSysInfo::getCFTimeout

構文

```
int getCFTimeout(void);
```

説明

データメンバのシステム構成データ領域からタイムアウト値の値を取得します。

戻り値

タイムアウト値を返します。

CDAQMXSysInfo::getCFWriteMode

構文

```
int getCFWriteMode(void);
```

説明

データメンバのシステム構成データ領域からCF書き込み種類の値を取得します。

戻り値

CF書き込み種類を返します。

CDAQMXSysInfo::getChNum

構文

```
int getChNum(int moduleNo);
```

引数

moduleNo　　モジュール番号を指定します。

説明

データメンバのシステム構成データ領域から指定されたモジュール番号に対応するモジュールのチャネル数を取得します。
存在しない場合，「0」を返します。

戻り値

チャネル数を返します。

参照

`getMXModuleData`

CDAQMXSysInfo::getFIFONo

構文

```
int getFIFONo(int moduleNo);
```

引数

moduleNo　　モジュール番号を指定します。

説明

データメンバのシステム構成データ領域から指定されたモジュール番号に対応するモジュールのFIFO番号を取得します。
存在しない場合，負の値を返します。

戻り値

FIFO番号を返します。

参照

`getMXModuleData`

CDAQMXSysInfo::getFrequency

構文

```
int getFrequency(void);
```

説明

データメンバのシステム構成データ領域から電源周波数の値を取得します。

戻り値

電源周波数を返します。

CDAQMXSysInfo::getIntegral

構文

```
int getIntegral(int moduleNo);
```

引数

moduleNo　　モジュール番号を指定します。

説明

データメンバのシステム構成データ領域から指定されたモジュール番号に対応するモジュールのA/D積分時間種類を取得します。
存在しない場合，「自動」を返します。

戻り値

A/D積分時間種類を返します。

参照

`getMXModuleData`

CDAQMXSysInfo::getInterval

構文

```
int getInterval(int moduleNo);
```

引数

moduleNo　　モジュール番号を指定します。

説明

データメンバのシステム構成データ領域から指定されたモジュール番号に対応するモジュールの周期種類を取得します。
存在しない場合，0を返します。

戻り値

周期種類を返します。

参照

`getMXModuleData`

CDAQMXSysInfo::getItemError

構文

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

説明

データメンバの設定項目番号領域の値を取得します。

戻り値

設定項目番号を返します。

CDAQMXSysInfo::getMAC

構文

```
unsigned char getMAC(int index);
```

引数

index　　バイト位置を指定します。

説明

データメンバのシステム構成データ領域からMACアドレスを取得します。
バイト単位で取得します。範囲外の場合，0を返します。
バイト位置は，0から始まる整数です。
MACアドレスの要素数は，個数値の定義があります。

戻り値

バイト値を返します。

CDAQMXSysInfo::getModuleSerial

構文

```
const char * getModuleSerial(int moduleNo);
```

引数

moduleNo　　モジュール番号を指定します。

説明

データメンバのシステム構成データ領域から指定されたモジュール番号に対応するモジュールのシリアル番号を取得します。
存在しない場合，NULLを返します。

戻り値

文字列へのポインタを返します。

参照

`getMXModuleData`

CDAQMXSysInfo::getModuleType

構文

```
int getModuleType(int moduleNo);
```

引数

moduleNo　　モジュール番号を指定します。

説明

データメンバのシステム構成データ領域から指定されたモジュール番号に対応するモジュールのモジュール種類を取得します。
存在しない場合，「モジュールなし」を返します。

戻り値

モジュール種類を返します。

参照

`getMXModuleData`

CDAQMXSysInfo::getModuleVersion

構文

```
int getModuleVersion(int moduleNo);
```

引数

moduleNo　　モジュール番号を指定します。

説明

データメンバのシステム構成データ領域から指定されたモジュール番号に対応するモジュールのモジュールバージョンを取得します。
存在しない場合，0を返します。

戻り値

モジュールバージョンを返します。

参照

`getMXModuleData`

CDAQMXSysInfo::getMXModuleData

構文

```
MXModuleData * getMXModuleData(int moduleNo);
```

引数

moduleNo　　モジュール番号を指定します。

説明

データメンバのシステム構成データ領域から指定されたモジュール番号の構造体を取得します。
存在しない場合，NULLを返します。

戻り値

構造体へのポインタを返します。

CDAQMXSysInfo::getMXSystemInfo

構文

```
void getMXSystemInfo(MXSystemInfo * pMXSystemInfo);
```

引数

pMXSystemInfo　　システム構成データの返却先を指定します。

説明

構造体でデータを取得します。
データメンバの内容を指定された構造体に格納します。

CDAQMXSysInfo::getOption

構文

```
int getOption(void);
```

説明

データメンバのシステム構成データ領域からオプションの値を取得します。

戻り値

オプションを返します。

CDAQMXSysInfo::getPartNo

構文

```
const char * getPartNo(void);
```

説明

データメンバのシステム構成データ領域からパート番号を取得します。

戻り値

文字列へのポインタを返します。

CDAQMXSysInfo::getRealType

構文

```
int getRealType(int moduleNo);
```

引数

moduleNo　　モジュール番号を指定します。

説明

データメンバのシステム構成データ領域から指定されたモジュール番号に対応するモジュールの実際のモジュール種類を取得します。
存在しない場合，「モジュールなし」を返します。

戻り値

モジュール種類を返します。

参照

`getMXModuleData`

CDAQMXSysInfo::getStandbyType

構文

```
int getStandbyType(int moduleNo);
```

引数

moduleNo　　モジュール番号を指定します。

説明

データメンバのシステム構成データ領域から指定されたモジュール番号に対応するモジュールの起動時モジュール種類を取得します。
存在しない場合，「モジュールなし」を返します。

戻り値

モジュール種類を返します。

参照

`getMXModuleData`

CDAQMXSysInfo::getStyle

構文

```
int getStyle(void);
```

説明

データメンバのシステム構成データ領域からスタイルの値を取得します。

戻り値

スタイルを返します。

CDAQMXSysInfo::getTempUnit

構文

```
int getTempUnit(void);
```

説明

データメンバのシステム構成データ領域から温度単位種類の値を取得します。

戻り値

温度単位種類を返します。

CDAQMXSysInfo::getTerminalType

構文

```
int getTerminalType(int moduleNo);
```

引数

moduleNo　　モジュール番号を指定します。

説明

データメンバのシステム構成データ領域から指定されたモジュール番号に対応するモジュールの端子種類を取得します。
存在しない場合，「クランプ」を返します。

戻り値

端子種類を返します。

参照

`getMXModuleData`

CDAQMXSysInfo::getUnitNo

構文

```
int getUnitNo(void);
```

説明

データメンバのシステム構成データ領域からユニット番号の値を取得します。

戻り値

ユニット番号を返します。

CDAQMXSysInfo::getUnitSerial

構文

```
const char * getUnitSerial(void);
```

説明

データメンバのシステム構成データ領域からユニットのシリアル番号を取得します。

戻り値

文字列へのポインタを返します。

CDAQMXSysInfo::getUnitType

構文

```
int getUnitType(void);
```

説明

データメンバのシステム構成データ領域からユニット種類の値を取得します。

戻り値

ユニット種類を返します。

CDAQMXSysInfo::initialize

構文

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

説明

データメンバを初期化します。
初期値は，原則0です。
タイムアウト値には既定値を設定します。

参照

```
initMXSystemInfo setCFTimeout
```

CDAQMXSysInfo::initMXSystemInfo

構文

```
static void initMXSystemInfo(MXSystemInfo * pMXSystemInfo);
```

引数

pMXSystemInfo システム構成データの領域を指定します。

説明

指定された領域を初期化します。
初期値は，原則0です。

CDAQMXSysInfo::isCorrect

構文

```
int isCorrect(void);
```

説明

妥当性の検証をします。
各設定項目をチェックします。
FIFO個数の制限をチェックします。
不正な値を検出した場合，「無効値」を返します。
不正な値を検出した場合，データメンバの設定項目番号領域に検出した場所を示す設定項目番号を格納します。

戻り値

有効無効値を返します。

参照

```
getCFTimeout getCFWriteMode getMXModuleData getTempUnit
getUnitNo
```

CDAQMXSysInfo::isModuleValid

構文

```
int isModuleValid(int moduleNo);
```

引数

moduleNo　　モジュール番号を指定します。

説明

データメンバのシステム構成データ領域から指定されたモジュール番号に対応するモジュールの有効無効値を取得します。
存在しない場合，「無効値」を返します。

戻り値

有効無効値を返します。

参照

```
getMXModuleData
```

CDAQMXSysInfo::isObject

構文

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

引数

classname　　クラス名を文字列で指定します。

説明

指定されたクラス名を継承しているかをチェックします。
引数が省略された場合，本クラスであるかをチェックします。
本クラスを継承したクラスが，自分自身のクラスをチェックするためには，オーバーライドする必要があります。
クラスを継承している場合，「有効値」(真)を返します。それ以外は，「無効値」(偽)を返します。

戻り値

有効無効値を返します。

CDAQMXSysInfo::operator=

構文

```
CDAQMXSysInfo & operator=(CDAQMXSysInfo & cMXSysInfo);
```

引数

cMXSysInfo　　代入するオブジェクトを指定します。

説明

指定されたオブジェクトのデータメンバを複写します。

戻り値

本オブジェクトへの参照を返します。

CDAQMXSysInfo::setCFTimeout

構文

```
void setCFTimeout(int timeout = 60);
```

引数

timeout　　タイムアウト値を指定します。

説明

データメンバのシステム構成データ領域に指定された値を格納します。
単位は秒です。

CDAQMXSysInfo::setCFWriteMode

構文

```
void setCFWriteMode(int iCFWriteMode);
```

引数

iCFWriteMode　CF書き込み種類を指定します。

説明

データメンバのシステム構成データ領域に指定された値を格納します。

CDAQMXSysInfo::setModule

構文

```
void setModule(int moduleNo, int iModuleType, int iChNum, int
iInterval, int iHz = DAQMX_INTEGRAL_AUTO);
```

引数

moduleNo　　モジュール番号を指定します。
iModuleType　モジュール種類を指定します。
iChNum　　チャネル数を指定します。
iInterval　　周期種類を指定します。
iHz　　A/D積分時間種類を指定します。

説明

データメンバのシステム構成データ領域に指定されたモジュール番号に対応するモジュールに，指定された値を格納します。
領域が存在しない場合，格納しません。

参照

`getMXModuleData`

CDAQMXSysInfo::setMXSystemInfo

構文

```
void setMXSystemInfo(MXSystemInfo * pMXSystemInfo);
```

引数

pMXSystemInfo　　システム構成データを指定します。

説明

構造体でデータを設定します。
データメンバに指定された構造体の内容を格納します。
指定がない場合，データメンバは初期化されます。

参照

`initialize`

CDAQMXSysInfo::setRealModule

構文

```
int setRealModule(int moduleNo);
```

引数

moduleNo　　モジュール番号を指定します。

説明

指定されたモジュール番号に対応するモジュールを実際のモジュール種類にします。
データメンバのシステム構成データ領域の指定されたモジュール番号に対応するモジュールの設定を更新します。
各設定値はモジュール種類の既定値になります。
実際のモジュール種類を返します。

戻り値

モジュール種類を返します。

参照

`getRealType setModule`

CDAQMXSysInfo::setTempUnit

構文

```
void setTempUnit(int iTempUnit);
```

引数

iTempUnit　　温度単位種類を指定します。

説明

データメンバのシステム構成データ領域に指定された値を格納します。

CDAQMXSysInfo::setUnitNo

構文

```
void setUnitNo(int unitNo);
```

引数

unitNo　　ユニット番号を指定します。

説明

データメンバのシステム構成データ領域に指定された値を格納します。

CDAQMXTransmitクラス

本クラスは，MX100での伝送出力データを格納するクラスです。
MXTransmit構造体のラッパクラスになります。
全チャネル分の伝送状態をまとめたものです。
伝送出力データ番号で各データにアクセスできます。伝送出力データ番号は，PWMデータ番号，または，AOデータ番号です。
伝送出力データの取得と設定のインタフェースとして使用するクラスです。

パブリックメンバ

構築・消滅

CDAQMXTransmit	オブジェクトを構築します。
~CDAQMXTransmit	オブジェクトを消滅します。

構造体操作

getMXTransmit	構造体でデータを取得します。
setMXTransmit	構造体でデータを設定します。
initMXTransmit	構造体のデータを初期化します。

データメンバ操作

initialize	データメンバを初期化します。
getTransmit	伝送状態を取得します。
setTransmit	伝送状態を設定します。

演算子

operator=	代入を実行します。

ユーティリティ

isObject	オブジェクトをチェックします。

プロテクトメンバ

データメンバ

m_MXTransmit	伝送状態の格納領域です。

プライベートメンバ

なし。

関数メンバ

CDAQMXTransmit::CDAQMXTransmit

構文

```
CDAQMXTransmit(MXTransmit * pMXTransmit = NULL);
virtual ~CDAQMXTransmit(void);
```

引数

pMXTransmit　伝送状態を指定します。

説明

オブジェクトを構築，消滅します。
構築時，データメンバに指定された値を設定します。指定がない場合，データメンバを初期化します。

参照

`setMXTransmit`

CDAQMXTransmit::getMXTransmit

構文

```
void getMXTransmit(MXTransmit * pMXTransmit);
```

引数

pMXTransmit　伝送状態の返却先を指定します。

説明

構造体でデータを取得します。
データメンバの内容を指定された構造体に格納します。

CDAQMXTransmit::getTransmit

構文

```
int getTransmit(int aopwmNo);
```

引数

aopwmNo　AO/PWMデータ番号を指定します。

説明

データメンバから指定されたデータ番号の示す伝送状態を取得します。
存在しない場合，「指定なし(不明)」の値を返します。

戻り値

伝送状態を返します。

CDAQMXTransmit::initialize

構文

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

説明

データメンバを初期化します。
初期値は，原則0です。

参照

```
initMXTransmit
```

CDAQMXTransmit::initMXTransmit

構文

```
static void initMXTransmit(MXTransmit * pMXTransmit);
```

引数

pMXTransmit　伝送状態の領域を指定します。

説明

指定された領域を初期化します。
初期値は，原則0です。

CDAQMXTransmit::isObject

構文

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

引数

classname　　クラス名を文字列で指定します。

説明

指定されたクラス名を継承しているかをチェックします。
引数が省略された場合，本クラスであるかをチェックします。
本クラスを継承したクラスが，自分自身のクラスをチェックするためには，オーバーライドする必要があります。
クラスを継承している場合，「有効値」(真)を返します。それ以外は，「無効値」(偽)を返します。

戻り値

有効無効値を返します。

CDAQMXTransmit::operator=

構文

```
CDAQMXTransmit & operator=(CDAQMXTransmit & cMXTransmit);
```

引数

cMXTransmit　代入するオブジェクトを指定します。

説明

指定されたオブジェクトのデータメンバを複写します。

戻り値

本オブジェクトへの参照を返します。

CDAQMXTransmit::setMXTransmit

構文

```
void setMXTransmit(MXTransmit * pMXTransmit);
```

引数

pMXTransmit　伝送状態を指定します。

説明

構造体でデータを設定します。
データメンバに指定された構造体の内容を格納します。
指定がない場合，データメンバは初期化されます。

参照

`initMXTransmit`

CDAQMXTransmit::setTransmit

構文

```
void setTransmit(int aopwmNo, int iTransmit);
```

引数

aopwmNo　AO/PWMデータ番号を指定します。
iTransmit　伝送状態を指定します。

説明

データメンバの指定されたデータ番号の示す領域に，指定された値を格納します。
データ番号に，「全AO/PWMデータ番号指定」をした場合，全データに同じ値を格納します。

3.1　機能と関数の対応 - MX100/Visual C

本APIでサポートする機能と，Visual Cの関数群の対応を示します。

通信機能

機能	関数
MX100と通信接続	openMX
MX100との通信を切断	closeMX
通信タイムアウトを設定	setTimeOutMX

Note

通信タイムアウトの設定を推奨しません。**理由**：データ取得時にタイムアウト時間に抵触して予期しない通信切断が発生する場合があります。

制御機能

FIFOの開始/停止

機能	関数
FIFOを開始	startFIFOMX
FIFOを停止	stopFIFOMX
FIFOの自動制御を設定	autoFIFOMX

そのほかの制御

機能	FIFO	関数
MX100に時刻情報(基準日時(1970年1月1日)からの時間)を秒数で設定	停止	setDateTimeMX
MX100に現在の日付/時刻を設定	停止	setDateTimeNowMX
CFへのデータ保存(バックアップ)のON/OFFを設定	継続	setBackupMX
CFをフォーマット	停止	formatCFMX
・ユニットのシステム再構築	停止	initSystemMX
・ユニットのシステム初期化	停止	
・ユニットのアラームリセット(アラームACK)	継続	
7セグメントLEDの表示を設定	継続	setSegmentMX

表の「FIFO」欄は，FIFO中に関数を実行したときの，FIFOの動作を示します。
停止：関数を実行するとFIFOを停止します。
継続：関数を実行してもFIFOを継続します。
バックアップの設定は，CF書き込み種類が変更された場合，FIFOは停止します。
バックアップの設定のCF書き込み種類は，設定変更の操作(一括取得したデータを変更して一括送信)をします。

Note

FIFOの自動制御を設定しておくと，関数の実行によりFIFOが停止したあと，FIFOを自動的に再開します。

設定機能

一括設定

機能	FIFO	関数
設定データを一括設定	停止	setConfigDataMX
設定データ(システム設定データ)を設定	停止	setSystemConfigMX
設定データ(チャネル設定データ)を設定	停止	setChConfigMX
DO(Digital Output)データを送信	継続	setDODataMX
AO/PWMデータを一括設定	継続	setAOPWMDataMX
伝送出力データ送信	継続	setTransmitMX

表の「FIFO」欄については，前ページの「そのほかの制御」の説明をご覧ください。設定データの設定でデータ不正のエラー番号が発生した場合，最後の検出箇所を設定項目番号で保持します。ユーティリティで取得できます。

個別設定

機能		FIFO	関数
初期バランスデータを設定		停止	setBalanceMX
出力チャネルデータを設定		停止	setOutputMX
初期バランスデータ	実行	停止	runBalanceMX
	リセット	停止	resetBalanceMX

設定変更

機能		FIFO	関数
レンジ設定	スキップ(未使用)	停止	setSKIPMX
	直流電圧入力	停止	setVOLTMX
	熱電対入力	停止	setTCMX
	測温抵抗体入力	停止	setRTDMX
	ディジタル入力(DI)	停止	setDIMX
	チャネル間差演算	停止	setDELTAMX
	リモートRJC	停止	setRRJCMX
	抵抗入力	停止	setRESMX
	ひずみ	停止	setSTRAINMX
	AO	停止	setAOMX
	PWM	停止	setPWMMX
	パルス	停止	setPULSEMX
	通信	停止	setCOMMX
チャネルに単位名を設定		停止	setScallingUnitMX
チャネルのタグを設定		停止	setTagMX
チャネルにコメントを設定		停止	setCommentMX
チャネルにアラームを設定		停止	setAlarmMX
チャネルで使用する基準接点補償(RJC)を設定		停止	setRJCTypeMX
チャネルにフィルタを設定		停止	setFilterMX

機能		FIFO	関数
チャネルにバーンアウト検出時動作を設定		停止	setBurnoutMX
アラーム出力を指定したDOチャネルに，関連づけるアラームを設定 (DOチャネルをアラーム出力に指定するときは，setDOTypeMX関数を用います。)		停止	setRefAlarmMX
測定周期を設定		停止	setIntervalMX
温度単位を設定		停止	setTempUnitMX
ユニットの識別番号を設定		停止	setUnitNoMX
タイムアウト値(通信切断時に，CFへのデータ保存を開始するまでの時間)を設定 タイムアウト値の算出については，付録3を参照してください。		停止	setSystemTimeoutMX
DOチャネルに関連づける信号の種類を設定		停止	setDOTypeMX
チャネルのAO種類を設定		停止	setAOTypeMX
チャネルのPWM種類を設定		停止	setPWMTypeMX
出力チャネルデータ	出力種類	停止	setOutputTypeMX
	選択値	停止	setChoiceMX
	パルス周期倍率	停止	setPulseTimeMX
DOデータを部分変更		継続	changeDODataMX
AO/PWMデータの部分変更		継続	changeAOPWMDataMX
初期バランスデータの部分変更		継続	changeBalanceMX
伝送出力データの部分変更		継続	changeTransmitMX
チャネルにチャタリングフィルタを設定		停止	setChatFilterMX

表の「FIFO」欄については，3-1ページの「そのほかの制御」の説明をご覧ください。データ不正のエラー番号が発生した場合，最後の検出箇所を設定項目番号で保持します。ユーティリティで取得できます。

データ取得機能

システムステータスデータ/システム構成データの取得

機能	関数
システムステータスデータを取得	getStatusDataMX
システム構成データを取得	getSystemConfigMX

設定データの取得

機能	関数
設定データを一括取得	getConfigDataMX
設定データの取得を宣言 チャネル設定データ以外の設定データを取得します。	talkConfigMX
チャネル設定データを取得 talkConfigMX関数で設定データの取得を宣言したあとにチャネル設定データを取得する関数です。	getChConfigMX

DOデータの取得

機能	関数
DOデータを一括取得	getDODataMX

AO/PWMデータ，伝送出力データの取得

機能	関数
AO/PWMデータ，伝送出力データを一括取得	getAOPWMDataMX

チャネル情報データの取得

機能	関数
チャネル情報データの取得を宣言	talkChInfoMX
チャネル情報データの取得	getChInfoMX

測定データの取得(チャネル指定)

機能	関数
指定したチャネルの最新のデータ範囲を取得	getChDataNoMX
指定したチャネルの測定データ取得を宣言	talkChDataMX
指定したチャネルの瞬時値取得を宣言	talkChDataInstMX
指定したチャネルの時刻情報をデータ番号ごとに取得	getTimeDataMX
指定したチャネルの測定データを取得	getChDataMX

測定データの取得(FIFO指定)

機能	関数
指定したFIFO番号の最新のデータ範囲を取得	getFIFODataNoMX
指定したFIFO番号の測定データ取得を宣言	talkFIFODataMX
指定したFIFO番号の瞬時値取得を宣言	talkFIFODataInstMX
指定したFIFO番号の時刻情報をデータ番号ごとに取得	getTimeDataMX
指定したFIFO番号の測定データを取得	getChDataMX

測定データは，FIFOが動作中にのみ取得することができます。

初期バランスデータの取得

機能	関数
初期バランスデータ　一括取得	getBalanceMX

出力チャネルデータの取得

機能	関数
出力チャネルデータ　一括取得	getOutputMX

ユーティリティ

機能	関数
指定したユーザカウント(ユーザーが定義した順序情報)を，次に発行するパケットに挿入	setUserTimeMX
通信で最後に受信したMX100固有エラーを取得	getLastErrorMX
測定値を倍精度浮動小数に変換	toDoubleValueMX
測定値を文字列に変換	toStringValueMX
アラーム種類の文字列を取得	toAlarmNameMX getAlarmNameMX
本APIのバージョン番号を取得	getVersionAPIMX
本APIのリビジョン番号を取得	getRevisionAPIMX
エラーメッセージ文字列を取得	getErrorMessageMX toErrorMessageMX
エラーメッセージ文字列の最大長を取得	getMaxLenErrorMessageMX
(指定したデータ番号)+(指定した数値)のデータ番号を生成	incrementDataNoMX
(指定したデータ番号)−(指定した数値)のデータ番号を生成	decrementDataNoMX
指定した2つのデータ番号を比較	compareDataNoMX
時刻情報を年月日時分秒の値に変換	toDateTimeMX
エラー検出した設定項目番号を取得	getItemErrorMX
アラーム文字列の最大長を取得	getMaxLenAlarmNameMX
AO/PWM　出力値を出力データ値に変換	toAOPWMValueMX
出力データ値を出力値に変換	toRealValueMX
データ番号の有効性チェック	isDataNoMX
スタイルバージョンに変換	toStyleVersionMX

3.2　プログラムーMX100/Visual Cー

インクルードファイルのパスを追加

プロジェクトに，インクルードファイル(DAQMX.h)のパスを追加します。追加方法は，ご使用の環境により異なります。

ソースファイルでの宣言

ソースファイルに宣言を記述します。

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

Note

共通部のインクルードファイル(DAQHandler.h)は，上記インクルードファイルから参照されているので，宣言を記述する必要はありません。

ロードライブラリの記述

本APIの実行可能モジュール(.dll)がプロセスとリンクできるようにするため，下記の記述をします。

本APIの実行可能モジュール(.dll)をアドレス空間内にマップします(LoadLibrary)。次に，実行可能モジュール内のエクスポート関数のアドレスを取得(GetProcAddress)します。

関数ポインタのコールバック型は，関数名に接頭語「DLL」をつけてすべて大文字にしたものです。本APIのインクルードファイルで定義されています。

```
HMODULE pDll = LoadLibrary("DAQMX");
DLLOPENMX openMX = (DLLOPENMX)GetProcAddress(pDll, "openMX");
```

測定データの取得

プログラム例1

測定データを取得するプログラムです。

```
////////////////////////////////////////////////////////////
// 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;
  //laod
  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 { //meaasured 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;
}
////////////////////////////////////////////////////////////////
```

説明

全般

データ取得は，FIFOを開始することで可能になります。取得範囲はFIFO番号とデータ番号で指定します。データ番号に対応する時刻と，測定データを個別に取得します。終了はフラグにより判断します。

インクルードファイルの記述

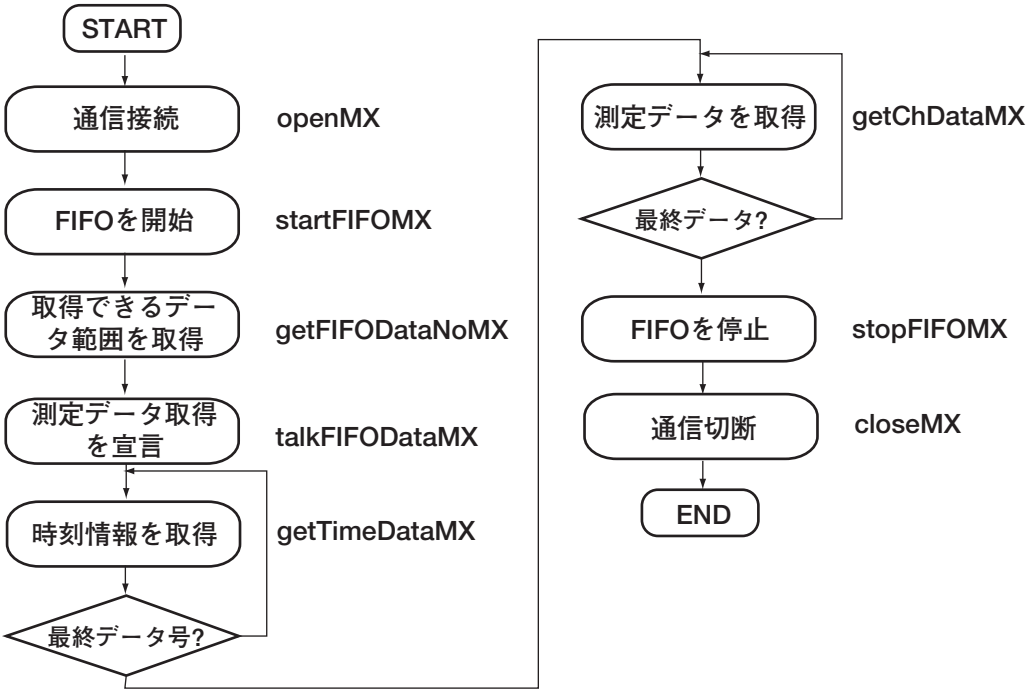
```
#include "DAQMX.h"
```

ロードライブラリの記述

`#ifdef WIN32`から`#endif //WIN32`までがロードライブラリの記述です。コールバック型(DLLOPENMXなど)を使用しています。

処理の流れ

下記のフローチャートでは，宣言部分を省略しています。



通信処理

最初に通信接続を行います。通信接続後，各関数が利用可能です。最後に終了処理として，通信切断を行います。

Note

- 約3分間アクセスがない場合，MX100が通信を切断します。長時間アクセスをしない場合には通信を切断し，必要なときに通信接続してください。
- 通信接続を継続したい場合は，適度にステータス取得を実行してください。

通信接続

```
openMX("192.168.1.12", &rc)
```

MX100のIPアドレスを指定しています。
通信用ポートは，通信用定数の「MX100の通信ポート番号」を指定したことになります。

FIFO開始

```
startFIFOMX(comm)
```

FIFOを開始します。

データ範囲の取得

```
getFIFODataNoMX(comm, 0, &startNo, &endNo)
```

指定したFIFO番号の，最後に取得したデータの次のデータから最新データまでの範囲を，データ番号で取得します。

トーカ

```
talkFIFODataMX(comm, 0, startNo, endNo)
```

データ範囲を指定して，FIFOデータの取得を宣言します(測定データ取得宣言)。

FIFOデータ時刻情報の取得

```
getTimeDataMX(comm, &dataNo, &datetime, &usertime, &flag)
```

指定範囲の時刻情報を，データ番号単位で取得します。
終了はフラグステータスの「最終データ」により判断します。

Note

usertimeは，ユーザーによる順序情報(ユーザーカウント)です。あらかじめ，setUserTimeMX関数で設定した値が返却されてきます。

FIFOデータの取得

```
getChDataMX(comm, &dataNo, &chinfo, &datainfo, &flag)
```

指定範囲の測定データを，チャネル単位で取得します。
終了はフラグステータスの「最終データ」により判断します。

FIFO停止

```
stopFIFOMX(comm)
```

FIFOを停止します。

通信切断

```
closeMX(comm)
```

通信を切断します。

設定データの取得/設定

プログラム例2

下記の3つを実行するプログラムです。このプログラムではまとめて記述していますが，それぞれ個別に記述して実行できます。

・ 設定データを一括取得
・ 設定データを一括設定
・ チャネルに直流電圧レンジを設定

```
////////////////////////////////////////////////////////////
// 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;
}
////////////////////////////////////////////////////////////
```

説明

設定データの一括取得

`getConfigDataMX(comm, &configdata)`

設定データの一括取得では，下記の設定データを取得できます。取得できる項目の詳細については，6.4節，「MX100の型」をご覧ください。

- システム構成データ：MXSystemInfo構造体に格納されます。
- ステータス：MXStatus構造体に格納されます。
- 基本設定：システムの基本的な設定内容です。

Note

設定データの取得方法には，talkConfigMXとgetChConfigMXを使用する方法もあります。talkConfigMXで設定データ取得宣言をして，システム構成データ，ステータス，ネットワーク情報データを取得し，getChConfigMXでチャネル設定データをチャネル単位で取得します。

設定データの一括設定

`setConfigDataMX(comm, &configdata)`

設定データの一括設定では，下記のデータを設定できます。設定できる項目の詳細については，6.4節，「MX100の型」をご覧ください。

- システム構成データ：MXSystemInfo構造体の内容です。
- 基本設定：システムの基本的な設定内容です。

チャネルに直流電圧レンジを設定

`setVOLTMX(comm, DAQMX_RANGE_VOLT_20MV, 1, 1, 0, 0, 0, 0, 0)`

チャネル番号1に，直流電圧レンジ「20mV」を設定します。スケーリングは使用しません。

エラー処理

- ほとんどの関数は，戻り値として，関数の処理結果の状態をエラー番号で返します(正常終了の場合は0)。
- エラー番号に対応するエラーメッセージ文字列を得ることができる関数(getErrorMessageMX)があります。また，エラーメッセージ文字列の最大長を得る関数(getMaxLenErrorMessageMX)もあります。
- MX100からのMX100固有エラーは，関数(getLastErrorMX)で取得できます。
- 設定でデータ不正のエラーになった場合，エラー検出した設定項目番号を関数で取得できます。

4.1　機能と関数の対応－MX100/Visual Basic－

本APIでサポートする機能と，Visual Basicの関数群の対応を示します。

通信機能

機能	関数
MX100と通信接続	openMX
MX100との通信を切断	closeMX
通信タイムアウトを設定	setTimeOutMX

Note

通信タイムアウトの設定を推奨しません。**理由**：データ取得時にタイムアウト時間に抵触して予期しない通信切断が発生する場合があります。

制御機能

FIFOの開始/停止

機能	関数
FIFOを開始	startFIFOMX
FIFOを停止	stopFIFOMX
FIFOの自動制御を設定	autoFIFOMX

そのほかの制御

機能	FIFO	関数
MX100に時刻情報(基準日時(1970年1月1日)からの時間)を秒数で設定	停止	setDateTimeMX
MX100に現在の日付/時刻を設定	停止	setDateTimeNowMX
CFへのデータ保存(バックアップ)のON/OFFを設定	継続	setBackupMX
CFをフォーマット	停止	formatCFMX
・ユニットのシステム再構築 ・ユニットのシステム初期化 ・ユニットのアラームリセット(アラームACK)	停止 停止 継続	initSystemMX
7セグメントLEDの表示を設定	継続	setSegmentMX

表の「FIFO」欄は，FIFO中に関数を実行したときの，FIFOの動作を示します。
停止：関数を実行するとFIFOを停止します。
継続：関数を実行してもFIFOは継続します。
バックアップの設定は，CF書き込み種類が変更された場合，FIFOは停止します。
バックアップの設定のCF書き込み種類は，設定変更の操作(一括取得したデータを変更して一括送信)をします。

Note

FIFOの自動制御を設定しておくと，関数の実行によりFIFOが停止したあと，FIFOを自動的に再開します。

設定機能

一括設定

機能	FIFO	関数
設定データ(システム設定データ)を設定	停止	setSystemConfigMX
設定データ(チャネル設定データ)を設定	停止	setChConfigMX
DO(Digital Output)データを一括設定	継続	setDODataMX
AO/PWMデータ送信	継続	setAOPWMDataMX
伝送出力データ送信	継続	setTransmitMX

表の「FIFO」欄については，前ページの「そのほかの制御」の説明をご覧ください。
設定データは一括して扱うことができません。
データ不正のエラー番号が発生した場合，最後の検出箇所を設定項目番号で保持します。ユーティリティで取得できます。

個別設定

機能		FIFO	関数
初期バランスデータを設定		停止	setBalanceMX
出力チャネルデータを設定		停止	setOutputMX
初期バランスデータ	実行	停止	runBalanceMX
	リセット	停止	resetBalanceMX

設定変更

機能		FIFO	関数
レンジ設定	スキップ(未使用)	停止	setSKIPMX
	直流電圧入力	停止	setVOLTMX
	熱電対入力	停止	setTCMX
	測温抵抗体入力	停止	setRTDMX
	ディジタル入力(DI)	停止	setDIMX
	チャネル間差演算	停止	setDELTAMX
	チャネルにリモートRJC	停止	setRRJCMX
	抵抗	停止	setRESMX
	ひずみ	停止	setSTRAINMX
	AO	停止	setAOMX
	PWM	停止	setPWMMX
	パルス	停止	setPULSEMX
	通信	停止	setCOMMX
チャネルに単位名を設定		停止	setScallingUnitMX
チャネルのタグを設定		停止	setTagMX
チャネルにコメントを設定		停止	setCommentMX
チャネルにアラームを設定		停止	setAlarmMX
チャネルで使用する基準接点補償(RJC)を設定		停止	setRJCTypeMX

機能		FIFO	関数
チャネルにフィルタを設定		停止	setFilterMX
チャネルにバーンアウト検出時動作を設定		停止	setBurnoutMX
アラーム出力を指定したDOチャネルに，関連づけるアラームを設定 (DOチャネルをアラーム出力に指定するときは，setDOTypeMX関数を用います。)		停止	setRefAlarmMX
測定周期を設定		停止	setIntervalMX
温度単位を設定		停止	setTempUnitMX
ユニットの識別番号を設定		停止	setUnitNoMX
タイムアウト値(通信切断時に，CFへのデータ保存を開始するまでの時間)を設定 タイムアウト値の算出については，付録3を参照してください。		停止	setSystemTimeoutMX
DOチャネルに関連づける信号の種類を設定		停止	setDOTypeMX
チャネルにAO種類を設定		停止	setAOTypeMX
チャネルにPWM種類を設定		停止	setPWMTypeMX
出力チャネルデータ	出力種類	停止	setOutputTypeMX
	選択値	停止	setChoiceMX
	パルス周期倍率	停止	setPulseTimeMX
DOデータを部分変更		継続	changeDODataMX
AO/PWMデータの部分変更		継続	changeAOPWMDataMX
初期バランスデータの部分変更		継続	changeBalanceMX
伝送出力データの部分変更		継続	changeTransmitMX
チャネルにチャタリングフィルタを設定		停止	setChatFilterMX

個別に自動送信しています。必ずFIFOは停止します。
表の「FIFO」欄については，4-1ページの「そのほかの制御」の説明をご覧ください。

データ取得機能

システムステータスデータ/システム構成データの取得

機能	関数
システムステータスデータを取得	getStatusDataMX
システム構成データを取得	getSystemConfigMX

設定データの取得

機能	関数
設定データの取得を宣言 チャネル設定データ以外の設定データを取得します。	talkConfigMX
チャネル設定データを取得 talkConfigMX関数で設定データの取得を宣言したあとにチャネル設定データを取得する関数です。	getChConfigMX

DOデータの取得

機能	関数
DOデータを一括取得	getDODataMX

AO/PWMデータ，伝送出力データの取得

機能	関数
AO/PWMデータ，伝送出力データの一括取得	getAOPWMDataMX

チャネル情報データの取得

機能	関数
チャネル情報データの取得を宣言	talkChInfoMX
チャネル情報データの取得	getChInfoMX

測定データの取得(チャネル指定)

機能	関数
指定したチャネルの最新のデータ範囲を取得	getChDataNoMX
指定したチャネルの測定データ取得を宣言	talkChDataVBMX
指定したチャネルの瞬時値取得を宣言	talkChDataInstMX
指定したチャネルの時刻情報をデータ番号ごとに取得	getTimeDataMX
指定したチャネルの測定データを取得	getChDataMX

測定データの取得(FIFO指定)

機能	関数
指定したFIFO番号の最新のデータ範囲を取得	getFIFODataNoMX
指定したFIFO番号の測定データ取得を宣言	talkFIFODataVBMX
指定したFIFO番号の瞬時値取得を宣言	talkFIFODataInstMX
指定したFIFO番号の時刻情報をデータ番号ごとに取得	getTimeDataMX
指定したFIFO番号の測定データを取得	getChDataMX

測定データは， FIFOが動作中にのみ取得することができます。

初期バランスデータ,

機能	関数
初期バランスデータの一括取得	getBalanceMX

出力チャネルデータの取得

機能	関数
出力チャネルデータの一括取得	getOutputMX

ユーティリティ

機能	関数
指定したユーザカウント(ユーザーが定義した順序情報)を，次に発行するパケットに挿入	setUserTimeVBMX
通信で最後に受信したMX100固有エラーを取得	getLastErrorMX
測定値を倍精度浮動小数に変換	toDoubleValueMX
測定値を文字列に変換	toStringValueMX
アラーム種類の文字列を取得	toAlarmNameMX
本APIのバージョン番号を取得	getVersionAPIMX
本APIのリビジョン番号を取得	getRevisionAPIMX
エラーメッセージ文字列を取得	toErrorMessageMX
エラーメッセージ文字列の最大長を取得	getMaxLenErrorMessageMX
(指定したデータ番号)+(指定した数値)のデータ番号を生成	incrementDataNoMX
(指定したデータ番号)ー(指定した数値)のデータ番号を生成	decrementDataNoMX
指定した2つのデータ番号を比較	compareDataNoMX
時刻情報を年月日時分秒の値に変換	toDateTimeMX
エラー検出した設定項目番号を取得	getItemErrorMX
アラーム文字列の最大長を取得	getMaxLenAlarmNameMX
AO/PWM　出力値を出力データ値に変換	toAOPWMValueMX
出力データ値を出力値に変換	toRealValueMX
データ番号の有効性チェック	isDataNoVBMX
スタイルバージョンに変換	toStyleVersionMX

4.2　プログラムーMX100/Visual Basicー

型，関数，定数の宣言

Visual Basic用の型，関数，定数を使用するためには，あらかじめ宣言をしておく必要があります。次の宣言記述方法があります。

全宣言の記述

プロジェクトにVisual Basic用標準モジュールライブラリファイル(DAQMX.bas)を追加すると，すべての型，関数，定数を宣言したことになります。

宣言の選択記述

Visual Studioに付属しているAPIビューアで，任意の型，関数，定数の宣言記述をコピーできます。この機能を使用するためには，APIビューアで，APIビューア用テキストファイル(DAQMX.txt)を読み込んでください。
APIビューアの使用方法については，Visual Studioの取扱説明書をご覧ください。

宣言の直接記述

記述例を示します。

```
Public Declare Function openMX Lib "DAQMX" (ByVal strAddress As
String, ByRef errorCode As Long) As Long
```

測定データの取得

プログラム例1

測定データを取得するプログラムです。

```
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
```

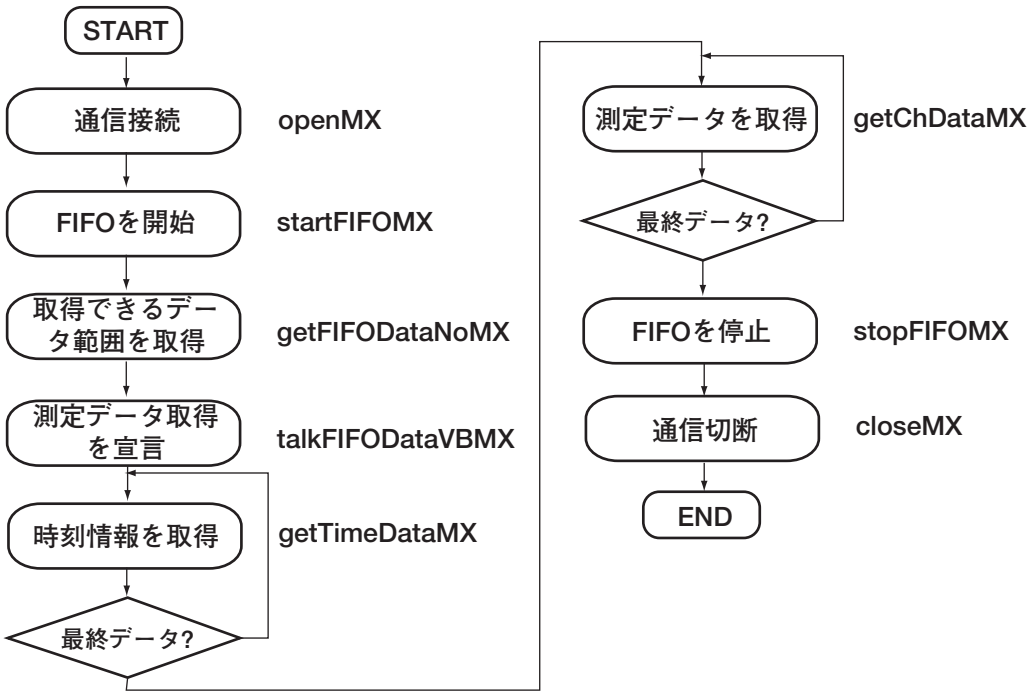
説明

全般

データ取得は，FIFOを開始することで可能になります。取得範囲はFIFO番号とデータ番号で指定します。データ番号に対応する時刻と，測定データを個別に取得します。終了はフラグにより判断します。

処理の流れ

下記のフローチャートでは，宣言部分を省略しています。



通信処理

最初に通信接続を行います。通信接続後，各関数が利用可能です。最後に終了処理として，通信切断を行います。

Note

約3分間アクセスがない場合，MX100が通信を切断します。長時間アクセスをしない場合には通信を切断し，必要なときに通信接続してください。通信接続を継続したいときは，適度にステータス取得を実行してください。

通信接続

`openMX(host, rc)`

MX100のIPアドレスを指定しています。

通信用ポートは，通信用定数の「MX100の通信ポート番号」を指定したことになります。

FIFO開始

`startFIFOMX(comm)`

FIFOを開始します。

データ範囲の取得

```
getFIFODataNoMX(comm, 0, startNo, endNo)
```

指定したFIFO番号の，最後に取得したデータの次のデータから最新データまでの範囲を，データ番号で取得します。

トーカ

```
talkFIFODataVBMX(comm, 0, startNo, endNo)
```

データ範囲を指定して，FIFOデータの取得を宣言します(測定データ取得宣言)。

FIFOデータ時刻情報の取得

```
getTimeDataMX(comm, dataNo, datetime, usertime, flag)
```

指定範囲の時刻情報を，データ番号単位で取得します。
終了はフラグステータスの「最終データ」により判断します。

Note

usertimeは，ユーザーによる順序情報(ユーザーカウント)です。あらかじめ，setUserTimeVBMX関数で設定した値が返却されてきます。

FIFOデータの取得

```
getChDataMX(comm, dataNo, chinfo, datainfo, flag)
```

指定範囲の測定データを，チャネル単位で取得します。
終了はフラグステータスの「最終データ」により判断します。

FIFO停止

```
stopFIFOMX(comm)
```

FIFOを停止します。

通信切断

```
closeMX(comm)
```

通信を切断します。

設定データの取得/設定

プログラム例2

下記の4つを実行するプログラムです。このプログラムではまとめて記述していますが，それぞれ個別に記述して実行できます。

- 設定データを取得
- チャネル設定データ以外の設定データを一括設定
- チャネル設定データを設定
- チャネルに直流電圧レンジを設定

```
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
```

説明

設定データの取得

`talkConfigMX(comm, sysinfo, status, netinfo)`

設定データの取得を宣言します。

チャネル設定データを除く設定データ(システム構成データ，ステータス，ネットワーク情報データ)を取得します。

システム構成データはMXSystemInfo構造体に，ステータスはMXStatus構造体に，ネットワーク情報データはMXNetInfo構造体に格納されます。

`getChConfigMX(comm, chconfig, flag)`

チャネル設定データをチャネルごとに取得します。

終了はフラグステータスの「最終データ」により判断します。

チャネル設定データ以外の設定データを一括設定

```
setSystemConfigMX(comm, sysinfo)
```

指定したMXSystemInfo構造体の内容を，MX100に設定します。

チャネル設定データを設定

```
setChConfigMX(comm, chconfig)
```

指定したMXChConfig構造体の内容を，MX100に設定します。

チャネルに直流電圧レンジを設定

```
setVOLTMX(comm, DAQMX_RANGE_VOLT_20MV, 1, 1, 0, 0, 0, 0, 0)
```

チャネル番号1に，直流電圧レンジ「20mV」を設定します。スケーリングは使用しません。

エラー処理

- ほとんどの関数は，戻り値として，関数の処理結果の状態をエラー番号で返します(正常終了の場合は0)。
- エラー番号に対応するエラーメッセージ文字列を得ることができる関数(toErrorMessageMX)があります。 また，エラーメッセージ文字列の最大長を得る関数(getMaxLenErrorMessageMX)もあります。
- MX100からのMX100固有エラーは，関数(getLastErrorMX)で取得できます。
- 設定でデータ不正のエラーになった場合，エラー検出した設定項目番号を関数で取得できます。

5.1　関数の詳細－MX100(Visual C/Visual Basic)－

ここでは，Visual CとVisual Basicで使用するMX100用関数について説明しています。関数は，関数名のアルファベット順で並んでいます。

定数，型については第6章をご覧ください。
MX100の用語については付録1をご覧ください。

ほとんどの関数は戻り値として，エラー番号を返します。正常終了の場合は，エラー番号「0」を返します。

autoFIFOMX

構文

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

宣言

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

引数

daqmx	機器記述子を指定します。
bAuto	有効無効値を指定します。

説明

FIFOの自動制御を設定します。

戻り値

エラー番号を返します。

エラー：

Not descriptor　機器記述子がありません。

参照

`CDAQMX::autoFIFO`

changeAOPWMDataMX

構文

```
int changeAOPWMDataMX(MXAOPWMData * pMXAOPWMData, int aopwmNo,
int bValid, int iAOPWMValue);
```

宣言

```
Public Declare Function changeAOPWMDataMX Lib "DAQMX" (ByRef
pMXAOPWMData As MXAOPWMData, ByVal aopwmNo As Long, ByVal
bValid As Long, ByVal iAOPWMValue As Long) As Long
```

引数

pMXAOPWMData	AO/PWMデータを指定します。
aopwmNo	AO/PWMデータ番号を指定します。
bValid	有効無効値を指定します。
iAOPWMValue	出力データ値を指定します。

説明

指定されたAO/PWMデータのデータを変更します。

- AO/PWMデータ番号に，定数値の「全AO/PWM番号指定」をすると，全データを変更します。
- AO/PWMデータ番号が，範囲外で無効な場合，指定は無視されます。
- 出力データ値は，実際の出力値を変換した値を指定します。
- 正常終了します。

戻り値

エラー番号を返します。

参照

```
CDAQMXAOPWMData::getMXAOPWMData
CDAQMXAOPWMData::setAOPWM
```

changeBalanceMX

構文

```
int changeBalanceMX(MXBalanceData * pMXBalanceData, int
balanceNo, int bValid, int iValue);
```

宣言

```
Public Declare Function changeBalanceMX Lib "DAQMX" (ByRef
pMXBalanceData As MXBalanceData, ByVal balanceNo As Long,
ByVal bValid As Long, ByVal iValue As Long) As Long
```

引数

pMXBalanceData	初期バランスデータを指定します。
balanceNo	初期バランスデータ番号を指定します。
bValid	有効無効値を指定します。
iValue	初期バランス値を指定します。

説明

指定された初期バランスデータのデータを変更します。

- 初期バランスデータ番号に，定数値の「全初期バランス番号指定」をすると，全データを変更します。
- 初期バランスデータ番号が，範囲外で無効な場合，指定は無視されます。
- 正常終了します。

戻り値

エラー番号を返します。

参照

```
CDAQMXBalanceData::getMXBalanceData
CDAQMXBalanceData::setBalance
```

changeDODataMX

構文

```
int changeDODataMX(MXDOData * pMXDOData, int doNo, int bValid,
int bONOFF);
```

宣言

```
Public Declare Function changeDODataMX Lib "DAQMX" (ByRef
pMXDOData As MXDOData, ByVal doNo As Long, ByVal bValid As
Long, ByVal bONOFF As Long) As Long
```

引数

pMXDOData	DOデータを指定します。
doNo	DOデータ番号を指定します。
bValid	有効無効値を指定します。
bONOFF	有効無効値を指定します。

説明

指定されたDOデータのDOデータ番号のデータを変更します。

- DOデータを送信した場合，引数bValidで「有効」に指定されたDOデータに，引数bONOFFで指定された値が出力されます。
- DOデータ番号が無効な場合，指定は無視されます。
- 正常終了します。

戻り値

エラー番号を返します。

参照

```
CDAQMXDOData::setDO
CDAQMXDOData::CDAQMXDOData
```

changeTransmitMX

構文

```
int changeTransmitMX(MXTransmit * pMXTransmit, int aopwmNo,
int iTrans);
```

宣言

```
Public Declare Function changeTransmitMX Lib "DAQMX" (ByRef
pMXTransmit As MXTransmit, ByVal pwmNo As Long, ByVal iTrans
As Long) As Long
```

引数

pMXTransmit	伝送出力データを指定します。
aopwmNo	AO/PWMデータ番号を指定します。
iTrans	伝送状態を指定します。

説明

指定された伝送出力データのデータを変更します。

- AO/PWMデータ番号に，定数値の「全AO/PWMデータ番号指定」をすると，全データを変更します。
- AO/PWMデータ番号が範囲外で無効な場合，指定は無視されます。
- 正常終了します。

戻り値

エラー番号を返します。

参照

```
CDAQMXTransmit::getMXTransmit
CDAQMXTransmit::setTransmit
```

closeMX

構文

```
int closeMX(DAQMX daqmx);
```

宣言

```
Public Declare Function closeMX Lib "DAQMX" (ByVal daqmx As
Long) As Long
```

引数

daqmx　　機器記述子を指定します。

説明

指定された機器記述子による通信を切断をします。
通信を切断すると，機器記述子の値は無意味です。機器記述子として使用しないでください。

戻り値

エラー番号を返します。
エラー：
Not descriptor　機器記述子がありません。

参照

`CDAQMX::close`

compareDataNoMX

構文

```
int compareDataNoMX(MXDataNo * prevDataNo, MXDataNo *
nextDataNo);
```

宣言

```
Public Declare Function compareDataNoMX Lib "DAQMX" (ByRef
prevDataNo As MXDataNo, ByRef nextDataNo As MXDataNo) As Long
```

引数

prevDataNo　　データ番号(前)を指定します。
nextDataNo　　データ番号(後)を指定します。

説明

指定されたデータ番号を比較します。

- データ番号が64ビットに対して，戻り値は32ビットなので，差分を返すわけではありません。
- Visual Cの場合，引数にNULLが指定されると，不定な値を返します。

戻り値

データ番号が同じ場合，0を返します。
データ番号(前)がデータ番号(後)より小さい場合，正の数を返します。
データ番号(前)がデータ番号(後)より大きい場合，負の数を返します。

decrementDataNoMX

構文

```
void decrementDataNoMX(MXDataNo * dataNo, int decrement);
```

宣言

```
Public Declare Sub decrementDataNoMX Lib "DAQMX" (ByRef dataNo
As MXDataNo, ByVal decrement As Long)
```

引数

dataNo	データ番号を指定します。
decrement	デクリメントする数値を指定します。

説明

指定されたデータ番号を指定された数だけデクリメントします。指定されたデータ番号の領域が変更されます。

formatCFMX

構文

```
int formatCFMX(DAQMX daqmx);
```

宣言

```
Public Declare Function formatCFMX Lib "DAQMX" (ByVal daqmx As
Long) As Long
```

引数

daqmx　　機器記述子を指定します。

説明

CF(Compact Flash)をフォーマットします。

・ 本関数は，応答に数秒以上の時間がかかることがあります。かかる時間はメディアにより異なります。

戻り値

エラー番号を返します。

エラー：

Not descriptor　機器記述子がありません。

参照

`CDAQMX::formatCF`

getAlarmNameMX [Visual Cのみ]

構文

```
const char * getAlarmNameMX(int iAlarmType);
```

引数

iAlarmType アラーム種類を指定します。

説明

指定されたアラーム種類に対応する文字列を取得します。

・ Visual Basicの場合， toAlarmNameMX関数を使用してください。

戻り値

文字列へのポインタを返します。

参照

`CDAQMXDataInfo::getAlarmName`

getAOPWMDataMX

構文

```
int getAOPWMDataMX(DAQMX daqmx, MXAOPWMData * pMXAOPWMData,
MXTransmit * pMXTransmit);
```

宣言

```
Public Declare Function getAOPWMDataMX Lib "DAQMX" (ByVal
daqmx As Long, ByRef pMXAOPWMData As MXAOPWMData, ByRef
pMXTransmit As MXTransmit) As Long
```

引数

daqmx	機器記述子を指定します。
pMXAOPWMData	AO/PWMデータの返却先を指定します。
pMXTransmit	伝送出力データの返却先を指定します。

説明

AO/PWMデータと伝送出力データを一括取得します。

・ 返却先が指定されていれば，取得したデータを指定先に格納します。

戻り値

エラー番号を返します。

エラー ：

Not descriptor	機器記述子がありません。

参照

```
CDAQMX::getAOPWMData
CDAQMXAOPWMData::getMXAOPWMData
CDAQMXTransmit::getMXTransmit
```

getBalanceMX

構文

```
int getBalanceMX(DAQMX daqmx, MXBalanceData * pMXBalanceData);
```

宣言

```
Public Declare Function getBalanceMX Lib "DAQMX" (ByVal daqmx
As Long, ByRef pMXBalanceData As MXBalanceData) As Long
```

引数

daqmx	機器記述子を指定します。
pMXBalanceData	初期バランスデータの返却先を指定します。

説明

初期バランスデータを一括取得します。

・ 返却先が指定されていれば， 初期バランスデータを指定先に格納します。

戻り値

エラー番号を返します。

エラー：

Not descriptor	機器記述子がありません。

参照

```
CDAQMX::getBalance
CDAQMXBalanceData::getMXBalanceData
```

getChConfigMX

構文

```
int getChConfigMX(DAQMX daqmx, MXChConfig * pMXChConfig, int *
pFlag);
```

宣言

```
Public Declare Function getChConfigMX Lib "DAQMX" (ByVal daqmx
As Long, ByRef pMXChConfig As MXChConfig, ByRef pFlag As Long)
As Long
```

引数

daqmx	機器記述子を指定します。
pMXChConfig	チャネル設定データの返却先を指定します。
pFlag	フラグの返却先を指定します。

説明

talkConfigMX関数で取得を宣言したチャネル設定データを，チャネル単位で取得します。

- 最終データを取得した場合，フラグにフラグステータスがセットされます。
- データ取得を終了するまでは，他関数で通信を行わないでください。本関数でデータ取得中は，他関数が正しく動作できません。

戻り値

エラー番号を返します。

エラー：

Not descriptor	機器記述子がありません。

参照

```
CDAQMX::getChConfig
CDAQMXChConfig::getMXChConfig
```

getChDataMX

構文

```
int getChDataMX(DAQMX daqmx, MXDataNo * pMXDataNo, MXChInfo *
pMXChInfo, MXDataInfo * pMXDataInfo, int * pFlag);
```

宣言

```
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
```

引数

daqmx	機器記述子を指定します。
pMXDataNo	データ番号の返却先を指定します。
pMXChInfo	チャネル情報データの返却先を指定します。
pMXDataInfo	測定データの返却先を指定します。
pFlag	フラグの返却先を指定します。

説明

データ取得開始関数(talkChDataMX関数，talkFIFODataMX関数など)で宣言したあと，さらにgetTimeDataMX関数で時刻情報データを取得したあと，測定データをデータごとに取得します。

- 最終データを取得した場合，フラグにフラグステータスがセットされます。
- データ取得を終了するまでは，他関数で通信を行わないでください。本関数でデータ取得中は，他関数が正しく動作できません。

戻り値

エラー番号を返します。

エラー：

Not descriptor　機器記述子がありません。

参照

```
CDAQMX::getChData
CDAQMXChInfo::getMXChInfo
CDAQMXDataInfo::getMXDataInfo
```

getChDataNoMX

構文

```
int getChDataNoMX(DAQMX daqmx, int chNo, MXDataNo *
startDataNo, MXDataNo * endDataNo);
```

宣言

```
Public Declare Function getChDataNoMX Lib "DAQMX" (ByVal daqmx
As Long, ByVal chNo As Long, ByRef startDataNo As MXDataNo,
ByRef endDataNo As MXDataNo) As Long
```

引数

daqmx	機器記述子を指定します。
chNo	チャネル番号を指定します。
startDataNo	開始データ番号の返却先を指定します。
endDataNo	終了データ番号の返却先を指定します。

説明

指定されたチャネルで，測定データを取得できるデータ範囲を取得します。

- 開始データ番号は， 最後に取得したデータ番号の次になります。
- 取得すべき測定データが存在しない場合， 各データ番号の返却先に負の数を返します。

  例：

  機器本体にデータ番号「10」から「49」までのデータが存在し， 最後に取得したデータ番号が「29」だとした場合， 開始データ番号に「30」，終了データ番号に「49」を返します。

  ここで，最後に取得したデータ番号が「49」だとした場合， 各データ番号に負の数を返し， 正常終了します。
- 返却されたデータ番号の有効性チェックには，isDataNoMX関数を参照してください。

戻り値

エラー番号を返します。

エラー：

Not descriptor　機器記述子がありません。

参照

CDAQMX::getChDataNo

getChInfoMX

構文

```
int getChInfoMX(DAQMX daqmx, MXChInfo * pMXChInfo, int *
pFlag);
```

宣言

```
Public Declare Function getChInfoMX Lib "DAQMX" (ByVal daqmx
As Long, ByRef pMXChInfo As MXChInfo, ByRef pFlag As Long) As
Long
```

引数

daqmx	機器記述子を指定します。
pMXChInfo	チャネル情報データの返却先を指定します。
pFlag	フラグの返却先を指定します。

説明

talkChInfoMX関数で取得を宣言したチャネル情報データを，チャネル単位で取得します。

- 最終データを取得した場合，フラグにフラグステータスがセットされます。
- データ取得を終了するまでは，他関数で通信を行わないでください。本関数でデータ取得中は，他関数が正しく動作できません。

戻り値

エラー番号を返します。

エラー：

Not descriptor　機器記述子がありません。

参照

`CDAQMX::getChInfo`

getConfigDataMX [Visual Cのみ]

構文

```
int getConfigDataMX(DAQMX daqmx, MXConfigData *
pMXConfigData);
```

引数

daqmx 機器記述子を指定します。
pMXConfigData 設定データの返却先を指定します。

説明

設定データを一括取得します。

- 返却先が指定されていれば， 設定データを指定先に格納します。
- Visual Basicの場合，各データ構造体ごとに個別に一括取得を行ってください。または，逐次取得の手順で取得してください。

戻り値

エラー番号を返します。
エラー：
Not descriptor 機器記述子がありません。

参照

```
CDAQMX::getConfig
CDAQMXConfig::getMXConfigData
```

getDODataMX

構文

```
int getDODataMX(DAQMX daqmx, MXDOData * pMXDOData);
```

宣言

```
Public Declare Function getDODataMX Lib "DAQMX" (ByVal daqmx
As Long, ByRef pMXDOData As MXDOData) As Long
```

引数

daqmx　　機器記述子を指定します。
pMXDOData　DOデータの返却先を指定します。

説明

DOデータを一括取得します。
返却先が指定されていれば，DOデータを指定先に格納します。

戻り値

エラー番号を返します。
エラー：
Not descriptor　機器記述子がありません。

参照

```
CDAQMX::getDOData
CDAQMXDOData::getMXDOData
```

getErrorMessageMX [Visual Cのみ]

構文

```
const char * getErrorMessageMX(int errCode);
```

引数

errCode エラー番号を指定します。

説明

エラー番号に対応するエラーメッセージ文字列を取得します。
Visual Basicの場合，toErrorMessageMX関数を使用してください。

戻り値

文字列の長さを返します。

参照

`CDAQMX::getErrorMessage`

getFIFODataNoMX

構文

```
int getFIFODataNoMX(DAQMX daqmx, int fifoNo, MXDataNo *
startDataNo, MXDataNo * endDataNo);
```

宣言

```
Public Declare Function getFIFODataNoMX Lib "DAQMX" (ByVal
daqmx As Long, ByVal fifoNo As Long, ByRef startDataNo As
MXDataNo, ByRef endDataNo As MXDataNo) As Long
```

引数

daqmx	機器記述子を指定します。
fifoNo	FIFO番号を指定します。
startDataNo	開始データ番号の返却先を指定します。
endDataNo	終了データ番号の返却先を指定します。

説明

指定されたFIFO番号で，測定データを取得できるデータ範囲を取得します。
開始データ番号は，最後に取得したデータ番号の次になります。

- 開始データ番号は， 最後に取得したデータ番号の次になります。
- 取得すべき測定データが存在しない場合， 各データ番号の返却先に負の数を返します。
  例：
  機器本体にデータ番号「10」から「49」までのデータが存在し， 最後に取得したデータ番号が「29」だとした場合， 開始データ番号に「30」，終了データ番号に「49」を返します。
  ここで，最後に取得したデータ番号が「49」だとした場合， 各データ番号に負の数を返し， 正常終了します。
- 返却されたデータ番号の有効性チェックには，isDataNoMX関数を参照してください。

戻り値

エラー番号を返します。
エラー：
Not descriptor　機器記述子がありません。

参照

`CDAQMX::getFIFODataNo`

getItemErrorMX

構文

```
int getItemErrorMX(DAQMX daqmx, int * itemErr);
```

宣言

```
Public Declare Function getItemErrorMX Lib "DAQMX" (ByVal
daqmx As Long, ByRef itemErr As Long) As Long
```

引数

daqmx	機器記述子を指定します。
itemErr	設定項目番号の返却先を指定します。

説明

最後にエラー検出した設定項目番号を取得します。

・ 指定された返却先に設定項目番号を返します。

戻り値

エラー番号を返します。

エラー:

Not descriptor　機器記述子がありません。

参照

`CDAQMX::getItemError`

getLastErrorMX

構文

```
int getLastErrorMX(DAQMX daqmx, int * lastErr);
```

宣言

```
Public Declare Function getLastErrorMX Lib "DAQMX" (ByVal
daqmx As Long, ByRef lastErr As Long) As Long
```

引数

daqmx	機器記述子を指定します。
lastErr	MX100固有エラーの返却先を指定します。

説明

最後に通信で受信したMX100固有エラーを取得します。

戻り値

エラー番号を返します。

エラー：

Not descriptor	機器記述子がありません。

参照

getMaxLenAlarmNameMX

構文

```
int getMaxLenAlarmNameMX(void);
```

宣言

```
Public Declare Function getMaxLenAlarmNameMX Lib "DAQMX" () As
Long
```

説明

アラーム種類の文字列の最大長を取得します。

・ 戻り値に終端は含まれません。

戻り値

文字列の長さを返します。

参照

`CDAQMXDataInfo::getMaxLenAlarmName`

getMaxLenErrorMessageMX

構文

```
int getMaxLenErrorMessageMX(void);
```

宣言

```
Public Declare Function getMaxLenErrorMessageMX Lib "DAQMX" ()
As Long
```

説明

エラーメッセージ文字列の最大長を取得します。

・ 戻り値に終端は含まれません。

戻り値

文字列の長さを返します。

参照

```
CDAQMX::getMaxLenErrorMessage
```

getOutputMX

構文

```
int getOutputMX(DAQMX daqmx, MXOutputData * pMXOutputData);
```

宣言

```
Public Declare Function getOutputMX Lib "DAQMX" (ByVal daqmx
As Long, ByRef pMXOutputData As MXOutputData) As Long
```

引数

daqmx	機器記述子を指定します。
pMXOutputData	出力チャネルデータの返却先を指定します。

説明

出力チャネルデータを取得します。

・ 返却先が指定されていれば，取得したデータを指定先に格納します。

戻り値

エラー番号を返します。

エラー：

Not descriptor	機器記述子がありません。

参照

```
CDAQMX::getMXConfig
CDAQMXConfig::getClassMXOutputData
CDAQMXOutputData::getMXOutputData
CDAQMXOutputData::setMXOutputData
```

getRevisionAPIMX

構文

```
const int getRevisionAPIMX(void);
```

宣言

```
Public Declare Function getRevisionAPIMX Lib "DAQMX" () As
Long
```

説明

本APIのリビジョン番号を取得します。

戻り値

本APIのリビジョン番号を整数値で返します。

参照

`CDAQMX::getRevisionAPIMX`

getStatusDataMX

構文

```
int getStatusDataMX(DAQMX daqmx, MXStatus * pMXStatus);
```

宣言

```
Public Declare Function getStatusDataMX Lib "DAQMX" (ByVal
daqmx As Long, ByRef pMXStatus As MXStatus) As Long
```

引数

daqmx　　機器記述子を指定します。
pMXStatus　　ステータスの返却先を指定します。

説明

ステータスを取得します。

・ 返却先が指定されていれば，ステータスを指定先に格納します。

戻り値

エラー番号を返します。
エラー：
Not descriptor　機器記述子がありません。

参照

`CDAQMX::getStatusData`
`CDAQMXStatus::getMXStatus`

getSystemConfigMX

構文

```
int getSystemConfigMX(DAQMX daqmx, MXSystemInfo * pSysInfo);
```

宣言

```
Public Declare Function getSystemConfigMX Lib "DAQMX" (ByVal
daqmx As Long, ByRef pSysInfo As MXSystemInfo) As Long
```

引数

daqmx	機器記述子を指定します。
pSysInfo	システム構成データの返却先を指定します。

説明

システム構成データを取得します。

・ 返却先が指定されていれば， システム構成データを指定先に格納します。

戻り値

エラー番号を返します。

エラー：

Not descriptor　機器記述子がありません。

参照

```
CDAQMX::getSystemConfig
CDAQMXSysInfo::getMXSystemInfo
```

getTimeDataMX

構文

```
int getTimeDataMX(DAQMX daqmx, MXDataNo * pMXDataNo,
MXDateTime * pMXDateTime, MXUserTime * pMXUserTime, int *
pFlag);
```

宣言

```
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
```

引数

daqmx	機器記述子を指定します。
pMXDataNo	データ番号の返却先を指定します。
pMXDateTime	時刻情報データの返却先を指定します。
pMXUserTime	ユーザーカウントの返却先を指定します。
pFlag	フラグの返却先を指定します。

説明

データ取得開始関数(talkChDataMX関数，talkFIFODataMX関数など)で取得を宣言した時刻情報データを，データ番号毎に取得します。

- 最終データを取得した場合，フラグにフラグステータスがセットされます。
- データ取得を終了するまでは，他関数で通信を行わないでください。本関数でデータ取得中は，他関数が正しく動作できません。

戻り値

エラー番号を返します。

エラー：

Not descriptor	機器記述子がありません。

参照

```
CDAQMX::getTimeData
CDAQMXDateTime::getMXDateTime
```

getVersionAPIMX

構文

```
const int getVersionAPIMX(void);
```

宣言

```
Public Declare Function getVersionAPIMX Lib "DAQMX" () As Long
```

説明

本APIのバージョン番号を取得します。

戻り値

本APIのバージョン番号を整数値で返します。

参照

`CDAQMX::getVersionAPI`

incrementDataNoMX

構文

```
void incrementDataNoMX(MXDataNo * dataNo, int increment);
```

宣言

```
Public Declare Sub incrementDataNoMX Lib "DAQMX" (ByRef dataNo
As MXDataNo, ByVal increment As Long)
```

引数

dataNo	データ番号を指定します。
increment	インクリメントする数値を指定します。

説明

指定されたデータ番号を指定された数だけインクリメントします。指定されたデータ番号の領域が変更されます。

initSystemMX

構文

```
int initSystemMX(DAQMX daqmx, int iCtrl);
```

宣言

```
Public Declare Function initSystemMX Lib "DAQMX" (ByVal daqmx
As Long, ByVal iCtrl As Long) As Long
```

引数

daqmx	機器記述子を指定します。
iCtrl	システム制御種類を指定します。

説明

指定されたシステム制御を実行します。

戻り値

エラー番号を返します。

エラー：

Not descriptor	機器記述子がありません。

参照

`CDAQMX::initSystem`

isDataNoMX [Visual Cのみ]

構文

```
int isDataNoMX(MXDataNo dataNo);
```

引数

dataNo　　データ番号を指定します。

説明

指定されたデータ番号が有効な番号かどうかをチェックします。

・ データ番号が0以上の場合，「有効」を返します。

戻り値

有効無効値を返します。

参照

`CDAQMXStatus::isDataNo`

isDataNoVBMX

構文

```
int isDataNoVBMX(MXDataNo * dataNo);
```

宣言

```
Public Declare Function isDataNoVBMX Lib "DAQMX" (ByRef dataNo
As MXDataNo) As Long
```

引数

dataNo　　データ番号を指定します。

説明

指定されたデータ番号が有効な番号かどうかをチェックします。

・ データ番号の指定が，参照指定であること以外は，isDataNoMX関数と同じです。

・ Visual Cの場合，データ番号にNULLを指定すると，「無効」を返します。

戻り値

有効無効値を返します。

参照

isDataNoMX

openMX

構文

```
DAQMX openMX(const char * strAddress, int * errorCode);
```

宣言

```
Public Declare Function openMX Lib "DAQMX" (ByVal strAddress
As String, ByRef errorCode As Long) As Long
```

引数

strAddress　　IPアドレスを文字列で指定します。
errorCode　　エラー番号の返却先を指定します。

説明

引数で指定されたIPアドレスの機器と通信接続をします。

- 機器記述子を作成し，戻り値として返却します。
- 返却先が指定されていれば，エラー番号を格納します。
- 失敗した場合，Visual CではNULL，Visual Basicでは0を返します。

戻り値

機器記述子を返します。
エラー：
Creating descriptor is failure　　機器記述子の作成に失敗しました。

参照

`CDAQMX::open`

resetBalanceMX

構文

```
int resetBalanceMX(DAQMX daqmx, MXBalanceData *
pMXBalanceData, MXBalanceResult * pMXBalanceResult);
```

宣言

```
Public Declare Function resetBalanceMX Lib "DAQMX" (ByVal
daqmx As Long, ByRef pMXBalanceData As MXBalanceData, ByRef
pMXBalanceResult As MXBalanceResult) As Long
```

引数

daqmx	機器記述子を指定します。
pMXBalanceData	初期バランスデータを指定します。
pMXBalanceResult	初期バランス結果の返却先を指定します。

説明

初期バランス値を初期化します。

- 初期バランスデータの有効/無効の項が有効であるチャネルに対して実行します。
- 初期バランス値が初期バランスデータに返却されます。また，チャネル別の初期バランス結果を指定された返却先に返却します。このとき，初期バランスデータの有効/無効の項に「有効」を指定したチャネルのデータだけが上書きされます。
- 本関数は，応答に5秒以上の時間がかかることがあります。

戻り値

エラー番号を返します。

エラー：

Not descriptor	機器記述子がありません。

参照

```
CDAQMX::restBalance
CDAQMXBalanceResult::getMXBalanceData
CDAQMXBalanceResult::getMXBalanceResult
```

runBalanceMX

構文

```
int runBalanceMX(DAQMX daqmx, MXBalanceData * pMXBalanceData,
MXBalanceResult * pMXBalanceResult);
```

宣言

```
Public Declare Function runBalanceMX Lib "DAQMX" (ByVal daqmx
As Long, ByRef pMXBalanceData As MXBalanceData, ByRef
pMXBalanceResult As MXBalanceResult) As Long
```

引数

daqmx	機器記述子を指定します。
pMXBalanceData	初期バランスデータを指定します。
pMXBalanceResult	初期バランス結果の返却先を指定します。

説明

初期バランスを実行します。

- 初期バランスデータの有効/無効の項が「有効」であるチャネルに対して実行します。
- 初期バランス値が初期バランスデータに返却されます。また，チャネル別の初期バランス結果を指定された返却先に返却します。このとき，初期バランスデータの有効/無効の項に「有効」を指定したチャネルのデータだけが上書きされます。
- 本関数は，応答に5秒以上の時間がかかることがあります。

戻り値

エラー番号を返します。

エラー：

Not descriptor	機器記述子がありません。

参照

```
CDAQMX::runBalance
CDAQMXBalanceResult::getMXBalanceData
CDAQMXBalanceResult::getMXBalanceResult
```

setAlarmMX

構文

```
int setAlarmMX(DAQMX daqmx, int levelNo, int startChNo, int
endChNo, int iAlarmType, int value, int histerisys);
```

宣言

```
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
```

引数

daqmx	機器記述子を指定します。
levelNo	アラームレベルを指定します。
startChNo	開始チャネル番号を指定します。
endChNo	終了チャネル番号を指定します。
iAlarmType	アラーム種類を指定します。
value	アラーム値を指定します。
histerisys	ヒステリシスを指定します。

説明

指定されたチャネル範囲にアラームを設定します。
設定データを一括取得して指定変更後，一括送信しています。

戻り値

エラー番号を返します。
エラー：

Not descriptor	機器記述子がありません。

参照

CDAQMX::getMXConfig CDAQMX::setMXConfig
CDAQMXChConfig::setAlarm
CDAQMXConfig::getClassMXChConfig

setAOMX

構文

```
int setAOMX(DAQMX daqmx, int iRangeAO, int startChNo, int
endChNo, int spanMin, int spanMax);
```

宣言

```
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
```

引数

daqmx	機器記述子を指定します。
iRangeAO	レンジ種類からAOレンジを指定します。
startChNo	開始チャネル番号を指定します。
endChNo	終了チャネル番号を指定します。
spanMin	スパン最小値を指定します。
spanMax	スパン最大値を指定します。

説明

指定されたチャネル範囲をAOレンジに設定します。

- スパン最小値とスパン最大値が等しい場合，省略されたものとみなします。
- 設定データを一括取得して指定変更後，一括送信しています。
- 対応していないモジュールのチャネルは無視します。
- 出力チャネルデータも変更されます。

戻り値

エラー番号を返します。

エラー：

Not descriptor　機器記述子がありません。

参照

```
CDAQMX::getMXConfig
CDAQMX::setMXConfig
CDAQMXConfig::setAO
```

setAOPWMDataMX

構文

```
int setAOPWMDataMX(DAQMX daqmx, MXAOPWMData * pMXAOPWMData);
```

宣言

```
Public Declare Function setAOPWMDataMX Lib "DAQMX" (ByVal
daqmx As Long, ByRef pMXAOPWMData As MXAOPWMData) As Long
```

引数

daqmx	機器記述子を指定します。
pMXAOPWMData	出力するAO/PWMデータを指定します。

説明

AO/PWMデータを一括送信します。

- コマンドAOとコマンドPWMのチャネルの出力を変更します。
- 送信するデータは，データ操作機能のAO/PWMデータの変更で作成できます。
- AO/PWMデータの有効/無効の項が「有効」に指定されたチャネルの指定値が出力されます。

戻り値

エラー番号を返します。

エラー：

Not descriptor	機器記述子がありません。

参照

`CDAQMX::setAOPWMData`

setAOTypeMX

構文

```
int setAOTypeMX(DAQMX daqmx, int aoNo, int iKind, int
refChNo);
```

宣言

```
Public Declare Function setAOTypeMX Lib "DAQMX" (ByVal daqmx
As Long, ByVal aoNo As Long, ByVal iKind As Long, ByVal
refChNo As Long) As Long
```

引数

daqmx	機器記述子を指定します。
aoNo	AOデータ番号を指定します。
iKind	AOチャネルの種類をチャネル種類で指定します。
refChNo	基準チャネル番号をチャネル番号で指定します。

説明

AOデータ番号のチャネルを， 指定されたチャネル種類に設定します。

- チャネルは，AOモジュール上でなければなりません。
- 指定できるチャネル種類は， AO，コマンドAOのいずれかです。
- 基準チャネル番号は，入力チャネルでなければなりません。
- コマンドAOを設定した場合，基準チャネル番号は無視されます。
- 設定データを一括取得して指定変更後，一括送信しています。

戻り値

エラー番号を返します。

エラー：

Not descriptor　機器記述子がありません。

参照

```
CDAQMX::getMXConfig
CDAQMX::setMXConfig
CDAQMXConfig::setAOType
```

setBackupMX

構文

```
int setBackupMX(DAQMX daqmx, int bBackup, int iCFWriteMode);
```

宣言

```
Public Declare Function setBackupMX Lib "DAQMX" (ByVal daqmx
As Long, ByVal bBackup As Long, ByVal iCFWriteMode As Long) As
Long
```

引数

daqmx	機器記述子を指定します。
bBackup	有効無効値を指定します。
iCFWriteMode	CF書き込み種類を指定します。

説明

バックアップ(CFへのデータ書き込み)を設定します。

・ CF書き込み種類は，既に設定されているものと異なる場合に設定します。

・ CF書き込み種類が設定される場合，FIFOは停止します。

戻り値

エラー番号を返します。

エラー：

Not descriptor　機器記述子がありません。

参照

```
CDAQMX::getMXConfig
CDAQMX::setBackup
CDAQMX::setMXConfig
CDAQMXConfig::getClassMXStatus
CDAQMXStatus::getCFWriteMode
CDAQMXStatus::setCFWriteMode
```

setBalanceMX

構文

```
int setBalanceMX(DAQMX daqmx, MXBalanceData * pMXBalanceData);
```

宣言

```
Public Declare Function setBalanceMX Lib "DAQMX" (ByVal daqmx
As Long, ByRef pMXBalanceData As MXBalanceData) As Long
```

引数

daqmx	機器記述子を指定します。
pMXBalanceData	初期バランスデータを指定します。

説明

初期バランスデータを設定します。

- 初期バランスデータの有効/無効の項が「有効」であるチャネルの初期バランス値を書き込みます。
- 設定データを一括取得して指定変更後，一括送信しています。

戻り値

エラー番号を返します。

エラー：

Not descriptor	機器記述子がありません。

参照

`CDAQMX::setBalance`

setBurnoutMX

構文

```
int setBurnoutMX(DAQMX daqmx, int iBurnout, int startChNo, int
endChNo);
```

宣言

```
Public Declare Function setBurnoutMX Lib "DAQMX" (ByVal daqmx
As Long, ByVal iBurnout As Long, ByVal startChNo As Long,
ByVal endChNo As Long) As Long
```

引数

daqmx	機器記述子を指定します。
iBurnout	バーンアウト種類を指定します。
startChNo	開始チャネル番号を指定します。
endChNo	終了チャネル番号を指定します。

説明

指定されたチャネル範囲にバーンアウト種類を設定します。

・ 設定データを一括取得して指定変更後，一括送信しています。

戻り値

エラー番号を返します。

エラー：

Not descriptor　機器記述子がありません。

参照

```
CDAQMX::getMXConfig
CDAQMX::setMXConfig
CDAQMXChConfig::setBurnout
CDAQMXConfig::getClassMXChConfig
```

setChatFilterMX

構文

```
int setChatFilterMX(DAQMX daqmx, int bChatFilter, int
startChNo, int endChNo);
```

宣言

```
Public Declare Function setChatFilterMX Lib "DAQMX" (ByVal
daqmx As Long, ByVal bChatFilter As Long, ByVal startChNo As
Long, ByVal endChNo As Long) As Long
```

引数

daqmx	機器記述子を指定します。
bChatFilter	有効無効値を指定します。
startChNo	開始チャネル番号を指定します。
endChNo	終了チャネル番号を指定します。

説明

指定されたチャネル範囲にチャタリングフィルタを設定します。

・設定データを一括取得して指定変更後，一括送信しています。

戻り値

エラー番号を返します。

エラー：

Not descriptor　機器記述子がありません。

参照

```
CDAQMX::getMXConfig
CDAQMX::setMXConfig
CDAQMXChConfig::setChatFilter
CDAQMXConfig::getClassMXChConfig
```

setChConfigMX

構文

```
int setChConfigMX(DAQMX daqmx, MXChConfig * pMXChConfig);
```

宣言

```
Public Declare Function setChConfigMX Lib "DAQMX" (ByVal daqmx
As Long, ByRef pMXChConfig As MXChConfig) As Long
```

引数

daqmx　　機器記述子を指定します。
pMXChConfig　チャネル設定データを指定します。

説明

チャネル設定データを設定します。

・ 設定データを一括取得して指定変更後，一括送信しています。

戻り値

エラー番号を返します。
エラー：
Not descriptor　機器記述子がありません。

参照

CDAQMX::getMXConfig
CDAQMX::setMXConfig
CDAQMXChConfigData::setMXChConfig
CDAQMXConfig::getClassMXChConfigData

setChoiceMX

構文

```
int setChoiceMX(DAQMX daqmx, int startChNo, int endChNo, int
idleChoice, int errorChoice, int presetValue);
```

宣言

```
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
```

引数

daqmx	機器記述子を指定します。
startChNo	開始チャネル番号を指定します。
endChNo	終了チャネル番号を指定します。
idleChoice	アイドル時の選択値を指定します。
errorChoice	エラー時の選択値を指定します。
presetValue	選択値が「指定値」の場合の出力値を指定します。

説明

出力チャネルデータのアイドル時とエラー時の出力を設定します。

- ユーザ指定の出力値は，データ値やスパンと同様に整数値で指定します。
- 設定データを一括取得して指定変更後，一括送信しています。

戻り値

エラー番号を返します。

エラー：

Not descriptor　機器記述子がありません。

参照

```
CDAQMX::getMXConfig
CDAQMX::setMXConfig
CDAQMXChConfig::setChoice
```

setCommentMX

構文

```
int setCommentMX(DAQMX daqmx, const char * strComment, int
startChNo, int endChNo);
```

宣言

```
Public Declare Function setCommentMX Lib "DAQMX" (ByVal daqmx
As Long, ByVal strComment As String, ByVal startChNo As Long,
ByVal endChNo As Long) As Long
```

引数

daqmx	機器記述子を指定します。
strComment	コメントを指定します。
startChNo	開始チャネル番号を指定します。
endChNo	終了チャネル番号を指定します。

説明

指定されたチャネル範囲にコメントを設定します。

・ 設定データを一括取得して指定変更後，一括送信しています。

戻り値

エラー番号を返します。

エラー：

Not descriptor　機器記述子がありません。

参照

```
CDAQMX::getMXConfig
CDAQMX::setMXConfig
CDAQMXChConfig::setComment
CDAQMXConfig::getClassMXChConfig
```

setCOMMX

構文

```
int setCOMMX(DAQMX daqmx, int iRangeCOM, int startChNo, int
endChNo, int spanMin, int spanMax, int scaleMin, int scaleMax,
int scalePoint);
```

宣言

```
Public Declare Function setCOMMX Lib "DAQMX" (ByVal daqmx As
Long, ByVal iRangeCOM 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
```

引数

daqmx	機器記述子を指定します。
iRangeCOM	レンジ種類から通信レンジを指定します。
startChNo	開始チャネル番号を指定します。
endChNo	終了チャネル番号を指定します。
spanMin	スパン最小値を指定します。
spanMax	スパン最大値を指定します。
scaleMin	スケール最小値を指定します。
scaleMax	スケール最大値を指定します。
scalePoint	スケールの小数点位置を指定します。

説明

指定されたチャネル範囲を通信レンジに設定します。

・スパン最小値とスパン最大値が等しい場合，省略されたものとみなします。

・スケール最小値とスケール最大値が等しい場合，省略されたものとみなします。

・設定データを一括取得して，指定変更後，一括送信しています。

・対応していないモジュールのチャネルは無視します。

・アラームの設定は初期化されます。

戻り値

エラー番号を返します。

エラー：

Not descriptor　機器記述子がありません。

参照

```
CDAQMX::getMXConfig
CDAQMX::setMXConfig
CDAQMXConfig::setCOM
CDAQMXConfig::setScalling
```

setConfigDataMX

構文

```
int setConfigDataMX(DAQMX daqmx, MXConfigData *
pMXConfigData);
```

引数

daqmx　　機器記述子を指定します。
pMXConfigData 設定データを指定します。

説明

設定データを一括送信します。

・ Visual Basicの場合，各データ構造体ごとに個別に設定を行ってください。

戻り値

エラー番号を返します。
エラー：
Not descriptor　機器記述子がありません。

参照

`CDAQMX::setMXConfig`

setDateTimeMX

構文

```
int setDateTimeMX(DAQMX daqmx, MXDateTime * pMXDateTime);
```

宣言

```
Public Declare Function setDateTimeMX Lib "DAQMX" (ByVal daqmx
As Long, ByRef pMXDateTime As MXDateTime) As Long
```

引数

daqmx　　機器記述子を指定します。
pMXDateTime　時刻情報データを指定します。

説明

機器本体に時刻情報データを設定します。

- ミリ秒は無視されます。
- Visual Cの場合，引数の時刻情報にNULLを指定すると，PCの現在の日付時刻を設定します。
- 本関数は，応答に1秒以上の時間がかかることがあります。

戻り値

エラー番号を返します。
エラー：
Not descriptor　機器記述子がありません。

参照

`CDAQMX::setDateTime`

setDateTimeNowMX

構文

```
int setDateTimeNowMX(DAQMX daqmx);
```

宣言

```
Public Declare Function setDateTimeNowMX Lib "DAQMX" (ByVal
daqmx As Long) As Long
```

引数

daqmx　　機器記述子を指定します。

説明

現在の日付時刻を設定します。

戻り値

エラー番号を返します。

参照

setDateTimeMX

setDELTAMX

構文

```
int setDELTAMX(DAQMX daqmx, int refChNo, int startChNo, int
endChNo, int spanMin, int spanMax, int scaleMin, int scaleMax,
int scalePoint, int iRange);
```

宣言

```
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
```

引数

daqmx	機器記述子を指定します。
refChNo	基準チャネルをチャネル番号で指定します。
startChNo	開始チャネル番号を指定します。
endChNo	終了チャネル番号を指定します。
spanMin	スパン最小値を指定します。
spanMax	スパン最大値を指定します。
scaleMin	スケール最小値を指定します。
scaleMax	スケール最大値を指定します。
scalePoint	スケール時の小数点位置を指定します。
iRange	自チャネルのレンジ種類を指定します。

説明

- 指定されたチャネル範囲にチャネル間差演算を設定します。
- スパン最小値とスパン最大値が等しい場合，省略されたものとみなします。
- スケール最小値とスケール最大値が等しい場合，省略されたものとみなします。
- 設定データを一括取得して，指定変更後，一括送信しています。
- 対応していないモジュールのチャネルは無視します。
- アラームの設定は初期化されます。
- 自チャネルの入力レンジの指定にレンジ種類の「参照レンジ」を指定すると，自チャネルの測定レンジを基準チャネルと同じレンジにします。

戻り値

エラー番号を返します。

エラー：

Not descriptor　機器記述子がありません。

参照

```
CDAQMX::getMXConfig
CDAQMX::setMXConfig
CDAQMXConfig::setDELTA
CDAQMXConfig::setScalling
```

setDIMX

構文

```
int setDIMX(DAQMX daqmx, int iRangeDI, int startChNo, int
endChNo, int spanMin, int spanMax, int scaleMin, int scaleMax,
int scalePoint);
```

宣言

```
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
```

引数

daqmx	機器記述子を指定します。
iRangeDI	ディジタル入力(DI)レンジを指定します。
startChNo	開始チャネル番号を指定します。
endChNo	終了チャネル番号を指定します。
spanMin	スパン最小値を指定します。
spanMax	スパン最大値を指定します。
scaleMin	スケール最小値を指定します。
scaleMax	スケール最大値を指定します。
scalePoint	スケール時の小数点位置を指定します。

説明

指定されたチャネル範囲をディジタル入力(DI)レンジに設定します。

- スパンの最小値と最大値が等しい場合，省略されたものとみなします。
- スケールの最小値と最大値が等しい場合，省略されたものとみなします。
- 設定データを一括取得して指定変更後，一括送信しています。
- 対応していないモジュールのチャネルは無視します。
- アラームの設定は初期化されます。

戻り値

エラー番号を返します。

エラー：

Not descriptor　機器記述子がありません。

参照

```
CDAQMX::getMXConfig
CDAQMX::setMXConfig
CDAQMXConfig::setDI
CDAQMXConfig::setScalling
```

setDODataMX

構文

```
int setDODataMX(DAQMX daqmx, MXDOData * pMXDOData);
```

宣言

```
Public Declare Function setDODataMX Lib "DAQMX" (ByVal daqmx
As Long, ByRef pMXDOData As MXDOData) As Long
```

引数

daqmx　　機器記述子を指定します。
pMXDOData　DOデータを指定します。

説明

DOデータを一括送信します。

- コマンドDOのチャネルの出力を変更します。
- 送信するデータは，データ操作機能のDOデータの変更で作成できます。
- DOデータ内の「有効」に指定されたON/OFF値が出力されます。

戻り値

エラー番号を返します。
エラー：
Not descriptor　機器記述子がありません。

参照

`CDAQMX::setDOData`

setDOTypeMX

構文

```
int setDOTypeMX(DAQMX daqmx, int doNo, int iKind, int
bDeenergize, int bHold);
```

宣言

```
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
```

引数

daqmx	機器記述子を指定します。
doNo	DOデータ番号を指定します。
iKind	DOチャネルの種類をチャネル種類で指定します。
bDeenergize	非励磁を有効無効値で指定します。
bHold	ホールドを有効無効値で指定します。

説明

DOデータ番号のチャネルを，指定されたチャネル種類に設定します。

- チャネルはDOモジュール上にあり，種類はDO(アラーム出力，コマンドDO，システムFail，システムError)でなければなりません。
- 参照アラームは，setRefAlarmMX関数で設定します。
- 設定データを一括取得して指定変更後，一括送信しています。

戻り値

エラー番号を返します。

エラー：

Not descriptor　機器記述子がありません。

参照

```
CDAQMX::getMXConfig
CDAQMX::setMXConfig
CDAQMXConfig::setDOType
```

setFilterMX

構文

```
int setFilterMX(DAQMX daqmx, int iFilter, int startChNo, int
endChNo);
```

宣言

```
Public Declare Function setFilterMX Lib "DAQMX" (ByVal daqmx
As Long, ByVal iFilter As Long, ByVal startChNo As Long, ByVal
endChNo As Long) As Long
```

引数

daqmx	機器記述子を指定します。
iFilter	フィルタ係数を指定します。
startChNo	開始チャネル番号を指定します。
endChNo	終了チャネル番号を指定します。

説明

指定されたチャネル範囲にフィルタ係数を設定します。

・ 設定データを一括取得して指定変更後，一括送信しています。

戻り値

エラー番号を返します。

エラー：

Not descriptor　機器記述子がありません。

参照

```
CDAQMX::getMXConfig
CDAQMX::setMXConfig
CDAQMXChConfig::setFilter
CDAQMXConfig::getClassMXChConfig
```

setIntervalMX

構文

```
int setIntervalMX(DAQMX daqmx, int moduleNo, int iInterval,
int iHz);
```

宣言

```
Public Declare Function setIntervalMX Lib "DAQMX" (ByVal daqmx
As Long, ByVal moduleNo As Long, ByVal iInterval As Long,
ByVal iHz As Long) As Long
```

引数

daqmx	機器記述子を指定します。
moduleNo	モジュール番号を指定します。
iInterval	周期種類を指定します。
iHz	A/D積分時間種類を指定します。

説明

指定されたモジュール番号のモジュールに指定された値を設定します。

- 設定データを一括取得して，指定変更後，一括送信しています。
- モジュール番号に， 定数値の「全モジュール番号指定」をすると， 全モジュールに設定します。

戻り値

エラー番号を返します。

エラー：

Not descriptor　機器記述子がありません。

参照

```
CDAQMX::getMXConfig
CDAQMX::setMXConfig
CDAQMXConfig::setInterval
```

setOutputMX

構文

```
int setOutputMX(DAQMX daqmx, MXOutputData * pMXOutputData);
```

宣言

```
Public Declare Function setOutputMX Lib "DAQMX" (ByVal daqmx
As Long, ByRef pMXOutputData As MXOutputData) As Long
```

引数

daqmx	機器記述子を指定します。
pMXOutputData	出力チャネルデータを指定します。

説明

出力チャネルデータを設定します。

・ 設定データを一括取得して指定変更後，一括送信しています。

戻り値

エラー番号を返します。

エラー：

Not descriptor	機器記述子がありません。

参照

```
CDAQMX::getMXConfig
CDAQMX::setMXConfig
CDAQMXConfig::getClassMXOutputData
CDAQMXOutputData::setMXOutputData
```

setOutputTypeMX

構文

```
int setOutputTypeMX(DAQMX daqmx, int iOutput, int startChNo,
int endChNo);
```

宣言

```
Public Declare Function setOutputTypeMX Lib "DAQMX" (ByVal
daqmx As Long, ByVal iOutput As Long, ByVal startChNo As Long,
ByVal endChNo As Long) As Long
```

引数

daqmx	機器記述子を指定します。
iOutput	出力種類を指定します。
startChNo	開始チャネル番号を指定します。
endChNo	終了チャネル番号を指定します。

説明

出力チャネルデータの出力種類を設定します。

- 各項目は既定値になります。
- 対応するチャネルの設定も変更されます。
- 設定データを一括取得して指定変更後，一括送信しています。

戻り値

エラー番号を返します。

エラー：

Not descriptor　機器記述子がありません。

参照

```
CDAQMX::getMXConfig
CDAQMX::setMXConfig
CDAQMXConfig::setAO
CDAQMXConfig::setPWM
```

setPULSEMX

構文

```
int setPULSEMX(DAQMX daqmx, int iRangePULSE, int startChNo,
int endChNo, int spanMin, int spanMax, int scaleMin, int
scaleMax, int scalePoint);
```

宣言

```
Public Declare Function setPULSEMX Lib "DAQMX" (ByVal daqmx As
Long, ByVal iRangePULSE 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
```

引数

daqmx	機器記述子を指定します。
iRangePULSE	レンジ種類からパルスレンジを指定します。
startChNo	開始チャネル番号を指定します。
endChNo	終了チャネル番号を指定します。
spanMin	スパン最小値を指定します。
spanMax	スパン最大値を指定します。
scaleMin	スケール最小値を指定します。
scaleMax	スケール最大値を指定します。
scalePoint	スケールの小数点位置を指定します。

説明

指定されたチャネル範囲をパルスレンジに設定します。

・スパン最小値とスパン最大値が等しい場合，省略されたものとみなします。

・スケール最小値とスケール最大値が等しい場合，省略されたものとみなします。

・設定データを一括取得して，指定変更後，一括送信しています。

・対応していないモジュールのチャネルは無視します。

・アラームの設定は初期化されます。

戻り値

エラー番号を返します。

エラー：

Not descriptor	機器記述子がありません。

参照

```
CDAQMX::getMXConfig
CDAQMX::setMXConfig
CDAQMXConfig::setPULSE
CDAQMXConfig::setScalling
```

setPulseTimeMX

構文

```
int setPulseTimeMX(DAQMX daqmx, int pulseTime, int startChNo,
int endChNo);
```

宣言

```
Public Declare Function setPulseTimeMX Lib "DAQMX" (ByVal
daqmx As Long, ByVal pulseTime As Long, ByVal startChNo As
Long, ByVal endChNo As Long) As Long
```

引数

daqmx	機器記述子を指定します。
pulseTime	パルス周期倍率を指定します。
startChNo	開始チャネル番号を指定します。
endChNo	終了チャネル番号を指定します。

説明

出力チャネルデータのパルス周期倍率を設定します。

・ 設定データを一括取得して指定変更後，一括送信しています。

戻り値

エラー番号を返します。

エラー：

Not descriptor　機器記述子がありません。

参照

```
CDAQMX::getMXConfig
CDAQMX::setMXConfig
CDAQMXChConfig::setPulseTime
```

setPWMMX

構文

```
int setPWMMX(DAQMX daqmx, int iRangePWM, int startChNo, int
endChNo, int spanMin, int spanMax);
```

宣言

```
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
```

引数

daqmx	機器記述子を指定します。
iRangePWM	レンジ種類からPWMレンジを指定します。
startChNo	開始チャネル番号を指定します。
endChNo	終了チャネル番号を指定します。
spanMin	スパン最小値を指定します。
spanMax	スパン最大値を指定します。

説明

指定されたチャネル範囲をPWMレンジに設定します。

- スパン最小値とスパン最大値が等しい場合，省略されたものとみなします。
- 設定データを一括取得して指定変更後，一括送信しています。
- 対応していないモジュールのチャネルは無視します。
- 出力チャネルデータも変更されます。

戻り値

エラー番号を返します。

エラー：

Not descriptor　機器記述子がありません。

参照

```
CDAQMX::getMXConfig
CDAQMX::setMXConfig
CDAQMXConfig::setPWM
```

setPWMTypeMX

構文

```
int setPWMTypeMX(DAQMX daqmx, int pwmNo, int iKind, int
refChNo);
```

宣言

```
Public Declare Function setPWMTypeMX Lib "DAQMX" (ByVal daqmx
As Long, ByVal pwmNo As Long, ByVal iKind As Long, ByVal
refChNo As Long) As Long
```

引数

daqmx	機器記述子を指定します。
pwmNo	PWMデータ番号を指定します。
iKind	PWMチャネルの種類をチャネル種類で指定します。
refChNo	基準チャネル番号をチャネル番号で指定します。

説明

PWMデータ番号のチャネルを， 指定されたチャネル種類に設定します。

- チャネルは，PWMモジュール上でなければなりません。
- 指定できるチャネル種類は， PWM，コマンドPWMのいずれかです。
- 基準チャネル番号は，入力チャネルでなければなりません。
- コマンドPWMを設定した場合，基準チャネル番号は無視されます。
- 設定データを一括取得して指定変更後，一括送信しています。

戻り値

エラー番号を返します。

エラー：

Not descriptor　機器記述子がありません。

参照

```
CDAQMX::getMXConfig
CDAQMX::setMXConfig
CDAQMXConfig::setPWMType
```

setRefAlarmMX

構文

```
int setRefAlarmMX(DAQMX daqmx, int refChNo, int startChNo, int
endChNo, int levelNo, int bValid);
```

宣言

```
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
```

引数

daqmx	機器記述子を指定します。
refChNo	参照チャネルをチャネル番号で指定します。
startChNo	開始チャネル番号を指定します。
endChNo	終了チャネル番号を指定します。
levelNo	アラームレベルを指定します。
bValid	有効無効値を指定します。

説明

指定されたチャネル範囲に参照アラームを設定します。

- 設定データを一括取得して指定変更後，一括送信しています。
- 参照チャネルに，定数値の「全参照チャネル番号指定」をすると，全チャネルを対象とします。
- アラームレベルに，定数値の「全アラームレベル番号指定」をすると，参照チャネルの全アラームレベルを対象とします。

戻り値

エラー番号を返します。

エラー：

Not descriptor　機器記述子がありません。

参照

```
CDAQMX::getMXConfig
CDAQMX::setMXConfig
CDAQMXChConfig::setRefAlarm
CDAQMXConfig::getClassMXChConfig
```

setRESMX

構文

```
int setRESMX(DAQMX daqmx, int iRangeRES, int startChNo, int
endChNo, int spanMin, int spanMax, int scaleMin, int scaleMax,
int scalePoint);
```

宣言

```
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
```

引数

daqmx	機器記述子を指定します。
iRangeRES	レンジ種類から抵抗レンジを指定します。
startChNo	開始チャネル番号を指定します。
endChNo	終了チャネル番号を指定します。
spanMin	スパン最小値を指定します。
spanMax	スパン最大値を指定します。
scaleMin	スケール最小値を指定します。
scaleMax	スケール最大値を指定します。
scalePoint	スケール時の小数点位置を指定します。

説明

指定されたチャネル範囲を抵抗レンジに設定します。

- スパン最小値とスパン最大値が等しい場合，省略されたものとみなします。
- スケール最小値とスケール最大値が等しい場合，省略されたものとみなします。
- 設定データを一括取得して指定変更後，一括送信しています。
- 対応していないモジュールのチャネルは無視します。
- アラームの設定は初期化されます。

戻り値

エラー番号を返します。

エラー

Not descriptor　機器記述子がありません。

参照

```
CDAQMX::getMXConfig
CDAQMX::setMXConfig
CDAQMXConfig::setRES
CDAQMXConfig::setScalling
```

setRJCTypeMX

構文

```
int setRJCTypeMX(DAQMX daqmx, int iRJCType, int startChNo, int
endChNo, int volt);
```

宣言

```
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
```

引数

daqmx	機器記述子を指定します。
iRJCType	RJC種類を指定します。
startChNo	開始チャネル番号を指定します。
endChNo	終了チャネル番号を指定します。
volt	RJC電圧値を指定します。

説明

指定されたチャネル範囲にRJC関連を設定します。

・ 設定データを一括取得して指定変更後，一括送信しています。

戻り値

エラー番号を返します。

エラー：

Not descriptor　機器記述子がありません。

参照

```
CDAQMX::getMXConfig
CDAQMX::setMXConfig
CDAQMXChConfig::setRJCType
CDAQMXConfig::getClassMXChConfig
```

setRRJCMX

構文

```
int setRRJCMX(DAQMX daqmx, int refChNo, int startChNo, int
endChNo, int spanMin, int spanMax);
```

宣言

```
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
```

引数

daqmx	機器記述子を指定します。
refChNo	参照チャネルをチャネル番号で指定します。
startChNo	開始チャネル番号を指定します。
endChNo	終了チャネル番号を指定します。
spanMin	スパン最小値を指定します。
spanMax	スパン最大値を指定します。

説明

指定されたチャネル範囲をリモートRJCに設定します。

- スパンの最小値と最大値が等しい場合，省略されたものとみなします。
- 設定データを一括取得して指定変更後，一括送信しています。
- 対応していないモジュールのチャネルは無視します。
- アラームの設定は初期化されます。

戻り値

エラー番号を返します。

エラー：

Not descriptor　機器記述子がありません。

参照

```
CDAQMX::getMXConfig CDAQMX::setMXConfig
CDAQMXConfig::setRRJC
```

setRTDMX

構文

```
int setRTDMX(DAQMX daqmx, int iRangeRTD, int startChNo, int
endChNo, int spanMin, int spanMax, int scaleMin, int scaleMax,
int scalePoint);
```

宣言

```
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
```

引数

daqmx	機器記述子を指定します。
iRangeRTD	測温抵抗体レンジを指定します。
startChNo	開始チャネル番号を指定します。
endChNo	終了チャネル番号を指定します。
spanMin	スパン最小値を指定します。
spanMax	スパン最大値を指定します。
scaleMin	スケール最小値を指定します。
scaleMax	スケール最大値を指定します。
scalePoint	スケール時の小数点位置を指定します。

説明

指定されたチャネル範囲を測温抵抗体レンジに設定します。

- スパンの最小値と最大値が等しい場合，省略されたものとみなします。
- スケールの最小値と最大値が等しい場合，省略されたものとみなします。
- 設定データを一括取得して指定変更後，一括送信しています。
- 対応していないモジュールのチャネルは無視します。
- アラームの設定は初期化されます。

戻り値

エラー番号を返します。

エラー：

Not descriptor　機器記述子がありません。

参照

```
CDAQMX::getMXConfig CDAQMX::setMXConfig
CDAQMXConfig::setRTD CDAQMXConfig::setScalling
```

setScallingUnitMX

構文

```
int setScallingUnitMX(DAQMX daqmx, const char * strUnit, int
startChNo, int endChNo);
```

宣言

```
Public Declare Function setScallingUnitMX Lib "DAQMX" (ByVal
daqmx As Long, ByVal strUnit As String, ByVal startChNo As
Long, ByVal endChNo As Long) As Long
```

引数

daqmx	機器記述子を指定します。
strUnit	単位名を指定します。
startChNo	開始チャネル番号を指定します。
endChNo	終了チャネル番号を指定します。

説明

指定されたチャネル範囲に単位名を設定します。

・ 設定データを一括取得して指定変更後，一括送信しています。

戻り値

エラー番号を返します。

エラー：

Not descriptor　機器記述子がありません。

参照

```
CDAQMX::getMXConfig
CDAQMX::setMXConfig
CDAQMXChConfig::setUnit
CDAQMXConfig::getClassMXChConfig
```

setSegmentMX

構文

```
int setSegmentMX(DAQMX daqmx, int iDispType, int dispTime,
MXSegment * newSegment, MXSegment * oldSegment);
```

宣言

```
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
```

引数

daqmx	機器記述子を指定します。
iDispType	表示種類を指定します。
dispTime	表示時間を指定します。
newSegment	表示パターンを指定します。
oldSegment	以前の表示パターンの返却先を指定します。

説明

7セグメントLEDの表示を設定します。

・ 返却先が指定されていれば，変更前の7セグメントLEDの表示パターンを格納します。

戻り値

エラー番号を返します。

エラー：

Not descriptor　機器記述子がありません。

参照

`CDAQMX::setSegment`
`CDAQMXSegment::getMXSegment`

setSKIPMX

構文

```
int setSKIPMX(DAQMX daqmx, int startChNo, int endChNo);
```

宣言

```
Public Declare Function setSKIPMX Lib "DAQMX" (ByVal daqmx As
Long, ByVal startChNo As Long, ByVal endChNo As Long) As Long
```

引数

daqmx	機器記述子を指定します。
startChNo	開始チャネル番号を指定します。
endChNo	終了チャネル番号を指定します。

説明

指定されたチャネル範囲をスキップ(未使用)に設定します。

・ 設定データを一括取得して指定変更後，一括送信しています。

・ 未使用チャネルに設定すると，本体のチャネルの設定は失われます。

戻り値

エラー番号を返します。

エラー：

Not descriptor　機器記述子がありません。

参照

```
CDAQMX::getMXConfig
CDAQMX::setMXConfig
CDAQMXConfig::setSKIP
```

setSTRAINMX

構文

```
int setSTRAINMX(DAQMX daqmx, int iRangeSTRAIN, int startChNo,
int endChNo, int spanMin, int spanMax, int scaleMin, int
scaleMax, int scalePoint);
```

宣言

```
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
```

引数

daqmx	機器記述子を指定します。
iRangeSTRAIN	レンジ種類からひずみレンジを指定します。
startChNo	開始チャネル番号を指定します。
endChNo	終了チャネル番号を指定します。
spanMin	スパン最小値を指定します。
spanMax	スパン最大値を指定します。
scaleMin	スケール最小値を指定します。
scaleMax	スケール最大値を指定します。
scalePoint	スケール時の小数点位置を指定します。

説明

指定されたチャネル範囲をひずみレンジに設定します。

- スパン最小値とスパン最大値が等しい場合，省略されたものとみなします。
- スケール最小値とスケール最大値が等しい場合，省略されたものとみなします。
- 設定データを一括取得して指定変更後，一括送信しています。
- 対応していないモジュールのチャネルは無視します。
- アラームの設定は初期化されます。

戻り値

エラー番号を返します。

エラー：

Not descriptor	機器記述子がありません。

参照

```
CDAQMX::getMXConfig
CDAQMX::setMXConfig
CDAQMXConfig::setScalling
CDAQMXConfig::setSTRAIN
```

setSystemConfigMX

構文

```
int setSystemConfigMX(DAQMX daqmx, MXSystemInfo *
pMXSystemInfo);
```

宣言

```
Public Declare Function setSystemConfigMX Lib "DAQMX" (ByVal
daqmx As Long, ByRef pMXSystemInfo As MXSystemInfo) As Long
```

引数

daqmx　　機器記述子を指定します。

pMXSystemInfo システム構成データを指定します。

説明

設定データの中の，システム構成データを設定します。

・ 設定データを一括取得して指定変更後，一括送信しています。

戻り値

エラー番号を返します。

エラー：

Not descriptor　機器記述子がありません。

参照

```
CDAQMX::getMXConfig
CDAQMX::setMXConfig
CDAQMXConfig::getClassMXSysInfo
CDAQMXSysInfo::setMXSystemInfo
```

setSystemTimeoutMX

構文

```
int setSystemTimeoutMX(DAQMX daqmx, int timeout);
```

宣言

```
Public Declare Function setSystemTimeoutMX Lib "DAQMX" (ByVal
daqmx As Long, ByVal timeout As Long) As Long
```

引数

daqmx	機器記述子を指定します。
timeout	タイムアウト値を秒数で指定します。

説明

タイムアウト値を設定します。

・ 設定データを一括取得して指定変更後，一括送信しています。

戻り値

エラー番号を返します。

エラー：

Not descriptor　機器記述子がありません。

参照

```
CDAQMX::getMXConfig
CDAQMX::setMXConfig
CDAQMXConfig::getClassMXSysInfo
CDAQMXSysInfo::setCFTimeout
```

setTagMX

構文

```
int setTagMX(DAQMX daqmx, const char * strTag, int startChNo,
int endChNo);
```

宣言

```
Public Declare Function setTagMX Lib "DAQMX" (ByVal daqmx As
Long, ByVal strTag As String, ByVal startChNo As Long, ByVal
endChNo As Long) As Long
```

引数

daqmx	機器記述子を指定します。
strTag	タグを指定します。
startChNo	開始チャネル番号を指定します。
endChNo	終了チャネル番号を指定します。

説明

指定されたチャネル範囲にタグを設定します。

・ 設定データを一括取得して指定変更後，一括送信しています。

戻り値

エラー番号を返します。

エラー：

Not descriptor　機器記述子がありません。

参照

```
CDAQMX::getMXConfig
CDAQMX::setMXConfig
CDAQMXChConfig::setTag
CDAQMXConfig::getClassMXChConfig
```

setTCMX

構文

```
int setTCMX(DAQMX daqmx, int iRangeTC, int startChNo, int
endChNo, int spanMin, int spanMax, int scaleMin, int scaleMax,
int scalePoint);
```

宣言

```
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
```

引数

daqmx	機器記述子を指定します。
iRangeTC	熱電対レンジを指定します。
startChNo	開始チャネル番号を指定します。
endChNo	終了チャネル番号を指定します。
spanMin	スパン最小値を指定します。
spanMax	スパン最大値を指定します。
scaleMin	スケール最小値を指定します。
scaleMax	スケール最大値を指定します。
scalePoint	スケールの小数点位置を指定します。

説明

指定されたチャネル範囲を熱電対レンジに設定します。

- スパンの最小値と最大値が等しい場合，省略されたものとみなします。
- スケールの最小値と最大値が等しい場合，省略されたものとみなします。
- 設定データを一括取得して指定変更後，一括送信しています。
- 対応していないモジュールのチャネルは無視します。
- アラームの設定は初期化されます。

戻り値

エラー番号を返します。

エラー：

Not descriptor　機器記述子がありません。

参照

```
CDAQMX::getMXConfig
CDAQMX::setMXConfig
CDAQMXConfig::setScalling
CDAQMXConfig::setTC
```

setTempUnitMX

構文

```
int setTempUnitMX(DAQMX daqmx, int iTempUnit);
```

宣言

```
Public Declare Function setTempUnitMX Lib "DAQMX" (ByVal daqmx
As Long, ByVal iTempUnit As Long) As Long
```

引数

daqmx　　機器記述子を指定します。
iTempUnit　　温度単位種類を指定します。

説明

温度単位種類を設定します。

・ 設定データを一括取得して指定変更後，一括送信しています。
・ 熱電対レンジ，測温抵抗体レンジのチャネル設定は初期化されます。

戻り値

エラー番号を返します。
エラー：
Notd escriptor　機器記述子がありません。

参照

CDAQMX::getMXConfig
CDAQMX::setMXConfig
CDAQMXConfig::setTempUnit

setTimeOutMX

構文

```
int setTimeOutMX(DAQMX daqmx, int seconds);
```

宣言

```
Public Declare Function setTimeOutMX Lib "DAQMX" (ByVal daqmx
As Long, ByVal seconds As Long) As Long
```

引数

daqmx	機器記述子を指定します。
seconds	通信のタイムアウト値を秒単位で指定します。

説明

機器との通信に対して，タイムアウトを設定します。

- 指定された値が負の場合，タイムアウトを無効にします。
- 使用を推奨しません(3.1節または4.1節を参照)。

戻り値

エラー番号を返します。

エラー：

Not descriptor　機器記述子がありません。

参照

`CDAQMX::setTimeOut`

setTransmitMX

構文

```
int setTransmitMX(DAQMX daqmx, MXTransmit ÅE   pMXTransmit);
```

宣言

```
Public Declare Function setTransmitMX Lib "DAQMX" (ByVal daqmx
As Long, ByRef pMXTransmit As MXTransmit) As Long
```

引数

daqmx　　機器記述子を指定します。
pMXTransmit　伝送出力データを指定します。

説明

伝送出力データを一括送信します。

- 伝送出力(AOとPWM)チャネルを指定された状態に変更します。
- 送信するデータは，データ操作機能の伝送出力データの変更で作成できます。
- 伝送出力チャネル以外は，指定は無視されます。
- 出力開始を指定したチャネルが既に出力中の場合，出力は継続します。

戻り値

エラー番号を返します。
エラー：
Not descriptor　　機器記述子がありません。

参照

`CDAQMX::setTransmit`

setUnitNoMX

構文

```
int setUnitNoMX(DAQMX daqmx, int unitNo);
```

宣言

```
Public Declare Function setUnitNoMX Lib "DAQMX" (ByVal daqmx
As Long, ByVal unitNo As Long) As Long
```

引数

daqmx　　機器記述子を指定します。
unitNo　　ユニット番号を指定します。

説明

ユニット番号を設定します。

・ 設定データを一括取得して指定変更後，一括送信しています。

戻り値

エラー番号を返します。
エラー：
Not descriptor　機器記述子がありません。

参照

CDAQMX::getMXConfig
CDAQMX::setMXConfig
CDAQMXConfig::getClassMXSysInfo
CDAQMXSysInfo::setUnitNo

setUserTimeMX [Visual Cのみ]

構文

```
int setUserTimeMX(DAQMX daqmx, MXUserTime userTime);
```

引数

daqmx	機器記述子を指定します。
userTime	ユーザーカウントを指定します。

説明

ユーザカウントを設定します。

・ 指定されたユーザカウントは，次から発行されるパケットに挿入されます。

・ Visual Basicの場合，setUserTimeVBMX関数を使用してください。

戻り値

エラー番号を返します。

エラー：

Not descriptor 機器記述子がありません。

参照

`CDAQMX::setUserTime`

setUserTimeVBMX

構文

```
int setUserTimeVBMX(DAQMX daqmx, MXUserTime * userTime);
```

宣言

```
Public Declare Function setUserTimeVBMX Lib "DAQMX" (ByVal
daqmx As Long, ByRef userTime As MXUserTime) As Long
```

引数

daqmx	機器記述子を指定します。
userTime	ユーザーカウントを指定します。

説明

ユーザカウントを設定します。

- ユーザカウントの指定が，参照指定であること以外は，setUserTimeMX関数と同じです。
- Visual Cの場合，ユーザカウントにNULLを指定すると，「0」とみなします。

戻り値

エラー番号を返します。

参照

`setUserTimeMX`

setVOLTMX

構文

```
int setVOLTMX(DAQMX daqmx, int iRangeVOLT, int startChNo, int
endChNo, int spanMin, int spanMax, int scaleMin, int scaleMax,
int scalePoint);
```

宣言

```
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
```

引数

daqmx	機器記述子を指定します。
iRangeVOLT	直流電圧レンジを指定します。
startChNo	開始チャネル番号を指定します。
endChNo	終了チャネル番号を指定します。
spanMin	スパン最小値を指定します。
spanMax	スパン最大値を指定します。
scaleMin	スケール最小値を指定します。
scaleMax	スケール最大値を指定します。
scalePoint	スケールの小数点位置を指定します。

説明

指定されたチャネル範囲を直流電圧レンジに設定します。

- スパンの最小値と最大値が等しい場合，省略されたものとみなします。
- スケールの最小値と最大値が等しい場合，省略されたものとみなします。
- 設定データを一括取得して指定変更後，一括送信しています。
- 対応していないモジュールのチャネルは無視します。
- アラームの設定は初期化されます。

戻り値

エラー番号を返します。

エラー：

Not descriptor　機器記述子がありません。

参照

```
CDAQMX::getMXConfig
CDAQMX::setMXConfig
CDAQMXConfig::setScalling
CDAQMXConfig::setVOLT
```

startFIFOMX

構文

```
int startFIFOMX(DAQMX daqmx);
```

宣言

```
Public Declare Function startFIFOMX Lib "DAQMX" (ByVal daqmx
As Long) As Long
```

引数

daqmx　　機器記述子を指定します。

説明

FIFOを開始します。

・既に開始している場合，FIFOを継続します。

・FIFO開始から測定データを取得できるようになるまで，時間がかかることがあります。測定周期により，かかる時間は異なります。

戻り値

エラー番号を返します。

エラー：

Not descriptor　機器記述子がありません。

参照

`CDAQMX::startFIFO`

stopFIFOMX

構文

```
int stopFIFOMX(DAQMX daqmx);
```

宣言

```
Public Declare Function stopFIFOMX Lib "DAQMX" (ByVal daqmx As
Long) As Long
```

引数

daqmx　　機器記述子を指定します。

説明

FIFOを停止します。

戻り値

エラー番号を返します。

エラー：

Not descriptor　機器記述子がありません。

参照

`CDAQMX::stopFIFO`

talkChDataInstMX

構文

```
int talkChDataInstMX(DAQMX daqmx, int chNo);
```

宣言

```
Public Declare Function talkChDataInstMX Lib "DAQMX" (ByVal
daqmx As Long, ByVal chNo As Long) As Long
```

引数

daqmx	機器記述子を指定します。
chNo	チャネル番号を指定します。

説明

指定されたチャネル番号の瞬時値の測定データの取得を宣言します。

・ 取得動作については，talkChDataMX関数を参照してください。

戻り値

エラー番号を返します。

参照

`talkChDataMX`

talkChDataMX [Visual Cのみ]

構文

```
int talkChDataMX(DAQMX daqmx, int chNo, MXDataNo startDataNo,
MXDataNo endDataNo);
```

引数

daqmx	機器記述子を指定します。
chNo	チャネル番号を指定します。
startDataNo	開始データ番号を指定します。
endDataNo	終了データ番号を指定します。

説明

指定されたチャネル番号の測定データの取得を宣言します。

- 取得する範囲は，開始/終了データ番号で指定します。
- 瞬時値を取得する場合，開始/終了データ番号に， 定数値の「瞬時値指定用データ番号」を指定します。または，talkChDataInstMX関数を使用します。
- 本関数の実行後，getTimeDataMX関数でデータ個数分のデータ時刻を取得します。次に，getChDataMX関数でデータ個数分の測定データを取得します。
- Visual Basicの場合，talkChDataVBMX関数を使用してください。

戻り値

エラー番号を返します。

エラー：

Not descriptor　機器記述子がありません。

参照

`CDAQMX::talkChData`

talkChDataVBMX

構文

```
int talkChDataVBMX(DAQMX daqmx, int chNo, MXDataNo *
startDataNo, MXDataNo * endDataNo);
```

宣言

```
Public Declare Function talkChDataVBMX Lib "DAQMX" (ByVal
daqmx As Long, ByVal chNo As Long, ByRef startDataNo As
MXDataNo, ByRef endDataNo As MXDataNo) As Long
```

引数

daqmx	機器記述子を指定します。
chNo	チャネル番号を指定します。
startDataNo	開始データ番号を指定します。
endDataNo	終了データ番号を指定します。

説明

指定されたチャネル番号の測定データの取得を宣言します。

- データ番号の指定が，参照指定であること以外は，talkChDataMX関数と同じです。
- Visual Cの場合，データ番号にNULLを指定すると，定数値の「瞬時値指定用データ番号」とみなします。
- 取得動作については，talkChDataMX関数を参照してください。

戻り値

エラー番号を返します。

参照

`talkChDataMX`

talkChInfoMX

構文

```
int talkChInfoMX(DAQMX daqmx, int startChNo, int endChNo);
```

宣言

```
Public Declare Function talkChInfoMX Lib "DAQMX" (ByVal daqmx
As Long, ByVal startChNo As Long, ByVal endChNo As Long) As
Long
```

引数

daqmx	機器記述子を指定します。
startChNo	開始チャネル番号を指定します。
endChNo	終了チャネル番号を指定します。

説明

チャネル情報データの取得を宣言します。

・本関数の実行後，チャネル毎のデータ取得には，getChInfoMXを使用します。

戻り値

エラー番号を返します。

エラー：

Not descriptor　機器記述子がありません。

参照

`CDAQMX::talkChInfo`

talkConfigMX

構文

```
int talkConfigMX(DAQMX daqmx, MXSystemInfo * pMXSystemInfo,
MXStatus * pMXStatus, MXNetInfo * pMXNetInfo);
```

宣言

```
Public Declare Function talkConfigMX Lib "DAQMX" (ByVal daqmx
As Long, ByRef pMXSystemInfo As MXSystemInfo, ByRef pMXStatus
As MXStatus, ByRef pMXNetInfo As MXNetInfo) As Long
```

引数

daqmx　　機器記述子を指定します。
pMXSystemInfo　システム構成データの返却先を指定します。
pMXStatus　　ステータスの返却先を指定します。
pMXNetInfo　　ネットワーク情報データの返却先を指定します。

説明

設定データの取得を宣言します。

- 設定データの内，システム構成データ，ステータス，ネットワーク情報データを指定された返却先に格納します。
- 本関数の実行後，getChConfigMX関数でチャネル個数分のチャネル設定データを取得してください。
- 設定データの内，初期バランスデータと出力チャネルデータは，別途各取得関数で取得します。

戻り値

エラー番号を返します。
エラー：
Not descriptor　機器記述子がありません。

参照

```
CDAQMX::talkConfig
CDAQMXNetInfo::getMXNetInfo
CDAQMXStatus::getMXStatus
CDAQMXSysInfo::getMXSystemInfo
```

talkFIFODataInstMX

構文

```
int talkFIFODataInstMX(DAQMX daqmx, int fifoNo);
```

宣言

```
Public Declare Function talkFIFODataInstMX Lib "DAQMX" (ByVal
daqmx As Long, ByVal fifoNo As Long) As Long
```

引数

daqmx	機器記述子を指定します。
fifoNo	FIFO番号を指定します。

説明

指定されたFIFO番号の瞬時値の測定データの取得を宣言します。

・ 取得動作については，talkChDataMX関数を参照してください。

戻り値

エラー番号を返します。

参照

`talkFIFODataMX`

talkFIFODataMX [Visual Cのみ]

構文

```
int talkFIFODataMX(DAQMX daqmx, int fifoNo, MXDataNo
startDataNo, MXDataNo endDataNo);
```

引数

daqmx	機器記述子を指定します。
fifoNo	FIFO番号を指定します。
startDataNo	開始データ番号を指定します。
endDataNo	終了データ番号を指定します。

説明

指定されたFIFO番号の測定データの取得を宣言します。

- 取得する範囲は，開始/終了データ番号で指定します。
- 瞬時値を取得する場合，開始/終了データ番号に，定数値の「瞬時値指定用データ番号」を指定します。または，talkFIFODataInstMX関数を使用します。
- 本関数の実行後，getTimeDataMX関数でデータ個数分のデータ時刻を取得します。次に，getChDataMX関数でデータ個数分の測定データを取得します。
- Visual Basicの場合，talkFIFODataVBMX関数を使用してください。

戻り値

エラー番号を返します。

エラー：

Not descriptor　機器記述子がありません。

参照

`CDAQMX::talkFIFOData`

talkFIFODataVBMX

構文

```
int talkFIFODataVBMX(DAQMX daqmx, int fifoNo, MXDataNo *
startDataNo, MXDataNo * endDataNo);
```

宣言

```
Public Declare Function talkFIFODataVBMX Lib "DAQMX" (ByVal
daqmx As Long, ByVal fifoNo As Long, ByRef startDataNo As
MXDataNo, ByRef endDataNo As MXDataNo) As Long
```

引数

daqmx	機器記述子を指定します。
fifoNo	FIFO番号を指定します。
startDataNo	開始データ番号を指定します。
endDataNo	終了データ番号を指定します。

説明

指定されたFIFO番号の測定データの取得を宣言します。

- データ番号の指定が，参照指定であること以外は，talkFIFODataMX関数と同じです。
- Visual Cの場合，データ番号にNULLを指定すると，定数値の「瞬時値指定用データ番号」とみなします。
- 取得動作については，talkFIFODataMX関数を参照してください。

戻り値

エラー番号を返します。

参照

`talkFIFODataMX`

toAlarmNameMX

構文

```
int toAlarmNameMX(int iAlarmType, char * strAlarm, int
lenAlarm);
```

宣言

```
Public Declare Function toAlarmNameMX Lib "DAQMX" (ByVal
iAlarmType As Long, ByVal strAlarm As String, ByVal lenAlarm
As Long) As Long
```

引数

iAlarmType	アラーム種類を指定します。
strAlarm	文字列を格納する領域を指定します。
lenAlarm	文字列を格納する領域のバイト数を指定します。

説明

指定されたアラーム種類に対応する文字列を，指定された領域に格納します。

- 領域に格納する文字列には，終端(NULL)も含まれます。
- 実際の文字列の長さが戻り値になります。戻り値に終端は含まれません。

戻り値

文字列の長さを返します。

参照

`CDAQMXDataInfo::getAlarmName`

toAOPWMValueMX

構文

```
int toAOPWMValueMX(double realValue, int iRangeAOPWM);
```

宣言

```
Public Declare Function toAOPWMValueMX Lib "DAQMX" (ByVal
realValue As Double, ByVal iRangeAOPWM As Long) As Long
```

引数

realValue	実際の出力値を指定します。
iRangeAOPWM	レンジ種類を指定します。

説明

実際の出力値を， 指定されたレンジ種類に従って，AO/PWMデータの出力データ値に変換します。

・ 有効なレンジ種類は，AOレンジとPWMレンジです。

・ 不明の場合，0を返します。

戻り値

出力データ値を返します。

参照

`CDAQMXAOPWMData::toAOPWMValue`

toDateTimeMX

構文

```
void toDateTimeMX(MXDateTime * pMXDateTime, int * pYear, int *
pMonth, int * pDay, int * pHour, int * pMinute, int *
pSecond);
```

宣言

```
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)
```

引数

pMXDateTime	時刻情報を指定します。
pYear	年の値の返却先を指定します。
pMonth	月の値の返却先を指定します。
pDay	日の値の返却先を指定します。
pHour	時の値の返却先を指定します。
pMinute	分の値の返却先を指定します。
pSecond	秒の値の返却先を指定します。

説明

指定された時刻情報内の 1970年01月01日からの秒数を年月日時分秒の値に変換します。

・ 年には4桁の数値を返します。 月には1から12を返します。 日には1から31を返します。 時には0から23を返します。 分には0から59を返します。 秒には0から59を返します。

参照

`CDAQDateTime::toLocalDateTime`

toDoubleValueMX

構文

```
double toDoubleValueMX(int dataValue, int point);
```

宣言

```
Public Declare Function toDoubleValueMX Lib "DAQMX" (ByVal
dataValue As Long, ByVal point As Long) As Double
```

引数

dataValue	データ値を指定します。
point	小数点位置を指定します。

説明

指定されたデータ値と小数点位置から測定値を生成します。

戻り値

測定値を倍精度浮動小数で返します。

参照

`CDAQDataInfo::toDoubleValue`

toErrorMessageMX

構文

```
int toErrorMessageMX(int errCode, char * errStr, int errLen);
```

宣言

```
Public Declare Function toErrorMessageMX Lib "DAQMX" (ByVal
errCode As Long, ByVal errStr As String, ByVal errLen As Long)
As Long
```

引数

errCode	エラー番号を指定します。
errStr	文字列を格納する領域を指定します。
errLen	文字列を格納する領域のバイト数を指定します。

説明

エラー番号に対応するエラーメッセージ文字列を，指定された領域に格納します。

- 領域に格納する文字列には，終端(NULL)も含まれます。
- 実際の文字列の長さが戻り値になります。戻り値に終端は含まれません。

戻り値

文字列の長さを返します。

参照

`getErrorMessageMX`

toRealValueMX

構文

```
double toRealValueMX(int iAOPWMValue, int iRangeAOPWM);
```

宣言

```
Public Declare Function toRealValueMX Lib "DAQMX" (ByVal
iAOPWMValue As Long, ByVal iRangeAOPWM As Long) As Double
```

引数

iAOPWMValue　出力データ値を指定します。
iRangeAOPWM レンジ種類を指定します。

説明

AO/PWMデータの出力データ値を，指定されたレンジ種類に従って，実際の出力値に変換します。

・ 有効なレンジ種類は，AOレンジとPWMレンジです。
・ 不明の場合，0を返します。

戻り値

実際の出力値を返します。

参照

`CDAQMXAOPWMData::toRealValue`

toStringValueMX

構文

```
int toStringValueMX(int dataValue, int point, char * strValue,
int lenValue);
```

宣言

```
Public Declare Function toStringValueMX Lib "DAQMX" (ByVal
dataValue As Long, ByVal point As Long, ByVal strValue As
String, ByVal lenValue As Long) As Long
```

引数

dataValue	データ値を指定します。
point	小数点位置を指定します。
strValue	文字列を格納する領域を指定します。
lenValue	文字列を格納する領域のバイト数を指定します。

説明

指定されたデータ値と小数点位置から測定値を生成します。

- 生成された測定値を文字列に変換して，指定された領域に格納します。
- 領域に格納する文字列には，終端(NULL)も含まれます。
- 実際の文字列の長さが戻り値になります。戻り値に終端は含まれません。

戻り値

文字列の長さを返します。

参照

`CDAQDataInfo::toStringValue`

toStyleVersionMX

構文

```
int toStyleVersionMX(int style)
```

宣言

```
Public Declare Function toStyleVersionMX Lib "DAQMX" (ByVal
style As Long) As Long
```

引数

style　　スタイルを指定します。

説明

システム構成データのスタイルからバージョン番号を取得します。

戻り値

スタイルバージョンを返します。

参照

CDAQMXSysInfo::toStyleVersion

6.1　MX100の定数の概要

本APIでは，以下の種類の定数を用意しています。定数は，Visual C++，Visual C，Visual Basicで共通です。

種類	説明	ページ
通信用定数	MX100の通信ポート番号	6-3
個数値	モジュール数など	6-3
最大値	タグ文字列最大長など	6-3
定数値	瞬時値指定用データ番号など	6-4
有効無効値	有効(ON)設定，無効(OFF)設定	6-4
フラグステータス	データ取得時に最終データを判別	6-4
データステータス値	測定データの状態	6-4
アラーム種類	上限アラームなど	6-5
システム制御種類	システム制御操作	6-5
チャネル種類	ユニバーサル入力，ディジタル入力など	6-5
スケール種類	スケールなしまたは線形スケール	6-6
モジュール種類	ユニバーサル入力4CHなど	6-6
チャネル数	4または10	6-6
周期種類	10ms～60000ms	6-7
フィルタ時定数	入力フィルタ時定数	6-7
RJC種類	内蔵ＲＪＣまたは外部RJC	6-7
バーンアウト種類	Off/Up/Down	6-7
ユニット種類	MX100	6-7
端子種類	ねじ端子または押し締め端子	6-8
A/D積分時間種類	自動，50Hz，または60Hz	6-8
温度単位種類	℃	6-8
CF書き込み種類	CFへのデータ書き込み方式	6-8
CFステータス種類	CFの状態	6-8
ユニットステータス値	ユニットの状態	6-8
FIFOステータス値	FIFOの状態	6-9
表示形式値	7セグメントLEDの表示形式	6-9
出力種類	出力レンジの種類	6-9
選択値	出力値の選択	6-9
伝送状態	伝送出力の状態	6-9
初期バランス結果	初期バランスの実行結果	6-9
オプション	オプションの有無	6-10
参照レンジ	チャネル間差演算チャネルの測定レンジとして基準チャネルの測定レンジを参照	6-10
直流電圧レンジ	20mVレンジなど	6-10
熱電対レンジ	Type Rなど	6-11
測温抵抗体(1mA)レンジ	Pt100など	6-12
測温抵抗体(2mA)レンジ	Pt100など	6-14
測温抵抗体(その他)レンジ	Pt500，Pt1000	6-15

種類	説明	ページ
抵抗レンジ	20Ω，200Ω，または2kΩ(0.25mA)	6-15
ディジタル入力(DI)レンジ	Levelまたは接点入力	6-15
ディジタル入力(DI)詳細レンジ	「ユニバーサル入力4CHモジュールの接点入力」など	6-16
ひずみレンジ	2000μひずみ，20000μひずみ，または200000μひずみ	6-16
AOレンジ	V出力，またはmA出力	6-16
PWMレンジ	PWM出力分解能　1msまたは10ms	6-16
通信レンジ	CAN Bus入力	6-16
パルスレンジ	パルス入力	6-16

6.2 MX100の定数

定数のニーモニックと意味を説明しています。MX100の機能の詳細については，それぞれのユーザーズマニュアルを参照してください。

通信用定数

ニーモニック	内容
DAQMX_COMMPORT	MX100の通信ポート番号です。

個数値

ニーモニック	内容
DAQMX_NUMMODULE	モジュール数です。
DAQMX_NUMCHANNEL	チャネル数です。
DAQMX_NUMDO	DOデータ数です。
DAQMX_NUMFIFO	FIFO数です。
DAQMX_NUMALARM	アラーム数です。R3.01から個数が「4」になりました。
DAQMX_NUMSEGMENT	7セグメントLED数です。
DAQMX_NUMMACADDR	MACアドレスの要素数(バイト数)です。
DAQMX_NUMAOPWM	AO/PWMデータ個数です。
DAQMX_NUMBALANCE	初期バランスデータ個数です。
DAQMX_NUMOUTPUT	出力チャネルデータ個数です。

最大値

ニーモニック	内容
DAQMX_MAXHOSTNAMELEN	ホスト名文字列最大長です。
DAQMX_MAXUNITLEN	単位名文字列最大長です。
DAQMX_MAXTAGLEN	タグ文字列最大長です。
DAQMX_MAXCOMMENTLEN	コメント文字列最大長です。
DAQMX_MAXSERIALLEN	MX100のシリアル番号文字列最大長です。
DAQMX_MAXPARTNOLEN	パート番号(ファームウエアの部品番号)文字列最大長です。
DAQMX_MAXDECIMALPOINT	小数点位置の最大値です。
DAQMX_MAXDISPTIME	7セグメントLED表示時間の最大値です。
DAQMX_MAXPULSETIME	パルス周期倍率の最大値です。

文字列の最大長は，終端(NULL)を含みません。

定数値

ニーモニック	内容
DAQMX_INSTANTANEOUS	瞬時値取得を指定するときのデータ番号。
DAQMX_REFCHNO_ALL	全参照チャネル番号指定。
DAQMX_LEVELNO_ALL	全アラームレベル番号指定。
DAQMX_DONO_ALL	全DO番号指定。
DAQMX_SEGMENTNO_ALL	7セグメントLEDの全セグメント番号指定。
DAQMX_CHNO_ALL	全チャネル番号指定。
DAQMX_MODULENO_ALL	全モジュール番号指定。
DAQMX_FIFONO_ALL	全FIFO番号指定。
DAQMX_AOPWMNO_ALL	全AO/PWMデータ番号指定。
DAQMX_BALANCENO_ALL	全初期バランス番号指定。
DAQMX_OUTPUTNO_ALL	全出力データ番号指定。
DAQMX_REFCHNO_NONE	未定義参照チャネル番号。

有効無効値

ニーモニック	内容
DAQMX_VALID_OFF	無効(OFF)値
DAQMX_VALID_ON	有効(ON)値

フラグステータス

論理OR演算で合成できます。

ニーモニック	内容
DAQMX_FLAG_OFF	全OFF。
DAQMX_FLAG_ENDDATA	チャネル単位またはデータ番号単位で取得するデータが最終データです。

データステータス値

ニーモニック	内容
DAQMX_DATA_UNKNOWN	不明状態です。
DAQMX_DATA_NORMAL	正常状態です。
DAQMX_DATA_PLUSOVER	プラスオーバ状態です。
DAQMX_DATA_MINUSOVER	マイナスオーバ状態です。
DAQMX_DATA_SKIP	スキップ(未使用)状態です。
DAQMX_DATA_ILLEGAL	不明な不正データ状態です。
DAQMX_DATA_NODATA	データなし状態です。
DAQMX_DATA_LACK	データ抜け状態です。
DAQMX_DATA_INVALID	不正状態です。

アラーム種類

□はスペースです。

ニーモニック	内容	文字列
DAQMX_ALARM_NONE	アラームなし	□□
DAQMX_ALARM_UPPER	上限アラーム	H□
DAQMX_ALARM_LOWER	下限アラーム	L□
DAQMX_ALARM_UPDIFF	差上限アラーム	dH
DAQMX_ALARM_LOWDIFF	差下限アラーム	dL

システム制御種類

ニーモニック	内容
DAQMX_SYSTEM_RECONSTRUCT	システム再構築
DAQMX_SYSTEM_INITOPE	システム初期化
DAQMX_SYSTEM_RESETALARM	アラームリセット(アラームACK)

チャネル種類

ニーモニック	内容
DAQMX_CHKIND_NONE	未使用
DAQMX_CHKIND_AI	AI*
DAQMX_CHKIND_AIDIFF	AI*(チャネル間差演算指定)
DAQMX_CHKIND_AIRJC	AI*(リモートRJC適用チャネル)
DAQMX_CHKIND_DI	DI*
DAQMX_CHKIND_DIDIFF	DI*(チャネル間差演算指定)
DAQMX_CHKIND_DO	DO*(アラーム出力指定)
DAQMX_CHKIND_DOCOM	DO*(コマンドDO指定)
DAQMX_CHKIND_DOFAIL	DO*(システムFail出力指定)
DAQMX_CHKIND_DOERR	DO*(システムError出力指定)
DAQMX_CHKIND_AO	AO*(伝送出力)
DAQMX_CHKIND_AOCOM	AO*(コマンドAO)
DAQMX_CHKIND_PWM	PWM*(伝送出力)
DAQMX_CHKIND_PWMCOM	PWM*(コマンドPWM)
DAQMX_CHKIND_PI	パルス入力
DAQMX_CHKIND_PIDIFF	パルス入力(チャネル間差演算指定)
DAQMX_CHKIND_CI	CAN Bus入力
DAQMX_CHKIND_CIDIFF	CAN Bus入力(チャネル間差演算指定)

* AI : Analog Input，直流電圧入力，TC入力など
AO : Analog Output, アナログ出力
DI : Digital Input，ディジタル入力
DO : Digital Output，ディジタル出力
PWM : Pulse Width Modulation，PWM出力

入力チャネルの場合，レンジ設定のレンジ種類によりチャネル種類は確定します。
出力チャネルの場合，チャネル設定でチャネル種類を設定します。

スケール種類

ニーモニック	内容
DAQMX_SCALE_NONE	スケールなし
DAQMX_SCALE_LINER	線形スケール

モジュール種類

ニーモニック	内容
DAQMX_MODULE_NONE	なし
DAQMX_MODULE_MX110UNVH04	4ch高速ユニバーサル入力モジュール
DAQMX_MODULE_MX110UNVM10	10ch中速ユニバーサル入力モジュール
DAQMX_MODULE_MX115D05H10	10ch高速ディジタル入力モジュール
DAQMX_MODULE_MX125MKCM10	10ch中速ディジタル出力モジュール
DAQMX_MODULE_MX110V4RM06	6ch中速4線式RTD抵抗入力モジュール
DAQMX_MODULE_MX112NDIM04	4chひずみ入力モジュール(NDIS)
DAQMX_MODULE_MX112B35M04	4chひずみ入力モジュール(350Ω)
DAQMX_MODULE_MX112B12M04	4chひずみ入力モジュール(20Ω)
DAQMX_MODULE_MX115D24H10	10ch高速ディジタル入力モジュール(DC24V)
DAQMX_MODULE_MX120VAOM08	8ch中速アナログ出力モジュール
DAQMX_MODULE_MX120PWMM08	8ch中速PWM出力モジュール
DAQMX_MODULE_HIDDEN	複数スロット幅を使用するモジュールが，モジュールを装着したスロット以外に占有するスロット(仮想モジュール部)
DAQMX_MODULE_MX114PLSM10	10chパルス入力モジュール
DAQMX_MODULE_MX110VTDL30	30ch中速DCV/TC/DI入力モジュール
DAQMX_MODULE_MX118CANM10	CAN Busモジュール 10ch*
DAQMX_MODULE_MX118CANM20	CAN Busモジュール 20ch*
DAQMX_MODULE_MX118CANM30	CAN Busモジュール 30ch*
DAQMX_MODULE_MX118CANSUB	CAN Busモジュールが，モジュールを装着したスロット以外に占有するスロット(仮想CAN Busモジュール部)
DAQMX_MODULE_MX118CANMERR	CAN Busモジュールの位置エラー
DAQMX_MODULE_MX118CANSERR	CAN Busモジュールが，モジュールを装着したスロット以外に占有するスロット(仮想CAN Busモジュール部)のエラー。

* CAN Busモジュールは，使用チャネル数で区別されます。

チャネル数

ニーモニック	内容
DAQMX_CHNUM_0	0
DAQMX_CHNUM_4	4
DAQMX_CHNUM_6	6
DAQMX_CHNUM_8	8
DAQMX_CHNUM_10	10
DAQMX_CHNUM_30	30

周期種類

ニーモニック	内容
DAQMX_INTERVAL_10	10msec
DAQMX_INTERVAL_50	50msec
DAQMX_INTERVAL_100	100msec
DAQMX_INTERVAL_200	200msec
DAQMX_INTERVAL_500	500msec
DAQMX_INTERVAL_1000	1000msec
DAQMX_INTERVAL_2000	2000msec
DAQMX_INTERVAL_5000	5000msec
DAQMX_INTERVAL_10000	10000msec
DAQMX_INTERVAL_20000	20000msec
DAQMX_INTERVAL_30000	30000msec
DAQMX_INTERVAL_60000	60000msec

フィルタ係数

ニーモニック	内容
DAQMX_FILTER_0	係数0
DAQMX_FILTER_5	係数5
DAQMX_FILTER_10	係数10
DAQMX_FILTER_20	係数20
DAQMX_FILTER_25	係数25
DAQMX_FILTER_40	係数40
DAQMX_FILTER_50	係数50
DAQMX_FILTER_100	係数100

RJC種類

ニーモニック	内容
DAQMX_RJC_INTERNAL	MX100のRJC機能
DAQMX_RJC_EXTERNAL	外部のRJC機能

バーンアウト種類

ニーモニック	内容
DAQMX_BURNOUT_OFF	バーンアウト検出機能なし
DAQMX_BURNOUT_UP	バーンアウト検出時，＋レンジオーバの表示
DAQMX_BURNOUT_DOWN	バーンアウト検出時，－レンジオーバの表示

ユニット種類論理

論理OR演算で合成されます。

ニーモニック	内容
DAQMX_UNITTYPE_NONE	不明
DAQMX_UNITTYPE_MX100	MX100

端子種類

ニーモニック	内容
DAQMX_TERMINAL_SCREW	ねじ端子
DAQMX_TERMINAL_CLAMP	押し締め端子
DAQMX_TERMINAL_NDIS	NDIS
DAQMX_TERMINAL_DSUB	D-SUB 9ピン

A/D積分時間種類

ニーモニック	内容
DAQMX_INTEGRAL_AUTO	自動(50Hz/60HzをMX100が自動設定)
DAQMX_INTEGRAL_50HZ	50Hz
DAQMX_INTEGRAL_60HZ	60Hz

温度単位種類

ニーモニック	内容
DAQMX_TEMPUNIT_C	℃

CF書き込み種類

ニーモニック	内容
DAQMX_CFWRITEMODE_ONCE	上書きなし(空き容量がなくなると書き込みを停止します)
DAQMX_CFWRITEMODE_FIFO	繰り返し(古いデータから順に上書きされます)

CFステータス種類

論理OR演算で合成されます。

ニーモニック	内容
DAQMX_CFSTATUS_NONE	全OFF
DAQMX_CFSTATUS_EXIST	存在の有無
DAQMX_CFSTATUS_USE	CFカードを使用可能です。
DAQMX_CFSTATUS_FORMAT	CFカードをフォーマット中です。

ユニットステータス値

ニーモニック	内容
DAQMX_UNITSTAT_NONE	不明
DAQMX_UNITSTAT_INIT	初期化中
DAQMX_UNITSTAT_STOP	停止中
DAQMX_UNITSTAT_RUN	測定中
DAQMX_UNITSTAT_BACKUP	測定中(バックアップ中)

FIFOステータス値

ニーモニック	内容
DAQMX_FIFOSTAT_NONE	不明
DAQMX_FIFOSTAT_INIT	初期化中
DAQMX_FIFOSTAT_STOP	停止中
DAQMX_FIFOSTAT_RUN	測定中
DAQMX_FIFOSTAT_BACKUP	測定中(バックアップ中)

表示形式値

ニーモニック	内容
DAQMX_DISPTYPE_NONE	未定義
DAQMX_DISPTYPE_ON	点灯
DAQMX_DISPTYPE_BLINK	点滅表示

出力種類

ニーモニック	内容	設定範囲
DAQMX_OUTPUT_NONE	出力なし	
DAQMX_OUTPUT_AO_10V	V出力	−11.000～11.000V
DAQMX_OUTPUT_AO_20MA	mA出力	0～22.000 mA
DAQMX_OUTPUT_PWM_1MS	PWM出力 分解能 1ms	0～100.000 %
DAQMX_OUTPUT_PWM_10MS	PWM出力 分解能 10ms	0～100.000 %

出力のレンジ種類と対応します。

選択値

ニーモニック	内容
DAQMX_CHOICE_PREV	前回値
DAQMX_CHOICE_PRESET	指定値

伝送状態

ニーモニック	内容
DAQMX_TRANSMIT_NONE	指定なし(不明)
DAQMX_TRANSMIT_RUN	出力開始(出力中)
DAQMX_TRANSMIT_STOP	出力停止

初期バランス結果

ニーモニック	内容
DAQMX_BALANCE_NONE	指定なし
DAQMX_BALANCE_DONE	正常終了
DAQMX_BALANCE_NG	範囲外
DAQMX_BALANCE_ERROR	エラー

オプション

ニーモニック	内容
DAQMX_OPTION_NONE	オプションなし
DAQMX_OPTION_DS	Dual Save(/DSオプション)

レンジ種類

参照レンジ

ニーモニック	内容
DAQMX_RANGE_REFERENCE	基準チャネルの測定レンジ

差演算チャネルの測定レンジとしてこの定数を指定すると，差演算チャネルの測定レンジが，基準チャネルの測定レンジと同じレンジに設定されます。
参照レンジは，差演算などで，参照する基準チャネルと同じレンジに設定したいときの指定に用います。

直流電圧レンジ

ニーモニック	内容	設定範囲
DAQMX_RANGE_VOLT_20MV	20mV	−20.000～20.000 mV
DAQMX_RANGE_VOLT_60MV	60mV	−60.00～60.00 mV
DAQMX_RANGE_VOLT_200MV	200mV	−200.00～200.00 mV
DAQMX_RANGE_VOLT_2V	2V	−2.0000～2.0000 V
DAQMX_RANGE_VOLT_6V	6V	−6.000～6.000 V
DAQMX_RANGE_VOLT_20V	20V	−20.000～20.000 V
DAQMX_RANGE_VOLT_100V	100V	−100.00～100.00 V
DAQMX_RANGE_VOLT_60MVH	60mV：高分解能	0.000～60.000 mV
DAQMX_RANGE_VOLT_1V	1V	−10000～1.0000 V
DAQMX_RANGE_VOLT_6VH	6V：高分解能	0.0000～6.0000 V

熱電対レンジ

ニーモニック	内容	設定範囲
DAQMX_RANGE_TC_R	R	0.0～1760.0°C
DAQMX_RANGE_TC_S	S	0.0～1760.0°C
DAQMX_RANGE_TC_B	B	0.0～1820.0°C
DAQMX_RANGE_TC_K	K	−200.0～1370.0°C
DAQMX_RANGE_TC_E	E	−200.0～800.0°C
DAQMX_RANGE_TC_J	J	−200.0～1100.0°C
DAQMX_RANGE_TC_T	T	−200.0～400.0°C
DAQMX_RANGE_TC_N	N	0.0～1300.0°C
DAQMX_RANGE_TC_W	W	0.0～2315.0°C
DAQMX_RANGE_TC_L	L	−200.0～900.0°C
DAQMX_RANGE_TC_U	U	−200.0～400.0°C
DAQMX_RANGE_TC_KP	KpAu7Fe	0.0～300.0K
DAQMX_RANGE_TC_PL	PLATINEL	0.0～1400.0°C
DAQMX_RANGE_TC_PR	PR40−20	0.0～1900.0°C
DAQMX_RANGE_TC_NNM	NiNiMo	0.0～1310.0°C
DAQMX_RANGE_TC_WR	WRe3−25	0.0～2400.0°C
DAQMX_RANGE_TC_WWR	W/WRe26	0.0～2400.0°C
DAQMX_RANGE_TC_AWG	Type−N(AWG14)	0.0～1300.0°C
DAQMX_RANGE_TC_XK	XK	−200.0 ～ 600.0°C

測温抵抗体(1mA)レンジ

ニーモニック	内容	設定範囲
DAQMX_RANGE_RTD_1MAPT	Pt100	−200.0～600.0°C
DAQMX_RANGE_RTD_1MAJPT	JPt100	−200.0～550.0°C
DAQMX_RANGE_RTD_1MAPTH	Pt100 :高分解能	−140.00～150.00°C
DAQMX_RANGE_RTD_1MAJPTH	JPt100 :高分解能	−140.00～150.00°C
DAQMX_RANGE_RTD_1MANIS	Ni100:SAMA	−200.0～250.0°C
DAQMX_RANGE_RTD_1MANID	Ni100:DIN	−60.0～180.0°C
DAQMX_RANGE_RTD_1MANI120	Ni120	−70.0～200.0°C
DAQMX_RANGE_RTD_1MAPT50	Pt50	−200.0～550.0°C
DAQMX_RANGE_RTD_1MACU10GE	Cu10:GE	−200.0～300.0°C
DAQMX_RANGE_RTD_1MACU10LN	Cu10:L&N	−200.0～300.0°C
DAQMX_RANGE_RTD_1MACU10WEED	Cu10:WEED	−200.0～300.0°C
DAQMX_RANGE_RTD_1MACU10BAILEY	Cu10:BAILEY	−200.0～300.0°C
DAQMX_RANGE_RTD_1MAJ263B	J263*B	0.0～300.0K
DAQMX_RANGE_RTD_1MACU10A392	Cu10 at 20°C a=0.00392	−200.0 300.0°C
DAQMX_RANGE_RTD_1MACU10A393	Cu10 at 20°C a=0.00393	−200.0～300.0°C
DAQMX_RANGE_RTD_1MACU25	Cu25 at 0°C a=0.00425	−200.0～300.0°C
DAQMX_RANGE_RTD_1MACU53	Cu53 at 0°C a=0.00426035	−50.0～150.0°C
DAQMX_RANGE_RTD_1MACU100	Cu100 at 0°C a=0.00425	−50.0～150.0°C

ニーモニック	内容	設定範囲
DAQMX_RANGE_RTD_1MAPT25	Pt25	−200.0～550.0°C
DAQMX_RANGE_RTD_1MACU10GEH	Cu10:GE :高分解能	−200.0～300.0°C
DAQMX_RANGE_RTD_1MACU10LNH	Cu10:L&N :高分解能	−500.0～500.0°C
DAQMX_RANGE_RTD_1MACU10WEEDH	Cu10:WEED :高分解能	−500.0～500.0°C
DAQMX_RANGE_RTD_1MACU10BAILEYH	Cu10:BAILEY :高分解能	−500.0～500.0°C
DAQMX_RANGE_RTD_1MAPTN	Pt100 :高耐ノイズ	−800.0～800.0°C
DAQMX_RANGE_RTD_1MAJPTN	Jpt100 :高耐ノイズ	−750.0～750.0°C
DAQMX_RANGE_RTD_1MAPTG	Pt100G	−200.0 ～ 600.0°C
DAQMX_RANGE_RTD_1MACU100G	Cu100G	−200.0 ～ 200.0°C
DAQMX_RANGE_RTD_1MACU50G	Cu50G	−200.0 ～ 200.0°C
DAQMX_RANGE_RTD_1MACU10G	Cu10G	−200.0 ～ 200.0°C

測温抵抗体(2mA)レンジ

ニーモニック	内容	設定範囲
DAQMX_RANGE_RTD_2MAPT	Pt100	−200.0～250.0°C
DAQMX_RANGE_RTD_2MAJPT	JPt100	−200.0～250.0°C
DAQMX_RANGE_RTD_2MAPTH	Pt100 :高分解能	−140.00～150.00°C
DAQMX_RANGE_RTD_2MAJPTH	JPt100 :高分解能	−140.00～150.00°C
DAQMX_RANGE_RTD_2MAPT50	Pt50	−200.0～550.0°C
DAQMX_RANGE_RTD_2MACU10GE	CU10:GE	−200.0～300.0°C
DAQMX_RANGE_RTD_2MACU10LN	Cu10:L&N	−200.0～300.0°C
DAQMX_RANGE_RTD_2MACU10WEED	Cu10:WEED	−200.0～300.0°C
DAQMX_RANGE_RTD_2MACU10BAILEY	Cu10:BAILEY	−200.0～300.0°C
DAQMX_RANGE_RTD_2MAJ263B	J263*B	0.0～300.0K
DAQMX_RANGE_RTD_2MACU10A392	Cu10 at 20°C a=0.00392	−200.0～300.0°C
DAQMX_RANGE_RTD_2MACU10A393	Cu10 at 20°C a=0.00393	−200.0～300.0°C
DAQMX_RANGE_RTD_2MACU25	Cu25 at 0°C a=0.00425	−200.0～300.0°C
DAQMX_RANGE_RTD_2MACU53	Cu53 at 0°C a=0.00426035	−50.0～150.0°C
DAQMX_RANGE_RTD_2MACU100	Cu100 at 0°C a=0.00425	−50.0～150.0°C
DAQMX_RANGE_RTD_2MAPT25	Pt25	−200.0～550.0°C
DAQMX_RANGE_RTD_2MACU10GEH	CU10:GE :高分解能	−200.0～300.0°C
DAQMX_RANGE_RTD_2MACU10LNH	Cu10:L&N :高分解能	−200.0～300.0°C
DAQMX_RANGE_RTD_2MACU10WEEDH	Cu10:WEED :高分解能	−200.0～300.0°C

ニーモニック	内容	設定範囲
DAQMX_RANGE_RTD_2MACU10BAILEYH	Cu10:BAILEY :高分解能	−200.0～300.0℃
DAQMX_RANGE_RTD_2MAPTN	Pt100 :高耐ノイズ	−200.0～250.0℃
DAQMX_RANGE_RTD_2MAJPTN	Jpt100 :高耐ノイズ	−200.0～250.0℃
DAQMX_RANGE_RTD_2MACU100G	Cu100G	−200.0 ～ 200.0℃
DAQMX_RANGE_RTD_2MACU50G	Cu50G	−200.0 ～ 200.0℃
DAQMX_RANGE_RTD_2MACU10G	Cu10G	−200.0 ～ 200.0℃

測温抵抗体(その他)のレンジ

ニーモニック	内容	設定範囲
DAQMX_RANGE_RTD_025MAPT500	0.25mA Pt500	−200.0～600.0 ℃
DAQMX_RANGE_RTD_025MAPT1K	0.25mA Pt1000	−200.0～600.0 ℃

抵抗レンジ

ニーモニック	内容	設定範囲
DAQMX_RANGE_RES_20	20Ω	0～20.000
DAQMX_RANGE_RES_200	200Ω	0～200.00
DAQMX_RANGE_RES_2K	2kΩ (0.25mA)	0～2000.0

ディジタル入力(DI)レンジ

ニーモニック	内容	設定範囲
DAQMX_RANGE_DI_LEVEL	Level	0：2.4V未満，1：2.4V以上
DAQMX_RANGE_DI_CONTACT	接点入力	0：open, 1：close

ディジタル入力(DI)詳細レンジ

ニーモニック	内容
DAQMX_RANGE_DI_LEVEL_AI	ユニバーサル入力モジュールのDI/Level
DAQMX_RANGE_DI_CONTACT_AI4	4chユニバーサル入力モジュールのDI/接点入力
DAQMX_RANGE_DI_CONTACT_AI10	10chユニバーサル入力モジュールのDI/接点入力
DAQMX_RANGE_DI_CONTACT_AI30	30chDCV/TC/DI入力モジュールのDI/接点入力
DAQMX_RANGE_DI_LEVEL_DI	ディジタル入力モジュールのDI/Level
DAQMX_RANGE_DI_CONTACT_DI	ディジタル入力モジュールのDI/接点入力
DAQMX_RANGE_DI_LEVEL_DI5V*	DI 5V ディジタル入力モジュール(5V用)のDI/接点入力
DAQMX_RANGE_DI_LEVEL_DI24V	DI 24V ディジタル入力モジュール(24V用)のDI/接点入力

* DAQMX_RANGE_DI_LEVEL_DIの別名称です。 24V用と区別するために定義されています。

ひずみレンジ

ニーモニック	内容	設定範囲
DAQMX_RANGE_STRAIN_2K	2000μSTR	−2000.0～2000.0 ±563200000
DAQMX_RANGE_STRAIN_20K	20000μSTR	−20000～20000 ±56320000
DAQMX_RANGE_STRAIN_200K	200000μSTR	−200000～200000 ±5632000

AOレンジ

ニーモニック	内容	設定範囲
DAQMX_RANGE_AO_10V	V出力	−10.000～10.000V
DAQMX_RANGE_AO_20MA	mA出力	0.000～20.000mA

PWMレンジ

ニーモニック	内容	設定範囲
DAQMX_RANGE_PWM_1MS	PWM出力 分解能 1ms	0～100.000 %
DAQMX_RANGE_PWM_10MS	PWM出力 分解能 10ms	0～100.000 %

通信レンジ

ニーモニック	内容	設定範囲
DAQMX_RANGE_COM_CAN	CAN Bus	−30000 ～ 30000

パルスレンジ

ニーモニック	内容	設定範囲
DAQMX_RANGE_PI_LEVEL	パルス/Level	0 ～ 30000
DAQMX_RANGE_PI_CONTACT	パルス/接点入力	0 ～ 30000

6.3 MX100の設定項目番号

設定データの構造体の各項目に一意になるように付けられた番号です。
設定データを送信するとき， 整合性のチェックでエラーを検出した場所を表す番号です。
定義ファイルが用意されています。

設定項目番号一覧

MXConfigData構造体の各項目領域を番号化したものです。
以下のファイルに定義が記述されています。必要に応じて読み込んでください。

- DAQMXItems.h
- DAQMXItems.bas
- DAQMXItems.txt
- DAQMXItems.cs
- DAQMXItems.vb

チャネル番号項目など，固定な内容で，読み出しても特に意味のないものは番号化されていません。
エラーが発生した場合，設定内容の値を確認してください。
文字列は，設定項目番号の値を項目名に変換した場合のものです。拡張用APIでだけ対応します。
Visual C#の場合，DAQMXItemsクラスの定数として定義してあります。

システム

ニーモニック	値	位置	内容	エラー	文字列
DAQMX_ITEM_NONE	−1	−	不明	−	−
DAQMX_ITEM_NETHOST	0	aNetInfo. aHost	ホスト名	−	Host Name
DAQMX_ITEM_NETADDRESS	1	aNetInfo.	IPアドレス aAddress	−	Address
DAQMX_ITEM_NETPORT	2	aNetInfo.	ポート番号 aPort	−	Port No
DAQMX_ITEM_NETSUBMASK	3	aNetInfo. aSubMask	サブネット マスク	−	Submask
DAQMX_ITEM_NETGATEWAY	4	aNetInfo. aGateway	GATEWAY アドレス	−	Gateway
DAQMX_ITEM_UNITTYPE	5	aSystemInfo. aUnit.aType	ユニット種類	−	Unit Type
DAQMX_ITEM_UNITSTYLE	6	aSystemInfo. aUnit.aStyle	スタイル	−	Unit Style
DAQMX_ITEM_UNITNO	7	aSystemInfo. aUnit.aNo	ユニット番号	範囲外の値です。	Unit No
DAQMX_ITEM_UNITTEMP	8	aSystemInfo. aUnit.aTempUnit	温度単位種類	範囲外の値です。	Temperature Unit

ニーモニック	値	位置	内容	エラー	文字列
DAQMX_ITEM_UNITCFTIMEOUT	9	aSystemInfo. aUnit.aCFTimeout	タイムアウト値	計算外の値です。付録3の「タイムアウト値の算出」を参照。	CF Timeout (s)
DAQMX_ITEM_UNITCFWRITEMODE	10	aSystemInfo. aUnit. aCFWriteMode	CF書き込み種類	範囲外の値です。	CF WriteMode
DAQMX_ITEM_UNITFREQUENCY	11	aSystemInfo. aUnit. aFrequency	電源周波数	—	Frequency
DAQMX_ITEM_UNITPARTNO	12	aSystemInfo. aUnit.aPartNo	パート番号	—	Part No
DAQMX_ITEM_UNITOPTION	13	aSystemInfo. aUnit.aProduct. aOption	オプション	—	Unit Option
DAQMX_ITEM_UNITSERIAL	14	aSystemInfo. aUnit.aProduct. aSerial	シリアル番号	—	Unit Serial
DAQMX_ITEM_UNITMAC	15	aSystemInfo. aUnit.aProduct. aMAC	MACアドレス	—	Mac Address
DAQMX_ITEM_UNITSTATUS	16	aStatus. aUnitStatus	ユニットステータス値	—	Unit Status
DAQMX_ITEM_CNTCONFIG	17	aStatus. aConfigCnt	設定番号	—	Count Config
DAQMX_ITEM_CNTTIME	18	aStatus. aTimeCnt	時刻番号	—	Count Time
DAQMX_ITEM_FIFONUM	19	aStatus. aFIFONum	FIFOの有効個数	—	FIFO Num
DAQMX_ITEM_BACKUP	20	aStatus. aBackup	バックアップの有無	—	Backup
DAQMX_ITEM_CFSTATUS	21	aStatus.aCFInfo. aStatus	CFステータス種類	—	CF Status
DAQMX_ITEM_CFSIZE	22	aStatus.aCFInfo. aSize	容量	—	CF Size (KB)
DAQMX_ITEM_CFREMAIN	23	aStatus.aCFInfo. aRemain	残容量	—	CF Remain (KB)
DAQMX_ITEM_STATUSTIME	2304	aStatus. aDateTime.aTime	秒数	—	Status Time
DAQMX_ITEM_STATUSMSEC	2305	aStatus. aDateTime. aMilliSecond	ミリ秒	—	Status Millisecond

「不明」の項目には文字列は存在しません。

FIFO

ニーモニック	値	位置	内容	文字列
DAQMX_ITEM_FIFOSTATUS	24	aStatus.aFIFOInfo[ ]. aStatus	FIFOステータス値	FIFO0 Status
DAQMX_ITEM_FIFOINTERVAL	28	aStatus.aFIFOInfo[ ]. aInterval	周期種類	FIFO0 Interval (msec)
DAQMX_ITEM_FIFOOLDNO	32	aStatus.aFIFOInfo[ ]. aOldNo	最古のデータ番号	FIFO0 Old No
DAQMX_ITEM_FIFONEWNO	36	aStatus.aFIFOInfo[ ]. aNewNo	最新のデータ番号	FIFO0 New No

実際の値は，表の値にFIFO番号(インデックス)を加算した値になります。
文字列「FIFO0」の0にはFIFO番号が入ります。

モジュール

ニーモニック	値	位置	内容	エラー	文字列
DAQMX_ITEM_MODULETYPE	40	aSystemInfo .aModule[ ]. aType	モジュール種類	範囲外の値です。	Module0 Type
DAQMX_ITEM_MODULECHNUM	48	aSystemInfo .aModule[ ]. aChNum	チャネル数	範囲外の値です。	Module0 Channel Num
DAQMX_ITEM_MODULEINTERVAL	56	aSystemInfo .aModule[ ]. aInterval	周期種類	範囲外の値です。FIFO個数を超えました。	Module0 Interval (msec)
DAQMX_ITEM_MODULEINTEGRAL	64	aSystemInfo .aModule[ ]. aIntegralTime	AD積分時間種類	範囲外の値です。	Module0 Integral
DAQMX_ITEM_MODULESTANDBY	'72	aSystemInfo .aModule[ ]. aStandbyType	起動時モジュール種類	ー	Module0 Standby Type
DAQMX_ITEM_MODULEREALTYPE	80	aSystemInfo .aModule[ ]. aRealType	実際のモジュール種類	ー	Module0 Real Type
DAQMX_ITEM_MODULESTATUS	88	aSystemInfo .aModule[ ]. aStatus	モジュール有効，無効	ー	Module0 Status
DAQMX_ITEM_MODULEVERSION	96	aSystemInfo .aModule[ ]. aVersion	モジュールバージョン	ー	Module0 Version
DAQMX_ITEM_MODULETERMINAL	104	aSystemInfox .aModule[ ]. aTerminalType	端子種類	ー	Module0 Terminal
DAQMX_ITEM_MODULEFIFONO	112	aSystemInfo .aModule[ ]. aFIFONo	FIFO番号	ー	Module0 FIFO No
DAQMX_ITEM_MODULESERIAL	120	aSystemInfo .aModule[ ]. aProduct.aSerial	シリアル番号	ー	Module0 Serial

実際の値は，表の値にモジュール番号(インデックス)を加算した値になります。
文字列「Module0」の0にはモジュール番号が入ります。

チャネル

ニーモニック	値	位置	内容	エラー	文字列
DAQMX_ITEM_CHVALID	128	aChConfigData. aChConfig[ ]. aChID.aValid	チャネルステータス	存在しないのに有効になっています。	Channel00 Valid
DAQMX_ITEM_CHPOINT	192	aChConfigData. aChConfig[ ]. aChID.aPoint	小数点位置	範囲外の値です。	Channel00 Point
DAQMX_ITEM_CHKIND	256	aChConfigData. aChConfig[ ]. aChID.aKind	チャネル種類	範囲外の値です。	Channel00 Kind
DAQMX_ITEM_CHRANGE	320	aChConfigData. aChConfig[ ]. aChID.aRange	レンジ種類	範囲外の値です。	Channel00 Range
DAQMX_ITEM_CHSCALE	384	aChConfigData. aChConfig[ ]. aChID.aScaleType	スケール種類	範囲外の値です。	Channel00 Scale

ニーモニック	値	位置	内容	エラー	文字列
DAQMX_ITEM_CHUNIT	338	aChConfigData.aChConfig[ ].aChID.aUnit	単位名	―	Channel00 Unit
DAQMX_ITEM_CHTAG	512	aChConfigData.aChConfig[ ].aChID.aTag	タグ	―	Channel00 Tag
DAQMX_ITEM_CHCOMMENT	576	aChConfigData.aChConfig[ ].aChID.aComment	コメント	―	Channel00 Comment
DAQMX_ITEM_ALARMTYPE	640	aChConfigData.aChConfig[ ].aChID.aAlarm[ ].aType	アラームレベル「1」または「2」のアラーム種類	範囲外の値です。	Channel00 Alarm0 Type
DAQMX_ITEM_ALARMON	768	aChConfigData.aChConfig[ ].aChID.aAlarm[ ].aON	アラームレベル「1」または「2」のOn値	範囲外の値です。	Channel00 Alarm0 On
DAQMX_ITEM_ALARMOFF	896	aChConfigData.aChConfig[ ].aChID.aAlarm[ ].aOFF	アラームレベル「1」または「2」のOff値	範囲外の値です。	Channel00 Alarm0 Off
DAQMX_ITEM_CHSPANMIN	1024	aChConfigData.aChConfig[ ].aAIDI.aSpanMin	スパン最小値	範囲外の値です。	Channel00 Span Min
DAQMX_ITEM_CHSPANMAX	1088	aChConfigData.aChConfig[ ].aAIDI.aSpanMax	スパン最大値	範囲外の値です。	Channel00 Span Max
DAQMX_ITEM_CHSCALEMIN	1152	aChConfigData.aChConfig[ ].aAIDI.aScaleMin	スケール最小値	範囲外の値です。	Channel00 Scale Min
DAQMX_ITEM_CHSCALEMAX	1216	aChConfigData.aChConfig[ ].aAIDI.aScaleMax	スケール最大値	範囲外の値です。	Channel00 Scale Max
DAQMX_ITEM_CHREFCHNO	1280	aChConfigData.aChConfig[ ].aAIDI.aRefChNo	基準チャネル番号	範囲外の値です。参照先のチャネルが入力チャネルではありません。	Channel00 Reference Channel No
DAQMX_ITEM_CHFILTER	1344	aChConfigData.aChConfig[ ].aAI.aFilter	フィルタ係数	範囲外の値です。	Channel00 Filter
DAQMX_ITEM_CHRJCTYPE	1408	aChConfigData.aChConfig[ ].aAI.aRJCType	RJC種類	範囲外の値です。	Channel00 RJC Type
DAQMX_ITEM_CHRJCVOLT	1472	aChConfigData.aChConfig[ ].aAI.aRJCVolt	RJC電圧値	範囲外の値です。	Channel00 RJC Volt
DAQMX_ITEM_CHBURNOUT	1536	aChConfigData.aChConfig[ ].aAI.aBurnout	バーンアウト	範囲外の値です。	Channel00 Burnout
DAQMX_ITEM_CHDEENERGIZE	1600	aChConfigData.aChConfig[ ].aDO.aDeenergize	非励磁	―	Channel00 Deenergize
DAQMX_ITEM_CHREFALARM	1664	aChConfigData.aChConfig[ ].aDO.aRefAlarm[*][ ]	全参照チャネルのアラームレベル「1」または「2」。(*)複数の参照チャネルをまとめて表す。	―	Channel00 Alarm0 Reference
DAQMX_ITEM_CHHOLD	1792	aChConfigData.aChConfig[ ].aDO.aHold	保持	―	Channel00 Hold

ニーモニック	値	位置	内容	エラー	文字列
DAQMX_ITEM_CHOUTPUTTYPE	1856	aOutputData. aOutput[ ]. aType	出力種類	範囲外の値です。レンジ種類と一致しません。	Channel00 Output Type
DAQMX_ITEM_CHIDLECHOICE	1920	aOutputData. aOutput[ ]. aIdleChoice	アイドル時の選択値	範囲外の値です。	Channel00 Idle Choice
DAQMX_ITEM_CHERRCHOICE	1984	aOutputData. aOutput[ ]. aErrorChoice	エラー時の選択値	範囲外の値です。	Channel00 Error Choice
DAQMX_ITEM_CHPRESETVALUE	2048	aOutputData. aOutput[ ]. aPresetValue	選択値が指定値の場合の値	範囲外の値です。	Channel00 Preset Value
DAQMX_ITEM_CHPULSETIME	2112	aOutputData. aOutput[ ]. aPulseTime	パルス周期倍率	範囲外の値です。	Channel00 Pulse Time
DAQMX_ITEM_CHBALANCEVALID	2176	aBalanceData. aBalance[ ]. aValid	有効／無効	ー	Channel00 Balance Valid
DAQMX_ITEM_CHBALANCEVALUE	2240	aBalanceData. aBalance[ ]. aValue	初期バランス値	範囲外の値です。	Channel00 Balance Value
DAQMX_ITEM_CHCHATFILTER	2368	aChConfigData. aChConfig[ ]. aAIDI.aChatFilter	チャタリングフィルタ	ー	Channel00 ChatFilter
DAQMX_ITEM_ALARMTYPE2	2432	aChConfigData. aChConfig[ ]. aChID.aAlarm[ ]. aType	アラームレベル「3」または「4」のアラーム種類	範囲外の値です。	Channel00 Alarm0 Type
DAQMX_ITEM_ALARMON2	2560	aChConfigData. aChConfig[ ]. aChID.aAlarm[ ]. aON	アラームレベル「3」または「4」のアラームOn値	範囲外の値です。	Channel00 Alarm0 On
DAQMX_ITEM_ALARMOFF2	2688	aChConfigData. aChConfig[ ]. aChID.aAlarm[ ]. aOFF	アラームレベル「3」または「4」のアラームOff値	範囲外の値です。	Channel00 Alarm0 Off
DAQMX_ITEM_CHREFALARM2	2816	aChConfigData. aChConfig[ ]. aDO. aRefAlarm[*][ ]	全参照チャネルのアラームレベル「3」または「4」。 (*)複数の参照チャネルをまとめて表す。	ー	Channel00 Alarm0 Reference

スパンとスケールは，最小値と最大値の関係をチェックしますので，検出位置としては，最小値を示す値が設定されます。
文字列「Channel00」の00にはチャネル番号が入ります。文字列「Alarm0」の0にはアラームレベル番号が入ります。

実際の値は，表の値にチャネル番号(インデックス)を加算した値になります。
アラームの場合，さらにアラームレベル番号(インデックス)が下位から7ビット目に付加されます。以下の表の値にチャネル番号(インデックス)を加算した値になります。

アラーム

ニーモニック	アラームレベル	値	内容
DAQMX_ITEM_ALARMTYPE	1	640	チャネル番号「1」，アラームレベル「1」のアラーム種類
	2	704	チャネル番号「1」，アラームレベル「2」のアラーム種類
DAQMX_ITEM_ALARMON	1	768	チャネル番号「1」，アラームレベル「1」のON値
	2	832	チャネル番号「1」，アラームレベル「2」のON値
DAQMX_ITEM_ALARMOFF	1	896	チャネル番号「1」，アラームレベル「1」のOFF値
	2	960	チャネル番号「1」，アラームレベル「2」のOFF値
DAQMX_ITEM_CHREFALARM	1	1664	チャネル番号「1」で，全参照チャネルのアラームレベル「1」 複数の参照チャネルをまとめて表します。
	2	1728	チャネル番号「1」で，全参照チャネルのアラームレベル「2」 複数の参照チャネルをまとめて表します。
DAQMX_ITEM_ALARMTYPE2	3	2432	チャネル番号「1」，アラームレベル「3」のアラーム種類
	4	2496	チャネル番号「1」，アラームレベル「4」のアラーム種類
DAQMX_ITEM_ALARMON2	3	2560	チャネル番号「1」，アラームレベル「3」のON値
	4	2624	チャネル番号「1」，アラームレベル「4」のON値
DAQMX_ITEM_ALARMOFF2	3	2688	チャネル番号「1」，アラームレベル「3」のOFF値
	4	2752	チャネル番号「1」，アラームレベル「4」のOFF値
DAQMX_ITEM_CHREFALARM2	3	2816	チャネル番号「1」で，全参照チャネルのアラームレベル「3」 複数の参照チャネルをまとめて表します。
	4	2880	チャネル番号「1」で，全参照チャネルのアラームレベル「4」 複数の参照チャネルをまとめて表します。

文字列について

項目内容の文字列による表記については，以下のようになります。

- 有効無効値は，「OFF」「ON」で表記されます。
- 文字列は，そのまま表記されます。
- 数値は，十進数で表記されます。ただし，ビットに意味のある場合などは16進数で表記されます。
- 参照チャネルのアラームレベルの表記は，「01，02，…」のように有効なチャネル番号で表されます。
- 選択肢は，各名称文字列で表記されます。定数のニーモニックを識別できる程度の簡略した文字列で表記されます。
- 論理OR演算で合成されている定数の場合，カンマ区切りで表記されます。
- ユニット種類の場合，文字列は「MX100 (0x0000)」と表記されます。括弧内は補足情報です。範囲外は，「0x00000000」と表記されます。
- IPアドレス，MACアドレス，バージョンなどは，ドット区切りで表記されます。

範囲

ニーモニック	内容
DAQMX_ITEM_ALL_START	全項目の先頭番号
DAQMX_ITEM_ALL_END_R1	スタイルバージョン1での全項目の最終番号
DAQMX_ITEM_ALL_END_R2	スタイルバージョン2での全項目の最終番号
DAQMX_ITEM_ALL_END_R3	スタイルバージョン3での全項目の最終番号
DAQMX_ITEM_ALL_END	最新スタイルバージョンでの全項目の最終番号

マスク

ニーモニック	内容
DAQMX_MASK_BYMODULE	モジュール番号マスク
DAQMX_MASK_BYCHANNEL	チャネル番号マスク
DAQMX_MASK_BYFIFO	FIFO番号マスク
DAQMX_MASK_BYALARM	アラームレベル番号マスク

アラームレベル番号マスクは，下位から7ビット目の位置を示すものであることに注意してください。

インデックス

ニーモニック	内容
DAQMX_MAX_INDEX_FIFO	FIFOの最大インデックス値
DAQMX_MAX_INDEX_MODULE	モジュールの最大インデックス値
DAQMX_MAX_INDEX_CHANNEL	チャネルの最大インデックス値

6.4 MX100の型の概要

下記のデータ型が装備されています。

型	説明	ページ
DAQMX	機器記述子	6-27
DAQINT64	64ビットデータの型定義です。 Visual C/Visual C++で使用できます。	6-27
MXINT64	64ビットデータをVisual BasicとVisual C/Visual C++で対応できる共用体です。	6-27
MXDataNo	DAQINT64の別名です。 Visual Basicでは，MXINT64のVisual Basic対応部です。 測定データのデータ番号用です。	6-27
MXUserTime	DAQINT64の別名です。 Visual Basicでは，MXINT64のVisual Basic対応部です。 ユーザーカウント用です。	6-27
MXDateTime	時刻情報の構造体です。	6-27
MXAlarm	アラーム設定情報の構造体です。	6-28
MXDataInfo	測定データの構造体です。	6-28
MXChConfigAIDI	AIおよびDIチャネルに関する情報の構造体です。	6-28
MXChConfigAI	AIチャネルに関する情報の構造体です。	6-29
MXChConfigDO	DOチャネルに関する情報の構造体です。	6-29
MXChID	チャネルに関する構造体です。	6-30
MXChConfig	チャネル設定情報の構造体です。	6-30
MXChConfigData	全チャネル数分のチャネル設定情報の構造体です。	6-31
MXChInfo	チャネル情報データの構造体です。	6-31
MXProductInfo	MX100製品情報の構造体です。	6-32
MXUnitData	ユニット情報の構造体です。	6-32
MXModuleData	モジュール情報の構造体です。	6-33
MXSystemInfo	システム構成データの構造体です。	6-34
MXCFInfo	CF情報の構造体です。	6-34
MXFIFOInfo	FIFOの構造体です。	6-34
MXStatus	ステータスの構造体です。	6-35
MXNetInfo	イーサネット接続情報の構造体です。	6-36
MXBalance	初期バランス有効/無効の構造体です。	6-36
MXBalanceData	初期バランスデータの構造体です。	6-36
MXBalanceResult	初期バランス実行結果の構造体です。	6-36
MXOutput	出力値の構造体です。	6-37
MXOutputData	出力チャネルデータの構造体です。	6-37
MXConfigData	設定データの構造体です。	6-37
MXDO	DO設定の構造体です。	6-38
MXDOData	DOデータの構造体です。	6-38
MXSegment	7セグメントLEDごとの表示パターンの構造体です。	6-38
MXAOPWM	パルス出力の構造体です。	6-38

型	説明	ページ
MXAOPWMData	個数分のAO/PWMデータの構造体です。	6-38
MXTransmit	個数分の伝送状態の構造体です。	6-38

型	説明
コールバック型	関数名に接頭辞「DLL」を付加し，大文字で記述します。 例. openMX関数のコールバック型：DLLOPENMX

コールバック型は，Visual Cを使用するときに，実行可能モジュール(.dll)とリンクするために使用します。

6.5　MX100の型

記述に関する説明

Visual C/Visual C++型，Visual Basic型

Visual C/Visual C++とVisual Basicでの型名を示します。
Visual C/Visual C++で符号なしのものも，Visual Basicでは符号ありになります。
Visual C/Visual C++の型で，配列の個数値は省略しています。

取得/変更

下記の標記で，取得できる項目，ユーザーが設定できる項目などを示しています。

システム

「システム構成データの取得」関数により取得されるデータです。
○：取得できる項目です。
△：関数内で設定される項目です。

基本設定

設定データのうち，基本設定に含まれるデータです。基本設定項目は，「設定データの一括取得」関数により取得できます。
「AI」はアナログ入力チャネル(ユニバーサル入力モジュール，4線式RTD抵抗入力モジュールの直流電圧入力チャネルなど)，「DI」はディジタル入力チャネル(ユニバーサル入力モジュール，4線式RTD抵抗入力モジュールのDIチャネル，ディジタル入力モジュールDIチャネル)，「DO」はディジタル出力(ディジタル出力モジュールのDO)，「AO」はアナログ出力チャネル，「PWM」はPWM出力チャネル，「PI」はパルス入力チャネル，「CI」はCAN Bus入力チャネルです。
○：取得できる項目です。
●：ユーザーが変更可能な項目です。
△：関数内で設定される項目です。

チャネル情報

「チャネル情報データの取得」関数により取得されるデータです。
○：取得できる項目です。

データ取得

「測定データの取得」関数により取得されるデータです。
△：関数内で設定される項目です。

ステータス

「ステータスの取得」関数により取得されるデータです。
○：取得できる項目です。

用語

型の説明には，MX100の機能を表す用語を使用しています。MX100に関する用語については，付録1で説明しています。

型の詳細説明

DAQMX

機器記述子を格納するための型です。
Visual C++/Visual Cでは R3.01より前はint型の別名，R3.01からはvoid*型の別名です。Visual BasicではLong型の別名です。

DAQINT64

64ビットデータの型定義です。
Visual C++/Visual Cで使用できます。Visual Basicではこの型を使用できません。

MXINT64

MXINT64構造体

Visual C/Visual C++型		名称		内容		Visual Basic型
struct	unsigned int	aVB	aLow	Visual Basic対応部	下部	Long
	unsigned int		aHigh	Visual Basic対応部	上部	Long
DAQINT64		aC		Visual C/Visual C++対応部	全体	―

Visual C/Visual C++とVisual Basicとの64ビットデータを対応させるための構造体です。Visual Basic対応部とVisual C/Visual C++対応部は，共用体になっています。Visual Basicでは使用できません。データ種類別に以下のVisual Basic対応部を定義しています。

Visual Basic対応部

型	名称	内容	Visual Basic型
unsigned int	aLow	下部	Long
unsigned int	aHigh	上部	Long

MXDataNo

データ番号用です。
Visual C/Visual C++では，DAQINT64の別名です。
Visual Basicでは，MXINT64のVisual Basic対応部と同じ内容です。

MXUserTime

ユーザカウント用です。
Visual C/Visual C++では，DAQINT64の別名です。
Visual Basicでは，MXINT64のVB対応部と同じ内容です。

MXDateTime

MXDateTime構造体

Visual C/Visual C++型	名称	内容	Visual Basic型
time_t	aTime	1970年01月01日からの秒数です。	Long
int	aMilliSecond	ミリ秒の値です。	Long

時刻情報データの構造体です。
Visual C++：ラッパクラスは，CDAQMXDateTimeです。

MXAlarm

MXAlarm構造体

Visual C/Visual C++型	名称	内容	Visual Basic型
int	aType	アラーム種類	Long
int	aReserve	未使用	Long
int	aON	On値(アラーム発生のしきい値)	Long
int	aOFF	Off値(アラーム停止のしきい値)	Long

アラーム設定情報の構造体です。On値とOff値は，小数点位置を適用して測定レンジに対応した値に変換することが必要です。

MXDataInfo

MXDataInfo構造体

Visual C/Visual C++型	名称	内容	Visual Basic型
int	aValue	データ値	Long
int	aStatus	データステータス値	Long
int []	aAlarm	個数分のアラーム(有無)	(1 To 4) As Long

測定データの構造体です。
Visual C++：ラッパクラスは，CDAQMXDataInfoです。

MXChConfigAIDI

MXChConfigAIDI構造体

Visual C/Visual C++型	名称	内容	Visual Basic型
int	aSpanMin	スパン最小値	Long
int	aSpanMax	スパン最大値	Long
int	aScaleMin	スケール最小値	Long
int	aScaleMax	スケール最大値	Long
int	aRefChNo	基準チャネル番号	Long
int	aChatFilter	チャタリングフィルタ(有無)	Long

取得/変更

名称	内容	基本設定					
		AI	DI	AO	PWM	PI	CI
aSpanMin	スパン最小値	○●	○●	○●	○●	○●	○●
aSpanMax	スパン最大値	○●	○●	○●	○●	○●	○●
aScaleMin	スケール最小値	○●	○●			○●	○●
aScaleMax	スケール最大値	○●	○●			○●	○●
aRefChNo	基準チャネル番号	○●	○●	○●	○●	○●	○●
aChatFilter	チャタリングフィルタ(有無)					○●	

AIチャネル，DIチャネルの設定データの構造体です。

MXChConfigAI

MXChConfigAI構造体

Visual C/Visual C++型	名称	内容	Visual Basic型
int	aFilter	フィルタ時定数	Long
int	aRJCType	RJC種類	Long
int	aRJCVolt	RJC電圧値	Long
int	aBurnout	バーンアウト	Long

取得/変更

名称	内容	基本設定		
		AI	PI	CI
aFilter	フィルタ時定数	○●	○●	○●
aRJCType	RJC種類	○●		
aRJCVolt	RJC電圧値	○●		
aBurnout	バーンアウト	○●		

AIチャネルの設定データの構造体です。

MXChConfigDO

MXChConfigDO構造体

Visual C/Visual C++型	名称	内容	Visual Basic型
int	aDeenergize	非励磁(有無)	Long
int	aHold	保持(有無)	Long
unsigned char [ ] [ ]	aRefAlarm	個数分の参照アラーム(有無)	(1 To 4, 1 To 60) As Byte

* 2次配列の順序はVisual C/Visual C++とVisual Basicでは逆になります。

取得/変更

名称	内容	基本設定
		DO
aDeenergize	非励磁(有無)	○●
aHold	保持(有無)	○●
aRefAlarm	個数分の参照アラーム(有無)	○●

DOチャネルの設定データの構造体です。

MXChID

MXChID構造体

Visual C/Visual C++型	名称	内容	Visual Basic型
int	aChNo	チャネル番号	Long
int	aPoint	小数点位置	Long
int	aValid	チャネルステータス(有無)	Long
int	aKind	チャネル種類	Long
int	aRange	レンジ種類	Long
int	aScaleType	スケール種類	Long
char []	aUnit	単位名	String * DAQMX_MAXUNITLEN
char	align1	未使用	(0 To 1) As Byte
char []	aTag	タグ	String * DAQMX_MAXTAGLEN
(なし)	aNULL	未使用(終端，Visual Basic用)	Byte
char []	aComment	コメント	String * DAQMX_MAXCOMMENTLEN
char	align2	未使用	(0 To 1) As Byte
MXAlarm [ ]	aAlarm	個数分のアラーム	(1 To 4) As MXAlarm

取得/変更

名称	内容	チャネル情報	基本設定 AI	DI	DO	AO	PWM	PI	CI	データ取得
aChNo	チャネル番号	○	△	△	△	△	△	△	△	△
aPoint	小数点位置	○	○●	○●		○●	○●	○●	○●	△
aValid	チャネルステータス(有無)	△	○●	○●	○●	○●	○●	○●	○●	△
aKind	チャネル種類	○	○●	○●	○●	○●	○●	○●	○●	
aRange	レンジ種類	○	○●	○●	○●	○●	○●	○●	○●	
aScaleType	スケール種類	○	○●	○●	○●	○●	○●	○●	○●	
aUnit	単位名	○	○●	○●	○●	○●	○●	○●	○●	
aTag	タグ	○	○●	○●	○●	○●	○●	○●	○●	
aComment	コメント	○	○●	○●	○●	○●	○●	○●	○●	
aAlarm	個数分のアラーム	○	○●	○●				○●	○●	

チャネル識別情報の構造体です。

MXChConfig

MXChConfig構造体

Visual C/Visual C++型	名称	内容	Visual Basic型
MXChID	aChID	チャネル識別情報	MXChID
MXChConfigAIDI	aAIDI	AI，DIの設定情報	MXChConfigAIDI
MXChConfigAI	aAI	AIの設定情報	MXChConfigAI
MXChConfigDO	aDO	DOの設定情報	MXChConfigDO

取得/変更

名称	内容	基本設定						
		AI	DI	DO	AO	PWM	PI	CI
aChID	チャネル識別情報	○●	○●	○●	○●	○●	○●	○●
aAIDI	AI，DIの設定情報	○●	○●		○●	○●	○●	○●
aAI	AIの設定情報	○●					○●	○●
aDO	DOの設定情報			○●				

チャネル設定情報の構造体です。

MXChConfigData

MXChConfigData構造体

Visual C/Visual C++型	名称	内容	Visual Basic型
MXChConfig [ ]	aChConfig	個数分のチャネル設定情報	(なし)

全チャネル設定情報を格納する構造体です。
Visual Basicでは使用できません。チャネル毎の領域を用意することが必要です。

MXChInfo

MXChInfo構造体

Visual C/Visual C++型	名称	内容	Visual Basic型
MXChID	aChID	チャネル識別情報	MXChID
int	aFIFONo	FIFO番号	Long
int	aFIFOIndex	FIFO内チャネル順序番号	Long
double	aOrigMin	基準最小値	Double
double	aOrigMax	基準最大値	Double
double	aDispMin	表示最小値	Double
double	aDispMax	表示最大値	Double
double	aRealMin	実範囲最小値	Double
double	aRealMax	実範囲最大値	Double

取得/変更

名称	内容	チャネル情報	データ取得
aChID	チャネル識別情報	○	△
aFIFONo	FIFO番号	○	△
aFIFOIndex	FIFO内チャネル順序番号	○	△
aOrigMin	基準最小値	○	
aOrigMax	基準最大値	○	
aDispMin	表示最小値	○	
aDispMax	表示最大値	○	
aRealMin	実範囲最小値	○	
aRealMax	実範囲最大値	○	

チャネル情報データの構造体です。
Visual C++：ラッパクラスは，CDAQMXChInfoです。

MXProductInfo

MXProductInfo構造体

Visual C/ Visual C++型	名称	内容	Visual Basic型
int	aOption	オプション（ユニット情報のみ)*	Long
int	aCheck	未使用	Long
char [ ]	aSerial	シリアル番号	String * DAQMX_MAXSERIALLEN
(なし)	aNULL	未使用(シリアル番号の終端，Visual Basic用)	Byte
unsigned char [ ]	aMAC	MACアドレス（ユニット情報のみ)*	(0 To 5) As Byte

*（ユニット情報のみ)：ユニットのみの情報で，モジュールにはありません。

取得/変更

名称	内容	システム	基本設定
aOption	オプション(ユニット情報のみ)	○	○
aSerial	シリアル番号	○	○
aMAC	MACアドレス(ユニット情報のみ)	○	○

MX製品情報の構造体です。

MXUnitData

MXUnitData構造体

Visual C/ Visual C++型	名称	内容	Visual Basic型
int	aType	ユニット種類	Long
int	aStyle	スタイル	Long
int	aNo	ユニット番号	Long
int	aTempUnit	温度単位種類	Long
int	aCFTimeout	タイムアウト値	Long
int	aCFWriteMode	CF書き込み種類	Long
int	aFrequency	電源周波数	Long
int	aReserve	未使用	Long
char [ ]	aPartNo	パート番号	String * DAQMX_MAXPARTNOLEN
(なし)	aNULL	未使用(パート番号の終端，Visual Basic用)	Byte
MXProductInfo	aProduct	プロダクト情報	MXProductInfo

取得/変更

名称	内容	システム	基本設定
aType	ユニット種類	○	○
aStyle	スタイル	○	○
aNo	ユニット番号	○	○●
aTempUnit	温度単位種類		○●
aCFTimeout	タイムアウト値		○●
aCFWriteMode	CF書き込み種類		○●
aFrequency	電源周波数	○	
aPartNo	パート番号	○	
aProduct	プロダクト情報	○	○

ユニット情報の構造体です。

MXModuleData

MXModuleData構造体

Visual C/Visual C++型	名称	内容	Visual Basic型
int	aType	モジュール種類	Long
int	aChNum	チャネル数	Long
int	aInterval	測定周期(msec)	Long
int	aIntegralTime	A/D積分時間種類	Long
int	aStandbyType	起動時モジュール種類	Long
int	aRealType	実際のモジュール種類	Long
int	aStatus	モジュール有効，無効(有無)	Long
int	aVersion	モジュールバージョン	Long
int	aTerminalType	端子種類	Long
int	aFIFONo	FIFO番号	Long
MXProductInfo	aProduct	プロダクト情報	MXProductInfo

取得/変更

名称	内容	システム	基本設定
aType	モジュール種類	○	○●
aChNum	チャネル数	○	○●
iaInterval	測定周期(msec)		○●
iaIntegralTime	A/D積分時間種類		○●
aStandbyType	起動時モジュール種類	○	
aRealType	実際のモジュール種類	○	
aStatus	モジュール有効，無効(有無)	△	
aVersion	モジュールバージョン	○	
aTerminalType	端子種類	○	
aFIFONo	FIFO番号	○	
aProduct	プロダクト情報	○	

モジュール情報の構造体です。

MXSystemInfo

MXSystemInfo構造体

Visual C/ Visual C++型	名称	内容	Visual Basic型
MXUnitData	aUnit	ユニットの情報	MXUnitData
MXModuleData [ ]	aModule	個数分のモジュール情報の配列	(0 To 5) As MXModuleData

取得/変更

名称	内容	システム	基本設定
aUnit	ユニットの情報	○	○
aModule	個数分のモジュール情報の配列	○	○

システム構成データの構造体です。
Visual C++：ラッパクラスは，CDAQMXSysInfoです。

MXCFInfo

MXCFInfo構造体

Visual C/ Visual C++型	名称	内容	Visual Basic型
int	aStatus	CFステータス種類	Long
int	aSize	容量(KB)	Long
int	aRemain	残容量(KB)	Long
int	aReserve	未使用	Long

CF情報の構造体です。

MXFIFOInfo

MXFIFOInfo構造体

Visual C/ Visual C++型	名称	内容	Visual Basic型
int	aNo	FIFO番号	Long
int	aStatus	FIFOステータス値	Long
int	aInterval	測定周期（msec）	Long
int	aReserve	未使用	Long
MXDataNo	aOldNo	最古のデータ番号	MXDataNo
MXDataNo	aNewNo	最新のデータ番号	MXDataNo

FIFO情報の構造体です。

MXStatus

MXStatus構造体

Visual C/ Visual C++型	名称	内容	Visual Basic型
int	aUnitStatus	ユニットステータス値	Long
int	aConfigCnt	設定番号	Long
int	aTimeCnt	時刻番号	Long
int	aFIFONum	FIFOの有効個数	Long
int	aBackup	バックアップの有無	Long
int	aReserve	未使用	Long
MXCFInfo	aCFInfo	CFステータス情報	MXCFInfo
MXFIFOInfo [ ]	aFIFOInfo	個数分のFIFO情報	(0 To 2) As MXFIFOInfo
MXDateTime	aDateTime	ステータス返却時刻	MXDateTime

取得/変更

名称	内容	ステータス	基本設定
aUnitStatus	ユニットステータス値	○	
aConfigCnt	設定番号	○	○
aTimeCnt	時刻番号	○	○
aFIFONum	FIFOの有効個数	○	
aBackup	バックアップの有無	○	
aCFInfo	CFステータス情報	○	
aFIFOInfo	個数分のFIFO情報	○	
aDateTime	ステータス返却時刻	○	

ステータスの構造体です。

Visual C++：ラッパクラスは，CDAQMXStatusです。

MXNetInfo

MXNetInfo構造体

Visual C/Visual C++型	名称	内容	Visual Basic型
unsigned int	aAddress	IPアドレス	Long
unsigned int	aPort	ポート番号	Long
unsigned int	aSubMask	サブネットマスク	Long
unsigned int	aGateway	GATEWAYアドレス	Long
char [ ]	aHost	ホスト名	String * DAQMX_MAXHOSTNAMELEN
char [ ]	align	未使用	(0 To 7) As Byte

取得/変更

名称	内容	基本設定
aAddress	IPアドレス	○
aPort	ポート番号	○
aSubMask	サブネットマスク	○
aGateway	GATEWAYアドレス	○
aHost	ホスト名	○

イーサネット通信設定に関する構造体です。

MXBalance

MXBalance構造体

Visual C/Visual C++型	名称	内容	Visual Basic型
int	aValid	有効/無効	Long
int	aValue	初期バランス値	Long

取得/変更

名称	内容	基本設定
aValid	有効/無効	△
aValue	初期バランス値	○●

MXBalanceData

Visual C/Visual C++型	名称	内容	Visual Basic型
MXBalance [ ]	aBalance	個数分の初期バランスデータ	(1 TO 60) As MXBalance

MXBalanceResult

MXBalanceResult構造体

Visual C/Visual C++型	名称	内容	Visual Basic型
int [ ]	aResult	個数分の初期バランス結果	(1 To 60) As Long

MXOutput

MXOutput構造体

Visual C/Visual C++型	名称	内容	Visual Basic型
int	aType	出力種類	Long
int	aIdleChoice	アイドル時の選択値	Long
int	aErrorChoice	エラー時の選択値	Long
int	aPresetValue	選択値が「指定値」の場合の値	Long
int	aPulseTime	パルス周期倍率	Long
int	aReserve	未使用	Long

取得/変更

名称	内容	基本設定AO	基本設定PWM
aType	出力種類	○●	○●
aIdleChoice	アイドル時の選択値	○●	○●
aErrorChoice	エラー時の選択値	○●	○●
aPresetValue	選択値が「指定値」の場合の値	○●	○●
aPulseTime	パルス周期倍率		○●

MXOutputData

MXOutputData構造体

Visual C/Visual C++型	名称	内容	Visual Basic型
MXOutput [ ]	aOutput	個数分の出力チャネルデータ (1 TO 60)	As MXOutput

MXConfigData

MXConfigData構造体

Visual C/Visual C++型	名称	内容	Visual Basic型
MXSystemInfo	aSystemInfo	システム構成データ	(なし)
MXStatus	aStatusInfo	ステータス	(なし)
MXNetInfo	aNetInfo	ネットワーク情報データ	(なし)
MXChConfigData	aChConfigData	チャネル設定データ	(なし)
MXBalanceData	aBalanceData	初期バランスデータ	(なし)
MXOutputData	aOutputData	出力チャネルデータ	(なし)

設定データの構造体です。
Visual Basicでは使用できません。個別に領域を用意することが必要です。
Visual C++：ラッパクラスは，CDAQMXConfigです。

MXDO

MXDO構造体

Visual C/Visual C++型	名称	内容	Visual Basic型
int	aValid	有効/無効（有無）	Long
int	aONOFF	ON/OFF（有無）	Long

DOチャネルの構造体です。

MXDOData

MXDOData構造体

Visual C/Visual C++型	名称	内容	Visual Basic型
MXDO [ ]	aDO	個数分のDOデータ	(1 To 60) As MXDO

DOデータの構造体です。

MXSegment

MXSegment構造体

Visual C/Visual C++型	名称	内容	Visual Basic型
int []	aPattern	7セグメントLEDごとの表示パターン	(0 To 1) As Long

7セグメントLED表示の構造体です。

MXAOPWM

MXAOPWM構造体

Visual C/Visual C++型	名称	内容	Visual Basic型
int	aValid	有効/無効（有無）	Long
int	aValue	出力データ値	Long

MXAOPWMData

MXAOPWMData構造体

Visual C/Visual C++型	名称	内容	Visual Basic型
MXAOPWM [ ]	aAOPWM	個数分のAO/PWMデータ	(1 To 60) As MXAOPWM

MXTransmit

MXTransmit構造体

Visual C/Visual C++型	名称	内容	Visual Basic型
int [ ]	aTrans	個数分の伝送状態	(1 To 60) As Long

7.1　DARWINのクラス

本APIは，下図のように，MX100/DARWIN共通のクラスとDARWIN専用のクラスで構成されています。MX100/DARWIN共通クラスの詳細については，2.4節をご覧ください。

- ■ CDAQChInfo
  - ・ CDAQDARWINChInfo
- ・ CDAQDARWINSysInfo
- ■ CDAQDataInfo
  - ・ CDAQDARWINDataInfo
- ■ CDAQDateTime
  - ・ CDAQDARWINDateTime
- ■ CDAQHandler
  - ・ CDAQDARWIN

■ ：MX100とDARWINに共通のクラスです。
・ ：DARWIN専用のクラスです。

CDAQChInfoクラス

チャネル情報データを格納する基底クラスです。

CDAQDARWINChInfoクラス

CDAQChInfoクラスの派生クラスです。チャネル情報データをDarwinChInfo構造体に格納します。

CDAQDARWINSysInfoクラス

システム構成データをDarwinSystemnfo構造体に格納するクラスです。

CDAQDataInfoクラス

測定データを格納する基底クラスです。

CDAQDARWINDataInfoクラス

CDAQDataInfoクラスの派生クラスです。測定データを格納します。

CDAQDateTimeクラス

時刻情報を格納する基底クラスです。

CDAQDARWINDateTimeクラス

CDAQDateTimeクラスの派生クラスです。時刻情報を格納します。

CDAQHandlerクラス

機器(MX100/DARWIN)本体と通信を行うハンドラの基底クラスです。

CDAQDARWINクラス

CDAQHandlerクラスの派生クラスです。DARWINシリーズに共通な通信機能を提供します。

Note

データ種類と取得方法

DARWINから取得するデータは，種別ごとにクラス化されています。設定データは，行単位で取得するため，クラス化されていません。

7.2　機能とクラス/関数メンバの対応ーDARWINー

本APIでサポートする機能と，クラスの対応を示します。

Note

本APIでは，DARWINシリーズ機器の共通機能の一部を提供しています。機種別の機能，セットアップモードの設定機能，A/D校正機能は実装されていません。DARWIN通信機能のコマンドを使用して，機能を追加することができます。

表中の「コマンド」とは，DARWIN通信機能のコマンドのことです。コマンドの詳細については，通信インターフェースユーザーズマニュアルをご覧ください。

通信機能

機能	クラスと関数メンバ
DARWINと通信接続	CDAQDARWIN::open
DARWINとの通信を切断	CDAQDARWIN::close
データを行単位で送信 特別にデータ送信を制御する場合に使用します。	CDAQDARWIN::sendLine
データを行単位で受信 特別にデータ受信を制御する場合に使用します。	CDAQDARWIN::receiveLine
バイト単位でのデータ受信 特別にデータ受信を制御する場合に使用します。	CDAQDARWIN::receiveByte
コマンドを送信し，応答を受信 機能コマンドを実装する場合に使用します。	CDAQDARWIN::runCommand
ステータスバイトを取得 ステータスバイト出力コマンドを送信し，応答を受信します。	CDAQDARWIN::getStatusByte
トリガコマンド(ESC T)を送信し，応答を受信 新たにトーカ機能を実装する場合に使用します。	CDAQDARWIN::sendTrigger
通信タイムアウトを設定。	CDAQDARWIN::setTimeOut

Note

通信タイムアウトの設定を推奨しません。**理由**：データ取得時にタイムアウト時間に抵触して予期しない通信切断が発生する場合があります。

制御機能

機能	コマンド	クラスと関数メンバ
設定モード切り替え	DS	CDAQDARWIN::transMode
システム再構築	RS	CDAQDARWIN::initSystem
RAMクリア(運転モード設定パラメータの初期化)	RC	
アラームリセット	AR	
日付時刻設定	SD	CDAQDARWIN::setDateTime
演算のスタート，ストップ	EX	CDAQDARWIN::compute
レポートのスタート，ストップ	DR	CDAQDARWIN::reporting
セットアップモード確定	XE	CDAQDARWIN::establish

設定機能

機能		コマンド	クラスと関数メンバ
レンジ設定	スキップ(未使用)	SR	CDAQDARWIN::setSKIP
	直流電圧入力	SR	CDAQDARWIN::setVOLT
	熱電対入力	SR	CDAQDARWIN::setTC
	測温抵抗体入力	SR	CDAQDARWIN::setRTD
	接点入力(DI)	SR	CDAQDARWIN::setDI
	チャネル間差演算	SR	CDAQDARWIN::setDELTA
	リモートRJC	SR	CDAQDARWIN::setRRJC
	直流電流	SR	CDAQDARWIN::setMA
	ひずみ	SR	CDAQDARWIN::setSTRAIN
	パルス	SR	CDAQDARWIN::setPULSE
	パワーモニタ	SR	CDAQDARWIN::setPOWER
スケーリングの単位を設定		SN	CDAQDARWIN::setScallingUnit
アラームを設定		SA	CDAQDARWIN::setAlarm

データ取得機能

機能	コマンド	クラスと関数メンバ
システム構成データを取得	TS，CF	CDAQDARWIN::getSystemConfig
チャネル情報データの取得を宣言	TS，LF	CDAQDARWIN::talkChInfo
チャネル情報データを取得		CDAQDARWIN::getChInfo
測定データの取得を宣言(ASCIIコード)	TS，FM	CDAQDARWIN::talkDataByASCII
測定データを取得(ASCIIコード)		CDAQDARWIN::getChDataByASCII
測定データの取得を宣言(バイナリコード)	TS，FM	CDAQDARWIN::talkDataByBinary
測定データを取得(バイナリコード)		CDAQDARWIN::getChDataByBinary
設定データの取得を宣言(運転モード)	TS，LF	CDAQDARWIN::talkOperationData
設定データを取得(運転モード)		CDAQDARWIN::getSetDataByLine
設定データの取得を宣言(セットアップモード)	TS，LF	CDAQDARWIN::talkSetupData
設定データを取得(セットアップモード)		CDAQDARWIN::getSetDataByLine
設定データの取得を宣言(A/D校正モード)	TS，LF	CDAQDARWIN::talkCalibrationData
設定データを取得(A/D校正モード)		CDAQDARWIN::getSetDataByLine
レポートステータスの取得	TS，RF	CDAQDARWIN::getReportStatus

ユーティリティ

機能		クラスと関数メンバ
測定値を倍精度浮動小数に変換		CDAQDARWINDataInfo::toDoubleValue
測定値を文字列に変換		CDAQDARWINDataInfo::toStringValue
アラーム	アラーム種類の文字列を取得	CDAQDARWINDataInfo::getAlarmName
	アラーム文字列の最大長を取得	CDAQDARWINDataInfo:: getMaxLenAlarmName
本APIのバージョン番号を取得		CDAQDARWIN::getVersionAPI
本APIのリビジョン番号を取得		CDAQDARWIN::getRevisionAPI
エラーメッセージ文字列を取得		CDAQDARWIN::getErrorMessage
エラーメッセージ文字列の最大長を取得		CDAQDARWIN::getMaxLenErrorMessage

機能コマンドの実装

DARWIN通信機能コマンドを使用して，機能コマンドを実装できます。使用できるDARWIN通信機能コマンドは下記のとおりです。

- DA100データアクイジションユニット用の通信コマンドすべて
- DC100データコレクタ用の通信コマンドすべて
- DR130，DR231，DR232，DR241，DR242ハイブリッドレコーダ用の通信コマンドすべて

7.3　プログラムーDARWIN/Visual C++ー

インクルードファイルのパスを追加

プロジェクトに，インクルードファイル(DAQDARWIN.h)のパスを追加します。追加方法は，ご使用の環境により異なります。

ソースファイルでの宣言

ソースファイルに宣言を記述します。

```
#include "DAQDARWIN.h"
```

Note

共通部のインクルードファイル(DAQHandler.h)は，上記インクルードファイルから参照されているので，宣言を記述する必要はありません。

ライブラリの指定

プロジェクトにライブラリ(DAQDARWIN.lib，DAQHandler.lib)を追加します。追加方法は，ご使用の環境により異なります。
すべてのクラスが使用可能になります。Visual C用の関数郡も使用できます。

測定データの取得

プログラム例1

測定データを取得するプログラムです。

```
////////////////////////////////////////////////////////////
// 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 { //meaasured data
    rc = daqdarwin.getChDataByBinary(datainfo, &flag);
  } while (! (flag & DAQDARWIN_FLAG_ENDDATA));
  //disconenct
  rc = daqdarwin.close();
  return rc;
}
////////////////////////////////////////////////////////////
```

説明

全般

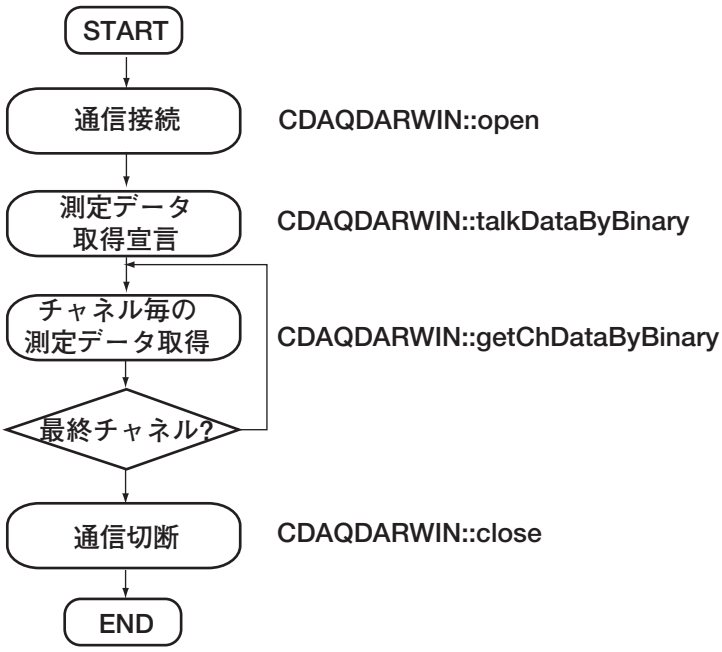
データを取得する場合，最初にトーカを実行し，チャネルまたは行単位でデータ取得を実行します。終了はフラグで判断します。

インクルードファイルの記述

```
#include "DAQDARWIN.h"
```

処理の流れ

下記のフローチャートでは，宣言部分を省略しています。



通信処理

最初に通信接続を行います。通信接続後，各関数メンバが利用可能です。最後に終了処理として，通信切断を行います。

通信接続

```
open("192.168.1.11")
```

DARWINのIPアドレスを指定しています。通信用ポートは，通信用定数の「DARWINの通信ポート番号」を指定したことになります。

Note

クラスの構築時に通信接続をすることも可能です。消滅時には，通信切断を行います。

トーカ

```
talkDataByBinary(0, 1, 0, 2, datetime)
```

サブユニット番号0/チャネル1，2の測定データ取得要求を送信し，時刻情報を取得します(測定データ取得宣言)。

測定データの取得

```
getChDataByBinary(datainfo, &flag)
```

測定データをチャネル単位で取得します。指定されたチャネルまで繰り返します。終了はフラグステータスの「最終データ」により判断します。

通信切断

```
close()
```

通信を切断します。

設定データの取得/設定

プログラム例2

下記の2つを実行するプログラムです。このプログラムではまとめて記述していますが，それぞれ個別に記述して実行できます。

・運転モードの設定データを取得
・チャネルに直流電圧レンジを設定

```
////////////////////////////////////////////////////////////
// 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;
}
////////////////////////////////////////////////////////////
```

説明

トーカ

```
talkOperationData(0, 1, 0, 2)
```

取得する設定データの種類(運転モードの設定データ)と，対象チャネル範囲(サブユニット番号0/チャネル1，2)を指定します。

運転モードの設定データを取得

```
getSetDataByLine(line, BUFSIZ, &len, &flag)
```

トーカ機能による出力を，行単位で取得します。
終了はフラグステータスの「最終データ」により判断します。

チャネルに直流電圧レンジを設定

```
setVOLT(DAQDARWIN_RANGE_VOLT_20MV, 0, 1, 2, 0, 0, 0, 0, 0)
```

サブユニット番号0/チャネル1，2の測定レンジを「20mV」に設定します。スケーリング機能は使用しません。
レンジ種類の指定には，「20mV」定数を使用しています。

機能コマンドの実装

プログラム例3

DARWINを運転モードに切り替えるプログラムです。DARWIN通信機能のDSコマンドを実行しています。

```
////////////////////////////////////////////////////////////
// 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;
}
////////////////////////////////////////////////////////////
```

説明

メッセージの作成

`sprintf(line, "DS%d", DAQDARWIN_MODE_OPE);`

DARWIN通信機能のDS0(運転モードに切り替え)コマンドメッセージを配列lineに格納します。

運転モードを指定するために，「運転モード」定数を使用しています。

メッセージの送信

`runCommand(line)`

コマンドメッセージを送信し，応答を受信します。本関数メンバが，メッセージにターミネータを付けて送信します。

トーカ機能の実装

プログラム例4

システム構成データを取得するプログラムです。DARWIN通信機能のTSコマンドとCFコマンドを実行しています。

```
////////////////////////////////////////////////////////////
// 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("192.168.1.11");
  //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] != 'E'));
  //disconnect
  rc = daqdarwin.close();
  return rc;
}
////////////////////////////////////////////////////////////
```

説明

トーカ

```
sprintf(line, "TS%d", DAQDARWIN_TALK_SYSINFODATA);
```

DARWIN通信機能のTS5(システム構成データの取得を指定)コマンドメッセージをlineに格納します。
システム構成データの出力指定には，「システム構成データの出力」定数を使用しています。

```
runCommand(line)
```

メッセージを送信し，応答を受信します。本関数メンバが，メッセージにターミネータを付けて送信します。

```
sendTrigger()
```

トリガ(機器トリガ)を送信します。

システム構成データの出力フォーマット指定

```
sendLine("CF0")
```

通信機能コマンドCF0(システム構築されたモジュール情報を指定)を送信します。本関数メンバが，メッセージにターミネータを付けて送信します。

データ取得

```
receiveLine(line, BUFSIZ, &len)
```

システム構成データを行単位で取得します。エンドマーク(E)が返されたときに終了します。

Note

receiveLine関数メンバは，単純にデータを受信する関数なので，ユーザーが終了の判断を記述する必要があります。

エラー処理

- ほとんどの関数メンバは，戻り値として，関数の処理結果の状態をエラー番号で返します。
- エラー番号に対応するエラーメッセージ文字列を得ることができる関数メンバ(getErrorMessage)があります。また，エラーメッセージ文字列の最大長を得る関数メンバ(getMaxLenErrorMessage)もあります。

7.4 DARWIN用クラス詳細

クラスは，クラス名のアルファベット順で並んでいます。

CDAQDARWINクラス

■ CDAQHandler
・ CDAQDARWIN

本クラスはCDAQHandlerクラスの派生クラスです。DARWINシリーズに共通な通信，データ取得，レンジ設定などの機能を提供します。

パブリックメンバ

構築・消滅

CDAQDARWIN	オブジェクトを構築します。
~CDAQDARWIN	オブジェクトを消滅します。

通信機能

runCommand	メッセージを送信し，応答を受信します。
getStatusByte	ステータスバイトを取得します。
sendTrigger	トリガを送信します。
receiveByte	バイナリデータをバイト単位で受信します。

制御機能

setDateTime	日付時刻を設定します。
transMode	操作モードの切り替えをします。
initSystem	システムを初期化します。
establish	セットアップモード確定を実行します。
compute	演算のスタート，ストップを実行します。
reporting	レポートのスタート，ストップを実行します。

データ取得機能

getSystemConfig	システム構成データを取得します。
talkChInfo	チャネル情報データを取得する宣言をします。
getChInfo	チャネル情報データを取得します。
talkDataByASCII	測定データをASCIIフォーマットで取得する宣言をします。
getChDataByASCII	測定データをASCIIフォーマットで取得します。
talkDataByBinary	測定データをバイナリフォーマットで取得する宣言をします。

getChDataByBinary	測定データをバイナリフォーマットで取得します。
talkOperationData	運転モードの設定データを取得する宣言をします。
talkSetupData	セットアップモードの設定データを取得する宣言をします。
talkCalibrationData	A/D校正モードの設定データを取得する宣言をします。
getSetDataByLine	設定データを取得します。
getReportStatus	レポートステータスを取得します。

設定機能

setSKIP	スキップ(未使用)を設定します。
setVOLT	直流電圧レンジを設定します。
setTC	熱電対レンジを設定します。
setRTD	測温抵抗体レンジを設定します。
setDI	接点入力(DI)レンジを設定します。
setDELTA	チャネル間差演算を設定します。
setRRJC	リモートRJCを設定します。
setScallingUnit	スケーリングの単位を設定します。
setAlarm	アラームを設定します。
setMA	直流電流レンジを設定します。
setSTRAIN	ひずみ入力を設定します。
setPULSE	パルス入力を設定します。
setPOWER	パワーモニタを設定します。

●オーバライドしたメンバ

通信機能

open	通信接続をします。

データ収集機能

getData	測定データを取得します。
getChannel	チャネル情報データを取得します。

ユーティリティ

isObject	オブジェクトをチェックします。

●継承するメンバ

CDAQHandler参照

```
close getErrorMessage getMaxLenErrorMessage
getRevisionAPIgetVersionAPIreceiveLine sendLine setTimeOut
```

プロテクトメンバ

通信機能

startTalker	トーカとしての機能を開始します。

ユーティリティ

checkAck	応答をチェックします。
getVersionDLL	本DLLのバージョンを取得します。
getRevisionDLL	本DLLのリビジョンを取得します。

●継承するメンバ

CDAQHandler参照

```
m_comm m_nRemainSize receive receiveRemain send
```

プライベートメンバ

なし

関数メンバ(アルファベット順)

CDAQDARWIN::CDAQDARWIN

構文

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

引数

strAddress	IPアドレスを文字列で指定します。
uiPort	ポート番号を指定します。
errCode	エラー番号の返却先を指定します。

説明

オブジェクトを構築，消滅します。

構築時，データメンバを初期化します。引数が指定されている場合，構築時に通信接続(open)を行います。返却先が指定されていれば，通信接続時のエラー番号を返します。

消滅時，データメンバの領域を開放します。通信記述子が存在する場合，通信切断(close)を行います。エラー番号は返却されません。

参照

```
CDAQHandler::CDAQHandler
```

CDAQDARWIN::checkAck

構文

```
int checkAck(const char * strAck, int lenAck);
```

引数

strAck　　応答を文字列で指定します。
lenAck　　応答のバイト数を指定します。

説明

引数で指定された文字列を応答としてチェックし，その結果を返します。

戻り値

エラー番号を返します。
エラー：
Success　　応答が，「処理が正常に行われた」を表しています。
Commands are not processed succesfully
　　応答が，「処理が正常に行われなかった」を表しています。
Not acknowledge　　応答ではありません。

CDAQDARWIN::compute

構文

```
int compute(int iCompute);
```

引数

iCompute　　演算処理を指定します。

説明

指定された演算処理を実行します。
「通信インターフェイス」のEXコマンドを実行します。
コマンドを生成して送信し，応答を受信します。
オプションの演算機能付き，またはパルスモジュール装着時の場合にだけ有効です。

戻り値

エラー番号を返します。

参照

`runCommand`

CDAQDARWIN::establish

構文

```
int establish(int iSetup = DAQDARWIN_SETUP_ABORT);
```

引数

iSetup　　セットアップ確定を指定します。

説明

指定されたセットアップ確定を実行します。
「通信インターフェイス」のXEコマンドを実行します。
コマンドを生成して送信し，応答を受信します。
セットアップモードで有効です。

戻り値

エラー番号を返します。

参照

runCommand

CDAQDARWIN::getChannel

構文

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

引数

chType　　チャネルタイプを指定します。
chNo　　チャネル番号を指定します。
cChInfo　　チャネル情報データの返却先を指定します。

説明

チャネル単位で，チャネル情報データを取得するための関数です。
指定されたチャネルのチャネル情報データを取得します。

戻り値

エラー番号を返します。

参照

getChInfo talkChInfo

CDAQDARWIN::getChDataByASCII

構文

```
int getChDataByASCII(CDAQDARWINDataInfo & cDARWINDataInfo, int
* pFlag);
```

引数

cDARWINDataInfo　測定データの返却先を指定します。
pFlag　フラグの返却先を指定します。

説明

ttalkDataByASCIIで宣言したトーカ機能による測定データの出力をチャネル単位で取得します。チャネル単位で受信した情報を解析して，返却先に格納します。
最終データを取得した場合，フラグにフラグステータスがセットされます。また，エラーで終了した場合もセットします。
データ取得を終了するまでは，他関数で通信を行わないでください。本関数でデータ取得中は，他の関数が正しく動作できません。

戻り値

エラー番号を返します。

参照

`checkAck receiveLine  CDAQDARWINDataInfo::setLine`

CDAQDARWIN::getChDataByBinary

構文

```
int getChDataByBinary(CDAQDARWINDataInfo & cDARWINDataInfo,
int * pFlag);
```

引数

cDARWINDataInfo　測定データの返却先を指定します。
pFlag　フラグの返却先を指定します。

説明

talkDataByBinaryで宣言したトーカ機能による測定データの出力をチャネル単位で取得します。
測定チャネルの場合6バイト，演算チャネルの場合8バイトを受信します。
チャネル単位で受信した情報を解析して，返却先に格納します。
データメンバの残りサイズを更新します。
最終データを取得した場合，フラグにフラグステータスがセットされます。また，エラーで終了した場合もセットします。
データ取得を終了するまでは，他関数で通信を行わないでください。本関数でデータ取得中は，他の関数が正しく動作できません。

戻り値

エラー番号を返します。
エラー：
Not data　出力バイト数が必要なサイズを満たしていません。

参照

`receive CDAQDARWINDataInfo::setByte`

CDAQDARWIN::getChInfo

構文

```
int getChInfo(CDAQDARWINChInfo & cDARWINChInfo, int * pFlag);
```

引数

cDARWINChInfo	チャネル情報データの返却先を指定します。
pFlag	フラグの返却先を指定します。

説明

talkChInfoで宣言したトーカ機能によるチャネル情報データの出力をチャネル単位で取得します。チャネル単位で受信した情報を解析して，返却先に格納します。
最終データを取得した場合，フラグにフラグステータスがセットされます。また，エラーで終了した場合もセットします。
データ取得を終了するまでは，他関数で通信を行わないでください。本関数でデータ取得中は，他関数が正しく動作できません。

戻り値

エラー番号を返します。

参照

```
checkAck receiveLine
CDAQDARWINChInfo::setLine
```

CDAQDARWIN::getData

構文

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

引数

chType	チャネルタイプを指定します。
chNo	チャネル番号を指定します。
cDateTime	時刻情報データの返却先を指定します。
cDataInfo	測定データの返却先を指定します。

t説明

チャネル単位で，瞬時値を取得するための関数です。
指定されたチャネルの測定データを，バイナリコードで取得します。

戻り値

エラー番号を返します。

参照

```
getChDataByBinary talkDataByBinary
```

CDAQDARWIN::getReportStatus

構文

```
int getReportStatus(int * pReportStatus);
```

引数

pReportStatus レポートステータスの返却先を指定します。

説明

レポートステータスを取得します。
トーカ機能としてのデータ取得の宣言とデータ出力の一連の動作を実行します。
レポートステータスを返却先に格納します。
「通信インターフェイス」のTSとRFコマンドを実行します。
コマンドを生成して送信し，データを受信します。

戻り値

エラー番号を返します。

参照

```
receive send startTalker
```

CDAQDARWIN::getRevisionDLL

構文

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

説明

本DLLのリビジョン番号を取得します。

戻り値

本DLLのリビジョン番号を返します。

CDAQDARWIN::getSetDataByLine

構文

```
int getSetDataByLine(char * strLine, int maxLine, int *
lenLine, int * pFlag);
```

引数

strLine	行単位の受信文字列を格納する領域を指定します。
maxLine	行単位の受信文字列を格納する領域のバイト数を指定します。
lenLine	実際に受信した文字列のバイト数の返却先を指定します。
pFlag	フラグの返却先を指定します。

説明

talkOperationData，talkSetupData，talkCalibrationDataで宣言したトーカ機能による出力を行単位で取得します。改行を除いた受信文字列を格納します。
返却先が指定されていれば，最終データを取得した場合，フラグにフラグステータスがセットされます。また，エラーで終了した場合もセットします。
データ取得を終了するまでは，他関数で通信を行わないでください。本関数でデータ取得中は，他の関数が正しく動作できません。

戻り値

エラー番号を返します。

参照

`receiveLine`

CDAQDARWIN::getStatusByte

構文

```
virtual int getStatusByte(int * pStatusByte);
```

引数

pStatusByte	ステータスバイトの返却先を指定します。

説明

ステータスバイトを取得します。
返却先が指定されていれば，ステータスバイトを整数値で返却先に格納します。
ステータスバイト出力コマンド(ESCS)を送信し，出力を受信します。

戻り値

エラー番号を返します。

参照

`checkAck receiveLine send`

CDAQDARWIN::getSystemConfig

構文

```
int getSystemConfig(CDAQDARWINSysInfo & cDARWINSysInfo);
```

引数

cDARWINSysInfo　　システム構成データの返却先を指定します。

説明

システム構成データを取得します。トーカ機能としてのデータ取得宣言と，データ出力の一連の動作を実行します。
測定周期とシステム構成データを返却先に格納します。
「通信インターフェイス」のTSとCFコマンドを実行します。
返却先を初期化してから，コマンドを生成して送信し，データを受信します。

戻り値

エラー番号を返します。

参照

```
checkAck receiveLine send startTalker
CDAQDARWINSysInfo::initialize CDAQDARWINSysInfo::setLine
```

CDAQDARWIN::getVersionDLL

構文

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

説明

本DLLのバージョン番号を取得します。

戻り値

本DLLのバージョン番号を返します。

CDAQDARWIN::initSystem

構文

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

引数

iCtrl　　システム制御種類を指定します。

説明

指定されたシステム制御種類の動作を実行します。
「通信インターフェイス」のRS，RC，ARコマンドのいずれかを実行します。
コマンドを生成して送信し，応答を受信します。

戻り値

エラー番号を返します。
エラー：
Not support　　指定された値が範囲外です。

参照

```
runCommand
```

CDAQDARWIN::isObject

構文

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

引数

classname　　クラス名を文字列で指定します。

説明

指定されたクラス名を継承しているかをチェックします。
省略された場合，本クラスであるかをチェックします。
本クラスを継承したクラスが，自分自身のクラスをチェックするためには，オーバーライドする必要があります。
クラスを継承している場合，「有効値」(真)を返します。それ以外は，「無効値」(偽)を返します。
本クラスと異なる場合，親クラスをチェックします。

戻り値

有効無効値を返します。

参照

CDAQHandler::isObject

CDAQDARWIN::open

構文

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

引数

strAddress　　IPアドレスを文字列で指定します。
uiPort　　ポート番号を指定します。

説明

引数で指定されたIPアドレスとポート番号の機器と通信接続をします。
ポート番号は省略可能で，省略時は「DARWINの通信ポート番号」になります。

戻り値

エラー番号を返します。

参照

CDAQHandler::open

CDAQDARWIN::receiveByte

構文

```
virtual int receiveByte(unsigned char * byteData, int maxData
= 1, int * lenData = NULL);
```

引数

byteData	受信データを格納する領域をバイト配列で指定します。
maxData	受信データのバイト数を指定します。
lenData	実際に受信したデータのバイト数の返却先を指定します。

説明

引数で指定された領域に， バイト数分になるまで受信データを格納します。
返却先が指定されていれば， 実際に受信したデータのバイト数を返します。

戻り値

エラー番号を返します。

参照

`receive`

CDAQDARWIN::reporting

構文

```
int reporting(int iReportRun);
```

引数

iReportRun	レポート実行種類を指定します。

説明

指定されたレポート実行種類を実行します。
「通信インターフェイス」のDRコマンドを実行します。
コマンドを生成して送信し， 応答を受信します。
オプションの演算機能付き，またはパルスモジュール装着時の場合にだけ有効です。

戻り値

エラー番号を返します。

参照

`runCommand`

CDAQDARWIN::runCommand

構文

```
virtual int runCommand(const char * strCmd)
```

引数

strCmd　　送信するコマンドメッセージを指定します。

説明

指定されたコマンドメッセージ送信し，応答を受信します。
送信時，本関数がコマンドメッセージにターミネータを付加するので， 指定するコマンドメッセージには，ターミネータを含まないでください。
複数コマンドの同時送信，ターミネータを含むコマンドメッセージには対応していません。
トーカ機能のデータ出力要求コマンドのように， 応答を返信しないコマンドには対応していません。
文字列はNULL文字を終端とします。

戻り値

エラー番号を返します。

参照

`checkAck receiveLine sendLine`

CDAQDARWIN::sendTrigger

構文

```
virtual int sendTrigger(void);
```

説明

トリガコマンド(ESCT)を送信し，応答を受信します。

戻り値

エラー番号を返します。

参照

`runCommand`

CDAQDARWIN::setAlarm

構文

```
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);
```

引数

levelNo	アラームレベルを指定します。
chType	チャネルタイプを指定します。
startChNo	開始チャネル番号を指定します。
endChNo	終了チャネル番号を指定します。
iAlarmType	アラーム種類をアラーム種類値で指定します。
value	アラーム値を指定します。
relayType	リレータイプを指定します。
relayNo	リレー番号を指定します。

説明

指定されたチャネル範囲(チャネルタイプ，開始チャネル番号，終了チャネル番号)のチャネルに，指定されたアラーム(アラームレベル，アラーム種類)とアラーム値を設定します。
リレー番号が0以下の場合，リレーは，OFFになります。
「通信インターフェイス」のSAコマンドを実行します。
コマンドを生成して送信し，応答を受信します。

戻り値

エラー番号を返します。

参照

```
runCommand
CDAQDARWINChInfo::toChRange
CDAQDARWINDataInfo::getAlarmName
CDAQDARWINSysInfo::toRelayName
```

CDAQDARWIN::setDateTime

構文

```
int setDateTime(CDAQDARWINDateTime * pcDARWINDateTime = NULL);
```

引数

pcDARWINDateTime 時刻情報データを指定します。

説明

機器本体に時刻情報データを設定します。
指定がNULLの場合，パソコンの現在の日付時刻を設定します。
「通信インターフェイス」のSDコマンドを実行します。
コマンドを生成して送信し，応答を受信します。

戻り値

エラー番号を返します。

参照

```
runCommand
CDAQDARWINDateTime::setNow CDAQDARWINDateTime::toString
```

CDAQDARWIN::setDELTA

構文

```
int setDELTA(int refChNo, int chType, int startChNo, int
endChNo = 0, int spanMin = 0, int spanMax = 0);
```

引数

refChNo	基準チャネルのチャネル番号を指定します。
chType	測定チャネルのチャネルタイプを指定します。
startChNo	開始チャネル番号を指定します。
endChNo	終了チャネル番号を指定します。
spanMin	スパンのレフト値を指定します。
spanMax	スパンのライト値を指定します。

説明

指定されたチャネル範囲(チャネルタイプ，開始チャネル番号，終了チャネル番号)の測定チャネルに，指定された基準チャネルとの差演算を設定します。
「通信インターフェイス」のSRコマンドを実行します。
コマンドを生成して送信し，応答を受信します。
レフト値とライト値が等しい場合，スパンは省略されたものとみなします。

戻り値

エラー番号を返します。

参照

```
runCommand
CDAQDARWINChInfo::toChRange
```

CDAQDARWIN::setDI

構文

```
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);
```

引数

iRangeDI	接点入力(DI)レンジを指定します。
chType	測定チャネルのチャネルタイプを指定します。
startChNo	開始チャネル番号を指定します。
endChNo	終了チャネル番号を指定します。
spanMin	スパンのレフト値を指定します。
spanMax	スパンのライト値を指定します。
scaleMin	スケールのレフト値を指定します。
scaleMax	スケールのライト値を指定します。
scalePoint	スケール時の小数点位置を指定します。

説明

指定されたチャネル範囲(チャネルタイプ，開始チャネル番号，終了チャネル番号)の測定チャネルに，指定された接点入力(DI)レンジを設定します。
「通信インターフェイス」のSRコマンドを実行します。
コマンドを生成して送信し，応答を受信します。
レフト値とライト値が等しい場合，スパン，スケールは省略されたものとみなします。

戻り値

エラー番号を返します。

参照

`runCommand`
`CDAQDARWINChInfo::toChRange`

CDAQDARWIN::setMA

構文

```
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);
```

引数

iRangeMA	直流電流レンジを指定します。
chType	測定チャネルのチャネルタイプを指定します。
startChNo	開始チャネル番号を指定します。
endChNo	終了チャネル番号を指定します。
spanMin	スパンのレフト値を指定します。
spanMax	スパンのライト値を指定します。
scaleMin	スケールのレフト値を指定します。
scaleMax	スケールのライト値を指定します。
scalePoint	スケール時の小数点位置を指定します。

説明

指定されたチャネル範囲(チャネルタイプ，開始チャネル番号，終了チャネル番号)の測定チャネルに，指定された直流電流レンジを設定します。
「通信インターフェイス」のSRコマンドを実行します。
コマンドを生成して送信し，応答を受信します。
レフト値とライト値が等しい場合，スパン，スケールは省略されたものとみなします。

戻り値

エラー番号を返します。

参照

`runCommand CDAQDARWINChInfo::toChRange`

CDAQDARWIN::setPOWER

構文

```
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);
```

引数

iRangeVOLT	パワーモニタレンジを指定します。
chType	測定チャネルのチャネルタイプを指定します。
chNo	チャネル番号を指定します。
iItem	パワー測定項目を指定します。
iWire	パワー接続方法を指定します。
spanMin	スパンのレフト値を指定します。
spanMax	スパンのライト値を指定します。
scaleMin	スケールのレフト値を指定します。
scaleMax	スケールのライト値を指定します。
scalePoint	スケール時の小数点位置を指定します。

説明

指定されたチャネル(チャネルタイプ，チャネル番号)の測定チャネルに， 指定されたパワーモニタレンジを設定します。
「通信インターフェイス」のSRコマンドを実行します。
コマンドを生成して送信し，応答を受信します。
レフト値とライト値が等しい場合，スパン，スケールは省略されたものとみなします。

戻り値

エラー番号を返します。

参照

`runCommand CDAQDARWINChInfo::toChRange`

CDAQDARWIN::setPULSE

構文

```
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);
```

引数

iRangePULSE	パルスレンジを指定します。
chType	測定チャネルのチャネルタイプを指定します。
startChNo	開始チャネル番号を指定します。
endChNo	終了チャネル番号を指定します。
spanMin	スパンのレフト値を指定します。
spanMax	スパンのライト値を指定します。
scaleMin	スケールのレフト値を指定します。
scaleMax	スケールのライト値を指定します。
scalePoint	スケール時の小数点位置を指定します。
bFilter	フィルタのON/OFFを有効無効値で指定します。

説明

指定されたチャネル範囲（チャネルタイプ，開始チャネル番号，終了チャネル番号）の測定チャネルに， 指定されたパルスレンジを設定します。
「通信インターフェイス」のSRコマンドを実行します。
コマンドを生成して送信し， 応答を受信します。
レフト値とライト値が等しい場合，スパン，スケールは省略されたものとみなします。

戻り値

エラー番号を返します。

参照

`runCommand CDAQDARWINChInfo::toChRange`

CDAQDARWIN::setRRJC

構文

```
int setRRJC(int refChNo, int chType, int startChNo, int
endChNo = 0, int spanMin = 0, int spanMax = 0);
```

引数

refChNo	基準チャネルのチャネル番号を指定します。
chType	測定チャネルのチャネルタイプを指定します。
startChNo	開始チャネル番号を指定します。
endChNo	終了チャネル番号を指定します。
spanMin	スパンのレフト値を指定します。
spanMax	スパンのライト値を指定します。

説明

指定されたチャネル範囲(チャネルタイプ，開始チャネル番号，終了チャネル番号)の測定チャネルに，指定された基準チャネルとのリモートRJCを設定します。
「通信インターフェイス」のSRコマンドを実行します。
コマンドを生成して送信し，応答を受信します。
レフト値とライト値が等しい場合，スパンは省略されたものとみなします。

戻り値

エラー番号を返します。

参照

`runCommand`
`CDAQDARWINChInfo::toChRange`

CDAQDARWIN::setRTD

構文

```
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);
```

引数

iRangeRTD	測温抵抗体レンジを指定します。
chType	測定チャネルのチャネルタイプを指定します。
startChNo	開始チャネル番号を指定します。
endChNo	終了チャネル番号を指定します。
spanMin	スパンのレフト値を指定します。
spanMax	スパンのライト値を指定します。
scaleMin	スケールのレフト値を指定します。
scaleMax	スケールのライト値を指定します。
scalePoint	スケール時の小数点位置を指定します。

説明

指定されたチャネル範囲(チャネルタイプ，開始チャネル番号，終了チャネル番号)の測定チャネルに，指定された測温抵抗体レンジを設定します。
「通信インターフェイス」のSRコマンドを実行します。
コマンドを生成して送信し，応答を受信します。
レフト値とライト値が等しい場合，スパン，スケールは省略されたものとみなします。

戻り値

エラー番号を返します。

参照

`runCommand`
`CDAQDARWINChInfo::toChRange`

CDAQDARWIN::setScallingUnit

構文

```
int setScallingUnit(const char * strUnit, int chType, int
startChNo, int endChNo = 0);
```

引数

strUnit	単位を文字列で指定します。
chType	チャネルタイプを指定します。
startChNo	開始チャネル番号を指定します。
endChNo	終了チャネル番号を指定します。

説明

指定されたチャネル範囲(チャネルタイプ，開始チャネル番号，終了チャネル番号)のチャネルに，指定された単位を設定します。
「通信インターフェイス」のSNコマンドを実行します。
コマンドを生成して送信し，応答を受信します。

戻り値

エラー番号を返します。
エラー：

Not support	文字列が範囲外です。指定がないか，文字列長が最大値を超えています。

参照

runCommand
CDAQDARWINChInfo::toChRange

CDAQDARWIN::setSKIP

構文

```
int setSKIP(int chType, int startChNo, int endChNo = 0);
```

引数

chType	測定チャネルのチャネルタイプを指定します。
startChNo	開始チャネル番号を指定します。
endChNo	終了チャネル番号を指定します。

説明

指定されたチャネル範囲(チャネルタイプ，開始チャネル番号，終了チャネル番号)の測定チャネルをスキップ(未使用)に設定します。
「通信インターフェイス」のSRコマンドを実行します。
コマンドを生成して送信し，応答を受信します。

戻り値

エラー番号を返します。

参照

runCommand
CDAQDARWINChInfo::toChRange

CDAQDARWIN::setSTRAIN

構文

```
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);
```

引数

iRangeSTRAIN	ひずみ入力レンジを指定します。
chType	測定チャネルのチャネルタイプを指定します。
startChNo	開始チャネル番号を指定します。
endChNo	終了チャネル番号を指定します。
spanMin	スパンのレフト値を指定します。
spanMax	スパンのライト値を指定します。
scaleMin	スケールのレフト値を指定します。
scaleMax	スケールのライト値を指定します。
scalePoint	スケール時の小数点位置を指定します。

説明

指定されたチャネル範囲(チャネルタイプ，開始チャネル番号，終了チャネル番号)の測定チャネルに，指定されたひずみ入力レンジを設定します。
「通信インターフェイス」のSRコマンドを実行します。
コマンドを生成して送信し，応答を受信します。
レフト値とライト値が等しい場合，スパン，スケールは省略されたものとみなします。

戻り値

エラー番号を返します。

参照

`runCommand CDAQDARWINChInfo::toChRange`

CDAQDARWIN::setTC

構文

```
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);
```

引数

iRangeTC	熱電対レンジを指定します。
chType	測定チャネルのチャネルタイプを指定します。
startChNo	開始チャネル番号を指定します。
endChNo	終了チャネル番号を指定します。
spanMin	スパンのレフト値を指定します。
spanMax	スパンのライト値を指定します。
scaleMin	スケールのレフト値を指定します。
scaleMax	スケールのライト値を指定します。
scalePoint	スケール時の小数点位置を指定します。

説明

指定されたチャネル範囲(チャネルタイプ，開始チャネル番号，終了チャネル番号)の測定チャネルに，指定された熱電対レンジを設定します。
「通信インターフェイス」のSRコマンドを実行します。
コマンドを生成して送信し，応答を受信します。
レフト値とライト値が等しい場合，スパン，スケールは省略されたものとみなします。

戻り値

エラー番号を返します。

参照

`runCommand`
`CDAQDARWINChInfo::toChRange`

CDAQDARWIN::setVOLT

構文

```
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);
```

引数

iRangeVOLT	直流電圧レンジを指定します。
chType	測定チャネルのチャネルタイプを指定します。
startChNo	開始チャネル番号を指定します。
endChNo	終了チャネル番号を指定します。
spanMin	スパンのレフト値を指定します。
spanMax	スパンのライト値を指定します。
scaleMin	スケールのレフト値を指定します。
scaleMax	スケールのライト値を指定します。
scalePoint	スケール時の小数点位置を指定します。

説明

指定されたチャネル範囲(チャネルタイプ，開始チャネル番号，終了チャネル番号)の測定チャネルに，指定された直流電圧レンジを設定します。
「通信インターフェイス」のSRコマンドを実行します。
コマンドを生成して送信し，応答を受信します。
レフト値とライト値が等しい場合，スパン，スケールは省略されたものとみなします。

戻り値

エラー番号を返します。

参照

`runCommand CDAQDARWINChInfo::toChRange`

CDAQDARWIN::startTalker

構文

```
virtual int startTalker(int iTalk);
```

引数

iTalk	トーカ機能種類を指定します。

説明

トーカ機能を使用する場合の開始処理を実行します。
「通信インターフェイス」のTSコマンドとトリガを実行します。
本関数の実行後，指定したトーカ機能に対応するデータ取得を行ってください。

戻り値

エラー番号を返します。

参照

`runCommand sendTrigger`

CDAQDARWIN::talkCalibrationData

構文

```
int talkCalibrationData(int startChType, int startChNo, int
endChType, int endChNo);
```

引数

startChType	開始チャネルタイプを指定します。
startChNo	開始チャネル番号を指定します。
endChType	終了チャネルタイプを指定します。
endChNo	終了チャネル番号を指定します。

説明

開始チャネル(開始チャネルタイプ，開始チャネル番号)から終了チャネル(終了チャネルタイプ，終了チャネル番号)までのセットアップモードの設定データを取得する宣言を実行します。
操作モードをA/D校正モードに切り替えておく必要があります。
「通信インターフェイス」のTSとLFコマンドを実行します。
本関数の実行後，行単位のデータ取得には，getSetDataByLineを使用します。

戻り値

エラー番号を返します。

参照

```
send startTalker CDAQDARWINChInfo::toChName
```

CDAQDARWIN::talkChInfo

構文

```
int talkChInfo(int startChType, int startChNo, int endChType,
int endChNo);
```

引数

startChType	開始チャネルタイプを指定します。
startChNo	開始チャネル番号を指定します。
endChType	終了チャネルタイプを指定します。
endChNo	終了チャネル番号を指定します。

説明

開始チャネル(開始チャネルタイプ，開始チャネル番号)から終了チャネル(終了チャネルタイプ，終了チャネル番号)までのチャネル情報データを取得する宣言を実行します。
「通信インターフェイス」のTSとLFコマンドを実行します。
コマンドを生成して送信します。
本関数の実行後，チャネル毎のデータ取得には，getChInfoを使用します。

戻り値

エラー番号を返します。

参照

```
send startTalker CDAQDARWINChInfo::toChName
```

CDAQDARWIN::talkDataByASCII

構文

```
int talkDataByASCII(int startChType, int startChNo, int
endChType, int endChNo, CDAQDARWINDateTime & cDARWINDateTime);
```

引数

startChType	開始チャネルタイプを指定します。
startChNo	開始チャネル番号を指定します。
endChType	終了チャネルタイプを指定します。
endChNo	終了チャネル番号を指定します。
cDARWINDateTime	時刻情報データの返却先を指定します。

説明

開始チャネル(開始チャネルタイプ，開始チャネル番号)から終了チャネル(終了チャネルタイプ，終了チャネル番号)までの測定データをASCIIフォーマットで取得する宣言を実行します。

測定データの時刻情報データを指定された返却先に格納します。

測定チャネルと演算チャネルは，別々に指定してください。開始チャネルタイプで判別します。

「通信インターフェイス」のTSとFMコマンドを実行します。

本関数の実行後，チャネル毎のデータ取得には，getChDataByASCIIを使用します。

戻り値

エラー番号を返します。

参照

checkAck receiveLine send startTalker
CDAQDARWINChInfo::toChName
CDAQDARWINDateTime::setLine

CDAQDARWIN::talkDataByBinary

構文

```
int talkDataByBinary(int startChType, int startChNo, int
endChType, int endChNo, CDAQDARWINDateTime & cDARWINDateTime);
```

引数

startChType	開始チャネルタイプを指定します。
startChNo	開始チャネル番号を指定します。
endChType	終了チャネルタイプを指定します。
endChNo	終了チャネル番号を指定します。
cDARWINDateTime	時刻情報データの返却先を指定します。

説明

開始チャネル(開始チャネルタイプ，開始チャネル番号)から終了チャネル(終了チャネルタイプ，終了チャネル番号)までの測定データをバイナリフォーマットで取得する宣言を実行します。
測定データの時刻情報データを指定された返却先に格納します。
測定チャネルと演算チャネルは，別々に指定してください。開始チャネルタイプで判別します。
バイト出力順序を上位バイト(MSB)からの出力にします。
データメンバの残りサイズ領域に出力バイト数の残りサイズを格納します。
「通信インターフェイス」のBO，TSとFMコマンドを実行します。
本関数の実行後，チャネル毎のデータ取得には，getChDataByBinaryを使用します。

戻り値

エラー番号を返します。
エラー：

Not data	出力バイト数が必要なサイズを満たしていません。

参照

```
checkAck receive runCommand send startTalker
CDAQDARWINChInfo::toChName
CDAQDARWINDateTime::setByte
```

CDAQDARWIN::talkOperationData

構文

```
int talkOperationData(int startChType, int startChNo, int
endChType, int endChNo);
```

引数

startChType	開始チャネルタイプを指定します。
startChNo	開始チャネル番号を指定します。
endChType	終了チャネルタイプを指定します。
endChNo	終了チャネル番号を指定します。

説明

開始チャネル(開始チャネルタイプ，開始チャネル番号)から終了チャネル(終了チャネルタイプ，終了チャネル番号)までの運転モードの設定データを取得する宣言を実行します。

「通信インターフェイス」のTSとLFコマンドを実行します。

本関数の実行後，行単位毎のデータ取得には，getSetDataByLineを使用します。

戻り値

エラー番号を返します。

参照

```
send startTalker
CDAQDARWINChInfo::toChName
```

CDAQDARWIN::talkSetupData

構文

```
int talkSetupData(int startChType, int startChNo, int
endChType, int endChNo);
```

引数

startChType	開始チャネルタイプを指定します。
startChNo	開始チャネル番号を指定します。
endChType	終了チャネルタイプを指定します。
endChNo	終了チャネル番号を指定します。

説明

開始チャネル(開始チャネルタイプ，開始チャネル番号)から終了チャネル(終了チャネルタイプ，終了チャネル番号)までのセットアップモードの設定データを取得する宣言を実行します。

「通信インターフェイス」のTSとLFコマンドを実行します。

本関数の実行後，行単位毎のデータ取得には，getSetDataByLineを使用します。

戻り値

エラー番号を返します。

参照

```
send startTalker
CDAQDARWINChInfo::toChName
```

CDAQDARWIN::transMode

構文

```
int transMode(int iMode);
```

引数

iMode　　操作モードを指定します。

説明

指定された操作モードに切り替えます。
「通信インターフェイス」のDSコマンドを実行します。
コマンドを生成して送信し，応答を受信します。

戻り値

エラー番号を返します。

参照

runCommand

CDAQDARWINChInfoクラス

■ CDAQChInfo
・ CDAQDARWINChInfo

本クラスはCDAQChInfoクラスの派生クラスです。
DARWINシリーズでのチャネル情報データを格納するクラスです。
DarwinChInfo構造体のラッパクラスになります。
トーカ機能のチャネル情報データの取得で取得される以下のデータを格納します。
・ チャネルタイプ
・ チャネル番号
・ 小数点位置
・ チャネルステータス
・ 単位名
以下の出力フォーマット文字列を解析して格納する関数メンバを提供します。

S1S2CCCUUUUUU,P

パブリックメンバ

構築・消滅

CDAQDARWINChInfo	オブジェクトを構築します。
~CDAQDARWINChInfo	オブジェクトを消滅します。

構造体操作

getDarwinChInfo	構造体でデータを取得します。
setDarwinChInfo	構造体でデータを設定します。
initDarwinChInfo	構造体のデータを初期化します。

データメンバ操作

getChStatus	チャネルステータスを取得します。
getUnit	単位名を取得します。
setChStatus	チャネルステータスを設定します。
setUnit	単位名を設定します。

出力フォーマット操作

setLine	チャネル情報データの出力フォーマット(行形式)のデータを解析して，データメンバに情報を格納します。

ユーティリティ

getChName	チャネルを文字列で取得します。
toChName	チャネルを文字列に変換します。
toChRange	チャネル範囲を文字列に変換します。
getStatusName	ステータス値を文字列で取得します。
toStatus	文字，または，データ値からデータステータス値に変換します。
toFlag	文字からフラグに変換します。
toChType	文字からチャネルタイプに変換します。

演算子

operator=	代入を実行します。

●オーバライドしたメンバ

データメンバ操作

initialize	データメンバを初期化します。

ユーティリティ

isObject	オブジェクトをチェックします。

●継承するメンバ

CDAQChInfo参照

```
getChNo getPoint getChType setChNo setChType setPoint
```

プロテクトメンバ

データメンバ

m_chStatus	チャネルステータスの格納領域です。
m_strUnit	単位名の格納領域です。

●継承するメンバ

CDAQChInfo参照

```
m_chNo m_chType m_point
```

プライベートメンバ

なし

関数メンバ(アルファベット順)

CDAQDARWINChInfo::CDAQDARWINChInfo

構文

```
CDAQDARWINChInfo(DarwinChInfo * pDarwinChInfo = NULL);
CDAQDARWINChInfo(int chType, int chNo, int point, const char *
strUnit, int iStatus = DAQDARWIN_DATA_UNKNOWN);
virtual ~CDAQDARWINChInfo(void);
```

引数

pDarwinChInfo	チャネル情報データを構造体で指定します。
chType	チャネルタイプを指定します。
chNo	チャネル番号を指定します。
point	小数点位置を指定します。
strUnit	単位名を指定します。
iStatus	チャネルステータスを指定します。

説明

オブジェクトを構築，消滅します。
構築時，指定されたデータをデータメンバに格納します。指定がない場合，データメンバを初期化します。

参照

```
setChStatus setDarwinChInfo setUnit
CDAQChInfo::CDAQChInfo
```

CDAQDARWINChInfo::getChName

構文

```
int getChName(char * strName, int lenName);
```

引数

strName	文字列を格納する領域を指定します。
lenName	文字列を格納する領域のバイト数を指定します。

説明

データメンバのチャネルタイプ領域とチャネル番号領域の値からチャネルの名称を文字列で作成し，指定された領域に格納します。

戻り値

生成された文字列のバイト数を返します。

参照

```
getChNo getChType toChName
```

CDAQDARWINChInfo::getChStatus

構文

```
int getChStatus(void);
```

説明

データメンバからチャネルステータス領域の値を取得します。

戻り値

チャネルステータスを返します。

CDAQDARWINChInfo::getDarwinChInfo

構文

```
void getDarwinChInfo(DarwinChInfo * pDarwinChInfo);
```

引数

pDarwinChInfo　チャネル情報データの返却先を指定します。

説明

構造体でデータを取得します。
データメンバの内容を，指定された構造体に格納します。

参照

```
getChNo getChStatus getChType getPoint getUnit
```

CDAQDARWINChInfo::getStatusName

構文

```
static const char * getStatusName(int iStatus);
```

引数

iStatus　　データステータス値を指定します。

説明

指定されたデータステータス値に対応する文字列を取得します。
範囲外の場合，「Unknown」になります。

戻り値

文字列へのポインタを返します。

CDAQDARWINChInfo::getUnit

構文

```
const char * getUnit(void);
```

説明

データメンバから単位名領域の単位名を取得します。

戻り値

文字列へのポインタを返します。

CDAQDARWINChInfo::initDarwinChInfo

構文

```
static void initDarwinChInfo(DarwinChInfo * pDarwinChInfo);
```

引数

pDarwinChInfo　チャネル情報データの領域を指定します。

説明

指定された領域を初期化します。
初期値は，　原則0です。

CDAQDARWINChInfo::initialize

構文

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

説明

データメンバを初期化します。初期値は，0です。

参照

```
setChStatus CDAQChInfo::initialize
```

CDAQDARWINChInfo::isObject

構文

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

引数

classname　　クラス名を文字列で指定します。

説明

指定されたクラス名を継承しているかをチェックします。
引数が省略された場合，本クラスであるかをチェックします。
本クラスを継承したクラスが，自分自身のクラスをチェックするためには，オーバーライドする必要があります。
クラスを継承している場合，「有効値」(真)を返します。それ以外は，「無効値」(偽)を返します。
本クラスと異なる場合，親クラスをチェックします。

戻り値

有効無効値を返します。

参照

```
CDAQChInfo::isObject
```

CDAQDARWINChInfo::operator=

構文

```
CDAQDARWINChInfo & operator=(CDAQDARWINChInfo &
cDARWINChInfo);
```

引数

cDARWINChInfo　　代入するオブジェクトを指定します。

説明

指定されたオブジェクトのデータメンバを複写します。

戻り値

本オブジェクトへの参照を返します。

CDAQDARWINChInfo::setChStatus

構文

```
void setChStatus(int iStatus);
```

引数

iStatus　　チャネルステータスを指定します。

説明

データメンバのチャネルステータス領域に指定された値を格納します。

CDAQDARWINChInfo::setDarwinChInfo

構文

```
void setDarwinChInfo(DarwinChInfo * pDarwinChInfo);
```

引数

pDarwinChInfo　チャネル情報データを指定します。

説明

構造体でデータを設定します。
データメンバに指定された構造体の内容を格納します。
指定がない場合，データメンバは初期化されます。

参照

```
initialize setChNo setChStatus setChType setPoint setUnit
```

CDAQDARWINChInfo::setLine

構文

```
int setLine(const char * strLine, int lenLine, int * pFlag);
```

引数

strLine	行を文字列で指定します。
lenLine	行のバイト数を指定します。
pFlag	フラグの返却先を指定します。

説明

指定された行を解析して，データメンバに情報を格納します。
行の形式は，チャネル情報データの出力フォーマットです。

戻り値

エラー番号を返します。
エラー：

Not data	入力データが短かすぎます。または，文字列が違います。

参照

`setChNo setChStatus setChType setPoint setUnit toChType toFlag toStatus`

CDAQDARWINChInfo::setUnit

構文

```
void setUnit(const char * strUnit)
```

引数

strUnit	単位名を指定します。

説明

データメンバの単位名領域に指定された値を格納します。

CDAQDARWINChInfo::toChName

構文

```
static int toChName(int chType, int chNo, char * strName, int
lenName);
```

引数

chType	チャネルタイプを指定します。
chNo	チャネル番号を指定します。
strName	文字列を格納する領域を指定します。
lenName	文字列を格納する領域のバイト数を指定します。

説明

指定されたチャネルタイプとチャネル番号からチャネルの名称を文字列で作成し，指定された領域に格納します。
例えば，チャネルタイプが0，チャネル番号が1，の場合，文字列は「001」になります。
定数として用意されているチャネル/リレータイプ以外に，サブユニット番号(0～5の整数)もチャネル/リレータイプとなります。

戻り値

生成された文字列のバイト数を返します。

CDAQDARWINChInfo::toChRange

構文

```
static int toChRange(int chType, int startChNo, int endChNo,
char * strName, int lenName);
```

引数

chType	チャネルタイプを指定します。
startChNo	開始チャネル番号を指定します。
endChNo	終了チャネル番号を指定します。
strName	文字列を格納する領域を指定します。
lenName	文字列を格納する領域のバイト数を指定します。

説明

指定されたチャネル範囲(チャネルタイプ，開始チャネル番号，終了チャネル番号)の名称を文字列で作成し，指定された領域に格納します。
終了チャネル番号が開始チャネル番号以下の場合，開始チャネル番号による単独のチャネルとみなします。
例えば，チャネルタイプが0，開始チャネル番号が1，終了チャネル番号が2，の場合，文字列は「001-02」になります。
定数として用意されているチャネル/リレータイプ以外に，サブユニット番号(0～5の整数)もチャネル/リレータイプとなります。

戻り値

生成された文字列のバイト数を返します。

参照

toChName

CDAQDARWINChInfo::toChType

構文

```
static int toChType(char cType);
```

引数

cType　　文字を指定します。

説明

指定された値からチャネルタイプに変換します。
範囲外の場合，0になります。
チャネル／リレータイプの1文字をチャネルタイプの値にします。

戻り値

チャネルタイプを返します。

CDAQDARWINChInfo::toFlag

構文

```
static int toFlag(char cFlag);
```

引数

cFlag　　文字を指定します。

説明

指定された値からフラグの値に変換します。
指定された文字が「E」の場合，「最終データ」になります。
範囲外の場合，「全OFF」になります。
指定する文字は，出力フォーマットの「S2」に対応します。

戻り値

フラグを返します。

CDAQDARWINChInfo::toStatus

構文

```
static int toStatus(char cStatus);
static int toStatus(int value);
```

引数

cStatus	文字を指定します。
value	2バイト分のデータ値を整数値で指定します。

説明

指定された値をデータステータス値に変換します。
範囲外の場合，文字指定では「データステータスがセットされていない状態」を，データ値指定では「正常状態」を返します。
文字指定の場合，オーバデータを指定すると，「プラスオーバ状態」を返します。
演算チャネルのデータステータスのデータ値は4バイトです。この値は同じ2バイト値の繰り返しになっています。データ値で指定する場合は，この2バイト値を指定します。
文字指定で指定する文字は，出力フォーマットの「S1」に対応します。
瞬時値データ読み込み時の文字(スペース)の場合，「瞬時値データ読み込み通信時の状態」を返します。
データ値指定の場合， 異常データ値以外は範囲外とし「正常状態」を返します。

戻り値

データステータス値を返します。

CDAQDARWINDataInfoクラス

■ CDAQDataInfo
・ CDAQDARWINDataInfo

本クラスはCDAQDataInfoクラスの派生クラスです。トーカ機能の測定データの取得で得られるデータを，チャネル単位で格納するクラスです。
本クラスはチャネル情報データとの関連と測定データから構成されています。
トーカ機能の測定データの取得で，コード(ASCIIまたはBinary)により，格納される情報が異なります。詳細は，11.4節をご覧ください。
また，チャネル情報データとの関連を設定しておけば，取得したチャネル情報データを格納します。

トーカ機能による測定データを格納する場合，コードに対応する下記の関数メンバを使用します。

コード	関数メンバ
ASCIIコード	setLine
バイナリコード	setByte

ASCIIコードの場合，以下の出力フォーマット文字列を解析して格納する関数メンバを提供します。

S1S2A1A1A2A2A3A3A4A4UUUUUUCCC,±DDDDDE−E

バイナリコードの場合， 6バイト(測定チャネル)，または8バイト(演算チャネル)のデータを解析して格納する関数メンバを提供します。アラームデータを含まないデータには対応しません。
チャネル情報データについては，メンバアクセスの関数メソッドでCDAQDARWINChInfoクラスを取得できます。トーカ機能のチャネル情報データの取得によるチャネル情報データのクラスと同じです。

パブリックメンバ

構築・消滅

CDAQDARWINDataInfo	オブジェクトを構築します。
~CDAQDARWINDataInfo	オブジェクトを消滅します。

構造体操作

getDarwinDataInfo	構造体でデータを取得します。
setDarwinDataInfo	構造体でデータを設定します。
initDarwinDataInfo	構造体のデータを初期化します。

データメンバ操作

getStatus	データステータスを取得します。
getAlarm	アラームを取得します。
setStatus	データステータスを設定します。
setAlarm	アラームを設定します。

出力フォーマット操作

setLine	文字列から測定データを格納します。
setByte	バイト配列から測定データを格納します。

関連付け

getClassDARWINChInfo	チャネル情報データとの関連を取得します。
setClassDARWINChInfo	チャネル情報データとの関連を設定します。

ユーティリティ機能

getAlarmName	アラーム種類の名称を取得します。
toAlarmType	文字列をアラーム種類に変換します。
getMaxLenAlarmName	アラーム種類の名称の最大長を取得します。

演算子

operator=	代入を実行します。

●オーバライドしたメンバ

データメンバ操作

initialize	データメンバを初期化します。

ユーティリティ

isObject	オブジェクトをチェックします。

●継承するメンバ

CDAQDataInfo参照

```
getClassChInfo getDoubleValue getStringValue getValue
setClassChInfo setValue toDoubleValue toStringValue
```

プロテクトメンバ

データメンバ

m_dataStatus	データステータスの格納領域です。
m_alarm	アラーム有無の格納領域です。

●継承するメンバ

CDAQDataInfo参照
`m_pChInfo m_value`

プライベートメンバ

なし

関数メンバ(アルファベット順)

CDAQDARWINDataInfo::CDAQDARWINDataInfo

構文

```
CDAQDARWINDataInfo(DarwinDataInfo * pDarwinDataInfo = NULL,
CDAQDARWINChInfo * pcDARWINChInfo = NULL);
virtual ~CDAQDARWINDataInfo(void);
```

引数

pDarwinDataInfo　　測定データを指定します。
pcDARWINChInfo　　チャネル情報データとの関連を指定します。

説明

オブジェクトを構築，消滅します。
構築時，指定されたデータをデータメンバに格納します。指定がない場合，データメンバを初期化します。

参照

`setClassDARWINChInfo setDarwinDataInfo`
`CDAQDataInfo::CDAQDataInfo`

CDAQDARWINDataInfo::getAlarm

構文

```
int getAlarm(int levelNo);
```

引数

levelNo　　アラームレベルを指定します。

説明

データメンバからアラーム有無領域の値を取得します。
指定されたアラームレベルに対応する値を返します。
アラームレベルが範囲外の場合，「アラームなし」を返します。
アラーム値はアラーム種類の値です。

戻り値

アラーム有無を返します。

CDAQDARWINDataInfo::getAlarmName

構文

```
static const char * getAlarmName(int iAlarmType);
```

引数

iAlarmType　　アラーム種類を指定します。

説明

指定されたアラーム種類に対応する文字列を取得します。
範囲外の場合，「アラームなし」と同じ文字列になります。
文字列は，左詰です。スペースで補完されています。

戻り値

文字列へのポインタを返します。

CDAQDARWINDataInfo::getClassDARWINChInfo

構文

```
CDAQDARWINChInfo * getClassDARWINChInfo(void);
```

説明

データメンバからチャネル情報データとの関連を取得します。

戻り値

チャネル情報データとの関連を返します。

参照

```
getClassChInfo
```

CDAQDARWINDataInfo::getDarwinDataInfo

構文

```
void getDarwinDataInfo(DarwinDataInfo * pDarwinDataInfo);
```

引数

pDarwinDataInfo　　測定データの返却先を指定します。

説明

構造体でデータを取得します。データメンバの内容を，指定された構造体に格納します。

参照

```
getAlarm getStatus getValue
```

CDAQDARWINDataInfo::getMaxLenAlarmName

構文

```
static int getMaxLenAlarmName(void);
```

説明

アラーム種類文字列の最大長を取得します。
戻り値には，終端は含まれません。

戻り値

文字列の最大長をバイト数で返します。

CDAQDARWINDataInfo::getStatus

構文

```
int getStatus(void);
```

説明

データメンバからデータステータス領域の値を取得します。

戻り値

データステータスを返します。

CDAQDARWINDataInfo::initDarwinDataInfo

構文

```
static void initDarwinDataInfo(DarwinDataInfo *
pDarwinDataInfo);
```

引数

pDarwinDataInfo 測定データの領域を指定します。

説明

指定された領域を初期化します。
初期値は，原則0です。

CDAQDARWINDataInfo::initialize

構文

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

説明

データメンバを初期化します。初期値は，原則0です。
データステータス領域は，「不明」にします。
アラーム有無領域は，「アラームなし」にします。。

参照

CDAQDataInfo::initialize

CDAQDARWINDataInfo::isObject

構文

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

引数

classname クラス名を文字列で指定します。

説明

指定されたクラス名を継承しているかをチェックします。
引数が省略された場合，本クラスであるかをチェックします。
本クラスを継承したクラスが，自分自身のクラスをチェックするためには，オーバーライドする必要があります。
クラスを継承している場合，「有効値」(真)を返します。それ以外は，「無効値」(偽)を返します。
本クラスと異なる場合，親クラスをチェックします。

戻り値

有効無効値を返します。

参照

CDAQDataInfo::isObject

CDAQDARWINDataInfo::operator=

構文

```
CDAQDARWINDataInfo & operator=(CDAQDARWINDataInfo &
cDARWINDataInfo);
```

引数

cDARWINDataInfo 代入するオブジェクトを指定します。

説明

指定されたオブジェクトのデータメンバを複写します。

戻り値

本オブジェクトへの参照を返します。

CDAQDARWINDataInfo::setAlarm

構文

```
void setAlarm(int levelNo, int iAlarmType);
```

引数

level アラームレベルを指定します。
iAlarmType アラーム種類を指定します。

説明

データメンバのアラーム有無領域に指定された値を格納します。
アラームレベルが範囲外の場合，何もしません。

CDAQDARWINDataInfo::setByte

構文

```
int setByte(const unsigned char pByte[], int numByte);
```

引数

pByte　　バイト配列の先頭ポインタを指定します。
numByte　　バイト配列のバイト数を指定します。

説明

指定されたバイト配列を解析してデータメンバに情報を格納します。
バイト配列の形式は，バイナリコードによる測定データの出力フォーマットです。
チャネル情報データとの関連が存在する場合，指定されたバイト配列から取得できる情報をチャネル情報データ領域に格納します。

戻り値

エラー番号を返します。
エラー：
Not data　　入力データが短かすぎます。

参照

`getClassDARWINChInfo setAlarm setStatus setValue CDAQDARWINChInfo::setChNo CDAQDARWINChInfo::setChType CDAQDARWINChInfo::toStatus`

CDAQDARWINDataInfo::setClassDARWINChInfo

構文

```
void setClassDARWINChInfo(CDAQDARWINChInfo * pcDARWINChInfo);
```

説明

データメンバのチャネル情報データとの関連を設定します。

参照

`setClassChInfo`

CDAQDARWINDataInfo::setDarwinDataInfo

構文

```
void setDarwinDataInfo(DarwinDataInfo * pDarwinDataInfo);
```

引数

pDarwinDataInfo　　測定データを指定します。

説明

構造体でデータを設定します。データメンバに，指定された構造体の内容を格納します。
指定がない場合，データメンバは初期化されます。

参照

`initialize setAlarm setStatus setValue`

CDAQDARWINDataInfo::setLine

構文

```
int setLine(const char * strLine, int lenLine, int * pFlag);
```

引数

strLine	行を文字列で指定します。
lenLine	行のバイト数を指定します。
pFlag	フラグの返却先を指定します。

説明

指定された行を解析してデータメンバに情報を格納します。
行の形式は，ASCIIコードによる測定データの出力フォーマットです。
チャネル情報データとの関連が存在する場合，指定された行から取得できる情報をチャネル情報データ領域に格納します。

戻り値

エラー番号を返します。
エラー：

Not data	入力データが短かすぎます。または，文字列が違います。

参照

```
getClassDARWINChInfo getStatus setAlarm setStatus setValue
toAlarmType CDAQDARWINChInfo::setChNo
CDAQDARWINChInfo::setChType CDAQDARWINChInfo::setPoint
CDAQDARWINChInfo::setUnit CDAQDARWINChInfo::toChType
CDAQDARWINChInfo::toFlag CDAQDARWINChInfo::toStatus
```

CDAQDARWINDataInfo::setStatus

構文

```
void setStatus(int iDataStatus);
```

引数

iDataStatus	データステータスを指定します。

説明

データメンバのデータステータス領域に指定された値を格納します。

CDAQDARWINDataInfo::toAlarmType

構文

```
static int toAlarmType(const char * strAlarm);
```

引数

strAlarm	アラーム種類の名称を指定します。

説明

指定された文字列をアラーム種類に変換します。
範囲外の場合，「アラームなし」を返します。

戻り値

アラーム種類を返します。

CDAQDARWINDateTimeクラス

■ CDAQDateTime
・ CDAQDARWINDateTime

本クラスはCDAQDateTimeクラスの派生クラスです。
DARWINシリーズでの時刻情報データを格納するクラスです。
DarwinDateTime構造体のラッパクラスになります。
トーカ機能の測定データの取得で取得される時刻情報データを格納します。
トーカ機能による測定データを格納する場合，コードに対応する下記の関数メンバを使用します。

コード	関数メンバ
ASCIIコード	setLine
バイナリコード	setByte

ASCIIコードの場合，以下の出力フォーマット文字列を解析して格納する関数メンバを提供します。

DATEYYMMDD
TIMEhhmmss

バイナリコードの場合，6バイト，または8バイト(瞬時値データ読み込み)のデータを解析して格納する関数メンバを提供します。

パブリックメンバ

構築・消滅

CDAQDARWINDateTime	オブジェクトを構築します。
~CDAQDARWINDateTime	オブジェクトを消滅します。

構造体操作

getDarwinDateTime	構造体でデータを取得します。
setDarwinDateTime	構造体でデータを設定します。
initDarwinDateTime	構造体のデータを初期化します。

データメンバ操作

getYear	年を取得します。
getMonth	月を取得します。
getDay	日を取得します。
getHour	時を取得します。
getMinute	分を取得します。
getSecond	秒を取得します。
getFullYear	4桁の年を取得します。

出力フォーマット操作

setLine	文字列から時刻情報データを格納します。
setByte	バイト配列から時刻情報データを格納します。

ユーティリティ

toString	時刻情報データを文字列に変換します。

演算子

operator=	代入を実行します。

●オーバライドしたメンバ

データメンバ操作

initialize	データメンバを初期化します。
setNow	現在の日付時刻を設定します。

ユーティリティ

isObject	オブジェクトをチェックします。

●継承するメンバ

CDAQDateTime参照

```
getMilliSecond getTime setMilliSecond setTime toLocalDateTime
```

プロテクトメンバ

データメンバ

m_DarwinDateTime	時刻情報データの格納領域です。

変換

toDateTime	データメンバを変換します。

●継承するメンバ

CDAQDateTime参照

```
m_milliSecond m_time
```

プライベートメンバ

なし

関数メンバ(アルファベット順)

CDAQDARWINDateTime::CDAQDARWINDateTime

構文

```
CDAQDARWINDateTime(DarwinDateTime * pDarwinDateTime = NULL);
CDAQDARWINDateTime(int iYaer, int iMonth, int iDay, int iHour
= 0, int iMinute = 0, int iSecond = 0);
virtual ~CDAQDARWINDateTime(void);
```

引数

pDarwinDateTime	時刻情報データを指定します。
iYaer	年の下2桁を指定します。
iMonth	月を指定します。
iDay	日を指定します。
iHour	時を指定します。
iMinute	分を指定します。
iSecond	秒を指定します。

説明

オブジェクトを構築，消滅します。
構築時，指定されたデータをデータメンバに格納します。指定がない場合，データメンバを初期化します。

参照

```
initialize setDarwinDateTime toDateTime
CDAQDateTime::CDAQDateTime
```

CDAQDARWINDateTime::getDarwinDateTime

構文

```
void getDarwinDateTime(DarwinDateTime * pDarwinDateTime);
```

引数

pDarwinDateTime	時刻情報データの返却先を指定します。

説明

構造体でデータを取得します。データメンバの内容を，指定された構造体に格納します。

CDAQDARWINDateTime::getDay

構文

```
int getDay(void);
```

説明

データメンバから日を取得します。

戻り値

日を返します。

CDAQDARWINDateTime::getFullYear

構文

```
int getFullYear(void);
```

説明

データメンバから年を取得します。
下2桁の値を補間して，4桁の値を返します。

戻り値

年を返します。

参照

getYear

CDAQDARWINDateTime::getHour

構文

```
int getHour(void);
```

説明

データメンバから時を取得します。

戻り値

時を返します。

CDAQDARWINDateTime::getMinute

構文

```
int getMinute(void);
```

説明

データメンバから分を取得します。

戻り値

分を返します。

CDAQDARWINDateTime::getMonth

構文

```
int getMonth(void);
```

説明

データメンバから月を取得します。

戻り値

月を返します。

CDAQDARWINDateTime::getSecond

構文

```
int getSecond(void);
```

説明

データメンバから秒を取得します。

戻り値

秒を返します。

CDAQDARWINDateTime::getYear

構文

```
int getYear(void);
```

説明

データメンバから年を取得します。
下2桁の値を返します。

戻り値

年を返します。

CDAQDARWINDateTime::initDarwinDateTime

構文

```
static void initDarwinDateTime(DarwinDateTime *
pDarwinDateTime);
```

引数

pDarwinDateTime 時刻情報データの領域を指定します。

説明

指定された領域を初期化します。
初期値は，原則0です。

CDAQDARWINDateTime::initialize

構文

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

説明

データメンバを初期化します。初期値は，原則0です。

参照

```
initDarwinDateTime CDAQDateTime::initialize
```

CDAQDARWINDateTime::isObject

構文

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

引数

classname クラス名を文字列で指定します。

説明

指定されたクラス名を継承しているかをチェックします。
引数が省略された場合，本クラスであるかをチェックします。
本クラスを継承したクラスが， 自分自身のクラスをチェックするためには，オーバーライドする必要があります。
クラスを継承している場合，「有効値」(真)を返します。それ以外は，「無効値」(偽)を返します。
本クラスと異なる場合，親クラスをチェックします。

戻り値

有効無効値を返します。

参照

CDAQDateTime::isObject

CDAQDARWINDateTime::operator=

構文

```
CDAQDARWINDateTime & operator=(CDAQDARWINDateTime &
cDARWINDateTime);
```

引数

cDARWINDateTime 代入するオブジェクトを指定します。

説明

指定されたオブジェクトのデータメンバを複写します。

戻り値

本オブジェクトへの参照を返します。

参照

toDateTime

CDAQDARWINDateTime::setByte

構文

```
int setByte(const unsigned char pByte[], int numByte);
```

引数

pByte　　バイト配列の先頭ポインタを指定します。
numByte　　バイト配列のバイト数を指定します。

説明

指定されたバイト配列を解析して，データメンバに情報を格納します。
バイト配列内の形式は，バイナリコードによる測定データの出力フォーマットの日付時刻の部分です。
指定が6バイトより多い場合，7バイト目をミリ秒として解釈します。この時，0.1秒台の値をミリ秒に変換します。

戻り値

エラー番号を返します。
エラー：
Not data　　入力データが短かすぎます

参照

`toDateTime`

CDAQDARWINDateTime::setDarwinDateTime

構文

```
void setDarwinDateTime(DarwinDateTime * pDarwinDateTime);
```

引数

pDarwinDateTime　　時刻情報データを指定します。

説明

構造体でデータを設定します。データメンバに指定された構造体の内容を格納します。
指定がない場合，データメンバは初期化されます。

参照

`initialize toDateTime`

CDAQDARWINDateTime::setLine

構文

```
int setLine(const char * strLine, int lenLine);
```

引数

strLine　　行を文字列で指定します。
lenLine　　行のバイト数を指定します。

説明

指定された行を解析して，データメンバに情報を格納します。
行の形式は，ASCIIコードによる測定データの出力フォーマットの先頭2行です。
日付と時刻は別々の行で指定します。
行の先頭が「D」の場合，年月日のフォーマットと解釈します。
行の先頭が「T」の場合，時分秒のフォーマットと解釈します。
それ以外は，「YY-MM-DD hh:mm:ss」のフォーマットと解釈します。

戻り値

エラー番号を返します。
エラー：
Not data　入力データが短かすぎます。または，文字列が違います。

参照

```
toDateTime
```

CDAQDARWINDateTime::setNow

構文

```
voidsetNow(void);
```

説明

現在の日付時刻を取得して，データメンバに格納します。

参照

```
initialize
CDAQDateTime::setNow
```

CDAQDARWINDateTime::toDateTime

構文

```
void toDateTime(void);
```

説明

データメンバの時刻情報データ領域の構造体の内容を1970年01月01日からの秒数に変換して秒領域に格納します。
年が70未満の場合は，2000年代に補間します。

参照

```
getDay getHour getMinute getMonth getSecond getYear
setMilliSecond setTime
```

CDAQDARWINDateTime::toString

構文

```
int toString(char * strDateTime, int lenDateTime);
```

引数

strDateTime　　文字列を格納する領域を指定します。
lenDateTime　　文字列を格納する領域のバイト数を指定します。

説明

データメンバから時刻情報データを文字列に変換して，指定された領域に格納します。
フォーマットは，「YY/MM/DD,hh:mm:ss」です。「通信インターフェイス」のSDコマンドの引数になります。

戻り値

生成された文字列のバイト数を返します。

CDAQDARWINSysInfoクラス

本クラスは，DARWINシリーズでのシステム構成データと測定周期を格納するクラスです。
DarwinSystemInfo構造体のラッパクラスになります。
トーカ機能のシステム構成データの取得で取得されるデータを格納します。
システム構成データ取得において，システム構成データを格納するインターフェイスとして使用するクラスです。

パブリックメンバ

構築・消滅

CDAQDARWINSysInfo	オブジェクトを構築します。
~CDAQDARWINSysInfo	オブジェクトを消滅します。

構造体操作

getDarwinSystemInfo	構造体でデータを取得します。
setDarwinSystemInfo	構造体でデータを設定します。
initDarwinSystemInfo	構造体のデータを初期化します。

データメンバ操作

initialize	データメンバを初期化します。
getInterval	測定周期を取得します。
isExist	ユニットの有無をチェックします。
getModuleName	モジュール名を取得します。
setLine	文字列からシステム構成データを格納します。
getModuleCode	モジュールの内部コードを取得します。

ユーティリティ

toRelayName	リレーを文字列に変換します。
isObject	オブジェクトをチェックします

演算子

operator=	代入を実行します。

プロテクトメンバ

データメンバ

m_nInterval	測定周期の格納領域です。
m_systemInfo	システム構成データの格納領域です。

メンバアクセス

getDarwinUnitInfo	ユニット情報の構造体を取得します。
getDarwinModuleInfo	モジュール情報の構造体を取得します。

プライベートメンバ

なし

関数メンバ(アルファベット順)

CDAQDARWINSysInfo::CDAQDARWINSysInfo

構文

```
CDAQDARWINSysInfo(double interval = 0.0, DarwinSystemInfo *
pDarwinSystemInfo = NULL);
virtual ~CDAQDARWINSysInfo(void);
```

引数

interval	測定周期を指定します。
pDarwinSystemInfo	システム構成データを指定します。

説明

オブジェクトを構築，消滅します。
構築時，指定されたデータをデータメンバに格納します。指定がない場合，データメンバを初期化します。

参照

`setDarwinSystemInfo`

CDAQDARWINSysInfo::getDarwinModuleInfo

構文

```
DarwinModuleInfo * getDarwinModuleInfo(int unitNo, int
slotNo);
```

引数

unitNo	ユニット番号を指定します。
slotNo	スロット番号を指定します。

説明

指定されたユニット番号とスロット番号に対応するデータメンバのシステム構成データ領域からモジュール情報の領域を取得します。
範囲外の場合，NULLを返します。

戻り値

構造体へのポインタを返します。

CDAQDARWINSysInfo::getDarwinSystemInfo

構文

```
void getDarwinSystemInfo(DarwinSystemInfo *
pDarwinSystemInfo);
```

引数

pDarwinSystemInfo　システム構成データの返却先を指定します。

説明

構造体でデータを取得します。データメンバの内容を，指定された構造体に格納します。

CDAQDARWINSysInfo::getDarwinUnitInfo

構文

```
DarwinUnitInfo * getDarwinUnitInfo(int unitNo);
```

引数

unitNo　　ユニット番号を指定します。

説明

指定されたユニット番号に対応するデータメンバのシステム構成データ領域からユニット情報の領域を取得します。
範囲外の場合，NULLを返します。

戻り値

構造体へのポインタを返します。

CDAQDARWINSysInfo::getInterval

構文

```
double getInterval(void);
```

説明

データメンバから測定周期領域の値を取得します。

戻り値

測定周期を返します。

CDAQDARWINSysInfo::getModuleCode

構文

```
int getModuleCode(int unitNo, int slotNo);
```

引数

unitNo　　ユニット番号を指定します。
slotNo　　スロット番号を指定します。

説明

データメンバのシステム構成データ領域から指定されたモジュールの内部コードを取得します。
存在しない場合，0を返します。

戻り値

内部コードを返します。

参照

`getDarwinModuleInfo`

CDAQDARWINSysInfo::getModuleName

構文

```
const char * getModuleName(int unitNo, int slotNo);
```

引数

unitNo　　ユニット番号を指定します。
slotNo　　スロット番号を指定します。

説明

データメンバから指定されたモジュールのモジュール名を取得します。
存在しない場合，NULLを返します。

戻り値

文字列へのポインタを返します。

参照

`getDarwinModuleInfo`

CDAQDARWINSysInfo::initDarwinSystemInfo

構文

```
static void initDarwinSystemInfo(DarwinSystemInfo *
pDarwinSystemInfo);
```

引数

pDarwinSystemInfo　システム構成データの領域を指定します。

説明

指定された領域を初期化します。
初期値は，原則0です。

CDAQDARWINSysInfo::initialize

構文

```
voidinitialize(void);
```

説明

データメンバを初期化します。初期値は，原則0です。
測定周期領域は初期化しません。

参照

```
initDarwinSystemInfo
```

CDAQDARWINSysInfo::isExist

構文

```
int isExist(int unitNo);
```

引数

unitNo　　ユニット番号を指定します。

説明

指定されたユニットの有効性をチェックします。
存在しない場合，「無効値」を返します。

戻り値

有効無効値を返します。

参照

```
getDarwinUnitInfo
```

CDAQDARWINSysInfo::isObject

構文

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

引数

classname　クラス名を文字列で指定します。

説明

指定されたクラス名を継承しているかをチェックします。
省略された場合，本クラスであるかをチェックします。
本クラスを継承したクラスが，自分自身のクラスをチェックするためには，オーバーライドする必要があります。
クラスを継承している場合，「有効値」(真)を返します。それ以外は，「無効値」(偽)を返します。

戻り値

有効無効値を返します。

CDAQDARWINSysInfo::operator=

構文

```
CDAQDARWINSysInfo & operator=(CDAQDARWINSysInfo &
cDARWINSysInfo);
```

引数

cDARWINSysInfo　代入するオブジェクトを指定します。

説明

指定されたオブジェクトのデータメンバを複写します。

戻り値

本オブジェクトへの参照を返します。

CDAQDARWINSysInfo::setDarwinSystemInfo

構文

```
void setDarwinSystemInfo(DarwinSystemInfo *
pDarwinSystemInfo);
```

引数

pDarwinSystemInfo　システム構成データを指定します。

説明

構造体でデータを設定します。データメンバに指定された構造体の内容を格納します。

指定がない場合，データメンバは初期化されます。

参照

`initialize`

CDAQDARWINSysInfo::setLine

構文

```
int setLine(const char * strLine, int lenLine, int * pFlag);
```

引数

strLine	行を文字列で指定します。
lenLine	行のバイト数を指定します。
pFlag	フラグの返却先を指定します。

説明

指定された行を解析して，データメンバに情報を格納します。
行の形式は，システム構成データの出力フォーマットです。
行の先頭が「M」の場合，測定周期のフォーマットと解釈します。
行の先頭が「E」の場合，最終行と解釈します。
行の先頭が「I」の場合，メインユニットと解釈します。
それ以外は，サブユニットと解釈します。

戻り値

エラー番号を返します。
エラー：

Not data	入力データが短かすぎます。または，文字列が違います。

参照

`getDarwinUnitInfo`

CDAQDARWINSysInfo::toRelayName

構文

```
static int toRelayName(int relayType, int relayNo, char *
strName, int lenName);
```

引数

relayType	リレータイプを指定します。
relayNo	リレー番号を指定します。
strName	文字列を格納する領域を指定します。
lenName	文字列を格納する領域のバイト数を指定します。

説明

指定されたリレータイプとリレー番号からリレーの名称を文字列で作成し，指定された領域に格納します。
リレー番号が0の場合，「OFF」になります。

戻り値

生成された文字列のバイト数を返します。

参照

`CDAQDARWINChInfo::toChName`

8.1　機能と関数の対応－DARWIN/Visual C－

本APIでサポートする機能と，Visual Cの関数群の対応を示します。

Note

本APIでは，DARWINシリーズ機器の共通機能の一部を定供しています。機種別の機能，セットアップモードの設定機能，A/D校正機能は実装されていません。DARWIN通信機能のコマンドを使用して，機能を追加することができます。

表中の「コマンド」とは，DARWIN通信機能のコマンドのことです。コマンドの詳細については，通信インターフェースユーザーズマニュアルをご覧ください。

通信機能

機能	関数
DARWINと通信接続	openDARWIN
DARWINとの通信を切断	closeDARWIN
データを行単位で送信 特別にデータ受信を制御する場合に使用します。	sendLineDARWIN
データを行単位で受信 特別にデータ受信を制御する場合に使用します。	receiveLineDARWIN
バイト単位でデータを受信します。 特別にデータ受信を制御する場合に使用します。	receiveByteDARWIN
コマンドを送信し，応答を受信 機能コマンドを実装する場合に使用します。	runCommandDARWIN
ステータスバイトを取得 ステータスバイト出力コマンドを送信し，応答を受信します。	getStatusByteDARWIN
トリガコマンド(ESC T)を送信し，応答を受信 新たにトーカ機能を実装する場合に使用します。	sendTriggerDARWIN
通信タイムアウトを設定	setTimeOutDARWIN

Note

通信タイムアウトの設定を推奨しません。**理由**：データ取得時にタイムアウト時間に抵触して予期しない通信切断が発生する場合があります。

制御機能

機能	コマンド	関数
操作モード切り替え	DS	transModeDARWIN
システム再構築	RS	initSystemDARWIN
RAMクリア(運転モード設定パラメータの初期化)	RC	
アラームリセット	AR	
日付時刻設定	SD	setDateTimeDARWIN
	SD	setDateTimeNowDARWIN
演算のスタート，ストップ	EX	computeDARWIN
レポートのスタート，ストップ	DR	reportingDARWIN
セットアップモード確定	XE	establishDARWIN

設定機能

機能		コマンド	関数
レンジ設定	スキップ(未使用)	SR	setSKIPDARWIN
	直流電圧入力	SR	setVOLTDARWIN
	熱電対入力	SR	setTCDARWIN
	測温抵抗体入力	SR	setRTDDARWIN
	接点入力(DI)	SR	setDIDARWIN
	チャネル間差演算	SR	setDELTADARWIN
	リモートRJC	SR	setRRJCDARWIN
	直流電流	SR	setMADARWIN
	ひずみ	SR	setSTRAINDARWIN
	パルス	SR	setPULSEDARWIN
	パワーモニタ	SR	setPOWERDARWIN
スケーリングの単位を設定		SN	setScallingUnitDARWIN
アラームを設定		SA	setAlarmDARWIN

データ取得機能

機能	コマンド	関数
システム構成データを取得	TS，CF	getSystemConfigDARWIN
チャネル情報データの取得を宣言	TS，LF	talkChInfoDARWIN
チャネル情報データを取得		getChInfoDARWIN
測定データの取得を宣言(ASCIIコード)	TS，FM	talkDataByASCIIDARWIN
測定データを取得(ASCIIコード)		getChDataByASCIIDARWIN
測定データの取得を宣言(バイナリコード)	TS，FM	talkDataByBinaryDARWIN
測定データを取得(バイナリコード)		getChDataByBinaryDARWIN
設定データの取得を宣言(運転モード)	TS，LF	talkOperationDataDARWIN
設定データを取得(運転モード)		getSetDataByLineDARWIN
設定データの取得を宣言(セットアップモード)	TS，LF	talkSetupDataDARWIN
設定データを取得(セットアップモード)		getSetDataByLineDARWIN
設定データの取得を宣言(A/D校正モード)	TS，LF	talkCalibrationDataDARWIN
設定データを取得(A/D校正モード)		getSetDataByLineDARWIN
レポートステータスの取得	TS,RF	getReportStatusDARWIN

ユーティリティ

機能		関数
測定値を倍精度浮動小数に変換		toDoubleValueDARWIN
測定値を文字列に変換		toStringValueDARWIN
アラーム	アラーム種類の文字列を取得	toAlarmNameDARWIN
		getAlarmNameDARWIN
	アラーム文字列の最大長を取得	getMaxLenAlarmNameDARWIN
本APIのバージョン番号を取得		getVersionAPIDARWIN
本APIのリビジョン番号を取得		getRevisionAPIDARWIN
エラーメッセージ文字列を取得		getErrorMessageDARWIN
エラーメッセージ文字列の最大長を取得		getMaxLenErrorMessageDARWIN

機能コマンドの実装

DARWIN通信機能コマンドを使用して，機能コマンドを実装できます。使用できるDARWIN通信機能コマンドは下記のとおりです。

- DA100データアクイジションユニット用の通信コマンドすべて
- DC100データコレクタ用の通信コマンドすべて
- DR130，DR231，DR232，DR241，DR242ハイブリッドレコーダ用の通信コマンドすべて

8.2　プログラムーDARWIN/Visual Cー

インクルードファイルのパスを追加

プロジェクトに，インクルードファイル(DAQDARWIN.h)のパスを追加します。追加方法は，ご使用の環境により異なります。

ソースファイルでの宣言

ソースファイルに宣言を記述します。

```
#include "DAQDARWIN.h"
```

Note

共通部のインクルードファイル(DAQHandler.h)は，上記インクルードファイルから参照されているので，宣言を記述する必要はありません。

ロードライブラリの記述

本APIの実行可能モジュール(.dll)がプロセスとリンクできるようにするため，下記の記述をします。

本APIの実行可能モジュール(.dll)をアドレス空間内にマップします(LoadLibrary)。次に，実行可能モジュール内のエクスポート関数のアドレスを取得(GetProcAddress)します。

関数ポインタのコールバック型は，関数名に接頭語「DLL」をつけてすべて大文字にしたものです。本APIのインクルードファイルで定義されています。

```
HMODULE pDll = LoadLibrary("DAQDARWIN");
DLLOPENDARWIN openDARWIN = (DLLOPENMX)GetProcAddress(pDll,
"openDARWIN");
```

測定データの取得

プログラム例1

測定データを取得するプログラムです。

```
////////////////////////////////////////////////////////////
// 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
  DLLOPENDARWIN openDARWIN;
  DLLCLOSEDARWIN closeDARWIN;
  DLLTALKDATABYBINARYDARWIN talkDataByBinaryDARWIN;
  DLLGETCHDATABYBINARYDARWIN getChDataByBinaryDARWIN;
  //laod
  pDll = LoadLibrary("DAQDARWIN");
  //get address
  openDARWIN = (DLLOPENDARWIN)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 { //meaasured data
    rc = getChDataByBinaryDARWIN(comm, &chinfo, &datainfo,
&flag);
  } while (! (flag & DAQDARWIN_FLAG_ENDDATA));
  //disconnect
  rc = closeDARWIN(comm);
#ifdef WIN32
  FreeLibrary(pDll);
```

```
#endif
  return rc;
}
////////////////////////////////////////////////////////////////
```

説明

全般

データを取得する場合，最初にトーカを実行し，チャネルまたは行単位でデータ取得を実行します。終了はフラグで判断します。

インクルードファイルの記述

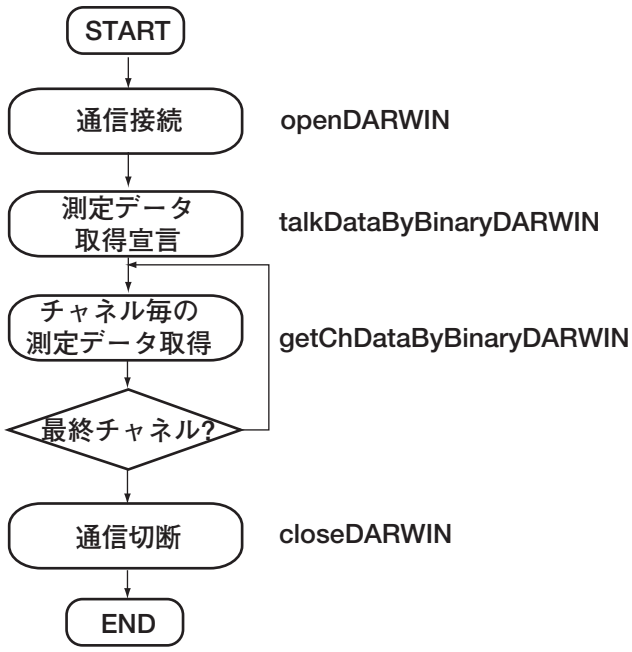
```
#include "DAQDARWIN.h"
```

ロードライブラリの記述

#ifdef WIN32から#endif //WIN32までがロードライブラリの記述です。コールバック型(DLLOPENDARWINなど)を使用しています。

処理の流れ

下記のフローチャートでは，宣言部分を省略しています。



通信処理

最初に通信接続を行います。通信接続後，各関数が利用可能です。最後に終了処理として，通信切断を行います。

通信接続

```
openDARWIN("192.168.1.11", &rc)
```

DARWINのIPアドレスを指定しています。
通信用ポートは，通信用定数の「DARWINの通信ポート番号」を指定したことになります。

トーカ

```
talkDataByBinaryDARWIN(comm, 0, 1, 0, 2, &datetime)
```

サブユニット番号0/チャネル1，2の測定データ取得要求を送信し，時刻情報を取得します(測定データ取得宣言)。

測定データの取得

```
getChDataByBinaryDARWIN(comm, &chinfo, &datainfo, &flag)
```

測定データを，チャネル単位で取得します。指定されたチャネルまで繰り返します。
終了はフラグステータスの「最終データ」により判断します。

通信切断

```
closeDARWIN(comm)
```

通信を切断します。

設定データの取得/設定

プログラム例2

下記の2つを実行するプログラムです。このプログラムではまとめて記述していますが，それぞれ個別に記述して実行できます。

・運転モードの設定データを取得
・チャネルに直流電圧レンジを設定

```
////////////////////////////////////////////////////////////
// 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;
  DLLSETVOLTDARWIN setVOLTDARWIN;
  //laod
  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");
  setVOLTDARWIN = (DLLSETVOLTDARWIN)GetProcAddress(pDll,
"setVOLTDARWIN");
#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;
}
////////////////////////////////////////////////////////////
```

説明

ロードライブラリの記述

#ifdef WIN32から#endif //WIN32までがロードライブラリの記述です。コールバック型(DLLOPENDARWINなど)を使用しています。

トーカ

`talkOperationDataDARWIN(comm, 0, 1, 0, 2)`

取得する設定データの種類(運転モードの設定データ)と，対象チャネル範囲(サブユニット番号0/チャネル1，2)を指定します。

運転モードの設定データを取得

`getSetDataByLineDARWIN(comm, line, BUFSIZ, &len, &flag)`

トーカ機能による出力を，行単位で取得します。
終了はフラグステータスの「最終データ」により判断します。

チャネルに直流電圧レンジを設定

`setVOLTDARWIN(comm, DAQDARWIN_RANGE_VOLT_20MV, 0, 1, 2, 0, 0, 0, 0, 0)`

サブユニット番号0/チャネル1，2の測定レンジを，「20mV」に設定します。スケーリング機能は使用しません。
レンジ種類の指定には，「20mV」定数を使用しています。

機能コマンドの実装

プログラム例3

DARWINを運転モードに切り替えるプログラムです。DARWIN通信機能のDSコマンドを使用しています。

```
////////////////////////////////////////////////////////////
// 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;
}
////////////////////////////////////////////////////////////
```

説明

メッセージの作成

```
sprintf(line, "DS%d", DAQDARWIN_MODE_OPE)
```

DARWIN通信機能のDS0(運転モードに切り替え)コマンドメッセージを配列lineに格納します。
運転モードを指定するために，「運転モード」定数を使用しています。

メッセージの送信

```
runCommandDARWIN(comm, line)
```

コマンドメッセージを送信し，応答を受信します。メッセージのバイト数の記述は省略しています。本関数が，メッセージにターミネータを付けて送信します。

トーカ機能の実装

プログラム例4

システム構成データを取得するプログラムです。DARWIN通信機能のTSコマンドとCFコマンドを実行しています。

```
////////////////////////////////////////////////////////////
// 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;
}
////////////////////////////////////////////////////////////
```

説明

ロードライブラリの記述

#ifdef WIN32からから#endif //WIN32までがロードライブラリの記述です。コールバック型(DLLOPENDARWINなど)を使用しています。

トーカ

`sprintf(line, "TS%d", DAQDARWIN_TALK_SYSINFODATA)`

DARWIN通信機能のTS5(システム構成データの取得を指定)コマンドメッセージをlineに格納します。

システム構成データの出力指定には，「システム構成データの出力」定数を使用しています。

`runCommandDARWIN(comm, line)`

メッセージを送信し，応答を受信します。本関数が，メッセージにターミネータを付けて送信します。

`sendTriggerDARWIN(comm)`

トリガ(機器トリガ)を送信します。

システム構成データの出力フォーマット指定

`sendLineDARWIN(comm, "CF0")`

通信機能コマンドCF0(システム構築されたモジュール情報を指定)を送信します。本関数が，メッセージにターミネータを付けて送信します。

データ取得

`receiveLineDARWIN(comm, line, BUFSIZ, &len)`

システム構成データを行単位で取得します。エンドマーク(E)が返されたときに終了します。

Note

receiveLine関数は，単純にデータを受信する関数なので，ユーザーが終了の判断を記述する必要があります。

エラー処理

- ほとんどの関数は，戻り値として，関数の処理結果の状態をエラー番号で返します。
- エラー番号に対応するエラーメッセージ文字列を得ることができる関数(getErrorMessageDARWIN)があります。また，エラーメッセージ文字列の最大長を得る関数(getMaxLenErrorMessageDARWIN)もあります。

9.1　機能と関数の対応－DARWIN/Visual Basic－

本APIでサポートする機能と，Visual Basicの関数群の対応を示します。

Note

本APIでは，DARWINシリーズ機器の共通機能の一部を定供しています。機種別の機能，セットアップモードの設定機能，A/D校正機能は実装されていません。DARWIN通信機能のコマンドを使用して，機能を追加することができます。

表中の「コマンド」とは，DARWIN通信機能のコマンドのことです。コマンドの詳細については，通信インターフェースユーザーズマニュアルをご覧ください。

通信機能

機能	関数
DARWINと通信接続	openDARWIN
DARWINとの通信を切断	closeDARWIN
データを行単位で送信 特別にデータ送信を制御する場合に使用します。	sendLineDARWIN
データを行単位で受信 特別にデータ受信を制御する場合に使用します。	receiveLineDARWIN
バイト単位でのデータ受信 特別にデータ受信を制御する場合に使用します。	receiveByteDARWIN
コマンドを送信し，応答を受信 機能コマンドを実装する場合に使用します。	runCommandDARWIN
ステータスバイトを取得 ステータスバイト出力コマンドを送信し，応答を受信します。	getStatusByteDARWIN
トリガコマンド(ESC T)を送信し，応答を受信 新たにトーカ機能を実装する場合に使用します。	sendTriggerDARWIN
通信タイムアウトを設定	setTimeOutDARWIN

Note

通信タイムアウトの設定を推奨しません。**理由**：データ取得時にタイムアウト時間に抵触して予期しない通信切断が発生する場合があります。

制御機能

機能	コマンド	関数
操作モード切り替え	DS	transModeDARWIN
システム再構築	RS	initSystemDARWIN
RAMクリア(運転モード設定パラメータの初期化)	RC	
アラームリセット	AR	
日付時刻設定	SD	setDateTimeDARWIN
	SD	setDateTimeNowDARWIN
演算のスタート，ストップ	EX	computeDARWIN
レポートのスタート，ストップ	DR	reportingDARWIN
セットアップモード確定	XE	establishDARWIN

設定機能

機能		コマンド	関数
レンジ設定	スキップ(未使用)	SR	setSKIPDARWIN
	直流電圧入力	SR	setVOLTDARWIN
	熱電対入力	SR	setTCDARWIN
	測温抵抗体入力	SR	setRTDDARWIN
	接点入力(DI)	SR	setDIDARWIN
	チャネル間差演算	SR	setDELTADARWIN
	リモートRJC	SR	setRRJCDARWIN
	直流電流	SR	setMADARWIN
	ひずみ	SR	setSTRAINDARWIN
	パルス	SR	setPULSEDARWIN
	パワーモニタ	SR	setPOWERDARWIN
スケーリングの単位を設定		SN	setScallingUnitDARWIN
アラームを設定		SA	setAlarmDARWIN

データ取得機能

機能	コマンド	関数
システム構成データを取得	TS，CF	getSystemConfigDARWIN
チャネル情報データの取得を宣言	TS，LF	talkChInfoDARWIN
チャネル情報データを取得		getChInfoDARWIN
測定データの取得を宣言(ASCIIコード)	TS，FM	talkDataByASCIIDARWIN
測定データを取得(ASCIIコード)		getChDataByASCIIDARWIN
測定データの取得を宣言(バイナリコード)	TS，FM	talkDataByBinaryDARWIN
測定データを取得(バイナリコード)		getChDataByBinaryDARWIN
設定データの取得を宣言(運転モード)	TS，LF	talkOperationDataDARWIN
設定データを取得(運転モード)		getSetDataByLineDARWIN
設定データの取得を宣言(セットアップモード)	TS，LF	talkSetupDataDARWIN
設定データを取得(セットアップモード)		getSetDataByLineDARWIN
設定データの取得を宣言(A/D校正モード)	TS，LF	talkCalibrationDataDARWIN
設定データを取得(A/D校正モード)		getSetDataByLineDARWIN
レポートステータスの取得	TS，RF	getReportStatusDARWIN

ユーティリティ

機能		関数
測定値を倍精度浮動小数に変換		toDoubleValueDARWIN
測定値を文字列に変換		toStringValueDARWIN
アラーム	アラーム種類の文字列を取得	toAlarmNameDARWIN
	アラーム文字列の最大長を取得	getMaxLenAlarmNameDARWIN
本APIのバージョン番号を取得		getVersionAPIDARWIN
本APIのリビジョン番号を取得		getRevisionAPIDARWIN
エラーメッセージ文字列を取得		toErrorMessageDARWIN
エラーメッセージ文字列の最大長を取得		getMaxLenErrorMessageDARWIN

機能コマンドの実装

DARWIN通信機能コマンドを使用して，機能コマンドを実装できます。使用できるDARWIN通信機能コマンドは下記のとおりです。

- DA100データアクイジションユニット用の通信コマンドすべて
- DC100データコレクタ用の通信コマンドすべて
- DR130，DR231，DR232，DR241，DR242ハイブリッドレコーダ用の通信コマンドすべて

9.2　プログラムーDARWIN/Visual Basicー

型，関数，定数の宣言

Visual Basic用の型，関数，定数を使用するためには，あらかじめ宣言しておく必要があります。次の記述方法があります。

全宣言の記述

プロジェクトにＶｉｓｕａｌ　Ｂａｓｉｃ用標準モジュールライブラリファイル(DAQDARWIN.bas)を追加すると，すべての型，関数，定数を宣言したことになります。

宣言の選択記述

Visual Studioに付属しているAPIビューアで，任意の型，関数，定数の宣言記述をコピーできます。この機能を使用するためには，APIビューアで，APIビューア用テキストファイル(DAQDARWIN.txt)を読み込んでください。
APIビューアの使用方法については，Visual Studioの取扱説明書をご覧ください。

宣言の直接記述

記述例を示します。

```
Public Declare Function openDARWIN Lib "DAQDARWIN" (ByVal
strAddress As String, ByRef errorCode As Long) As Long
```

測定データの取得

プログラム例1

測定データを取得するプログラムです。

```
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
```

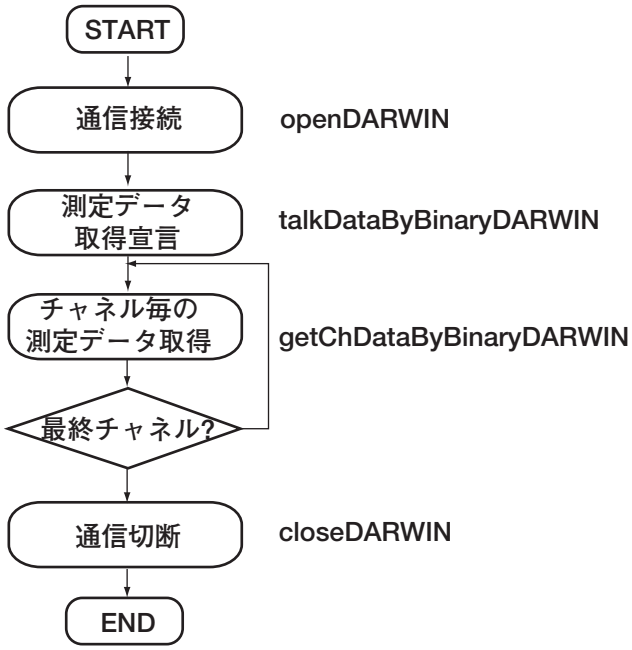
説明

全般

データを取得する場合，最初にトーカを実行し，チャネルまたは行単位でデータ取得を実行します。終了はフラグで判断します。

処理の流れ

下記のフローチャートでは，宣言部分を省略しています。



通信処理

最初に通信接続を行います。通信接続後，各関数が利用可能です。最後に終了処理として，通信切断を行います。

通信接続

```
openDARWIN(host, rc)
```

DARWINのIPアドレスを指定しています。通信用ポートは，通信用定数の「DARWINの通信ポート番号」を指定したことになります。

トーカ

```
talkDataByBinaryDARWIN(comm, 0, 1, 0, 2, datetime)
```

サブユニット番号0/チャネル1，2の測定データ取得要求を送信し，時刻情報を取得します(測定データ取得宣言)。

測定データの取得

```
getChDataByBinaryDARWIN(comm, chinfo, datainfo, flag)
```

測定データをチャネル単位で取得します。指定されたチャネルまで繰り返します。終了はフラグステータスの「最終データ」により判断します。

通信切断

```
closeDARWIN(comm)
```

通信を切断します。

設定データの取得/設定

プログラム例2

下記の2つを実行するプログラムです。このプログラムではまとめて記述していますが，それぞれ個別に記述して実行できます。

・ 運転モードの設定データを取得
・ チャネルに直流電圧レンジを設定

```
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
```

説明

トーカ

`talkOperationDataDARWIN(comm, 0, 1, 0, 2)`

取得する設定データの種類(運転モードの設定データ)と，対象チャネル範囲(サブユニット番号0/チャネル1，2)を指定します。

運転モードの設定データを取得

`getSetDataByLineDARWIN(comm, line, 256, lenLine, flag)`

トーカ機能による出力を，行単位で256バイトの領域に取得します。
終了はフラグステータスの「最終データ」により判断します。

チャネルに直流電圧レンジを設定

`setVOLTDARWIN(comm, DAQDARWIN_RANGE_VOLT_20MV, 0, 1, 2, 0, 0, 0, 0, 0)`

サブユニット番号0/チャネル1，2の測定レンジを，「20mV」に設定します。スケーリング機能は使用しません。
レンジ種類の指定には，「20mV」定数を使用しています。

機能コマンドの実装

プログラム例3

DARWINを運転モードに切り替えるプログラムです。DARWIN通信機能のDSコマンドを使用しています。

```
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
```

説明

メッセージの送信

`runCommandDARWIN(comm, line)`

コマンドメッセージを送信し，応答を受信します。本関数が，メッセージにターミネータを付けて送信します。

トーカ機能の実装

プログラム例4

システム構成データを取得するプログラムです。DARWIN通信機能のTSコマンドとCFコマンドを実行しています。

```
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
```

説明

トーカ

`runCommandDARWIN(comm, "TS5")`

DARWIN通信機能のTS5(システム構成データの取得を指定)コマンドメッセージを送信し，応答を受信します。本関数が，メッセージにターミネータを付けて送信します。

`sendTriggerDARWIN(comm)`

トリガ(機器トリガ)を送信します。

システム構成データの出力フォーマット指定

`sendLineDARWIN(comm, "CF0")`

通信機能コマンドCF0(システム構築されたモジュール情報を指定)を送信します。本関数が，メッセージにターミネータを付けて送信します。

データ取得

`receiveLineDARWIN(comm, line, 256, lenLine)`

システム構成データを行単位で256バイトの領域に取得します。エンドマーク(E)が返されたときに終了します。

Note

receiveLine関数は，単純にデータを受信する関数なので，ユーザーが終了の判断を記述する必要があります。

エラー処理

- ほとんどの関数は，戻り値として，関数の処理結果の状態をエラー番号で返します。
- エラー番号に対応するエラーメッセージ文字列を得ることができる関数(toErrorMessageDARWIN)があります。また，エラーメッセージ文字列の最大長を得る関数(getMaxLenErrorMessageDARWIN)もあります。

10.1 関数の詳細ーDARWIN(Visual C/Visual Basic)ー

ここでは，Visual CとVisual Basicで使用するDARWIN用関数について説明しています。関数は，関数名のアルファベット順で並んでいます。

定数，型については第11章をご覧ください。
DARWINに関する用語については付録2をご覧ください。

ほとんどの関数は戻り値として，エラー番号を返します。正常終了の場合は，エラー番号「0」を返します。

closeDARWIN

構文

```
int closeDARWIN(DAQDARWIN daqdarwin);
```

宣言

```
Public Declare Function closeDARWIN Lib "DAQDARWIN" (ByVal
daqdarwin As Long) As Long
```

引数

daqdarwin　　機器記述子を指定します。

説明

指定された機器記述子による通信切断をします。

・ 通信を切断すると，機器記述子の値は無意味になります。

・ 切断後は，機器記述子の値は使用しないでください。

戻り値

エラー番号を返します。

エラー：

Not descriptor　機器記述子がありません。

参照

`CDAQDARWIN::close`

computeDARWIN

構文

```
int computeDARWIN(DAQDARWIN daqdarwin, int iCompute);
```

宣言

```
Public Declare Function computeDARWIN Lib "DAQDARWIN" (ByVal
daqdarwin As Long, ByVal iCompute As Long) As Long
```

引数

daqdarwin　　機器記述子を指定します。
iCompute　　演算処理を指定します。

説明

演算のスタート／ストップを実行します。

・ 演算オプションがある場合に有効です。
・ 本関数は 「通信インターフェイス」のEXコマンドを実行します。

戻り値

エラー番号を返します。
エラー：
Not descriptor　機器記述子がありません。

参照

`CDAQDARWIN::compute`

establishDARWIN

構文

```
int establishDARWIN(DAQDARWIN daqdarwin, int iSetup);
```

宣言

```
Public Declare Function establishDARWIN Lib "DAQDARWIN" (ByVal
daqdarwin As Long, ByVal iSetup As Long) As Long
```

引数

daqdarwin	機器記述子を指定します。
iSetup	セットアップ確定を指定します。

説明

セットアップモードの設定内容を確定します。

・ セットアップモードでのみ有効です。

・ 本関数は「通信インターフェイス」のXEコマンドを実行します。

戻り値

エラー番号を返します。

エラー：

Not descriptor	機器記述子がありません。

参照

`CDAQDARWIN::establish`

getAlarmNameDARWIN [Visual Cのみ]

構文

```
const char * getAlarmNameDARWIN(int iAlarmType);
```

引数

iAlarmType アラーム種類を指定します。

説明

指定されたアラーム種類に対応する文字列を取得します。

・ Visual Basicの場合，toAlarmNameDARWIN関数を使用してください。

戻り値

アラーム種類文字列へのポインタを返します。

参照

`CDAQDARWINDataInfo::getAlarmName`

getChDataByASCIIDARWIN

構文

```
int getChDataByASCIIDARWIN(DAQDARWIN daqdarwin, DarwinChInfo *
pDarwinChInfo, DarwinDataInfo * pDarwinDataInfo, int * pFlag);
```

宣言

```
Public Declare Function getChDataByASCIIDARWIN Lib "DAQDARWIN"
(ByVal daqdarwin As Long, ByRef pDarwinChInfo As DarwinChInfo,
ByRef pDarwinDataInfo As DarwinDataInfo, ByRef pFlag As Long)
As Long
```

引数

daqdarwin	機器記述子を指定します。
pDarwinChInfo	チャネル情報データの返却先を指定します。
pDarwinDataInfo	測定データの返却先を指定します。
pFlag	フラグの返却先を指定します。

説明

talkDataByASCIIDARWIN関数で宣言したトーカ機能による測定データの出力をチャネル単位で取得します。

- チャネル単位で受信した情報を解析して，構造体に格納します。
- 返却先が指定されていれば，チャネル情報データや測定データを指定先に格納します。
- 最終データを取得した場合，フラグにフラグステータスをセットします。また，エラーで終了した場合もフラグステータスをセットします。
- データ取得を終了するまで，他の関数で通信を行わないでください。本関数でデータ取得中は，他の関数が正しく動作できません。

戻り値

エラー番号を返します。

エラー：

Not descriptor	機器記述子がありません。

参照

```
CDAQDARWIN::getChDataByASCII
CDAQDARWINChInfo::getDarwinChInfo
CDAQDARWINDataInfo::getDarwinDataInfo
```

getChDataByBinaryDARWIN

構文

```
int getChDataByBinaryDARWIN(DAQDARWIN daqdarwin, DarwinChInfo
* pDarwinChInfo, DarwinDataInfo * pDarwinDataInfo, int *
pFlag);
```

宣言

```
Public Declare Function getChDataByBinaryDARWIN Lib
"DAQDARWIN" (ByVal daqdarwin As Long, ByRef pDarwinChInfo As
DarwinChInfo, ByRef pDarwinDataInfo As DarwinDataInfo, ByRef
pFlag As Long) As Long
```

引数

daqdarwin	機器記述子を指定します。
pDarwinChInfo	チャネル情報データの返却先を指定します。
pDarwinDataInfo	測定データの返却先を指定します。
pFlag	フラグの返却先を指定します。

説明

talkDataByBinaryDARWIN関数のトーカ機能による出力をチャネルごとに取得します。

- チャネル単位で受信した情報を解析して，構造体に格納します。
- 返却先が指定されていれば，チャネル情報データや測定データを指定先に格納します。
- 最終データを取得した場合，フラグにフラグステータスをセットします。また，エラーで終了した場合もフラグステータスをセットします。
- データ取得を終了するまで，他関数で通信を行わないでください。本関数でデータ取得中は，他の関数が正しく動作できません。

戻り値

エラー番号を返します。

エラー：

Not descriptor	機器記述子がありません。

参照

```
CDAQDARWIN::getChDataByBinary
CDAQDARWINChInfo::getDarwinChInfo
CDAQDARWINDataInfo::getDarwinDataInfo
```

getChInfoDARWIN

構文

```
int getChInfoDARWIN(DAQDARWIN daqdarwin, DarwinChInfo *
pDarwinChInfo, int * pFlag);
```

宣言

```
Public Declare Function getChInfoDARWIN Lib "DAQDARWIN" (ByVal
daqdarwin As Long, ByRef pDarwinChInfo As DarwinChInfo, ByRef
pFlag As Long) As Long
```

引数

daqdarwin	機器記述子を指定します。
pDarwinChInfo	チャネル情報データの返却先を指定します。
pFlag	フラグの返却先を指定します。

説明

talkChInfoDARWIN関数で宣言したトーカ機能によるチャネル情報データの出力をチャネル単位で取得します。

- チャネル単位で受信した情報を解析して，構造体に格納します。
- 返却先が指定されていれば，チャネル情報データを指定先に格納します。
- 最終データを取得した場合，フラグにフラグステータスがセットされます。また，エラーで終了した場合もフラグステータスをセットします。
- データ取得を終了するまで，他関数で通信を行わないでください。本関数でデータ取得中は，他関数が正しく動作できません。

戻り値

エラー番号を返します。

エラー：

Not descriptor　機器記述子がありません。

参照

```
CDAQDARWIN::getChInfo
CDAQDARWINChInfo::getDarwinChInfo
```

getErrorMessageDARWIN [Visual Cのみ]

構文

```
const char * getErrorMessageDARWIN(int errCode);
```

引数

errCode エラー番号を指定します。

説明

エラー番号に対応するエラーメッセージ文字列を取得します。

・ Visual Basicの場合，toErrorMessageDARWIN関数を使用してください。

戻り値

文字列へのポインタを返します。

参照

`CDAQDARWIN::getErrorMessage`

getMaxLenAlarmNameDARWIN

構文

```
int getMaxLenAlarmNameDARWIN(void);
```

宣言

```
Public Declare Function getMaxLenAlarmNameDARWIN Lib
"DAQDARWIN" () As Long
```

説明

アラーム種類文字列の最大長を取得します。

・ 戻り値に終端は含まれません。

戻り値

文字列の長さを返します。

参照

CDAQDARWINDataInfo::getMaxLenAlarmName

getMaxLenErrorMessageDARWIN

構文

```
int getMaxLenErrorMessageDARWIN(void);
```

宣言

```
Public Declare Function getMaxLenErrorMessageDARWIN Lib
"DAQDARWIN" () As Long
```

説明

エラーメッセージ文字列の最大長を取得します。

・ 戻り値に終端は含まれません。

戻り値

文字列の長さを返します。

参照

`CDAQDARWIN::getMaxLenErrorMessage`

getReportStatusDARWIN

構文

```
int getReportStatusDARWIN(DAQDARWIN daqdarwin, int *
pReportStatus);
```

宣言

```
Public Declare Function getReportStatusDARWIN Lib "DAQDARWIN"
(ByVal daqdarwin As Long, ByRef pReportStatus As Long) As Long
```

引数

daqdarwin　　機器記述子を指定します。
pReportStatus　レポートステータスの返却先を指定します。

説明

レポートステータスを取得します。

・ トーカ機能でレポートステータスの出力を受信します。
・ 返却先が指定されていれば，レポートステータスを指定先に格納します。
・ 本関数は「通信インターフェイス」のTS，RFコマンドを実行します。

戻り値

エラー番号を返します。
エラー
Not descriptor　機器記述子がありません。

参照

`CDAQDARWIN::getReportStatus`

getRevisionAPIDARWIN

構文

```
const int getRevisionAPIDARWIN(void);
```

宣言

```
Public Declare Function getRevisionAPIDARWIN Lib "DAQDARWIN"
() As Long
```

説明

本APIのリビジョン番号を取得します。

戻り値

リビジョン番号を返します。

参照

CDAQDARWIN::getRevisionAPI

getSetDataByLineDARWIN

構文

```
int getSetDataByLineDARWIN(DAQDARWIN daqdarwin, char *
strLine, int maxLine, int * lenLine, int * pFlag);
```

宣言

```
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
```

引数

daqdarwin	機器記述子を指定します。
strLine	行単位の受信文字列を格納する領域を指定します。
maxLine	行単位の受信文字列を格納する領域のバイト数を指定します。
lenLine	実際に受信した文字列のバイト数の返却先を指定します。
pFlag	フラグの返却先を指定します。

説明

設定データを取得する宣言を実行した後，トーカ機能による出力を行単位で取得します。

- 改行を除いた受信文字列を格納します。
- 最終データを取得した場合，フラグにフラグステータスをセットします。また，エラーで終了した場合もフラグステータスをセットします。
- データ取得を終了するまで，他関数で通信を行わないでください。本関数でデータ取得中は，他の関数が正しく動作できません。

戻り値

エラー番号を返します。

エラー：

Not descriptor　機器記述子がありません。

参照

`CDAQDARWIN::getSetDataByLine`

getStatusByteDARWIN

構文

```
int getStatusByteDARWIN(DAQDARWIN daqdarwin, int *
pStatusByte);
```

宣言

```
Public Declare Function getStatusByteDARWIN Lib "DAQDARWIN"
(ByVal daqdarwin As Long, ByRef pStatusByte As Long) As Long
```

引数

daqdarwin　　機器記述子を指定します。
pStatusByte　ステータスバイトの返却先を指定します。

説明

ステータスの出力コマンド(ESC S)を送信し，ステータスバイトを取得します。
返却先が指定されていれば，ステータスバイトを整数値で指定先に格納します。

戻り値

エラー番号を返します。
エラー:
Not descriptor　機器記述子がありません。

参照

`CDAQDARWIN::getStatusByte`

getSystemConfigDARWIN

構文

```
int getSystemConfigDARWIN(DAQDARWIN daqdarwin, double *
interval, DarwinSystemInfo * pDarwinSystemInfo);
```

宣言

```
Public Declare Function getSystemConfigDARWIN Lib "DAQDARWIN"
(ByVal daqdarwin As Long, ByRef interval As Double, ByRef
pDarwinSystemInfo As DarwinSystemInfo) As Long
```

引数

daqdarwin	機器記述子を指定します。
interval	測定周期の返却先を指定します。
pDarwinSystemInfo	システム構成データの返却先を指定します。

説明

システム構成データを取得します。

- トーカでシステム構成データを受信します。
- 返却先が指定されていれば，測定周期とシステム構成データを指定先に格納します。
- 本関数は「通信インターフェイス」のTSとCFコマンドを実行します。

戻り値

エラー番号を返します。

エラー：

Not descriptor	機器記述子がありません。

参照

```
CDAQDARWIN::getSystemConfig
CDAQDARWINSysInfo::getDarwinSystemInfo
CDAQDARWINSysInfo::getInterval
```

getVersionAPIDARWIN

構文

```
const int getVersionAPIDARWIN(void);
```

宣言

```
Public Declare Function getVersionAPIDARWIN Lib "DAQDARWIN" ()
As Long
```

説明

本APIのバージョン番号を取得します。

戻り値

バージョン番号を返します。

参照

`CDAQDARWIN::getVersionAPI`

initSystemDARWIN

構文

```
int initSystemDARWIN(DAQDARWIN daqdarwin, int iCtrl);
```

宣言

```
Public Declare Function initSystemDARWIN Lib "DAQDARWIN"
(ByVal daqdarwin As Long, ByVal iCtrl As Long) As Long
```

引数

daqdarwin	機器記述子を指定します。
iCtrl	システム制御種類を指定します。

説明

指定されたシステム制御種類の動作を実行します。
本関数は「通信インターフェイス」のRS，RC，ARコマンドのいずれかを実行します。

戻り値

エラー番号を返します。
エラー：
Not descriptor　機器記述子がありません。

参照

`CDAQDARWIN::initSystem`

openDARWIN

構文

```
DAQDARWIN openDARWIN(const char * strAddress, int *
errorCode);
```

宣言

```
Public Declare Function openDARWIN Lib "DAQDARWIN" (ByVal
strAddress As String, ByRef errorCode As Long) As Long
```

引数

strAddress　　IPアドレスを文字列で指定します。
errorCode　　エラー番号の返却先を指定します。

説明

引数で指定されたIPアドレスの機器と通信接続をします。

- 機器記述子を作成し，戻り値として返却します。
- 返却先が指定されていれば，エラー番号を指定先に格納します。
- ポート番号は固定で通信用定数の「通信ポート番号」になります。
- 失敗した場合，Visual CではNULL，Visual Basicでは0を返します。

戻り値

機器記述子を返します。

エラー

Creating descriptor is failure　　機器記述子の作成に失敗しました。

参照

`CDAQDARWIN::open`

receiveByteDARWIN

構文

```
int receiveByteDARWIN(DAQDARWIN daqdarwin, unsigned char *
byteData, int maxData, int * lenData)
```

宣言

```
Public Declare Function receiveByteDARWIN Lib "DAQDARWIN"
(ByVal daqdarwin As Long, ByRef byteData As Byte, ByVal
maxData As Long, ByRef lenData As Long) As Long
```

引数

daqdarwin	機器記述子を指定します。
byteData	受信バイトデータを格納する領域を指定します。
maxData	受信データのバイト数を指定します。
lenData	実際に受信したデータのバイト数の返却先を指定します。

説明

引数で指定された領域に，バイト数分になるまで受信データを格納します。

- 返却先が指定されていれば，実際に受信したデータのバイト数を返します。
- 複数バイトのデータがある場合，本関数を繰り返し使用します。
- データ取得を終了するまで，他の関数で通信を行わないでください。本関数でデータ取得中は，他の関数が正しく動作できません。
- データ終了の判断は，ユーザが独自に行う必要があります。
- 機種特有のトーカ機能を実装する場合に，バイナリ出力を受信するのに使用します。

戻り値

エラー番号を返します。

エラー：

Not descriptor　機器記述子がありません。

参照

`CDAQDARWIN::receiveByte`

receiveLineDARWIN

構文

```
int receiveLineDARWIN(DAQDARWIN daqdarwin, char * strLine, int
maxLine, int * lenLine);
```

宣言

```
Public Declare Function receiveLineDARWIN Lib "DAQDARWIN"
(ByVal daqdarwin As Long, ByVal strLine As String, ByVal
maxLine As Long, ByRef lenLine As Long) As Long
```

引数

daqdarwin	機器記述子を指定します。
strLine	受信文字列を格納する領域を指定します。
maxLine	受信文字列を格納する領域のバイト数を指定します。
lenLine	実際に受信した文字列のバイト数の返却先を指定します。

説明

引数で指定された受信文字列を格納する領域に，改行を検出するまで，または，バイト数分になるまで受信します。

- 格納する領域には，改行を除いた受信文字列を格納します。
- 返却先が指定されていれば，実際に受信し格納した文字列のバイト数を指定先に格納します。
- 複数行のデータがある場合，本関数を繰り返し使用します。
- データ取得を終了するまで，他の関数で通信を行わないでください。本関数でデータ取得中は，他の関数が正しく動作できません。
- データ終了の判断は，ユーザが独自に行う必要があります。

戻り値

エラー番号を返します。

エラー：

Not descriptor　機器記述子がありません。

参照

`CDAQDARWIN::receiveLine`

reportingDARWIN

構文

```
int reportingDARWIN(DAQDARWIN daqdarwin, int iReportRun);
```

宣言

```
Public Declare Function reportingDARWIN Lib "DAQDARWIN" (ByVal
daqdarwin As Long, ByVal iReportRun As Long) As Long
```

引数

daqdarwin　　機器記述子を指定します。
iReportRun　　レポート実行種類を指定します。

説明

レポートのスタート/ストップを実行します。

・ レポートオプションがある場合に有効です。
・ 本関数は「通信インターフェイス」のDRコマンドを実行します。

戻り値

エラー番号を返します。
エラー：
Not descripto　機器記述子がありません。

参照

`CDAQDARWIN::reporting`

runCommandDARWIN

構文

```
int runCommandDARWIN(DAQDARWIN daqdarwin, const char *
strCmd);
```

宣言

```
Public Declare Function runCommandDARWIN Lib "DAQDARWIN"
(ByVal daqdarwin As Long, ByVal strCmd As String) As Long
```

引数

daqdarwin	機器記述子を指定します。
strCmd	送信するコマンドメッセージを文字列で指定します。

説明

指定されたコマンドメッセージとターミネータを送信し，応答を受信します。

- 送信時，本関数がコマンドメッセージにターミネータを付加するので，指定するコマンドメッセージにはターミネータを含まないでください。
- 複数コマンドの同時送信，ターミネータを含むコマンドメッセージには対応していません。
- トーカ機能のデータ出力要求コマンドのように，応答を返信しないコマンドには対応していません。

戻り値

エラー番号を返します。

エラー：

Not descriptor　機器記述子がありません。

参照

`CDAQDARWIN::runCommand`

sendLineDARWIN

構文

```
int sendLineDARWIN(DAQDARWIN daqdarwin, const char * strLine);
```

宣言

```
Public Declare Function sendLineDARWIN Lib "DAQDARWIN" (ByVal
daqdarwin As Long, ByVal strLine As String) As Long
```

引数

daqdarwin	機器記述子を指定します。
strLine	送信文字列を指定します。

説明

引数で指定された送信文字列を送信します。

- ターミネータを付加して送信します。
- 本関数は応答を受信しません。別途，受信のための関数で返信されるデータを受信してください。

戻り値

エラー番号を返します。

エラー：

Not descriptor　機器記述子がありません。

参照

`CDAQDARWIN::sendLine`

sendTriggerDARWIN

構文

```
int sendTriggerDARWIN(DAQDARWIN daqdarwin);
```

宣言

```
Public Declare Function sendTriggerDARWIN Lib "DAQDARWIN"
(ByVal daqdarwin As Long) As Long
```

引数

daqdarwin　　機器記述子を指定します。

説明

トリガコマンド(ESC T)を送信し，応答を受信します。

戻り値

エラー番号を返します。

エラー：

Not descriptor　機器記述子がありません。

参照

CDAQDARWIN::sendTrigger

setAlarmDARWIN

構文

```
int setAlarmDARWIN(DAQDARWIN daqdarwin, int levelNo, int
chType, int startChNo, int endChNo, int iAlarmType, int value,
int relayType, int relayNo);
```

宣言

```
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
```

引数

daqdarwin	機器記述子を指定します。
levelNo	アラームレベルを指定します。
chType	チャネルタイプを指定します。
startChNo	開始チャネル番号を指定します。
endChNo	終了チャネル番号を指定します。
iAlarmType	アラーム種類を指定します。
value	アラームのデータ値を指定します。
relayType	リレータイプを指定します。
relayNo	リレー番号を指定します。

説明

指定されたチャネル範囲(チャネルタイプ，開始チャネル番号，終了チャネル番号で指定)のチャネルに，指定されたアラーム(アラームレベル，アラーム種類)とアラーム値を設定します。

・ リレーの指定で，リレー番号が0以下の場合，リレーは指定されません(OFF)。
・ 本関数は「通信インターフェイス」のSAコマンドを実行します。

戻り値

エラー番号を返します。
エラー：
Not descriptor　機器記述子がありません。

参照

`CDAQDARWIN::setAlarm`

setDateTimeDARWIN

構文

```
int setDateTimeDARWIN(DAQDARWIN daqdarwin, DarwinDateTime *
pDarwinDateTime);
```

宣言

```
Public Declare Function setDateTimeDARWIN Lib "DAQDARWIN"
(ByVal daqdarwin As Long, ByRef pDarwinDateTime As
DarwinDateTime) As Long
```

引数

daqdarwin	機器記述子を指定します。
pDarwinDateTime	時刻情報データを指定します。

説明

機器本体に日付時刻を設定します。

- Visual Cの場合，引数の時刻情報データにNULLを指定すると，PCの現在の日付時刻を設定します。
- 本関数は「通信インターフェイス」のSDコマンドを実行します。

戻り値

エラー番号を返します。

エラー：

Not descriptor	機器記述子がありません。

参照

`CDAQDARWIN::setDateTime`

setDateTimeNowDARWIN

構文

```
int setDateTimeNowDARWIN(DAQDARWIN daqdarwin);
```

宣言

```
Public Declare Function setDateTimeNowDARWIN Lib "DAQDARWIN"
(ByVal daqdarwin As Long) As Long
```

引数

daqdarwin　　機器記述子を指定します。

説明

現在の日付時刻を設定します。

戻り値

エラー番号を返します。

参照

setDateTimeDARWIN

setDELTADARWIN

構文

```
int setDELTADARWIN(DAQDARWIN daqdarwin, int refChNo, int
chType, int startChNo, int endChNo, int spanMin, int spanMax);
```

宣言

```
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
```

引数

daqdarwin	機器記述子を指定します。
refChNo	基準チャネルのチャネル番号を指定します。
chType	チャネルタイプを指定します。
startChNo	開始チャネル番号を指定します。
endChNo	終了チャネル番号を指定します。
spanMin	スパンのレフト値を指定します。
spanMax	スパンのライト値を指定します。

説明

指定されたチャネル範囲(チャネルタイプ，開始チャネル番号，終了チャネル番号で指定)のチャネルに，指定された基準チャネルとの差演算を設定します。

- スパンの指定で，レフト値とライト値が等しい場合，省略されたものとみなします。
- 本関数は「通信インターフェイス」のSRコマンドを実行します。

戻り値

エラー番号を返します。

エラー：

Not descriptor　機器記述子がありません。

参照

`CDAQDARWIN::setDELTA`

setDIDARWIN

構文

```
int setDIDARWIN(DAQDARWIN daqdarwin, int iRangeDI, int chType,
int startChNo, int endChNo, int spanMin, int spanMax, int
scaleMin, int scaleMax, int scalePoint);
```

宣言

```
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
```

引数

daqdarwin	機器記述子を指定します。
iRangeDI	接点レンジを指定します。
chType	チャネルタイプを指定します。
startChNo	開始チャネル番号を指定します。
endChNo	終了チャネル番号を指定します。
spanMin	スパンのレフト値を指定します。
spanMax	スパンのライト値を指定します。
scaleMin	スケールのレフト値を指定します。
scaleMax	スケールのライト値を指定します。
scalePoint	スケールの小数点位置を指定します。

説明

指定されたチャネル範囲(チャネルタイプ，開始チャネル番号，終了チャネル番号で指定)のチャネルに，指定された接点レンジを設定します。

- スパン，スケールの指定で，レフト値とライト値が等しい場合，省略されたものとみなします。
- 本関数は「通信インターフェイス」のSRコマンドを実行します。

戻り値

エラー番号を返します。

エラー：

Not descriptor　機器記述子がありません。

参照

CDAQDARWIN::setDI

setMADARWIN

構文

```
int setMADARWIN(DAQDARWIN daqdarwin, int iRangeMA, int chType,
int startChNo, int endChNo, int spanMin, int spanMax, int
scaleMin, int scaleMax, int scalePoint);
```

宣言

```
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
```

引数

daqdarwin	機器記述子を指定します。
iRangeMA	直流電流レンジを指定します。
chType	チャネルタイプを指定します。
startChNo	開始チャネル番号を指定します。
endChNo	終了チャネル番号を指定します。
spanMin	スパンのレフト値を指定します。
spanMax	スパンのライト値を指定します。
scaleMin	スケールのレフト値を指定します。
scaleMax	スケールのライト値を指定します。
scalePoint	スケールの小数点位置を指定します。

説明

指定されたチャネル範囲(チャネルタイプ，開始チャネル番号，終了チャネル番号で指定)のチャネルに，指定された直流電流レンジを設定します。

- スパン，スケールの指定で，レフト値とライト値が等しい場合，省略されたものとみなします。
- 本関数は「通信インターフェイス」のSRコマンドを実行します。

戻り値

エラー番号を返します。

エラー：

Not descriptor　機器記述子がありません。

参照

CDAQDARWIN::setMA

setPOWERDARWIN

構文

```
int setPOWERDARWIN(DAQDARWIN daqdarwin, int iRangePOWER, int
chType, int chNo, int iItem, int iWire, int spanMin, int
spanMax, int scaleMin, int scaleMax, int scalePoint);
```

宣言

```
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
```

引数

daqdarwin	機器記述子を指定します。
iRangePOWER	パワーモニタレンジを指定します。
chType	チャネルタイプを指定します。
startChNo	チャネル番号を指定します。
iItem	パワー測定項目を指定します。
iWire	パワー接続方法指定します。
spanMin	スパンのレフト値を指定します。
spanMax	スパンのライト値を指定します。
scaleMin	スケールのレフト値を指定します。
scaleMax	スケールのライト値を指定します。
scalePoint	スケールの小数点位置を指定します。

説明

指定されたチャネル(チャネルタイプ，チャネル番号で指定)に，指定されたパワーレンジを設定します。

- スパン，スケールの指定で，レフト値とライト値が等しい場合，省略されたものとみなします。
- 本関数は「通信インターフェイス」のSRコマンドを実行します。

戻り値

エラー番号を返します。

エラー：

Not descriptor　機器記述子がありません。

参照

CDAQDARWIN::setPOWER

setPULSEDARWIN

構文

```
int setPULSEDARWIN(DAQDARWIN daqdarwin, int iRangePULSE, int
chType, int startChNo, int endChNo, int spanMin, int spanMax,
int scaleMin, int scaleMax, int scalePoint, int bFilter);
```

宣言

```
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
```

引数

daqdarwin	機器記述子を指定します。
iRangePULSE	パルスレンジを指定します。
chType	チャネルタイプを指定します。
startChNo	開始チャネル番号を指定します。
endChNo	終了チャネル番号を指定します。
spanMin	スパンのレフト値を指定します。
spanMax	スパンのライト値を指定します。
scaleMin	スケールのレフト値を指定します。
scaleMax	スケールのライト値を指定します。
scalePoint	スケールの小数点位置を指定します。
bFilter	フィルタを有効無効値で指定します。

説明

指定されたチャネル範囲(チャネルタイプ，開始チャネル番号，終了チャネル番号で指定)のチャネルに，指定されたパルスレンジを設定します。

- スパン，スケールの指定で，レフト値とライト値が等しい場合，省略されたものとみなします。
- 本関数は「通信インターフェイス」のSRコマンドを実行します。

戻り値

エラー番号を返します。

エラー：

Not descriptor　機器記述子がありません。

参照

`CDAQDARWIN::setPULSE`

setRRJCDARWIN

構文

```
int setRRJCDARWIN(DAQDARWIN daqdarwin, int refChNo, int
chType, int startChNo, int endChNo, int spanMin, int spanMax);
```

宣言

```
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
```

引数

daqdarwin	機器記述子を指定します。
refChNo	基準チャネルのチャネル番号を指定します。
chType	チャネルタイプを指定します。
startChNo	開始チャネル番号を指定します。
endChNo	終了チャネル番号を指定します。
spanMin	スパンのレフト値を指定します。
spanMax	スパンのライト値を指定します。

説明

指定されたチャネル範囲(チャネルタイプ，開始チャネル番号，終了チャネル番号で指定)のチャネルに，指定された基準チャネルとのリモートRJCを設定します。

- スパンの指定で，レフト値とライト値が等しい場合，省略されたものとみなします。
- 本関数は「通信インターフェイス」のSRコマンドを実行します。

戻り値

エラー番号を返します。

エラー：

Not descriptor　機器記述子がありません。

参照

`CDAQDARWIN::setRRJC`

setRTDDARWIN

構文

```
int setRTDDARWIN(DAQDARWIN daqdarwin, int iRangeRTD, int
chType, int startChNo, int endChNo, int spanMin, int spanMax,
int scaleMin, int scaleMax, int scalePoint);
```

宣言

```
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
```

引数

daqdarwin	機器記述子を指定します。
iRangeRTD	測温抵抗体レンジを指定します。
chType	チャネルタイプを指定します。
startChNo	開始チャネル番号を指定します。
endChNo	終了チャネル番号を指定します。
spanMin	スパンのレフト値を指定します。
spanMax	スパンのライト値を指定します。
scaleMin	スケールのレフト値を指定します。
scaleMax	スケールのライト値を指定します。
scalePoint	スケールの小数点位置を指定します。

説明

指定されたチャネル範囲(チャネルタイプ，開始チャネル番号，終了チャネル番号で指定)のチャネルに，指定された測温抵抗体レンジを設定します。

- スパン，スケールの指定で，レフト値とライト値が等しい場合，省略されたものとみなします。
- 本関数は「通信インターフェイス」のSRコマンドを実行します。

戻り値

エラー番号を返します。

エラー：

Not descriptor　機器記述子がありません。

参照

CDAQDARWIN::setRTD

setScallingUnitDARWIN

構文

```
int setScallingUnitDARWIN(DAQDARWIN daqdarwin, const char *
strUnit, int chType, int startChNo, int endChNo);
```

宣言

```
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 111
```

引数

daqdarwin	機器記述子を指定します。
strUnit	単位名を文字列で指定します。
chType	チャネルタイプを指定します。
startChNo	開始チャネル番号を指定します。
endChNo	終了チャネル番号を指定します。

説明

指定されたチャネル範囲(チャネルタイプ，開始チャネル番号，終了チャネル番号で指定)のチャネルに，指定された単位を設定します。
本関数は「通信インターフェイス」のSNコマンドを実行します。

戻り値

エラー番号を返します。
エラー：
Not descriptor　機器記述子がありません。

参照

`CDAQDARWIN::setScallingUnit`

setSKIPDARWIN

構文

```
int setSKIPDARWIN(DAQDARWIN daqdarwin, int chType, int
startChNo, int endChNo);
```

宣言

```
Public Declare Function setSKIPDARWIN Lib "DAQDARWIN" (ByVal
daqdarwin As Long, ByVal chType As Long, ByVal startChNo As
Long, ByVal endChNo As Long) As Long
```

引数

daqdarwin	機器記述子を指定します。
chType	チャネルタイプを指定します。
startChNo	開始チャネル番号を指定します。
endChNo	終了チャネル番号を指定します。

説明

指定されたチャネル範囲(チャネルタイプ，開始チャネル番号，終了チャネル番号で指定)のチャネルをスキップ(未使用)に設定します。
本関数は「通信インターフェイス」のSRコマンドを実行します。

戻り値

エラー番号を返します。
エラー：
Not descriptor　機器記述子がありません。

参照

`CDAQDARWIN::setSKIP`

setSTRAINDARWIN

構文

```
int setSTRAINDARWIN(DAQDARWIN daqdarwin, int iRangeSTRAIN, int
chType, int startChNo, int endChNo, int spanMin, int spanMax,
int scaleMin, int scaleMax, int scalePoint);
```

宣言

```
Public Declare Function setSTRAINDARWIN 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
```

引数

daqdarwin	機器記述子を指定します。
iRangeSTRAIN	ひずみ入力レンジを指定します。
chType	チャネルタイプを指定します。
startChNo	開始チャネル番号を指定します。
endChNo	終了チャネル番号を指定します。
spanMin	スパンのレフト値を指定します。
spanMax	スパンのライト値を指定します。
scaleMin	スケールのレフト値を指定します。
scaleMax	スケールのライト値を指定します。
scalePoint	スケールの小数点位置を指定します。

説明

指定されたチャネル範囲(チャネルタイプ，開始チャネル番号，終了チャネル番号で指定)のチャネルに，指定されたひずみレンジを設定します。

スパン，スケールの指定で，レフト値とライト値が等しい場合，省略されたものとみなします。

本関数は「通信インターフェイス」のSRコマンドを実行します。

戻り値

エラー番号を返します。

エラー：

Not descriptor　機器記述子がありません。

参照

CDAQDARWIN::setSTRAIN

setTCDARWIN

構文

```
int setTCDARWIN(DAQDARWIN daqdarwin, int iRangeTC, int chType,
int startChNo, int endChNo, int spanMin, int spanMax, int
scaleMin, int scaleMax, int scalePoint);
```

宣言

```
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
```

引数

daqdarwin	機器記述子を指定します。
iRangeTC	熱電対レンジを指定します。
chType	チャネルタイプを指定します。
startChNo	開始チャネル番号を指定します。
endChNo	終了チャネル番号を指定します。
spanMin	スパンのレフト値を指定します。
spanMax	スパンのライト値を指定します。
scaleMin	スケールのレフト値を指定します。
scaleMax	スケールのライト値を指定します。
scalePoint	スケールの小数点位置を指定します。

説明

指定されたチャネル範囲(チャネルタイプ，開始チャネル番号，終了チャネル番号)のチャネルに，指定された熱電対レンジを設定します。

- スパン，スケールの指定で，レフト値とライト値が等しい場合，省略されたものとみなします。
- 本関数は「通信インターフェイス」のSRコマンドを実行します。

戻り値

エラー番号を返します。

エラー：

Not descriptor　機器記述子がありません。

参照

`CDAQDARWIN::setTC`

setTimeOutDARWIN

構文

```
int setTimeOutDARWIN(DAQDARWIN daqdarwin, int seconds);
```

宣言

```
Public Declare Function setTimeOutDARWIN Lib "DAQDARWIN"
(ByVal daqdarwin As Long, ByVal seconds As Long) As Long
```

引数

daqdarwin	機器記述子を指定します。
seconds	通信のタイムアウト値を秒単位で指定します。

説明

機器との通信に対して，タイムアウトを設定します。

- 指定された値が負の場合，タイムアウトを無効にします。
- 使用を推奨しません。

戻り値

エラー番号を返します。

エラー：

Not descriptor　機器記述子がありません。

参照

`CDAQDARWIN::setTimeOut`

setVOLTDARWIN

構文

```
int setVOLTDARWIN(DAQDARWIN daqdarwin, int iRangeVOLT, int
chType, int startChNo, int endChNo, int spanMin, int spanMax,
int scaleMin, int scaleMax, int scalePoint);
```

宣言

```
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
```

引数

daqdarwin	機器記述子を指定します。
iRangeVOLT	直流電圧レンジを指定します。
chType	チャネルタイプを指定します。
startChNo	開始チャネル番号を指定します。
endChNo	終了チャネル番号を指定します。
spanMin	スパンのレフト値を指定します。
spanMax	スパンのライト値を指定します。
scaleMin	スケールのレフト値を指定します。
scaleMax	スケールのライト値を指定します。
scalePoint	スケールの小数点位置を指定します。

説明

指定されたチャネル範囲(チャネルタイプ，開始チャネル番号，終了チャネル番号で指定)のチャネルに，指定された直流電圧レンジを設定します。

- スパン，スケールの指定で，レフト値とライト値が等しい場合，省略されたものとみなします。
- 本関数は「通信インターフェイス」のSRコマンドを実行します。

戻り値

エラー番号を返します。

エラー：

Not descriptor　機器記述子がありません。

参照

`CDAQDARWIN::setVOLT`

talkCalibrationDataDARWIN

構文

```
int talkCalibrationDataDARWIN(DAQDARWIN daqdarwin, int
startChType, int startChNo, int endChType, int endChNo);
```

宣言

```
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
```

引数

daqdarwin	機器記述子を指定します。
startChType	開始チャネルタイプを指定します。
startChNo	開始チャネル番号を指定します。
endChType	終了チャネルタイプを指定します。
endChNo	終了チャネル番号を指定します。

説明

開始チャネル(開始チャネルタイプ，開始チャネル番号)から終了チャネル(終了チャネルタイプ，終了チャネル番号)までのA/D校正モードの設定データを取得する宣言を実行します。

- 操作モードを，「A/D校正モード」に切り替えておく必要があります。
- 本関数は「通信インターフェイス」のTSとLFコマンドを実行します。
- 本関数の実行後，行単位のデータ取得には，getSetDataByLineDARWIN関数を使用します。

戻り値

エラー番号を返します。

エラー：

Not descriptor　機器記述子がありません。

参照

`CDAQDARWIN::talkCalibrationData`

talkChInfoDARWIN

構文

```
int talkChInfoDARWIN(DAQDARWIN daqdarwin, int startChType, int
startChNo, int endChType, int endChNo);
```

宣言

```
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
```

引数

daqdarwin	機器記述子を指定します。
startChType	開始チャネルタイプを指定します。
startChNo	開始チャネル番号を指定します。
endChType	終了チャネルタイプを指定します。
endChNo	終了チャネル番号を指定します。

説明

開始チャネル(開始チャネルタイプ，開始チャネル番号)から終了チャネル(終了チャネルタイプ，終了チャネル番号)までのチャネル情報データを取得する宣言を実行します。

- 本関数は「通信インターフェイス」のTSとLFコマンドを実行します。
- 本関数の実行後，チャネルごとのデータ取得には，getChInfoDARWIN関数を使用します。

戻り値

エラー番号を返します。

エラー：

Not descriptor　機器記述子がありません。

参照

`CDAQDARWIN::talkChInfo`

talkDataByASCIIDARWIN

構文

```
int talkDataByASCIIDARWIN(DAQDARWIN daqdarwin, int
startChType, int startChNo, int endChType, int endChNo,
DarwinDateTime * pDarwinDateTime);
```

宣言

```
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
```

引数

daqdarwin	機器記述子を指定します。
startChType	開始チャネルタイプを指定します。
startChNo	開始チャネル番号を指定します。
endChType	終了チャネルタイプを指定します。
endChNo	終了チャネル番号を指定します。
pDarwinDateTime	時刻情報データの返却先を指定します。

説明

開始チャネル(開始チャネルタイプ，開始チャネル番号)から終了チャネル(終了チャネルタイプ，終了チャネル番号)までの測定データをASCIIフォーマットで取得する宣言を実行します。

- 返却先が指定されていれば，測定データの時刻情報を指定先に格納します。
- 測定チャネルと演算チャネルは，別々に指定してください。
- 本関数は「通信インターフェイス」のTSとFMコマンドを実行します。
- 本関数の実行後，チャネル毎のデータ取得には，getChDataByASCIIDARWIN関数を使用します。

戻り値

エラー番号を返します。

エラー：

Not descriptor	機器記述子がありません。

参照

```
CDAQDARWIN::talkDataByASCII
CDAQDARWINDateTime::getDarwinDateTime
```

talkDataByBinaryDARWIN

構文

```
int talkDataByBinaryDARWIN(DAQDARWIN daqdarwin, int
startChType, int startChNo, int endChType, int endChNo,
DarwinDateTime * pDarwinDateTime);
```

宣言

```
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
```

引数

daqdarwin	機器記述子を指定します。
startChType	開始チャネルタイプを指定します。
startChNo	開始チャネル番号を指定します。
endChType	終了チャネルタイプを指定します。
endChNo	終了チャネル番号を指定します。
pDarwinDateTime	時刻情報データの返却先を指定します。

説明

開始チャネル(開始チャネルタイプ，開始チャネル番号)から終了チャネル(終了チャネルタイプ，終了チャネル番号)までの測定データをバイナリフォーマットで取得する宣言を実行します。

- 返却先が指定されていれば，測定データの時刻情報データを指定先に格納します。
- 測定チャネルと演算チャネルは，別々に指定してください。
- バイト出力順序は，MSBに指定されます。
- 本関数は「通信インターフェイス」のTS，FMコマンドを実行します。
- 本関数の実行後，チャネルごとのデータ取得には，getChDataByBinaryDARWIN関数を使用します。

戻り値

エラー番号を返します。

エラー：

Not descriptor	機器記述子がありません。

参照

```
CDAQDARWIN::talkDataByBinary
CDAQDARWINDateTime::getDarwinDateTime
```

talkOperationDataDARWIN

構文

```
int talkOperationDataDARWIN(DAQDARWIN daqdarwin, int
startChType, int startChNo, int endChType, int endChNo);
```

宣言

```
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
```

引数

daqdarwin	機器記述子を指定します。
startChType	開始チャネルタイプを指定します。
startChNo	開始チャネル番号を指定します。
endChType	終了チャネルタイプを指定します。
endChNo	終了チャネル番号を指定します。

説明

開始チャネル(開始チャネルタイプ，開始チャネル番号)から終了チャネル(終了チャネルタイプ，終了チャネル番号)までの運転モードの設定データを取得する宣言を実行します。

- 本関数は「通信インターフェイス」のTSとLFコマンドを実行します。
- 本関数の実行後，行単位毎のデータ取得には，getSetDataByLineDARWIN関数を使用します。

戻り値

エラー番号を返します。

エラー：

Not descriptor　機器記述子がありません。

参照

`CDAQDARWIN::talkOperationData`

talkSetupDataDARWIN

構文

```
int talkSetupDataDARWIN(DAQDARWIN daqdarwin, int startChType,
int startChNo, int endChType, int endChNo);
```

宣言

```
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
```

引数

daqdarwin	機器記述子を指定します。
startChType	開始チャネルタイプを指定します。
startChNo	開始チャネル番号を指定します。
endChType	終了チャネルタイプを指定します。
endChNo	終了チャネル番号を指定します。

説明

開始チャネル(開始チャネルタイプ，開始チャネル番号)から終了チャネル(終了チャネルタイプ，終了チャネル番号)までのセットアップモードの設定データを取得する宣言を実行します。

- 「通信インターフェイス」のTSとLFコマンドを実行します。
- 本関数の実行後，行単位毎のデータ取得には，getSetDataByLineDARWIN関数を使用します。

戻り値

エラー番号を返します。

エラー：

Not descriptor　機器記述子がありません。

参照

`CDAQDARWIN::talkSetupData`

toAlarmNameDARWIN

構文

```
int toAlarmNameDARWIN(int iAlarmType, char * strAlarm, int
lenAlarm);
```

宣言

```
Public Declare Function toAlarmNameDARWIN Lib "DAQDARWIN"
(ByVal iAlarmType As Long, ByVal strAlarm As String, ByVal
lenAlarm As Long) As Long
```

引数

iAlarmType	アラーム種類を指定します。
strAlarm	文字列を格納する領域を指定します。
lenAlarm	文字列を格納する領域のバイト数を指定します。

説明

指定されたアラーム種類に対応する文字列を，指定された領域に格納します。

- 領域に格納する文字列には，終端(NULL)も含まれます。
- 実際の文字列の長さが戻り値になります。戻り値に終端は含まれません。

戻り値

文字列の長さを返します。

参照

`getAlarmNameDARWIN`

toDoubleValueDARWIN

構文

```
double toDoubleValueDARWIN(int dataValue, int point);
```

宣言

```
Public Declare Function toDoubleValueDARWIN Lib "DAQDARWIN"
(ByVal dataValue As Long, ByVal point As Long) As Double
```

引数

dataValue	データ値を指定します。
point	小数点位置を指定します。

説明

指定されたデータ値と小数点位置から測定値を生成します。

戻り値

測定値を倍精度浮動小数で返します。

参照

`CDAQDARWINDataInfo::toDoubleValue`

toErrorMessageDARWIN

構文

```
int toErrorMessageDARWIN(int errCode, char * errStr, int
errLen);
```

宣言

```
Public Declare Function toErrorMessageDARWIN Lib "DAQDARWIN"
(ByVal errCode As Long, ByVal errStr As String, ByVal errLen
As Long) As Long
```

引数

errCode	エラー番号を指定します。
errStr	文字列を格納する領域を指定します。
errLen	文字列を格納する領域のバイト数を指定します。

説明

エラー番号に対応するエラーメッセージ文字列を，指定された領域に格納します。

- 領域に格納する文字列には，終端(NULL)も含まれます。
- 実際の文字列の長さが戻り値になります。戻り値に終端は含まれません。

戻り値

文字列の長さを返します。

参照

`getErrorMessageDARWIN`

toStringValueDARWIN

構文

```
int toStringValueDARWIN(int dataValue, int point, char *
strValue, int lenValue);
```

宣言

```
Public Declare Function toStringValueDARWIN Lib "DAQDARWIN"
(ByVal dataValue As Long, ByVal point As Long, ByVal strValue
As String, ByVal lenValue As Long) As Long
```

引数

dataValue	データ値を指定します。
point	小数点位置を指定します。
strValue	文字列を格納する領域を指定します。
lenValue	文字列を格納する領域のバイト数を指定します。

説明

指定されたデータ値と小数点位置から測定値を生成します。

- 生成された測定値を文字列に変換して，指定された領域に格納します。
- 領域に格納する文字列には，終端(NULL)も含まれます。
- 実際の文字列の長さが戻り値になります。戻り値に終端は含まれません。

戻り値

文字列の長さを返します。

参照

`CDAQDARWINDataInfo::toStringValue`

transModeDARWIN

構文

```
int transModeDARWIN(DAQDARWIN daqdarwin, int iMode);
```

宣言

```
Public Declare Function transModeDARWIN Lib "DAQDARWIN" (ByVal
daqdarwin As Long, ByVal iMode As Long) As Long
```

引数

daqdarwin　　機器記述子を指定します。
iMode　　操作モードを指定します。

説明

指定された操作モードに切り替えます。

・ 本関数は「通信インターフェイス」のDSコマンドを実行します。

戻り値

エラー番号を返します。
エラー：
Not descriptor　機器記述子がありません。

参照

`CDAQDARWIN::transMode`

11.1 DARWINの定数の概要

以下の種類の定数が用意されています。定数は，Visual C++，Visual C，Visual Basicで共通です。

種類	説明	ページ
通信用定数	DARWINの通信ポート番号	11-2
個数値	サブユニット数など	11-2
最大値	チャネル名文字列最大長など	11-2
文字列	ターミネータ文字列	11-2
有効無効値	有効(ON)設定，無効(OFF)設定	11-2
フラグステータス	データ取得時に最終データを判別	11-3
データステータス値	測定データの状態	11-3
アラーム種類	上限アラームなど	11-3
システム制御種類	システム制御操作	11-3
チャネルタイプ/リレータイプ	チャネルやリレーの種類	11-4
操作モード	運転，セットアップ，A/D校正モード	11-4
トーカ機能種類	出力データに対応するトーカ	11-4
ステータスバイト	各種状態	11-5
セットアップ確定	破棄，確定	11-5
ユニット番号	拡張モデル，スタンドアロンモデル	11-5
演算処理	演算のスタート，ストップ，クリアなど	11-5
レポート実行種類	レポートのスタート，ストップ	11-5
レポート種類	時報，日報，月報，ステータス	11-6
レポートステータス	レポート種類の全無効，最新情報，有効	11-6
直流電圧レンジ	20mVレンジなど	11-6
熱電対レンジ	Type Rなど	11-6
測温抵抗体レンジ	Pt100: 1mAなど	11-7
接点入力(DI)レンジ	電圧入力または接点入力	11-7
ひずみ入力レンジ	2k，20k，200k	11-7
パルスレンジ	GATE，RATE	11-7
パワーモニタレンジ	25V 0.5A，25V 5A，250V 0.5A，250V 5A	11-8
直流電流レンジ	20mA	11-8
パワー接続方法	単相2線式など	11-8
パワー測定項目	実効電流1など	11-9

11.2 DARWINの定数

定数のニーモニックと意味を説明しています。DARWINの用語については， 付録2をご覧ください。

通信用定数

ニーモニック	内容
DAQDARWIN_COMMPORT	DARWINの通信ポート番号です。

個数値

個数値には，対象となるモジュールやユニットのそれぞれの個数値が設定されます。

ニーモニック	内容
DAQDARWIN_NUMCHANNEL	チャネル個数です。
DAQDARWIN_NUMALARM	アラーム個数です。
DAQDARWIN_NUMUNIT	サブユニット個数です。
DAQDARWIN_NUMSLOT	サブユニットごとのスロット個数です。
DAQDARWIN_NUMTERM	スロット(モジュール)ごとの端子個数です。

最大値

ニーモニック	内容
DAQDARWIN_MAXCHNAMELEN	チャネル名文字列最大長です。
DAQDARWIN_MAXCHRANGLEN	チャネル範囲名文字列最大長です。
DAQDARWIN_MAXUNITLEN	単位名文字列最大長です。
DAQDARWIN_MAXMODULELEN	モジュール名文字列最大長です。
DAQDARWIN_MAXRELAYLEN	リレー名文字列最大長です。チャネル名文字列最大長と同じです。リレーとは，アラーム出力モジュールまたはDI/DOモジュールの出力リレーのことです。
DAQDARWIN_MAXDECIMALPOINT	小数点位置の最大値です。

文字列の最大長は，終端(NULL)を含みません。

文字列

ニーモニック	内容
DAQDARWIN_TERMINATE	ターミネータ文字列です。

有効無効値

ニーモニック	内容
DAQDARWIN_VALID_OFF	無効(OFF)値
DAQDARWIN_VALID_ON	有効(ON)値

フラグステータス

ニーモニック	内容
DAQDARWIN_FLAG_OFF	全OFF。
DAQDARWIN_FLAG_ENDDATA	ASCIIコードや行単位で取得するデータ行が最終データです。

論理OR演算で合成できます。

データステータス値

ニーモニック	内容
DAQDARWIN_UNKNWON	データステータスがセットされていない状態です。
DAQDARWIN_DATA_NORMAL	正常です。
DAQDARWIN_DATA_DIFFINPUT	チャネル間差演算状態です。
DAQDARWIN_DATA_PLUSOVER	プラスオーバ状態です。
DAQDARWIN_DATA_MINUSOVER	マイナスオーバ状態です。
DAQDARWIN_DATA_SKIP	スキップ(未使用)状態です。
DAQDARWIN_DATA_ILLEGAL	不明な不正データ状態です。
DAQDARWIN_DATA_ABNORMAL	異常データ状態です。
DAQDARWIN_DATA_NODATA	データなし状態です。
DAQDARWIN_DATA_READER	瞬時値データ読み込み通信時の状態です。

瞬時値データ読み込み通信時の状態は，瞬時値データ読み込み通信ポート使用時，チャネル情報データ取得によるチャネルステータスです。

アラーム種類

□はスペースを示します。

ニーモニック	内容	文字列
DAQDARWIN_ALARM_NONE	アラームなし(アラームOFF)	□□
DAQDARWIN_ALARM_UPPER	上限アラーム	H□
DAQDARWIN_ALARM_LOWER	下限アラーム	L□
DAQDARWIN_ALARM_UPDIFF	差上限アラーム	dH
DAQDARWIN_ALARM_LOWDIFF	差下限アラーム	dL
DAQDARWIN_ALARM_INCRATE	変化率上昇限アラーム	RH
DAQDARWIN_ALARM_DECRATE	変化率下降限アラーム	RL

システム制御種類

システム制御動作を指定するときに使用できます。

ニーモニック	内容
DAQDARWIN_SYSTEM_RECONSTRUCT	システム再構築
DAQDARWIN_SYSTEM_INITOPE	運転モードの設定初期化
DAQDARWIN_SYSTEM_RESETALARM	アラームリセット

チャネル/リレータイプ

チャネル，リレー，通信入力，演算定数のタイプ値です。チャネルやリレーを指定するときに使用できます。記述を簡略にするために，1文字による定義もあります。

ニーモニック	1文字	内容	チャネル	リレー
DAQDARWIN_CHTYPE_MAINUNIT	I	拡張モデルのメインユニットを表す値です。	-	あり
DAQDARWIN_CHTYPE_STANDALONE		スタンドアロンモデルのユニットを表す値です。サブユニット番号の0と同じです。	あり	あり
DAQDARWIN_CHTYPE_MATHTYPE	A	演算チャネルを表す値です。	あり	-
DAQDARWIN_CHTYPE_SWITCH	S	内部スイッチを表す値です。	-	あり
DAQDARWIN_CHTYPE_COMMDATA	C	通信入力を表す値です。	-	-
DAQDARWIN_CHTYPE_CONSTANT	K	演算定数を表す値です。	-	-
DAQDARWIN_CHTYPE_REPORT	R	レポートを表す値です。	-	-

あり　：そのタイプのチャネル/リレーが存在します。
-　　：そのタイプのチャネル/リレーはありません。

Note

拡張モデルに接続されるサブユニットを識別するサブユニット番号もタイプ値です。サブユニット番号は0から5の整数値です。付録2を参照してください。

操作モード

ニーモニック	内容
DAQDARWIN_MODE_OPE	運転モード
DAQDARWIN_MODE_SETUP	セットアップモード
DAQDARWIN_MODE_CALIB	A/D校正モード

トーカ機能種類

ニーモニック	内容
DAQDARWIN_TALK_MEASUREDDATA	測定データ，演算データの出力
DAQDARWIN_TALK_OPEDATA	運転モードの設定データの出力
DAQDARWIN_TALK_CHINFODATA	チャネル情報データの出力
DAQDARWIN_TALK_SYSINFODATA	システム構成データの出力
DAQDARWIN_TALK_CALIBDATA	校正データ(A/D校正モードの設定データ)の出力
DAQDARWIN_TALK_SETUPDATA	セットアップモードの設定データの出力
DAQDARWIN_TALK_REPORTDATA	レポートステータスの出力

ステータスバイト値

論理OR演算で合成できます。

ニーモニック	内容
DAQDARWIN_STATUS_OFF	全ステータスバイトが無効の場合の値
DAQDARWIN_STATUS_ADCONV	A/D変換終了
DAQDARWIN_STATUS_SYNTAX	コマンド文法エラー
DAQDARWIN_STATUS_TIMER	内部タイマ起動/レポート作成
DAQDARWIN_STATUS_MEDIA	メディアへのアクセス(DC100)
DAQDARWIN_STATUS_RELEASE	演算中の測定抜け
DAQDARWIN_STATUS_ALL	全ステータスバイトを有効にするマスク値
DAQDARWIN_STATUS_SRQ	SRQ

ステータスバイト値の意味の詳細については，DARWIN機器の通信インターフェースユーザーズマニュアルをご覧ください。

セットアップ確定

ニーモニック	内容
DAQDARWIN_SETUP_ABORT	破棄
DAQDARWIN_SETUP_STORE	確定

ユニット番号

ニーモニック	内容
DAQDARWIN_UNITNO_MAINUNIT	拡張モデルのメインユニット
DAQDARWIN_UNITNO_STANDALONE	スタンドアロンモデルのユニット

サブユニット番号は，数値です。 チャネル/リレータイプを参照。

演算処理

ニーモニック	内容
DAQDARWIN_COMPUTE_START	演算のスタート
DAQDARWIN_COMPUTE_STOP	演算のストップ
DAQDARWIN_COMPUTE_RESTART	演算データクリア後，再度スタート
DAQDARWIN_COMPUTE_CLEAR	演算データクリア
DAQDARWIN_COMPUTE_RELEASE	測定抜けのステータス表示解除

レポート実行種類

ニーモニック	内容
DAQDARWIN_REPORT_RUN_START	レポートのスタート
DAQDARWIN_REPORT_RUN_STOP	レポートのストップ

レポート種類

ニーモニック	内容
DAQDARWIN_REPORT_HOURLY	時報
DAQDARWIN_REPORT_DAILY	日報
DAQDARWIN_REPORT_MONTHLY	月報
DAQDARWIN_REPORT_STATUS	ステータス

レポートステータス

論理OR演算で合成できます。

ニーモニック	内容
DAQDARWIN_REPSTATUS_NONE	全無効
DAQDARWIN_REPSTATUS_HOURLY_NEW	最新時報
DAQDARWIN_REPSTATUS_HOURLY_VALID	時報の有効
DAQDARWIN_REPSTATUS_DAILY_NEW	最新日報
DAQDARWIN_REPSTATUS_DAILY_VALID	日報の有効
DAQDARWIN_REPSTATUS_MONTHLY_NEW	最新月報
DAQDARWIN_REPSTATUS_MONTHLY_VALID	月報の有効

直流電圧レンジ

ニーモニック	内容	設定範囲
DAQDARWIN_RANGE_VOLT_20MV	20mV	−20.000～20.000 mV
DAQDARWIN_RANGE_VOLT_60MV	60mV	−60.00～60.00 mV
DAQDARWIN_RANGE_VOLT_200MV	200mV	−200.00～200.00 mV
DAQDARWIN_RANGE_VOLT_2V	2V	−2.0000～2.0000 V
DAQDARWIN_RANGE_VOLT_6V	6V	−6.000～6.000 V
DAQDARWIN_RANGE_VOLT_20V	20V	−20.000～20.000 V
DAQDARWIN_RANGE_VOLT_50V	50V	−50.00～50.00 V

熱電対レンジ

ニーモニック	内容	設定範囲
DAQDARWIN_RANGE_TC_R	R	0.0～1760.0 ℃
DAQDARWIN_RANGE_TC_S	S	0.0～1760.0 ℃
DAQDARWIN_RANGE_TC_B	B	0.0～1820.0 ℃
DAQDARWIN_RANGE_TC_K	K	−200.0～1370.0 ℃
DAQDARWIN_RANGE_TC_E	E	−200.0～800.0 ℃
DAQDARWIN_RANGE_TC_J	J	−200.0～1100.0 ℃
DAQDARWIN_RANGE_TC_T	T	−200.0～400.0 ℃
DAQDARWIN_RANGE_TC_N	N	0.0～1300.0 ℃
DAQDARWIN_RANGE_TC_W	W	0.0～2315.0 ℃
DAQDARWIN_RANGE_TC_L	L	−200.0～900.0 ℃
DAQDARWIN_RANGE_TC_U	U	−200.0～400.0 ℃
DAQDARWIN_RANGE_TC_KP	KpAu7Fe	0.0～300.0 K

測温抵抗体レンジ

ニーモニック	内容	設定範囲
DAQDARWIN_RANGE_RTD_1MAPT	Pt100:1mA	−200.0～600.0 ℃
DAQDARWIN_RANGE_RTD_2MAPT	Pt100:2mA	−200.0～250.0 ℃
DAQDARWIN_RANGE_RTD_1MAJPT	JPt100:1mA	−200.0～550.0 ℃
DAQDARWIN_RANGE_RTD_2MAJPT	JPt100:2mA	−200.0～250.0 ℃
DAQDARWIN_RANGE_RTD_2MAPT50	Pt50:2mA	−200.0～550.0 ℃
DAQDARWIN_RANGE_RTD_1MAPTH	Pt100:1mA−H	−140.00～150.00 ℃
DAQDARWIN_RANGE_RTD_2MAPTH	Pt100:2mA−H	−70.00～70.00 ℃
DAQDARWIN_RANGE_RTD_1MAJPTH	JPt100:1mA−H	−140.00～150.00 ℃
DAQDARWIN_RANGE_RTD_2MAJPTH	JPt100:2mA−H	−70.00～70.00 ℃
DAQDARWIN_RANGE_RTD_1MANIS	Ni100:1mA−S	−200.0～250.0 ℃
DAQDARWIN_RANGE_RTD_1MANID	Ni100:1mA−D	−60.0～180.0 ℃
DAQDARWIN_RANGE_RTD_1MANI120	Ni120:1mA	−70.0～200.0 ℃
DAQDARWIN_RANGE_RTD_CU10GE	Cu10:GE	−200.0～300.0 ℃
DAQDARWIN_RANGE_RTD_CU10LN	Cu10:L&N	−200.0～300.0 ℃
DAQDARWIN_RANGE_RTD_CU10WEED	Cu10:WEED	−200.0～300.0 ℃
DAQDARWIN_RANGE_RTD_CU10BAILEY	Cu10:BAILEY	−200.0～300.0 ℃
DAQDARWIN_RANGE_RTD_J263B	J263*B	−0.0～300.0 K

接点入力(DI)レンジ

ニーモニック	内容	設定範囲
DAQDARWIN_RANGE_DI_LEVEL	電圧入力	0：2.4V未満，1：2.4V以上
DAQDARWIN_RANGE_DI_CONTACT	接点入力	0：open, 1：close

ひずみ入力レンジ

ニーモニック	内容	設定範囲
DAQDARWIN_RANGE_STRAIN_2K	2k	−2000～2000μひずみ(1ゲージ法) −1000～1000μひずみ(2ゲージ法) −500～500μひずみ(4ゲージ法)
DAQDARWIN_RANGE_STRAIN_20K	20k	−20000～20000μひずみ(1ゲージ法) −10000～10000μひずみ(2ゲージ法) −5000～5000μひずみ(4ゲージ法)
DAQDARWIN_RANGE_STRAIN_200K	200k	−200000～200000μひずみ(1ゲージ法) −100000～100000μひずみ(2ゲージ法) −50000～50000μひずみ(4ゲージ法)

パルスレンジ

ニーモニック	内容	設定範囲
DAQDARWIN_RANGE_PULSE_RATE	RATE	0 ～ 30000
DAQDARWIN_RANGE_PULSE_GATE	GATE	0 ～ 30000

パワーモニタレンジ

ニーモニック	内容	設定範囲
DAQDARWIN_RANGE_POWER_25V05A	25V 0.5A	電圧25V，電流0.5A
DAQDARWIN_RANGE_POWER_25V5A	25V 5A	電圧25V，電流5A
DAQDARWIN_RANGE_POWER_250V05A	250V 0.5A	電圧250V，電流0.5A
DAQDARWIN_RANGE_POWER_250V5A	250V 5A	電圧250V，電流5A

直流電流レンジ

ニーモニック	内容	設定範囲
DAQDARWIN_RANGE_MA_20MA	20mA	−20.000～20.000mA

パワー接続方法

ニーモニック	内容
DAQDARWIN_WIRE_1PH2W	単相2線式
DAQDARWIN_WIRE_1PH3W	単相3線式(3線式用だけ)
DAQDARWIN_WIRE_3PH3W2I	3相3線式(2電圧2電流 3線式用だけ)
DAQDARWIN_WIRE_3PH3W3I	3相3線式(3電圧3電流 3線式用だけ)
DAQDARWIN_WIRE_3PH4W	3相4線式(3線式用だけ)

パワー測定項目

ニーモニック	内容
DAQDARWIN_POWERITEM_I0	(I1+I2+I3)/3
DAQDARWIN_POWERITEM_I1	実効電流1
DAQDARWIN_POWERITEM_I2	実効電流2
DAQDARWIN_POWERITEM_I3	実効電流3
DAQDARWIN_POWERITEM_I13	(I1+I3)/2
DAQDARWIN_POWERITEM_P0	P1+P2+P3
DAQDARWIN_POWERITEM_P1	有効電力1
DAQDARWIN_POWERITEM_P2	有効電力2
DAQDARWIN_POWERITEM_P3	有効電力3
DAQDARWIN_POWERITEM_P13	P1+P3
DAQDARWIN_POWERITEM_PF0	$P0/(P0^2+VAR0^2)^{1/2}=P0/VA0$
DAQDARWIN_POWERITEM_PF1	力率1
DAQDARWIN_POWERITEM_PF2	力率2
DAQDARWIN_POWERITEM_PF3	力率3
DAQDARWIN_POWERITEM_PF13	$P13/(P13^2+VAR132)^{1/2}=P13/VA13$
DAQDARWIN_POWERITEM_PH0	$\tan^{-1}(VAR0/P0)$
DAQDARWIN_POWERITEM_PH1	位相1
DAQDARWIN_POWERITEM_PH2	位相2
DAQDARWIN_POWERITEM_PH3	位相3
DAQDARWIN_POWERITEM_PH13	$\tan^{-1}(VAR13/P13)$
DAQDARWIN_POWERITEM_V0	(V1+V2+V3)/3
DAQDARWIN_POWERITEM_V1	実効電力1
DAQDARWIN_POWERITEM_V2	実効電力2
DAQDARWIN_POWERITEM_V3	実効電力3
DAQDARWIN_POWERITEM_V13	(V1+V3)/2
DAQDARWIN_POWERITEM_VA0	VA1+VA2+VA3
DAQDARWIN_POWERITEM_VA1	皮相電力1
DAQDARWIN_POWERITEM_VA2	皮相電力2
DAQDARWIN_POWERITEM_VA3	皮相電力3
DAQDARWIN_POWERITEM_VA13	VA1+VA3
DAQDARWIN_POWERITEM_VAR0	VAR1+VAR2+VAR3
DAQDARWIN_POWERITEM_VAR1	無効電力1
DAQDARWIN_POWERITEM_VAR2	無効電力2
DAQDARWIN_POWERITEM_VAR3	無効電力3
DAQDARWIN_POWERITEM_VAR13	VAR1+VAR3
DAQDARWIN_POWERITEM_FREQ	周波数

11.3 DARWINの型の概要

下記のデータ型が装備されています。

型	説明	ページ
DAQDARWIN	機器記述子の型です。	11-12
DarwinDateTime	時刻情報の構造体です。	11-12
DarwinChInfo	チャネル情報データの構造体です。	11-12
DarwinDataInfo	測定データの構造体です。	11-13
DarwinModuleInfo	モジュール情報の構造体です。	11-13
DarwinUnitInfo	ユニット情報の構造体です。	11-14
DarwinSystemInfo	システム構成データの構造体です。	11-14

型	説明
コールバック型	関数名に接頭辞「DLL」を付加し，大文字で記述します。 例.　openDARWIN関数のコールバック型：DLLOPENDARWIN

コールバック型は，Visual Cを使用のときに，実行可能モジュール(.dll)とリンクさせるために使用します。

11.4 DARWINの型

記述に関する説明

Visual C/Visual C++型，VB(Visual Basic)型

Visual C/Visual C++とVisual Basicでの型名を示します。
Visual C/Visual C++で符号なしのものも，Visual Basicでは符号ありになります。
Visual C/Visual C++の型で，配列の個数値は省略しています。

取得

下記の標記で，取得できる項目，ユーザーが設定できる項目などを示しています。

トーカ

「測定データの取得」に関するトーカ関数により取得されるデータです。
○：取得できる項目です。

チャネル情報

「チャネル情報データの取得」関数により取得されるデータです。
○：取得できる項目です。

ASCII

「ASCIIコードで測定データを取得」する関数により取得されるデータです。
○：取得できる項目です。

バイナリ

「バイナリコードで測定データを取得」する関数により取得されるデータです。
○：取得できる項目です。

用語

型の説明には，DARWINの機能を表す用語を使用しています。DARWINに関する用語については，付録2をご覧ください。

DAQDARWIN

機器記述子を格納するための型です。
Visual BasicではLong型，Visual Cではintで扱います。

DarwinDateTime

DarwinDateTime構造体

Visual C/Visual C++型	名称	内容	Visual Basic型
char	aYear	年の下2桁(0～99)	Byte
char	aMonth	月(1～12)	Byte
char	aDay	日(1～31)	Byte
char	aHour	時(0～23)	Byte
char	aMinute	分(0～59)	Byte
char	aSecond	秒(0～59)	Byte
short	aMilliSecond	未使用	Integer

取得

名称	内容	トーカ
aYear	年の下2桁(0～99)	○
aMonth	月(1～12)	○
aDay	日(1～31)	○
aHour	時(0～23)	○
aMinute	分(0～59)	○
aSecond	秒(0～59)	○

時刻情報データの構造体です。
Visual C++：ラッパクラスは，CDAQDARWINDateTimeです。
ミリ秒は，コマンドサポートのポートでは未使用です。

DarwinChInfo

DarwinChInfo構造体

Visual C/ Visual C++型	名称	内容	Visual Basic型
int	aChNo	チャネル番号	Long
int	aPoint	小数点位置	Long
int	aStatus	チャネルステータス	Long
int	aChType	チャネルタイプ	Long
char [ ]	aUnit	単位名	String * DAQDARWIN_MAXUNITLEN
char	align	未使用です。	(0 To 1) As Byte

取得

名称	内容	ASCII	バイナリ	チャネル情報
aChNo	チャネル番号	○	○	○
aPoint	小数点位置	○	-	○
aStatus	チャネルステータス	-	-	○
aChType	チャネルタイプ	○	○	○
aUnit	単位名	○	-	○

チャネル情報データの構造体です。
Visual C++：ラッパクラスは，CDAQDARWINChInfoです。

Note

バイナリコードで測定データを取得する関数の場合，小数点位置と単位情報は取得しません。測定データを工業量に変換するには，別途小数点位置情報を取得してください。

DarwinDataInfo

DarwinDataInfo構造体

Visual C/ Visual C++型	名称	内容	Visual Basic型
int	aValue	データ値	Long
int	aStatus	データステータス	Long
int [ ]	aAlarm	アラームレベルの個数値分のアラームの配列	(1 To 4) As Long

取得

名称	内容	ASCII	バイナリ
aValue	データ値	○	○
aStatus	データステータス	○	○
aAlarm	アラームレベルの個数値分のアラームの配列	○	○

測定データの構造体です。
ラッパクラスはCDAQDARWINDataInfoです。

DarwinModuleInfo

DarwinModuleInfo構造体

Visual C/ Visual C++型	名称	内容	Visual Basic型
int	aSlotNo	スロット番号	Long
int	aInternalCode	内部コード	Long
char [ ]	aName	モジュール名	String * DAQDARWIN_MAXMODULELEN
char	align	未使用です。	(0 To 1) As Byte

モジュール情報の構造体です。

DarwinUnitInfo

DarwinUnitInfo構造体

Visual C/ Visual C++型	名称	内容	Visual Basic型
int	aExist	情報の有効/無効	Long
int	aUnitNo	ユニット番号	Long
DarwinModuleInfo [ ]	aModule	スロットの個数値分のモジュール情報の配列	(0 To 5) As DarwinModuleInfo

ユニット情報の構造体です。

DarwinSystemInfo

DarwinSystemInfo構造体

Visual C/ Visual C++型	名称	内容	Visual Basic型
DarwinUnitInfo	aMainUnit	メインユニットのユニット情報	DarwinUnitInfo
DarwinUnitInfo [ ]	aSubUnit	サブユニットの個数値分のユニット情報の配列	(0 To 5) As DarwinUnitInfo

システム構成データの構造体です。
Visual C++：ラッパクラスは，CDAQDARWINSysInfoです。

12.1 MX100のクラス

本拡張APIは，APIに以下のクラスを追加した構成になります。

- ■ CDAQMXConfig
  - ・ CDAQMXItemConfig
- ■ CDAQHandler
  - ■ CDAQMX
    - ・ CDAQMX100
- ・ CDAQMXDataBuffer
- ・ CDAQMXList
  - ・ CDAQMXAOPWMList
  - ・ CDAQMXBalanceList
  - ・ CDAQMXDOList
  - ・ CDAQMXTransmitList

■ ：APIのクラスです。
・ ：拡張APIで追加されたクラスです。

CDAQMX100クラス

本体であるHandlerクラスです。状態遷移を実行します。

CDAQMXAOPWMListクラス

コマンドAO/PWM用チャネルのデータを管理するクラスです。

CDAQMXBalanceListクラス

初期バランスデータを管理するクラスです。

CDAQMXDataBufferクラス

チャネル単位の測定データを保持するクラスです。

CDAQMXDOListクラス

コマンドDO用チャネルのデータを管理するクラスです。

CDAQMXItemConfigクラス

設定項目で扱える設定データのクラスです。

CDAQMXListクラス

ユーザデータをリスト管理する共通クラスです。

CDAQMXTransmitListクラス

伝送出力データを管理するクラスです。

注意事項

各種データを内部に保持するため，メモリ消費が大きくなります。状態遷移時は状態を取得するため，パフォーマンスが低下することがあります。また，メモリやディスク容量が不足すると，正しくデータを保持することができなくなります。

状態遷移関数以外で通信を行うと，正しく動作しなくなることがあります。
MX100本体は，アクセスがないと通信を切断します。
通信操作を継続したい場合は，適度にステータスデータ取得を実行してください。

以下の機能は，制限事項として，サポートしません。

- FIFOの自動制御は使用できません。
- ユーザカウントは使用できません。
- データ番号は使用できません。
- 7セグメントLEDの状態は取得できません。
- CFカードのタイムアウト値は設定できません。
- 通信タイムアウトは設定できません。通信接続後，180秒に設定されます。

12.2 機能とクラス/関数メンバの対応ーMX100ー

本拡張APIでサポートする機能と，クラスの関数メンバの対応を示します。
状態遷移関数と取得関数の2種類あります。
状態遷移関数はMX100本体を制御します。データ取得機能で測定データを取得すると測定点が1点分だけ進みます(拡張APIの状態が遷移します)。
取得関数は項目値を返します。データ値取得を使用すると，拡張APIが保持している現在の状態のデータ値を返します(拡張APIの状態は遷移しません)。

状態遷移関数

表の「FIFO」欄は，FIFO中に関数メンバを実行したときの，FIFOの動作を示します。
停止：関数メンバを実行するとFIFOを停止します。
継続：関数メンバを実行してもFIFOを継続します。

通信機能

機能	FIFO	クラスと関数メンバ
MX100と通信接続	継続	CDAQMX100:: open
MX100との通信を切断	継続	CDAQMX100:: close

FIFOの開始/停止

機能	FIFO	クラスと関数メンバ
FIFOを開始	継続	CDAQMX100:: measStart
FIFOを停止	停止	CDAQMX100:: measStop

制御機能

機能		FIFO	クラスと関数メンバ
日付時刻設定 現在時刻		停止	CDAQMX100:: setDateTime
バックアップ 有効無効の設定		継続	CDAQMX100:: switchBackup
CFカードのフォーマット		停止	CDAQMX100:: formatCF
ユニット	システムの再構築	停止	CDAQMX100:: reconstruct
	システムの初期化	停止	CDAQMX100:: initSetValue
	アラームリセット（アラームACK）	停止	CDAQMX100:: ackAlarm
7セグメントLEDの表示		継続	CDAQMX100:: displaySegment
保持データの初期化	チャネル指定	継続	CDAQMX100:: initDataCh
	FIFO指定	継続	CDAQMX100:: initDataFIFO

制御機能は，通信の最後に状態更新を行います。
各データの送信，設定機能については， データ操作機能を参照してください。

設定機能

機能			FIFO	クラスと関数メンバ
設定データ	一括設定（一括送信）	全設定データ	停止	CDAQMX100:: sendConfig
		基本設定データ	停止	CDAQMX100:: sendConfig
	個別設定	システム構成データ	停止	CDAQMX100:: sendConfig
		チャネル設定データ	停止	CDAQMX100:: sendConfig
		初期バランスデータ	停止	CDAQMX100:: sendConfig
		出力チャネルデータ	停止	CDAQMX100:: sendConfig
初期バランスデータ	実行		停止	CDAQMX100:: initBalance
	リセット		停止	CDAQMX100:: clearBalance

設定データの設定機能は，保持しているデータを送信します。
任意の初期バランスデータを設定する場合，データ操作機能で初期バランスデータの送信機能を参照してください。

設定変更機能

設定機能は，設定送信して，状態更新を行います。
単独チャネル毎の設定なので，設定できなかった場合，原則エラーを返します。
データ値，または，測定値(倍精度浮動小数)での指定ができます。

レンジ設定

機能	FIFO	クラスと関数メンバ
スキップ	停止	CDAQMX100:: setRange
直流電圧入力	停止	CDAQMX100:: setRange
熱電対入力	停止	CDAQMX100:: setRange
測温抵抗体	停止	CDAQMX100:: setRange
ディジタル入力	停止	CDAQMX100:: setRange
抵抗	停止	CDAQMX100:: setRange
ひずみ	停止	CDAQMX100:: setRange
AO	停止	CDAQMX100:: setRange
PWM	停止	CDAQMX100:: setRange
チャネル間差演算	停止	CDAQMX100:: setChDELTA
リモートRJC	停止	CDAQMX100:: setChRRJC
パルス	停止	CDAQMX100:: setRange
通信	停止	CDAQMX100:: setRange

チャネル設定

機能		FIFO	クラスと関数メンバ
単位名		停止	CDAQMX100:: setChUnit
タグ		停止	CDAQMX100:: setChTag
コメント		停止	CDAQMX100:: setChComment
AI/DI/AO/PWM	スパン	停止	CDAQMX100:: setSpan
AI/DI	スケール	停止	CDAQMX100:: setScale
	アラーム	停止	CDAQMX100:: setAlarm
	ヒステリシス	停止	CDAQMX100:: setHisteresys
AI	フィルタ係数	停止	CDAQMX100:: setFilter
	基準接点補償(RJC)	停止	CDAQMX100:: setRJCType
	バーンアウト	停止	CDAQMX100:: setBurnout
DO	非励磁	停止	CDAQMX100:: setDeenergize
	保持	停止	CDAQMX100:: setHold
	参照アラーム	停止	CDAQMX100:: setRefAlarm
チャネル種類	DO種類	停止	CDAQMX100:: setChKind
	AO種類	停止	CDAQMX100:: setChKind
	PWM種類	停止	CDAQMX100:: setChKind
PI	チャタリングフィルタ	停止	CDAQMX100:: setChatFilter

モジュール設定

機能	FIFO	クラスと関数メンバ
周期種類	停止	CDAQMX100:: setInterval
A/D積分時間種類	停止	CDAQMX100:: setIntegral

ユニット設定

機能	FIFO	クラスと関数メンバ
ユニット番号	停止	CDAQMX100:: setUnitNo
温度単位種類	停止	CDAQMX100:: setUnitTemp
CF書き込み種類	停止	CDAQMX100:: setCFWriteMode

出力チャネルデータ

機能	FIFO	クラスと関数メンバ
出力種類	停止	CDAQMX100:: setOutputType
選択値	停止	CDAQMX100:: setChoice
パルス周期倍率	停止	CDAQMX100:: setPulseTime

データ操作機能

DOデータ

機能		FIFO	クラスと関数メンバ
作成		継続	CDAQMXDOList:: create
削除		継続	CDAQMXDOList:: del
部分変更	ユーザ指定	継続	CDAQMXDOList:: change
	コピー	継続	CDAQMXDOList:: copy
送信	既存指定	継続	CDAQMX100:: commandDO
	変更指定	継続	CDAQMX100:: switchDO

AO/PWMデータ

機能		FIFO	クラスと関数メンバ
作成		継続	CDAQMXAOPWMList:: create
削除		継続	CDAQMXAOPWMList:: del
部分変更	出力データ値	継続	CDAQMXAOPWMList:: change
	実出力値	継続	CDAQMX100:: changeAOPWMValue
	コピー	継続	CDAQMXAOPWMList:: copy
送信		継続	CDAQMX100:: commandAOPWM

初期バランスデータ

機能		FIFO	クラスと関数メンバ
作成		継続	CDAQMXBalanceList:: create
削除		継続	CDAQMXBalanceList:: del
部分変更	ユーザ指定	継続	CDAQMXBalanceList:: change
	コピー	継続	CDAQMXBalanceList:: copy
送信		停止	CDAQMX100::reloadBalance

伝送出力データ

機能		FIFO	クラスと関数メンバ
作成		継続	CDAQMXTransmitList:: create
削除		継続	CDAQMXTransmitList:: del
部分変更	ユーザ指定	継続	CDAQMXTransmitList:: change
	コピー	継続	CDAQMXTransmitList:: copy
送信	既存指定	継続	CDAQMX100:: commandTransmit
	変更指定	継続	CDAQMX100:: switchTransmit

各データ識別子で操作します。
CDAQMX100クラスからたどって操作します。
送信以外は，状態更新(通信)を行いません。

取得機能

機能			FIFO	クラスと関数メンバ
ステータスデータ			継続	CDAQMX100:: updateStatus
システム構成データ			継続	CDAQMX100:: updateSystem
設定データ			継続	CDAQMX100:: updateConfig
出力データ	DOデータ		継続	CDAQMX100:: updateDOData
	AO/PWMデータ 伝送出力データ		継続	CDAQMX100:: updateAOPWMData
チャネル情報データ			継続	CDAQMX100:: updateInfoCh
測定データ	チャネル指定	FIFO値	継続	CDAQMX100:: measDataCh
		瞬時値	継続	CDAQMX100:: measInstCh
	FIFO指定	FIFO値	継続	CDAQMX100:: measDataFIFO
		瞬時値	継続	CDAQMX100:: measInstFIFO
初期バランスデータ			継続	CDAQMX100:: updateBalance
出力チャネルデータ			継続	CDAQMX100:: updateOutput

データ取得は，本拡張API内部で一括取得が行われます。
収集によって，状態更新も行われます。
チャネル情報データや設定データ(システム構成データ，初期バランスデータ，出力チャネルデータを含む)は，内部で保持されていますが，ユーザが明示的に保持しているデータを更新できます。

設定項目機能

機能		FIFO	クラスと関数メンバ
設定データ	一括受信	継続	CDAQMX100:: getItemAll
	一括送信	停止	CDAQMX100:: setItemAll
設定項目	読み出し	継続	CDAQMXItemConfig:: readItem
	書き込み	継続	CDAQMXItemConfig:: writeItem
	初期化	継続	CDAQMXItemConfig:: initialize

設定項目の読み出し，書き込み，初期化は，保持している領域へのアクセスで，領域の整合性チェックをしません。また，状態更新(通信)を行いません。

取得関数

各データは種類により各クラスに格納されています。CDAQMX100クラスからたどって取得します。

測定データ

データ名		クラスと関数メンバ
データ値		CDAQMXDataInfo:: getValue
データステータス値		CDAQMXDataInfo:: getStatus
アラーム(有無)		CDAQMXDataInfo:: isAlarm
測定値	倍精度浮動小数	CDAQMXDataInfo:: getDoubleValue
	文字列	CDAQMXDataInfo:: getStringValue
時刻	秒数	CDAQMXDateTime:: getTime
	ミリ秒	CDAQMXDateTime:: getMilliSecond
有効データ(有無)		CDAQMXDataBuffer:: isCurrent

「CDAQMX100::getClassMXDataBuffer」から「CDAQMXDataBuffer::currentDataInfo」と「CDAQMXDataBuffer::currentDateTime」でたどって取得します。

チャネル情報データ

データ名	クラスと関数メンバ
FIFO番号	CDAQMXChInfo:: getFIFONo
FIFO内チャネル順序番号	CDAQMXChInfo:: getFIFOIndex
表示最小値	CDAQMXChInfo:: getDisplayMin
表示最大値	CDAQMXChInfo:: getDisplayMax
実範囲最小値	CDAQMXChInfo:: getRealMin
実範囲最大値	CDAQMXChInfo:: getRealMax

「CDAQMX100::getClassMXDataBuffer」から「CDAQMXDataBuffer::getClassMXChInfo」でたどって取得します。

チャネル設定データ

データ名				クラスと関数メンバ
チャネルステータス(有無)				CDAQMXChConfig:: isValid
小数点位置				CDAQMXChConfig:: getPoint
チャネル種類				CDAQMXChConfig:: getKind
レンジ種類				CDAQMXChConfig:: getRange
スケール種類				CDAQMXChConfig:: getScale
単位名				CDAQMXChConfig:: getUnit
タグ				CDAQMXChConfig:: getTag
コメント				CDAQMXChConfig:: getComment
AI/DI/AO/PWM	スパン	最小値	データ値	CDAQMXChConfig:: getSpanMin
			測定値	CDAQMXItemConfig:: getDoubleSpanMin
		最大値	データ値	CDAQMXChConfig:: getSpanMax
			測定値	CDAQMXItemConfig:: getDoubleSpanMax
AI/DI	スケール	最小値	データ値	CDAQMXChConfig:: getScaleMin
			測定値	CDAQMXItemConfig:: getDoubleScaleMin
		最大値	データ値	CDAQMXChConfig:: getScaleMax
			測定値	CDAQMXItemConfig:: getDoubleScaleMax
	アラーム種類			CDAQMXChConfig:: getAlarmType
	アラーム値(ON値)		データ値	CDAQMXChConfig:: getAlarmValueON
			測定値	CDAQMXItemConfig:: getDoubleAlarmON
	アラーム値(OFF値)		データ値	CDAQMXChConfig:: getAlarmValueOFF
			測定値	CDAQMXItemConfig:: getDoubleAlarmOFF
	ヒステリシス		データ値	CDAQMXItemConfig:: getHisterisys
			測定値	CDAQMXItemConfig:: getDoubleHisterisys
AI	フィルタ係数			CDAQMXChConfig:: getFilter
	RJC種類			CDAQMXChConfig:: getRJCType
	RJC電圧値			CDAQMXChConfig:: getRJCVolt
	バーンアウト			CDAQMXChConfig:: getBurnout
DO	非励磁			CDAQMXChConfig:: isDeenergize
	保持			CDAQMXChConfig:: isHold
	参照アラーム			CDAQMXChConfig:: isRefAlarm
チャネル間差演算/リモートRJC/AO/PWM				
	基準チャネル番号			CDAQMXChConfig:: getRefChNo
初期バランスデータ		有効無効値		CDAQMXBalanceData:: getBalanceValid
		初期バランス値		CDAQMXBalanceData:: getBalanceValue
出力チャネルデータ		出力種類		CDAQMXOutputData:: getOutputType
		アイドル時の選択値		CDAQMXOutputData:: getIdleChoice
		エラー時の選択値		CDAQMXOutputData:: getErrorChoice
		選択値が「指定値」の場合の値		
			データ値	CDAQMXOutputData:: getPresetValue
			測定値	CDAQMXItemConfig:: getDoublePresetValue
		パルス周期倍率		CDAQMXOutputData:: getPulseTime
PI	チャタリングフィルタ			CDAQMXChConfig:: isChatFilter

「CDAQMX100::getClassMXItemConfig」から「CDAQMXItemConfig::getClassMXChConfig」でたどって取得します。
初期バランスデータは，カレントデータと同じになります。「CDAQMX100::getClassMXItemConfig」から「CDAQMXItemConfig::getClassMXBalanceData」でたどって取得します。
出力チャネルデータは，「CDAQMX100::getClassMXItemConfig」から「CDAQMXItemConfig::getClassMXOutputData」でたどって取得します。

ネットワーク情報データ

データ名	クラスと関数メンバ
ホスト名	CDAQMXNetInfo:: getHost
IPアドレス	CDAQMXNetInfo:: getAddress
ポート番号	CDAQMXNetInfo:: getPort
サブネットマスク	CDAQMXNetInfo:: getSubMask
Gatewayアドレス	CDAQMXNetInfo:: getGateway

「CDAQMX100::getClassMXItemConfig」から「CDAQMXItemConfig::getClassMXNetInfo」でたどって取得します。

システム構成データ

データ名		クラスと関数メンバ
モジュール	モジュール種類	CDAQMXSysInfo:: getModuleType
	チャネル数	CDAQMXSysInfo:: getChNum
	周期種類	CDAQMXSysInfo:: getInterval
	AD積分時間種類	CDAQMXSysInfo:: getIntegral
	有効無効値	CDAQMXSysInfo:: isModuleValid
	起動時モジュール種類	CDAQMXSysInfo:: getStandbyType
	実際のモジュール種類	CDAQMXSysInfo:: getRealType
	端子種類	CDAQMXSysInfo:: getTerminalType
	バージョン	CDAQMXSysInfo:: getModuleVersion
	FIFO番号	CDAQMXSysInfo:: getFIFONo
	シリアル番号	CDAQMXSysInfo:: getModuleSerial
ユニット	ユニット種類	CDAQMXSysInfo:: getUnitType
	スタイル	CDAQMXSysInfo:: getStyle
	ユニット番号	CDAQMXSysInfo:: getUnitNo
	温度単位種類	CDAQMXSysInfo:: getTempUnit
	電源周波数	CDAQMXSysInfo:: getFrequency
	パート番号	CDAQMXSysInfo:: getPartNo
	オプション	CDAQMXSysInfo:: getOption
	シリアル番号	CDAQMXSysInfo:: getUnitSerial
	MACアドレス	CDAQMXSysInfo.getMAC
	CF書き込み種類	CDAQMXSysInfo:: getCFWriteMode

「CDAQMX100::getClassMXItemConfig」から「CDAQMXItemConfig::getClassMXSysInfo」でたどって取得します。

ステータスデータ

データ名		クラスと関数メンバ
ユニットステータス値		CDAQMXStatus:: getUnitStatus
FIFOの有効個数		CDAQMXStatus:: getFIFONum
バックアップ(有無)		CDAQMXStatus:: isBackup
FIFO	FIFOステータス値	CDAQMXStatus:: getFIFOStatus
	周期種類	CDAQMXStatus:: getInterval
CF	CFステータス種類	CDAQMXStatus:: getCFStatus
	サイズ	CDAQMXStatus:: getCFSize
	残容量	CDAQMXStatus:: getCFRemain
ステータス返却時刻	秒数	CDAQMXStatus:: getTime
	ミリ秒	CDAQMXStatus:: getMilliSecond

「CDAQMX100::getClassMXItemConfig」から「CDAQMXItemConfig:: getClassMXStatus」でたどって取得します。

カレントデータ

データ名		クラスと関数メンバ
DOデータ	有効無効値	CDAQMXDOData:: getDOValid
	ON/OFF状態	CDAQMXDOData:: getDOONOFF
AO/PWMデータ	有効無効値	CDAQMXAOPWMData:: getAOPWMValid
	出力データ値	CDAQMXAOPWMData:: getAOPWMValue
	出力値	CDAQMX100:: currentDoubleAOPWMValue
初期バランスデータ	有効無効値	CDAQMXBalanceResult:: getBalanceValid
	初期バランス値	CDAQMXBalanceResult:: getBalanceValue
	初期バランス結果	CDAQMXBalanceResult:: getResult
伝送出力データ	伝送状態	CDAQMXTransmit:: getTransmit

データ取得機能で取得された各データの状態です。
初期バランスデータの初期バランス結果は，設定機能による実行結果です。
DOデータは，「CDAQMX100::getClassMXDOList」から「CDAQMXDOList:: getCurrent」でたどって取得します。
AO/PWMデータは，「CDAQMX100::getClassMXAOPWMList」から「CDAQMXAOPWMList::getCurrent」でたどって取得します。
初期バランスデータは，「CDAQMX100::getClassMXBalanceList」から「CDAQMXBalanceList::getCurrent」でたどって取得します。
伝送出力データは，「CDAQMX100::getClassMXTransmitList」から「CDAQMXTransmitList::getCurrent」でたどって取得します。
DOデータ， AO/PWMデータなど，実際に出力されている出力状態をカレントデータとして取得できます。ただし，データを送信した直後は，設定した値が返却されて，実際の出力は次のタイミングになることがあります。
保持しているデータは，状態更新で取得したときの値です。取得関数を呼び出した時刻のデータではありません。

ユーザデータ

データ名		クラスと関数メンバ
DOデータ	有効無効値	CDAQMXDOData:: getDOValid
	ON/OFF状態	CDAQMXDOData:: getDOONOFF
AO/PWMデータ	有効無効値	CDAQMXAOPWMData:: getAOPWMValid
	出力データ値	CDAQMXAOPWMData:: getAOPWMValue
	出力値	CDAQMX100:: userDoubleAOPWMValue
初期バランス	有効無効値	CDAQMXBalanceData:: getBalanceValid
データ	初期バランス値	CDAQMXBalanceData:: getBalanceValue
伝送出力データ	伝送状態	CDAQMXTransmit:: getTransmit

ユーザがデータ操作機能で作成したデータの値を取得します。
DOデータは，「CDAQMX100::getClassMXDOList」から「CDAQMXDOList::getClassMXDOData」でたどって取得します。
AO/PWMデータは，「CDAQMX100::getClassMXAOPWMList」から「CDAQMXAOPWMList::getClassMXAOPWMData」でたどって取得します。
初期バランスデータは，「CDAQMX100::getClassMXBalanceList」から「CDAQMXBalanceList::getClassMXBalanceData」でたどって取得します。
伝送出力データは，「CDAQMX100::getClassMXTransmitList」から「CDAQMXTransmitList::getClassMXTransmit」でたどって取得します。

ユーティリティ

機能/データ名		クラスと関数メンバ
残りデータ個数	チャネル単位で取得	CDAQMXDataBuffer:: getDataNum
	FIFO単位で取得	CDAQMX100:: getDataNum
エラー	MX固有エラーの取得	CDAQMX100:: getLastError
	エラーメッセージ文字列を取得	CDAQMX100:: getErrorMessage
	エラーメッセージ文字列の最大長を取得	CDAQMX100:: getMaxLenErrorMessage
	エラー検出した設定項目番号を取得	CDAQMX100:: getItemError
FIFO情報から，チャネル番号に変換		CDAQMX100:: toChNo
レンジ種類別の小数点位置を取得		CDAQMXConfig:: getRangePoint
測定値	倍精度浮動小数に変換	CDAQMXDataInfo:: toDoubleValue
	文字列に変換	CDAQMXDataInfo:: toStringValue
アラーム	アラーム種類の文字列を取得	CDAQMXDataInfo:: getAlarmName
	アラーム文字列の最大長を取得	CDAQMXDataInfo:: getMaxLenAlarmName
本APIのバージョン番号を取得		CDAQMX100:: getVersionAPI
本APIのリビジョン番号を取得		CDAQMX100:: getRevisionAPI
IPアドレスのパート分割を取得		CDAQMXNetInfo:: getPart
AO/PWM	出力値を出力データ値に変換	CDAQMXAOPWMData:: toAOPWMValue
	出力データ値を出力値に変換	CDAQMXAOPWMData:: toRealValue
設定項目	設定項目番号から設定項目文字列を取得	CDAQMXItemConfig:: toItemName
	設定項目文字列から設定項目番号を取得	CDAQMXItemConfig:: getItemNo
	設定項目文字列の最大長を取得	CDAQMXItemConfig:: getMaxLenItemName
スタイルバージョンに変換		CDAQMXSysInfo:: toStyleVersion

12.3 プログラムーMX100/Visual C++ー

インクルードファイルのパスを追加

プロジェクトに，インクルードファイル(DAQMX100.h)のパスを追加します。追加方法は，ご使用の環境により異なります。

ソースファイルでの宣言

ソースファイルに宣言を記述します。

```
#include "DAQMX100.h"
```

Note

共通部のインクルードファイル(DAQHandler.h)，MX100部のインクルードファイル(DAQMX.h)は，上記インクルードファイルから参照されているので，宣言を記述する必要はありません。

ライブラリの指定

プロジェクトにライブラリ(DAQMX100.lib，DAQMX.lib，DAQHandler.lib)を追加します。追加方法は，ご使用の環境により異なります。
すべてのクラスが使用可能になります。Visual C用の関数郡も使用できます。

測定データの取得

プログラム例

```
////////////////////////////////////////////////////////////////
// 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;
}
////////////////////////////////////////////////////////////////
```

説明

全般

データ取得は，FIFOを開始することで可能になります。MX100のチャネル1のFIFOデータのうち，取得可能な分の測定データを一度に取得し，領域に格納します。その中から，現在状態の先頭の計測点の測定値データ(1点)を取得し，終了します。

通信接続

```
rc = daqmx100.open("192.168.1.12");
```

MX100のIPアドレスを指定しています。
通信用ポートは，通信用定数DAQMX_COMMPORT(MX100の通信ポート番号)を指定したことになります。

FIFO開始

```
daqmx100.measStart()
```

MX100でFIFOを開始します。

チャネル1の測定データの取得

```
rc = daqmx100.measDataCh(1);
```

MX100から，チャネル1の取得可能な分の測定データを一度に取得し，領域に格納します。先頭の計測点を現在状態とします。

測定値の取得

```
value = ((daqmx100.getClassMXDataBuffer(1))->
currentDataInfo())->getValue();
```

チャネル単位の測定データを格納している領域から，現在状態の測定データをたどって，チャネル1の現在状態の測定値を取得します。

FIFO停止

```
rc = daqmx100.measStop();
```

FIFOを停止します。

通信切断

```
rc = daqmx100.close();
```

通信を切断します。

参考

サンプルプログラムでは，measDataChを一度だけ実行して終了しています。measDataChを繰り返して実行すると，実行されるごとに，計測点をひとつ進めて現在状態とします。格納している計測点の最後まで到達したら，続く取得可能な分のデータを取得します。

設定データの読み出しと書き込み

プログラム例

```
////////////////////////////////////////////////////////////
// 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;
}
////////////////////////////////////////////////////////////
```

説明

全般

全設定項目の読み出しと書き込みのプログラム例です。下記の4つを実行します。

・MX100から設定データを一括受信

・設定データ領域の設定データを1項目ずつ取得

・設定データを1項目ずつ設定データ領域に書き込む

・MX100に設定データを一括送信

先頭番号から最終番号まで，1項目ずつ取得と書き込みをしています。

文字列領域はサイズに余裕を持って用意してください。

項目番号と項目文字列の組を保存，ロードすることで設定データをバックアップすることも可能になります。

設定項目番号については，6.3節を参照してください。

通信接続

```
rc = daqmx100.open("192.168.1.12");
```

ＭＸ１００のＩＰアドレスを指定しています。通信用ポートは，通信用定数DAQMX_COMMPORT(MX100の通信ポート番号)を指定したことになります。

設定データの一括受信

```
rc = daqmx100.getItemAll();
```

MX100の設定データの全項目を一括受信し，設定データ領域に格納します。

設定データを1項目ずつ取得

```
realLen = (daqmx100.getClassMXItemConfig()).readItem(i,
strItem, BUFSIZ);
```

設定データ領域から項目番号「i」の内容を取得します。

設定データを1項目ずつ書き込む

```
rc = (daqmx100.getClassMXItemConfig()).writeItem(i,
strItem);
```

設定データ領域の項目番号「i」に，strItemの内容を書き込みます。

設定データの一括送信

```
rc = daqmx100.setItemAll();
```

設定データの全項目をMX100に一括送信します。

通信切断

```
rc = daqmx100.close();
```

通信を切断します。

12.4 MX100用クラス詳細

クラスは，クラス名のアルファベット順で並んでいます。

CDAQMX100クラス

CDAQHandler
CDAQMX
CDAQMX100

本クラスは，CDAQMXの派生クラスです。
本クラスは，MX100と通信を行い，取得したデータを保持するクラスです。
状態遷移関数をサポートします。原則，関数実行後，状態を更新し保持します。
設定機能は，各設定実行後，設定データを再受信して状態更新します。設定機能と設定項目機能の一括送信はFIFOを停止します。以下のデータで保持できます。

- 設定データ
- チャネル情報データ
- 測定データ
- 時刻情報データ

コマンドDOなどユーザが動的に操作するデータを保持，管理できます。
あらかじめ送信するデータを作成しておきデータ識別子で指定して容易に送信することができます。以下のデータを保持できます。

- DOデータ
- AP/PWMデータ
- 伝送出力データ
- 初期バランスデータ

パブリックメンバ

構築・消滅

CDAQMX100	オブジェクトを構築します。
~CDAQMX100	オブジェクトを消滅します。

FIFO機能

measStart	データ収集を開始します。
measStop	データ収集を停止します。

制御機能

switchBackup	バックアップを切り替えます。
reconstruct	システムを再構築します。
initSetValue	システムを初期化します。
ackAlarm	アラームリセットを実行します。
displaySegment	7セグメントLEDを表示させます。
sendConfig	設定データを一括送信します。
initBalance	初期バランスを実行します。
clearBalance	初期バランス値を初期化します。

設定機能

setRange	レンジを設定します。
setChDELTA	チャネル間差演算を設定します。
setChRRJC	リモートRJCを設定します。
setChUnit	単位名を設定します。
setChTag	タグを設定します。
setChComment	コメントを設定します。
setSpan	スパンを設定します。
setScale	スケールを設定します。
setAlarm	アラームを設定します。
setHisterisys	ヒステリシスを設定します。
setFilter	フィルタ時定数を設定します。
setRJCType	RJC種類を設定します。
setBurnout	バーンアウト種類を設定します。
setDeenergize	非励磁を設定します。
setHold	保持を設定します。
setRefAlarm	参照アラームを設定します。
setChKind	チャネル種類を設定します。
setInterval	周期種類を設定します。
setIntegral	A/D積分時間種類を設定します。
setUnitNo	ユニット番号を設定します。
setUnitTemp	温度単位種類を設定します。
setCFWriteMode	CF書き込み種類を設定します。
setOutputType	出力種類を設定します。
setChoice	選択値を設定します。
setPulseTime	パルス周期倍率を設定します。
setChatFilter	チャタリングフィルタを設定します。

データ更新機能

updateStatus	ステータスデータを更新します。
updateSystem	システム構成データを更新します。
updateConfig	設定データを更新します。
updateDOData	現在のDOデータを更新します。
updateAOPWMData	現在のAO/PWMデータを更新します。
updateBalance	現在の初期バランスデータを更新します。
updateOutput	現在の出力チャネルデータを更新します。
updateInfoCh	チャネル情報データを更新します。

データ取得機能

measDataCh	チャネル単位でFIFO値を収集します。
measDataFIFO	FIFO単位でFIFO値を収集します。
measInstCh	チャネル単位で瞬時値を収集します。
measInstFIFO	FIFO単位で瞬時値を収集します。

データ操作機能

commandDO	DOデータを送信します。
switchDO	DOデータを切り替えます。
changeAOPWMValue	AO/PWMデータを変更します。
commandAOPWM	AO/PWMデータを送信します。
reloadBalance	初期バランスデータを送信します。
commandTransmit	伝送出力データを送信します。
switchTransmit	伝送出力データを切り替えます。
currentDoubleAOPWMValue	現在の出力データ値を取得します。
userDoubleAOPWMValue	ユーザ作成の出力データ値を取得します。

設定項目機能

getItemAll	設定データを一括受信します。
setItemAll	設定データを一括送信します。

データメンバ操作

getClassMXItemConfig	設定データを取得します。
getClassMXDataBuffer	チャネルごとの各種情報を取得します。
getClassMXDOList	DOデータの管理情報を取得します。
getClassMXAOPWMList	AO/PWMデータの管理情報を取得します。
getClassMXTransmitList	伝送出力データの管理情報を取得します。
getClassMXBalanceList	初期バランスデータの管理情報を取得します。

ユーティリティ

initDataCh	チャネル指定で保持データを初期化します。
initDataFIFO	FIFO指定で保持データを初期化します。
getDataNum	残りのデータ個数を取得します。
toChNo	チャネル番号を取得します。

●オーバライドしたメンバ

通信機能

open	通信接続をします。

制御機能

setDateTime	時刻情報を設定します。
formatCF	CFカードをフォーマットします。

ユーティリティ

isObject	オブジェクトをチェックします。

●継承するメンバ

CDAQHandler参照

```
closeget getErrorMessage getMaxLenErrorMessage getRevisionAPI
getVersionAPI receiveLine sendLine setTimeOut
```

CDAQMX参照

```
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
```

プロテクトメンバ

データメンバ

m_cMXItemConfig	設定データの格納領域です。
m_cMXDataBuffer	チャネルごとの各種情報の格納領域です。
m_cMXDOList	DOデータの管理領域です。
m_cMXAOPWMList	AO/PWMデータの管理領域です。
m_cMXTransmitList	伝送出力データの管理領域です。
m_cMXBalanceList	初期バランスデータの管理領域です。

データメンバ操作

measClear	測定データ取得のためのデータメンバを初期化します。
userClear	管理領域のデータメンバを初期化します。

データ更新機能

updateAll	状態，情報データをすべて更新します。
updateRenew	状態を更新します。

データ取得機能

getDataCh	チャネル単位でFIFO値を取得します。
getDataFIFO	FIFO単位でFIFO値を取得します。
getInstCh	チャネル単位で瞬時値を取得します。
getInstFIFO	FIFO単位で瞬時値を取得します。

ユーティリティ

nextFIFO	FIFO単位でカレントインデックス番号を次に進めます。
getVersionMX100DLL	本DLLのバージョンを取得します。
getRevisionMX100DLL	本DLLのリビジョンを取得します。

●継承するメンバ

CDAQHandler参照

```
m_comm m_nRemainSize receive receiveRemain send
```

CDAQMX参照

```
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
```

プライベートメンバ

なし。

関数メンバ(アルファベット順)

CDAQMX100::ackAlarm

構文

```
int ackAlarm(void);
```

説明

アラームリセットを実行します。
実行に成功したら，状態を更新します。

戻り値

エラー番号を返します。

参照

```
initSystem
updateRenew
```

CDAQMX100::CDAQMX100

構文

```
CDAQMX100(void);
CDAQMX100(const char * strAddress, unsigned int uiPort =
DAQMX_COMMPORT, int * errCode = NULL);
virtual ~CDAQMX100(void);
```

引数

strAddress	IPアドレスを文字列で指定します。
uiPort	ポート番号を指定します。
errCode	エラー番号の返却先を指定します。

説明

オブジェクトを構築，消滅します。
構築時，データメンバを初期化します。初期値は，原則0(NULL)です。引数が指定されている場合，通信接続を行います。返却先が指定されていれば，通信接続時のエラー番号を返します。
消滅時，データメンバの領域を開放します。通信記述子が存在する場合，通信切断(close)を行います。エラー番号は返却されません。

参照

```
measClear open userClear
CDAQMX::CDAQMX
```

CDAQMX100::changeAOPWMValue

構文

```
void changeAOPWMValue(int idAOPWM, int aopwmNo, int bValid,
double realValue);
```

引数

idAOPWM	AO/PWMデータ識別子を指定します。
aopwmNo	AO/PWMデータ番号を指定します。
bValid	有効/無効を有効無効値で指定します。
realValue	実際の出力値を指定します。

説明

指定されたAO/PWMデータ識別子のAO/PWMデータを変更します。
指定された実際の出力値を出力データ値に変換して格納します。

参照

```
getClassMXAOPWMList getClassMXItemConfig
CDAQMXAOPWMData::toAOPWMValue
CDAQMXAOPWMList::change
CDAQMXItemConfig::getClassMXOutputData
CDAQMXItemConfig::getRangePoint
CDAQMXOutputData::getOutputType
```

CDAQMX100::clearBalance

構文

```
int clearBalance(void);
```

説明

初期バランスをリセットします。
データメンバの初期バランスデータ管理領域の現在データに結果を格納します。
実行に成功したら，状態を更新します。

戻り値

エラー番号を返します。

参照

```
getClassMXBalanceList resetBalance updateRenew
CDAQMXBalanceList::getCurrent
```

CDAQMX100::commandAOPWM

構文

```
int commandAOPWM(int idAOPWM);
int commandAOPWM(CDAQMXAOPWMData & cMXAOPWMData);
```

引数

idAOPWM	AO/PWMデータ識別子を指定します。
cMXAOPWMData	AO/PWMデータを指定します。

説明

指定されたAO/PWMデータを送信します。
実行に成功したら，状態を更新します。

戻り値

エラー番号を返します。
エラー：

Not Data	データがありません。

参照

```
getClassMXAOPWMList setAOPWMData updateRenew
CDAQMXAOPWMList::getClassMXAOPWMData
```

CDAQMX100::commandDO

構文

```
int commandDO(int idDO);
int commandDO(CDAQMXDOData & cMXDOData);
```

引数

idDO	DOデータ識別子を指定します。
cMXDOData	DOデータを指定します。

説明

指定されたDOデータを送信します。
実行に成功したら，状態を更新します。

戻り値

エラー番号を返します。
エラー：

Not Data	データがありません。

参照

```
getClassMXDOList setDOData updateRenew
CDAQMXDOList::getClassMXDOData
```

CDAQMX100::commandTransmit

構文

```
int commandTransmit(int idTrans);
int commandTransmit(CDAQMXTransmit & cMXTransmit);
```

引数

idTrans　　伝送出力データ識別子を指定します。
cMXTransmit　伝送出力データを指定します。

説明

指定された伝送出力データを送信します。
実行に成功したら，状態を更新します。

戻り値

エラー番号を返します。
エラー：
Not Data　　データがありません。

参照

```
getClassMXTransmitList setTransmit updateRenew
CDAQMXTransmitList::getClassMXTransmit
```

CDAQMX100::currentDoubleAOPWMValue

構文

```
double currentDoubleAOPWMValue(int aopwmNo);
```

引数

aopwmNo　　AO/PWMデータ番号を指定します。

説明

データメンバのAO/PWMデータ管理領域の現在のデータから，指定されたAO/PWMデータ番号の出力データ値を実際の出力値で取得します。
存在しない場合，0.0を返します。

戻り値

実際の出力値を返します。

参照

```
getClassMXAOPWMList getClassMXItemConfig getOutputRange
CDAQMXAOPWMList::getCurrent
CDAQMXAOPWMData::getAOPWMValue
CDAQMXAOPWMData::toRealValue
```

CDAQMX100::displaySegment

構文

```
int displaySegment(int dispPattern0, int dispPattern1, int
dispType, int dispTime);
```

引数

dispPattern0	セグメント番号0の表示パターンを指定します。
dispPattern1	セグメント番号1の表示パターンを指定します。
dispType	表示形式を指定します。
dispTime	表示時間を指定します。

説明

7セグメントLEDの表示を設定します。
設定前の表示パターンは返却しません。
実行に成功したら，状態を更新します。

戻り値

エラー番号を返します。

参照

`setSegment updateRenew`

CDAQMX100::formatCF

構文

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

説明

CFカードをフォーマットします。
実行に成功したら，状態を更新します。

戻り値

エラー番号を返します。

参照

`updateRenew`
`CDAQMX::formatCF`

CDAQMX100::CDAQMXItemConfig

構文

```
CDAQMXItemConfig & getClassMXItemConfig(void);
```

説明

データメンバから設定データ領域のオブジェクトを取得します。

戻り値

オブジェクトへの参照を返します。

CDAQMX100::getClassMXAOPWMList

構文

```
CDAQMXAOPWMList & getClassMXAOPWMList(void);
```

説明

データメンバからAO/PWMデータ管理領域のオブジェクトを取得します。

戻り値

オブジェクトへの参照を返します。

CDAQMX100::getClassMXBalanceList

構文

```
CDAQMXBalanceList & getClassMXBalanceList(void);
```

説明

データメンバから初期バランスデータ管理領域のオブジェクトを取得します。

戻り値

オブジェクトへの参照を返します。

CDAQMX100::getClassMXDataBuffer

構文

```
CDAQMXDataBuffer * getClassMXDataBuffer(int chNo);
```

引数

chNo　　チャネル番号を指定します。

説明

データメンバからチャネル単位の測定データ を保持する格納領域をオブジェクトで取得します。指定されたチャネル番号のオブジェクトを返します。
存在しない場合，NULLを返します。

戻り値

オブジェクトへのポインタを返します。

CDAQMX100::getClassMXDOList

構文

```
CDAQMXDOList & getClassMXDOList(void);
```

説明

データメンバからDOデータ管理領域のオブジェクトを取得します。

戻り値

オブジェクトへの参照を返します。

CDAQMX100::getClassMXItemConfig

構文

```
CDAQMXItemConfig & getClassMXItemConfig(void);
```

説明

データメンバから設定データ領域のオブジェクトを取得します。

戻り値

オブジェクトへの参照を返します。

CDAQMX100::getClassMXTransmitList

構文

```
CDAQMXTransmitList & getClassMXTransmitList(void);
```

説明

データメンバから伝送出力データ管理領域のオブジェクトを取得します。

戻り値

オブジェクトへの参照を返します。

CDAQMX100::getDataCh

構文

```
int getDataCh(int chNo, int * bComm);
```

引数

chNo	チャネル番号を指定します。
bComm	通信が行われたか否かの返却先を指定します。

説明

指定されたチャネル番号の測定データを取得します。
データメンバのチャネルごとの各種情報領域のカレントインデックスを次に進めます。
保持しているデータがなくなったら，通信を実行して新たなデータを取得します。
チャネル単位でFIFO値を収集し，データメンバに格納します。
返却先が指定されていれば，実際に通信が実行されたか否かを有効無効値で返却します。

戻り値

エラー番号を返します。
エラー：

Not Data	チャネルごと各種情報領域が存在しません。

参照

```
getChData getChDataNo getClassMXDataBuffer getTimeData
talkChData
CDAQMXDataBuffer::create
CDAQMXDataBuffer::next
CDAQMXDataBuffer::setDataInfo
CDAQMXDataBuffer::setDateTime
CDAQMXStatus::isDataNo
```

CDAQMX100::getDataFIFO

構文

```
int getDataFIFO(int fifoNo, int * bComm);
```

引数

fifoNo　　FIFO番号を指定します。
bComm　　通信が行われたか否かの返却先を指定します。

説明

指定されたFIFO番号の測定データを取得します。
データメンバのチャネルごとの各種情報領域のカレントインデックスを次に進めます。
保持しているデータがなくなったら，通信を実行して新たなデータを取得します。
FIFO単位でFIFO値を収集し，データメンバに格納します。
返却先が指定されていれば，実際に通信が実行されたか否かを有効無効値で返却します。

戻り値

エラー番号を返します。
エラー：
Not Data　　チャネルごと各種情報領域が存在しません。

参照

```
getChData getClassMXDataBuffer getFIFODataNo getTimeData
nextFIFO searchChNo talkFIFOData
CDAQMXDataBuffer::create
CDAQMXDataBuffer::setDataInfo
CDAQMXDataBuffer::setDateTime
CDAQMXStatus::isDataNo
```

CDAQMX100::getDataNum

構文

```
int getDataNum(int fifoNo);
```

引数

fifoNo　　FIFO番号を指定します。

説明

データメンバのチャネルごとの各種情報領域から，指定されたFIFO番号の残りのデータ個数を取得します。
FIFO内チャネルの中で最小値を返します。
存在しない場合，0を返します。

戻り値

残りのデータ個数を返します。

参照

```
getClassMXDataBuffer
CDAQMXDataBuffer::getDataNum
```

CDAQMX100::getInstCh

構文

```
int getInstCh(int chNo);
```

引数

chNo　　チャネル番号を指定します。

説明

指定されたチャネル番号の測定データを受信します。
チャネル単位で瞬時値を収集し，データメンバに格納します。

戻り値

エラー番号を返します。
エラー：
Not Data　　チャネルごと各種情報領域が存在しません。

参照

```
getChData getClassMXDataBuffer getTimeData talkChData
CDAQMXDataBuffer::create
CDAQMXDataBuffer::setDataInfo
CDAQMXDataBuffer::setDateTime
```

CDAQMX100::getInstFIFO

構文

```
int getInstFIFO(int fifoNo);
```

引数

fifoNo　　FIFO番号を指定します。

説明

指定されたFIFO番号の測定データを受信します。
FIFO単位で瞬時値を収集し，データメンバに格納します。

戻り値

エラー番号を返します。

参照

```
getChData getClassMXDataBuffer getTimeData searchChNo
talkFIFOData
CDAQMXDataBuffer::create
CDAQMXDataBuffer::setDataInfo
CDAQMXDataBuffer::setDateTime
```

CDAQMX100::getItemAll

構文

```
int getItemAll(void);
```

説明

設定データを受信し，データメンバに格納します。
実行に成功したら，状態と情報データをすべて更新します。

戻り値

エラー番号を返します。

参照

```
getClassMXItemConfig getConfig updateAll
```

CDAQMX100::getRevisionMX100DLL

構文

```
static const int getRevisionMX100DLL(void);
```

説明

本DLLのリビジョン番号を取得します。

戻り値

本DLLのリビジョン番号を返します。

CDAQMX100::getVersionMX100DLL

構文

```
static const int getVersionMX100DLL(void);
```

説明

本DLLのバージョン番号を取得します。

戻り値

本DLLのバージョン番号を返します。

CDAQMX100::initBalance

構文

```
int initBalance(void);
```

説明

初期バランスを実行します。
データメンバの初期バランスデータ管理領域の現在データに結果を格納します。
実行に成功したら，状態を更新します。

戻り値

エラー番号を返します。

参照

```
getClassMXBalanceList runBalance updateRenew
CDAQMXBalanceList::getCurrent
```

CDAQMX100::initDataCh

構文

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

引数

chNo　　チャネル番号を指定します。

説明

指定されたチャネルの各種情報を初期化します。
チャネル番号に，定数値の「全チャネル番号指定」をすると，全チャネルを処理します。
実行に成功したら，状態を更新します。

参照

```
clearLastDataNoCh getClassMXDataBuffer updateRenew
CDAQMXDataBuffer::initialize
```

CDAQMX100::initDataFIFO

構文

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

引数

fifoNo　　FIFO番号を指定します。

説明

指定されたFIFO内のチャネルの各種情報を初期化します。
FIFO番号に，定数値の「全FIFO番号指定」をすると，全FIFOを処理します。
実行に成功したら，状態を更新します。

参照

```
clearLastDataNoFIFO getClassMXDataBuffer updateRenew
CDAQMXDataBuffer::initialize
```

CDAQMX100::initSetValue

構文

```
int initSetValue(void);
```

説明

システムの初期化をします。
実行に成功したら，状態，情報データをすべて更新します。

戻り値

エラー番号を返します。

参照

```
initSystem updateAll
```

CDAQMX100::isObject

構文

```
virtual int isObject(const char * classname = "CDAQMX100");
```

引数

classname　　クラス名を文字列で指定します。

説明

指定されたクラス名を継承しているかをチェックします。
省略された場合，本クラスであるかをチェックします。
本クラスを継承したクラスが，自分自身のクラスをチェックするためには，オーバーライドする必要があります。
クラスを継承している場合，「有効値」(真)を返します。それ以外は，「無効値」(偽)を返します。
本クラスと異なる場合，親クラスをチェックします。

戻り値

有効無効値を返します。

参照

`CDAQMX::isObject`

CDAQMX100::measClear

構文

```
void measClear(void);
```

説明

測定データ取得のための以下のデータメンバを初期化します。

- 設定データ領域
- チャネルごとの各種情報領域

参照

`getClassMXItemConfig`
`CDAQMXDataBuffer::initialize`
`CDAQMXItemConfig::initialize`

CDAQMX100::measDataCh

構文

```
int measDataCh(int chNo = DAQMX100_CHNO_ALL);
```

引数

chNo　　チャネル番号を指定します。

説明

指定されたチャネル番号の測定データのカレント測定点を一点だけ次に進めます。
チャネル番号に，定数値の「全チャネル番号指定」をすると，有効な全チャネルを処理します。
通信を実行したら，状態を更新します。

戻り値

エラー番号を返します。

参照

```
getDataCh updateRenew
```

CDAQMX100::measDataFIFO

構文

```
int measDataFIFO(int fifoNo = DAQMX100_FIFONO_ALL);
```

引数

fifoNo　　FIFO番号を指定します。

説明

指定されたFIFO番号の各チャネルの測定データのカレント測定点を一点だけ次に進めます。
FIFO番号に，定数値の「全FIFO番号指定」をすると，有効な全FIFOを処理します。
通信を実行したら，状態を更新します。

戻り値

エラー番号を返します。

参照

```
getDataFIFO updateRenew
```

CDAQMX100::measInstCh

構文

```
int measInstCh(int chNo = DAQMX100_CHNO_ALL);
```

引数

chNo　　チャネル番号を指定します。

説明

指定されたチャネル番号の測定データを受信します。
チャネル単位で瞬時値を収集し，データメンバに格納します。
チャネル番号に，定数値の「全チャネル番号指定」をすると，有効な全チャネルを処理します。
実行に成功したら，状態を更新します。

戻り値

エラー番号を返します。

参照

```
getInstCh updateRenew
```

CDAQMX100::measInstFIFO

構文

```
int measInstFIFO(int fifoNo = DAQMX100_FIFONO_ALL);
```

引数

fifoNo　　FIFO番号を指定します。

説明

指定されたFIFO番号の測定データを受信します。
FIFO単位で瞬時値を収集し，データメンバに格納します。
FIFO番号に，定数値の「全FIFO番号指定」をすると，有効な全FIFOを処理します。
実行に成功したら，状態を更新します。

戻り値

エラー番号を返します。

参照

```
getInstFIFO updateRenew
```

CDAQMX100::measStart

構文

```
int measStart(void);
```

説明

データ収集を開始します。
FIFOを開始します。
実行に成功したら，状態を更新します。

戻り値

エラー番号を返します。

参照

```
startFIFO updateRenew
```

CDAQMX100::measStop

構文

```
int measStop(void);
```

説明

データ収集を停止します。
FIFOを停止します。
実行に成功したら，状態を更新します。

戻り値

エラー番号を返します。

参照

```
stopFIFO updateRenew
```

CDAQMX100::nextFIFO

構文

```
int nextFIFO(int fifoNo);
```

引数

fifoNo　　FIFO番号を指定します。

説明

指定されたFIFO番号のチャネルのカレントインデックス番号をインクリメントします。
インクリメントの結果を有効無効値で返します。インクリメントしてデータのないチャネルが存在したら，「無効値」を返します。

戻り値

有効無効値を返します。

参照

```
getClassMXDataBuffer
CDAQMXDataBuffer::next
```

CDAQMX100::open

構文

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

引数

strAddress	IPアドレスを文字列で指定します。
uiPort	ポート番号を指定します。

説明

引数で指定されたIPアドレスとポート番号の機器と通信接続をします。
ポート番号は省略可能で，省略時は通信用定数の「MX100の通信ポート番号」になります。
測定データ取得のためのデータメンバを初期化し，接続に成功した場合，それらを取得して格納します。
通信タイムアウトを3分(計測器が自動切断する時間)に設定します。

戻り値

エラー番号を返します。

参照

```
close measClear setTimeOut updateAll
CDAQMX::open
```

CDAQMX100::reconstruct

構文

```
int reconstruct(void);
```

説明

システムを再構築します。
実行に成功したら，状態，情報データをすべて更新します。

戻り値

エラー番号を返します。

参照

```
initSystem updateAll
```

CDAQMX100::reloadBalance

構文

```
int reloadBalance(int idBalance);
int reloadBalance(CDAQMXBalanceData & cMXBalanceData);
```

引数

idBalance	初期バランスデータ識別子を指定します。
cMXBalanceData	初期バランスデータを指定します。

説明

指定された初期バランスデータを送信します。
実行に成功したら，状態を更新します。

戻り値

エラー番号を返します。
エラー：

Not Data	データがありません。

参照

`getClassMXBalanceList setBalance updateRenew CDAQMXBalanceList::getClassMXBalanceData`

CDAQMX100::sendConfig

構文

```
int sendConfig(void);
```

説明

データメンバの設定データ領域を送信します。

戻り値

エラー番号を返します。

参照

`setItemAll`

CDAQMX100::setAlarm

構文

```
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);
```

引数

chNo	チャネル番号を指定します。
levelNo	アラームレベルを指定します。
iAlarmType	アラーム種類を指定します。
valueON	アラーム発生のしきい値(On値)を指定します。
valueOFF	アラーム停止のしきい値(Off値)を指定します。

説明

指定されたチャネル番号のアラームレベルにアラームを設定します。
データメンバの設定データ領域を変更して，一括送信します。
チャネル番号に定数値の「全チャネル番号指定」をすると，全チャネルを処理します。
アラームレベルに定数値の「全アラームレベル番号指定」をすると，全アラームレベルを処理します。

戻り値

エラー番号を返します。

参照

```
getClassMXItemConfig setItemAll
CDAQMXChConfig::getPoint
CDAQMXChConfig::setAlarmValue
CDAQMXItemConfig::getClassMXChConfig
```

CDAQMX100::setBurnout

構文

```
int setBurnout(int chNo, int iBurnout);
```

引数

chNo	チャネル番号を指定します。
iBurnout	バーンアウト種類を指定します。

説明

指定されたチャネル番号のチャネルにバーンアウト種類を設定します。
データメンバの設定データ領域を変更して，一括送信します。
チャネル番号に定数値の「全チャネル番号指定」をすると，全チャネルを処理します。

戻り値

エラー番号を返します。

参照

```
getClassMXItemConfig setItemAll
CDAQMXChConfig::setBurnout
CDAQMXItemConfig::getClassMXChConfig
```

CDAQMX100::setCFWriteMode

構文

```
int setCFWriteMode(int iCFWriteMode);
```

引数

iCFWriteMode　CF書き込み種類を指定します。

説明

CF書き込み種類を設定します。
データメンバの設定データ領域を変更して，一括送信します。

戻り値

エラー番号を返します。

参照

```
getClassMXItemConfig setItemAll
CDAQMXItemConfig::getClassMXSysInfo
CDAQMXSysInfo::setCFWriteMode
```

CDAQMX100::setChatFilter

構文

```
int setChatFilter(int chNo, int bChatFilter);
```

引数

chNo　　チャネル番号を指定します。
bChatFilter　　チャタリングフィルタを有効無効値で指定します。

説明

指定されたチャネル番号のチャネルにチャタリングフィルタを設定します。
データメンバの設定データ領域を変更して，一括送信します。
チャネル番号に定数値の「全チャネル番号指定」をすると，全チャネルを処理します。

戻り値

エラー番号を返します。

参照

```
getClassMXItemConfig
setItemAll
CDAQMXChConfig::setChatFilter
CDAQMXItemConfig::getClassMXChConfig
```

CDAQMX100::setChComment

構文

```
int setChComment(int chNo, const char * strComment);
```

引数

chNo	チャネル番号を指定します。
strComment	コメントを指定します。

説明

指定されたチャネル番号のチャネルにコメントを設定します。
データメンバの設定データ領域を変更して，一括送信します。
チャネル番号に定数値の「全チャネル番号指定」をすると，全チャネルを処理します。

戻り値

エラー番号を返します。

参照

```
getClassMXItemConfig setItemAll
CDAQMXChConfig::setComment
CDAQMXItemConfig::getClassMXChConfig
```

CDAQMX100::setChDELTA

構文

```
int setChDELTA(int chNo, int refChNo, int iRange =
DAQMX_RANGE_REFERENCE);
```

引数

chNo	チャネル番号を指定します。
refChNo	基準チャネルのチャネル番号を指定します。
iRange	自チャネルのレンジ種類を指定します。

説明

指定された基準チャネルとの差演算を設定します。
レンジ以外のチャネルの設定値は，既定値になります。
データメンバの設定データ領域を変更して，一括送信します。
チャネル番号に定数値の「全チャネル番号指定」をすると，全チャネルを処理します。
レンジ種類に「参照チャネル」を指定した場合，自チャネルの測定レンジに参照するチャネル番号のレンジを適用します。

戻り値

エラー番号を返します。

参照

```
getClassMXItemConfig setItemAll
CDAQMXItemConfig::getClassMXSysInfo
CDAQMXItemConfig::setDELTA
CDAQMXSysInfo::getModuleType
```

CDAQMX100::setChKind

構文

```
int setChKind(int chNo, int iKind, int refChNo =
DAQMX_REFCHNO_NONE);
```

引数

chNo	チャネル番号を指定します。
iKind	チャネル種類を指定します。
refChNo	参照するチャネル番号を指定します。

説明

指定されたチャネル番号のチャネルにチャネル種類を設定します。
チャネルの設定値は，既定値になります。
データメンバの設定データ領域を変更して，一括送信します。
チャネル番号に定数値の「全チャネル番号指定」をすると，全チャネルを処理します。
参照するチャネル番号は，チャネルの種類が「AI(チャネル間差)」，「DI(チャネル間差)」，「AI(リモートRJC)」，「AO(伝送出力)」，「PWM(伝送出力)」の場合に有効です。

戻り値

エラー番号を返します。

参照

```
getClassMXItemConfig setItemAll
CDAQMXItemConfig::setAOType
CDAQMXItemConfig::setDELTA
CDAQMXItemConfig::setDI
CDAQMXItemConfig::setDOType
CDAQMXItemConfig::setPWMType
CDAQMXItemConfig::setRRJC
CDAQMXItemConfig::setSKIP
CDAQMXItemConfig::setVOLT
```

CDAQMX100::setChoice

構文

```
int setChoice(int outputNo, int idleChoice, int errorChoice,
int presetValue);
int setChoice(int outputNo, int idleChoice, int errorChoice,
double presetValue);
```

引数

outputNo	出力チャネルデータ番号を指定します。
idleChoice	アイドル時の選択値を指定します。
errorChoice	エラー時の選択値を指定します。
presetValue	選択値が「指定値」の場合の値を指定します。

説明

指定された出力チャネルデータ番号の出力チャネルデータに選択値を設定します。
データメンバの設定データ領域を変更して，一括送信します。
出力チャネルデータ番号に定数値の「全出力データ番号指定」をすると，全チャネルを処理します。

戻り値

エラー番号を返します。

参照

```
getClassMXItemConfig setItemAll
CDAQMXItemConfig::getClassMXOutputData
CDAQMXItemConfig::getRangePoint
CDAQMXOutputData::setChoice
```

CDAQMX100::setChRRJC

構文

```
int setChRRJC(int chNo, int refChNo);
```

引数

chNo	チャネル番号を指定します。
refChNo	参照するチャネル番号を指定します。

説明

指定された参照チャネルとのリモートRJCを設定します。
レンジ以外のチャネルの設定値は，既定値になります。
データメンバの設定データ領域を変更して，一括送信します。
チャネル番号に定数値の「全チャネル番号指定」をすると，全チャネルを処理します。

戻り値

エラー番号を返します。

参照

```
getClassMXItemConfig setItemAll
CDAQMXItemConfig::setRRJC
```

CDAQMX100::setChTag

構文

```
int setChTag(int chNo, const char * strTag);
```

引数

chNo	チャネル番号を指定します。
strTag	タグを指定します。

説明

指定されたチャネル番号のチャネルにタグを設定します。
データメンバの設定データ領域を変更して，一括送信します。
チャネル番号に定数値の「全チャネル番号指定」をすると，全チャネルを処理します。

戻り値

エラー番号を返します。

参照

```
getClassMXItemConfig setItemAll
CDAQMXChConfig::setTag
CDAQMXItemConfig::getClassMXChConfig
```

CDAQMX100::setChUnit

構文

```
int setChUnit(int chNo, const char * strUnit);
```

引数

chNo	チャネル番号を指定します。
strUnit	単位名を指定します。

説明

指定されたチャネル番号のチャネルに単位名を設定します。
データメンバの設定データ領域を変更して，一括送信します。
チャネル番号に定数値の「全チャネル番号指定」をすると，全チャネルを処理します。

戻り値

エラー番号を返します。

参照

```
getClassMXItemConfig setItemAll
CDAQMXChConfig::setUnit
CDAQMXItemConfig::getClassMXChConfig
```

CDAQMX100::setDateTime

構文

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

引数

pcMXDateTime 時刻情報データを指定します。

説明

機器本体に時刻情報データを設定します。
実行に成功したら，状態を更新します。

戻り値

エラー番号を返します。

参照

`updateRenew`
`CDAQMX::setDateTime`

CDAQMX100::setDeenergize

構文

```
int setDeenergize(int doNo, int bDeenergize);
```

引数

doNo	DOデータ番号を指定します。
bDeenergize	非励磁を有効無効値で指定します。

説明

指定されたDOデータ番号のチャネルに非励磁を設定します。
データメンバの設定データ領域を変更して，一括送信します。
チャネル番号に定数値の「全チャネル番号指定」をすると，全チャネルを処理します。

戻り値

エラー番号を返します。

参照

`getClassMXItemConfig setItemAll`
`CDAQMXChConfig::setDeenergize`
`CDAQMXItemConfig::getClassMXChConfig`

CDAQMX100::setFilter

構文

```
int setFilter(int chNo, int iFilter);
```

引数

chNo	チャネル番号を指定します。
iFilter	フィルタ係数を指定します。

説明

指定されたチャネル番号のチャネルにフィルタ係数を設定します。
データメンバの設定データ領域を変更して，一括送信します。
チャネル番号に定数値の「全チャネル番号指定」をすると，全チャネルを処理します。

戻り値

エラー番号を返します。

参照

```
getClassMXItemConfig setItemAll
CDAQMXChConfig::setFilter
CDAQMXItemConfig::getClassMXChConfig
```

CDAQMX100::setHisterisys

構文

```
int setHisterisys(int chNo, int levelNo, double histerisys);
int setHisterisys(int chNo, int levelNo, int histerisys = 0);
```

引数

chNo	チャネル番号を指定します。
levelNo	アラームレベルを指定します。
histerisys	ヒステリシスを指定します。

説明

指定されたチャネル番号のアラームレベルにヒステリシスを設定します。
データメンバの設定データ領域を変更して，一括送信します。
チャネル番号に定数値の「全チャネル番号指定」をすると，全チャネルを処理します。
アラームレベルに定数値の「全アラームレベル番号指定」をすると，全アラームレベルを処理します。

戻り値

エラー番号を返します。

参照

```
getClassMXItemConfig setItemAll
CDAQMXChConfig::getAlarmType
CDAQMXChConfig::getAlarmValueON
CDAQMXChConfig::getPoint
CDAQMXChConfig::setAlarm
CDAQMXItemConfig::getClassMXChConfig
```

CDAQMX100::setHold

構文

```
int setHold(int doNo, int bHold);
```

引数

doNo	DOデータ番号を指定します。
bHold	保持を有効無効値で指定します。

説明

指定されたDOデータ番号のチャネルに保持を設定します。
データメンバの設定データ領域を変更して，一括送信します。
チャネル番号に定数値の「全チャネル番号指定」をすると，全チャネルを処理します。

戻り値

エラー番号を返します。

参照

```
getClassMXItemConfig setItemAll
CDAQMXChConfig::setHold
CDAQMXItemConfig::getClassMXChConfig
```

CDAQMX100::setIntegral

構文

```
int setIntegral(int moduleNo, int iHz);
```

引数

moduleNo	モジュール番号を指定します。
iHz	A/D積分時間種類を指定します。

説明

指定されたモジュール番号のモジュールにA/D積分時間種類を設定します。
データメンバの設定データ領域を変更して，一括送信します。
モジュール番号に定数値の「全モジュール番号指定」をすると，全モジュールを処理します。

戻り値

エラー番号を返します。

参照

```
getClassMXItemConfig setItemAll
CDAQMXItemConfig::getClassMXSysInfo
CDAQMXItemConfig::setInterval
CDAQMXSysInfo::getInterval
```

CDAQMX100::setInterval

構文

```
int setInterval(int moduleNo, int iInterval);
```

引数

moduleNo	モジュール番号を指定します。
iInterval	周期種類を指定します。

説明

指定されたモジュール番号のモジュールに周期種類を設定します。
データメンバの設定データ領域を変更して，一括送信します。
モジュール番号に定数値の「全モジュール番号指定」をすると，全モジュールを処理します。

戻り値

エラー番号を返します。

参照

```
getClassMXItemConfig setItemAll
CDAQMXItemConfig::getClassMXSysInfo
CDAQMXItemConfig::setInterval
CDAQMXSysInfo::getIntegral
```

CDAQMX100::setItemAll

構文

```
int setItemAll(void);
```

説明

データメンバの設定データ領域を送信します。
実行に成功したら，状態と情報データをすべて更新します。

戻り値

エラー番号を返します。

参照

```
getClassMXItemConfig setConfig updateAll
```

CDAQMX100::setOutputType

構文

```
int setOutputType(int outputNo, int iOutput);
```

引数

outputNo　　出力チャネルデータ番号を指定します。
iOutput　　出力種類を指定します。

説明

指定された出力チャネルデータ番号の出力チャネルデータに出力種類を設定します。
出力種類以外の設定値は，既定値になります。
データメンバの設定データ領域を変更して，一括送信します。
出力チャネルデータ番号に定数値の「全出力データ番号指定」をすると，全チャネルを処理します。

戻り値

エラー番号を返します。

参照

```
getClassMXItemConfig setItemAll
CDAQMXItemConfig::setAO
CDAQMXItemConfig::setPWM
```

CDAQMX100::setPulseTime

構文

```
int setPulseTime(int outputNo, int pulseTime);
```

引数

outputNo　　出力チャネルデータ番号を指定します。
pulseTime　　パルス周期倍率を指定します。

説明

指定された出力チャネルデータ番号の出力チャネルデータにパルス周期倍率を設定します。
データメンバの設定データ領域を変更して，一括送信します。
出力チャネルデータ番号に定数値の「全出力データ番号指定」をすると，全チャネルを処理します。

戻り値

エラー番号を返します。

参照

```
getClassMXItemConfig setItemAll
CDAQMXItemConfig::getClassMXOutputData
CDAQMXOutputData::setPulseTime
```

CDAQMX100::setRange

構文

```
int setRange(int chNo, int iRange);
```

引数

chNo	チャネル番号を指定します。
iRange	レンジ種類を指定します。

説明

レンジを設定します。
接点レンジとSKIPレンジの種類については，新たな定数の定義を参照してください。
レンジ以外のチャネルの設定値は，既定値になります。
データメンバの設定データ領域を変更して，一括送信します。
チャネル番号に定数値の「全チャネル番号指定」をすると，全チャネルを処理します。

戻り値

エラー番号を返します。

参照

```
getClassMXItemConfig setItemAll
CDAQMXItemConfig::setAO
CDAQMXItemConfig::setDI
CDAQMXItemConfig::setPWM
CDAQMXItemConfig::setRES
CDAQMXItemConfig::setRTD
CDAQMXItemConfig::setSKIP
CDAQMXItemConfig::setSTRAIN
CDAQMXItemConfig::setTC
CDAQMXItemConfig::setVOLT
```

CDAQMX100::setRefAlarm

構文

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

引数

doNo	DOデータ番号を指定します。
refChNo	参照するチャネル番号を指定します。
levelNo	アラームレベルを指定します。
bValid	有効無効値を指定します。

説明

指定されたDOデータ番号のチャネルに参照アラームを設定します。
参照アラームは，参照するチャネル番号とアラームレベルで指定します。
チャネル番号に定数値の「全チャネル番号指定」をすると，全チャネルを処理します。
アラームレベルに定数値の「全アラームレベル番号指定」をすると，全アラームレベルを処理します。

戻り値

エラー番号を返します。

参照

```
getClassMXItemConfig setItemAll
CDAQMXChConfig::setRefAlarm
CDAQMXItemConfig::getClassMXChConfig
```

CDAQMX100::setRJCType

構文

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

引数

chNo	チャネル番号を指定します。
iRJCType	RJC種類を指定します。
volt	RJC電圧値を指定します。

説明

指定されたチャネル番号のチャネルにRJC種類を設定します。
データメンバの設定データ領域を変更して，一括送信します。
チャネル番号に定数値の「全チャネル番号指定」をすると，全チャネルを処理します。

戻り値

エラー番号を返します。

参照

```
getClassMXItemConfig setItemAll
CDAQMXChConfig::setRJCType
CDAQMXItemConfig::getClassMXChConfig
```

CDAQMX100::setScale

構文

```
int setScale(int chNo, double scaleMin, double scaleMax, int
scalePoint);
int setScale(int chNo, int scaleMin = 0, int scaleMax = 0, int
scalePoint = 0);
```

引数

chNo	チャネル番号を指定します。
scaleMin	スケール最小値を指定します。
scaleMax	スケール最大値を指定します。
scalePoint	小数点位置を指定します。

説明

指定されたチャネル番号のチャネルにスケールを設定します。
データメンバの設定データ領域を変更して，一括送信します。
チャネル番号に定数値の「全チャネル番号指定」をすると，全チャネルを処理します。

戻り値

エラー番号を返します。

参照

```
getClassMXItemConfig setItemAll
CDAQMXItemConfig::setScalling
```

CDAQMX100::setSpan

構文

```
int setSpan(int chNo, double spanMin, double spanMax);
int setSpan(int chNo, int spanMin = 0, int spanMax = 0);
```

引数

chNo	チャネル番号を指定します。
spanMin	スパン最小値を指定します。
spanMax	スパン最大値を指定します。

説明

指定されたチャネル番号のチャネルにスパンを設定します。
データメンバの設定データ領域を変更して，一括送信します。
チャネル番号に定数値の「全チャネル番号指定」をすると，全チャネルを処理します。

戻り値

エラー番号を返します。

参照

```
getClassMXItemConfig setItemAll
CDAQMXChConfig::setSpan
CDAQMXItemConfig::getClassMXChConfig
CDAQMXItemConfig::getClassMXSysInfo
CDAQMXItemConfig::getSpanPoint
CDAQMXSysInfo::getTempUnit
```

CDAQMX100::setUnitNo

構文

```
int setUnitNo(int unitNo);
```

引数

unitNo　　ユニット番号を指定します。

説明

ユニット番号を設定します。
データメンバの設定データ領域を変更して，一括送信します。

戻り値

エラー番号を返します。

参照

```
getClassMXItemConfig setItemAll
CDAQMXItemConfig::getClassMXSysInfo
CDAQMXSysInfo::setUnitNo
```

CDAQMX100::setUnitTemp

構文

```
int setUnitTemp(int iTempUnit);
```

引数

iTempUnit　　温度単位種類を指定します。

説明

温度単位種類を設定します。
影響を受けるチャネルの設定値が変更されます。
データメンバの設定データ領域を変更して，一括送信します。

戻り値

エラー番号を返します。

参照

```
getClassMXItemConfig setItemAll
CDAQMXItemConfig::setTempUnit
```

CDAQMX100::switchBackup

構文

```
int switchBackup(int bBackup);
```

引数

bBackup　　バックアップを有効無効値で指定します。

説明

バックアップを切り替えます。
実行に成功したら，状態を更新します。

戻り値

エラー番号を返します。

参照

```
setBackup updateRenew
```

CDAQMX100::switchDO

構文

```
int switchDO(int idDO, int bONOFF);
```

引数

idDO　　DOデータ識別子を指定します。
bONOFF　　ON/OFFを有効無効値で指定します。

説明

指定されたDOデータを送信します。
DOデータの有効チャネルを指定されたON/OFF値に変更して送信します。
実行に成功したら，状態を更新します。

戻り値

エラー番号を返します。
エラー：
Not Data　　データがありません。

参照

```
commandDO getClassMXDOList
CDAQMXDOData::setDOONOFF
CDAQMXDOList::getClassMXDOData
```

CDAQMX100::switchTransmit

構文

```
int switchTransmit(int idTrans, int iTransmit);
```

引数

idTrans　　伝送出力データ識別子を指定します。
iTransmit　　伝送状態を指定します。

説明

指定された伝送出力データを送信します。
伝送出力データの全チャネルを指定された伝送状態に変更して送信します。
実行に成功したら，状態を更新します。

戻り値

エラー番号を返します。
エラー：
Not Data　　データがありません。

参照

commandTransmit getClassMXTransmitList
CDAQMXTransmit::setTransmit
CDAQMXTransmitList::getClassMXTransmit

CDAQMX100::toChNo

構文

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

引数

fifoNo　　FIFO番号を指定します。
fifoIndex　　FIFO内チャネル順序番号を指定します。

説明

指定された情報からチャネル番号を取得します。
存在しない場合，0を返します。

戻り値

チャネル番号を返します。

参照

searchChNo

CDAQMX100::updateAll

構文

```
int updateAll(void);
```

説明

状態，情報データをすべて更新します。
以下の情報を取得し，データメンバに格納します。

- 設定データ
- チャネル情報データ
- 随時変化している機器本体の状態

戻り値

エラー番号を返します。

参照

```
updateConfig updateInfoCh updateRenew
```

CDAQMX100::updateAOPWMData

構文

```
int updateAOPWMData(void);
```

説明

AO/PWMデータと伝送出力データを受信し，データメンバに格納します。
AO/PWMデータ管理領域と伝送出力管理領域の現在データに格納されます。
通信パケットバージョンが本機能に対応していない場合，何もせずに正常終了します。

戻り値

エラー番号を返します。

参照

```
getAOPWMData getClassMXAOPWMList getClassMXTransmitList
getPacketVersion
CDAQMXAOPWMList::getCurrent
CDAQMXTransmitList::getCurrent
```

CDAQMX100::updateBalance

構文

```
int updateBalance(void);
```

説明

初期バランスデータを受信し，データメンバに格納します。
初期バランスデータ管理領域の現在データに格納されます。
設定データ領域の初期バランスデータへも複写します。
通信パケットバージョンが本機能に対応していない場合，何もせずに正常終了します。

戻り値

エラー番号を返します。

参照

```
getBalance getClassMXBalanceList getClassMXItemConfig
getPacketVersion
CDAQMXBalanceList::getCurrent
CDAQMXItemConfig::getClassMXBalanceData
```

CDAQMX100::updateConfig

構文

```
int updateConfig(void);
```

説明

設定データを受信し，データメンバに格納します。
初期バランスデータを初期バランスデータ管理領域の現在データに複写します。

戻り値

エラー番号を返します。

参照

```
getClassMXBalanceList getClassMXItemConfig getConfig
CDAQMXBalanceList::getCurrent
CDAQMXItemConfig::getClassMXBalanceData
```

CDAQMX100::updateDOData

構文

```
int updateDOData(void);
```

説明

DOデータを受信し，データメンバに格納します。
DOデータ管理領域の現在データに格納されます。

戻り値

エラー番号を返します。

参照

```
getClassMXDOList getDOData
CDAQMXDOList::getCurrent
```

CDAQMX100::updateInfoCh

構文

```
int updateInfoCh(int chNo = DAQMX_CHNO_ALL);
```

引数

chNo　　チャネル番号を指定します。

説明

指定されたチャネル番号のチャネル情報データを受信し，データメンバに格納します。

チャネル番号に，定数値の「全チャネル番号指定」をすると，全チャネルを処理します。

戻り値

エラー番号を返します。

参照

```
getChInfo getClassMXDataBuffer talkChInfo
CDAQMXDataBuffer::setChInfo
```

CDAQMX100::updateOutput

構文

```
int updateOutput(void);
```

説明

出力チャネルデータを受信し，データメンバに格納します。

戻り値

エラー番号を返します。

参照

```
getClassMXItemConfig getOutput
CDAQMXItemConfig::getClassMXOutputData
```

CDAQMX100::updateRenew

構文

```
int updateRenew(void);
```

説明

状態を更新します。

随時変化している機器本体の以下の状態を取得し，データメンバに格納します。

- ステータスデータ
- 現在のDOデータ
- 現在のAO/PWMデータと伝送出力データ

戻り値

エラー番号を返します。

参照

```
updateAOPWMData updateDOData updateStatus
```

CDAQMX100::updateStatus

構文

```
int updateStatus(void);
```

説明

ステータスデータを受信し，データメンバに格納します。

戻り値

エラー番号を返します。

参照

```
getClassMXItemConfig getStatusData
CDAQMXItemConfig::getClassMXStatus
```

CDAQMX100::updateSystem

構文

```
int updateSystem(void);
```

説明

システム構成データを受信し，データメンバに格納します。

戻り値

エラー番号を返します。

参照

```
getClassMXItemConfig getSystemConfig
CDAQMXItemConfig::getClassMXSysInfo
```

CDAQMX100::userClear

構文

```
void userClear(void);
```

説明

管理領域の以下のデータメンバから，現在のデータを初期化します。

- DOデータ管理領域
- AO/PWMデータ管理領域
- 伝送出力データ管理領域
- 初期バランスデータ管理領域

参照

```
getClassMXAOPWMList getClassMXBalanceList getClassMXDOList
getClassMXTransmitList
CDAQMXAOPWMList::initCurrent
CDAQMXBalanceList::initCurrent
CDAQMXDOList::initCurrent
CDAQMXTransmitList::initCurrent
```

CDAQMX100::userDoubleAOPWMValue

構文

```
double userDoubleAOPWMValue(int idAOPWM, int aopwmNo);
```

引数

idAOPWM　　AO/PWMデータ識別子を指定します。
aopwmNo　　AO/PWMデータ番号を指定します。

説明

データメンバのAO/PWMデータ管理領域の指定されたAO/PWMデータ識別子のAO/PWMデータから，指定されたAO/PWMデータ番号の出力データ値を実際の出力値で取得します。
存在しない場合，0.0を返します。

戻り値

実際の出力値を返します。

参照

```
getClassMXAOPWMList getClassMXItemConfig getOutputRange
CDAQMXAOPWMList::getClassMXAOPWMData
CDAQMXAOPWMData::getAOPWMValue
CDAQMXAOPWMData::toRealValue
```

CDAQMXAOPWMListクラス

CDAQMXList
　CDAQMXAOPWMList

本クラスは，コマンドAO/PWMの出力であるAO/PWMデータを管理するクラスです。あらかじめ送信するAO/PWMデータを作成，保持することができます。データ識別子で識別します。

パブリックメンバ

構築・消滅

CDAQMXAOPWMList	オブジェクトを構築します。
~CDAQMXAOPWMList	オブジェクトを消滅します。

データメンバ操作

add	AO/PWMデータを追加します。
change	AO/PWMデータを変更します
copyData	AO/PWMデータを複写します。
getClassMXAOPWMData	AO/PWMデータを取得します。
initCurrent	現在のAO/PWMデータを初期化します。
getCurrent	現在のAO/PWMデータを取得します。

●オーバライドしたメンバ

データメンバ操作

create	AO/PWMデータを作成します。
copy	AO/PWMデータを複写します。

●継承するメンバ

CDAQMXList参照

```
del
getMaxNo
getNum
initialize
isData
```

プロテクトメンバ

データメンバ

m_cCurrent　　現在のAO/PWMデータを格納する領域です。

●継承するメンバ

CDAQMXList参照

```
m_list
m_num
addData
deldata
getData
```

プライベートメンバ

なし。

関数メンバ

CDAQMXAOPWMList::add

構文

```
int add(CDAQMXAOPWMData * pcMXAOPWMData);
```

引数

pcMXAOPWMData　データをポインタで指定します。

説明

指定されたデータをリストに追加し，データ識別子を生成します。
追加できなかった場合，負の値を返します。

戻り値

データ識別子を返します。

参照

addData

CDAQMXAOPWMList::CDAQMXAOPWMList

構文

```
CDAQMXAOPWMList(void);
virtual ~CDAQMXAOPWMList(void);
```

説明

オブジェクトを構築，消滅します。
構築時，データメンバを初期化します。
消滅時，リスト内のデータを削除します。

参照

```
initCurrent
CDAQMXList::CDAQMXList
```

CDAQMXAOPWMList::change

構文

```
void change(int idAOPWM, int aopwmNo, int bValid, int
iAOPWMValue = 0);
```

引数

idAOPWM	AO/PWMデータ識別子を指定します。
aopwmNo	AO/PWMデータ番号を指定します。
bValid	有効/無効を有効無効値で指定します。
iAOPWMValue	出力データ値を指定します。

説明

指定されたAO/PWMデータ識別子のAO/PWMデータを変更します。
データ識別子に定数値の「全データ識別子指定」をした場合，リストにあるすべてのデータを処理します。
AO/PWMデータ番号に定数値の「全AO/PWMデータ番号指定」をした場合，データ内のすべてを処理します。

参照

```
getClassMXAOPWMData
CDAQMXAOPWMData::setAOPWM
```

CDAQMXAOPWMList::copy

構文

```
virtual void copy(int idxNo, int idxSrc);
```

引数

idxNo	複写先のデータ識別子を指定します。
idxSrc	複写元のデータ識別子を指定します。

説明

複写元から複写先へデータ識別子の示すデータの内容を複写します。

参照

```
copyData
getClassMXAOPWMData
```

CDAQMXAOPWMList::copyData

構文

```
void copyData(int idAOPWM, CDAQMXAOPWMData * pcMXAOPWMData);
```

引数

idAOPWM　　AO/PWMデータ識別子を指定します。
pcMXAOPWMData　データをポインタで指定します。

説明

指定されたデータ識別子のデータにポインタで指定されたデータを複写します。
データ識別子に定数値の「全データ識別子指定」をした場合，リストにあるすべてのデータを処理します。

参照

`getClassMXAOPWMData`

CDAQMXAOPWMList::create

構文

```
virtual int create(void);
```

説明

データを作成し，リストに追加します。

戻り値

データ識別子を返します。

参照

`add`
`CDAQMXAOPWMData::CDAQMXAOPWMData`

CDAQMXAOPWMList::getClassMXAOPWMData

構文

```
CDAQMXAOPWMData * getClassMXAOPWMData(int idAOPWM);
```

引数

idAOPWM　　AO/PWMデータ識別子を指定します。

説明

指定されたデータ識別子のデータを取得します。
データ識別子に定数値の「カレントデータ指定」をした場合，データメンバから現在のデータ領域を取得します。
存在しない場合，NULLを返します。

戻り値

データへのポインタを返します。

参照

`getCurrent getData`

CDAQMXAOPWMList::getCurrent

構文

```
CDAQMXAOPWMData & getCurrent(void);
```

説明

データメンバから現在のデータ領域を取得します。

戻り値

データへの参照を返します。

CDAQMXAOPWMList::initCurrent

構文

```
void initCurrent(void);
```

説明

データメンバの現在のデータ領域を初期化します。

参照

```
getCurrent
CDAQMXAOPWMData::initialize
```

CDAQMXBalanceListクラス

CDAQMXList
CDAQMXBalanceList

本クラスは，初期バランスデータを管理するクラスです。
あらかじめ送信する初期バランスデータを作成，保持することができます。データ識別子で識別します。
現在のデータには，結果も保存できます。

パブリックメンバ

構築・消滅

CDAQMXBalanceList	オブジェクトを構築します。
~CDAQMXBalanceList	オブジェクトを消滅します。

データメンバ操作

add	初期バランスデータを追加します。
change	初期バランスデータを変更します。
copyData	初期バランスデータを複写します。
getClassMXBalanceData	初期バランスデータを取得します。
initCurrent	現在の初期バランスデータを初期化します。
getCurrent	現在の初期バランスデータを取得します。

●オーバライドしたメンバ

データメンバ操作

create	初期バランスデータを作成します。
copy	初期バランスデータを複写します。

●継承するメンバ

CDAQMXList参照
del
getMaxNo
getNum
initialize
isData

プロテクトメンバ

データメンバ

m_cCurrent	現在の初期バランスデータを格納する領域です。

●継承するメンバ

CDAQMXList参照
m_list
m_num
addData
delData
getData

プライベートメンバ

なし。

関数メンバ

CDAQMXBalanceList::add

構文

```
int add(CDAQMXBalanceData * pcMXBalanceData);
```

引数

pcMXBalanceData　データをポインタで指定します。

説明

指定されたデータをリストに追加し，データ識別子を生成します。
追加できなかった場合，負の値を返します。

戻り値

データ識別子を返します。

参照

addData

CDAQMXBalanceList::CDAQMXBalanceList

構文

```
CDAQMXBalanceList(void);
virtual ~CDAQMXBalanceList(void);
```

説明

オブジェクトを構築，消滅します。
構築時，データメンバを初期化します。
消滅時，リスト内のデータを削除します。

参照

initCurrent
CDAQMXList::CDAQMXList

CDAQMXBalanceList::change

構文

```
void change(int idBalance, int balanceNo, int bValid, int
iValue = 0);
```

引数

idBalance	初期バランスデータ識別子を指定します。
balanceNo	初期バランスデータ番号を指定します。
bValid	有効/無効を有効無効値で指定します。
iValue	初期バランス値を指定します。

説明

指定された初期バランスデータ識別子の初期バランスデータを変更します。
データ識別子に定数値の「全データ識別子指定」をした場合，リストにあるすべてのデータを処理します。
初期バランスデータ番号に定数値の「全初期バランス番号指定」をした場合，データ内のすべてを処理します。

参照

```
getClassMXBalanceData
CDAQMXBalanceData::setBalance
```

CDAQMXBalanceList::copy

構文

```
virtual void copy(int idxNo, int idxSrc);
```

引数

idxNo	複写先のデータ識別子を指定します。
idxSrc	複写元のデータ識別子を指定します。

説明

複写元から複写先へデータ識別子の示すデータの内容を複写します。

参照

```
copyData
getClassMXBalanceData
```

CDAQMXBalanceList::copyData

構文

```
void copyData(int idBalance, CDAQMXBalanceData *
pcMXBalanceData);
```

引数

idBalance　　初期バランスデータ識別子を指定します。
pcMXBalanceData　データをポインタで指定します。

説明

指定されたデータ識別子のデータにポインタで指定されたデータを複写します。
データ識別子に定数値の「全データ識別子指定」をした場合，リストにあるすべてのデータを処理します。

参照

`getClassMXBalanceData`

CDAQMXBalanceList::create

構文

```
virtual int create(void);
```

説明

データを作成し，リストに追加します。

戻り値

データ識別子を返します。

参照

`add`
`CDAQMXBalanceData::CDAQMXBalanceData`

CDAQMXBalanceList::getClassMXBalanceData

構文

```
CDAQMXBalanceData * getClassMXBalanceData(int idBalance);
```

引数

idBalance　　初期バランスデータ識別子を指定します。

説明

指定されたデータ識別子のデータを取得します。
データ識別子に定数値の「カレントデータ指定」をした場合，データメンバから現在のデータ領域を取得します。
存在しない場合，NULLを返します。

戻り値

データへのポインタを返します。

参照

`getCurrent`
`getData`

CDAQMXBalanceList::getCurrent

構文

```
CDAQMXBalanceResult & getCurrent(void);
```

説明

データメンバから現在のデータ領域を取得します。

戻り値

データへの参照を返します。

CDAQMXBalanceList::initCurrent

構文

```
void initCurrent(void);
```

説明

データメンバの現在のデータ領域を初期化します。

参照

```
getCurrent
CDAQMXBalanceResult::initialize
```

CDAQMXDataBufferクラス

本クラスは，MX100のチャネルごとの各種情報をまとめたクラスです。
FIFOによる複数データを格納することができます。リストで管理されます。格納位置順序番号(インデックス番号)で指定します。現在位置をカレントインデックス番号で示すことができます。
以下のデータを格納することができます。

- チャネル情報データ
- 測定データ
- 時刻情報データ

パブリックメンバ

構築・消滅

CDAQMXDataBuffer	オブジェクトを構築します。
~CDAQMXDataBuffer	オブジェクトを消滅します。

データメンバ操作

initialize	データメンバを初期化します。
getClassMXChInfo	チャネル情報データを取得します。
create	格納領域を構築します。
setChInfo	チャネル情報データを格納します。
setDataInfo	測定データを格納します。
setDateTime	時刻情報データを格納します。
currentDataInfo	現在の測定データを取得します。
currentDateTime	現在の時刻情報データを取得します。

ユーティリティ

next	カレントインデックス番号を次に進めます。
getDataNum	残りのデータ個数を取得します。
isCurrent	現在のデータが存在するかをチェックします。

プロテクトメンバ

データメンバ

m_cMXChInfo	チャネル情報データの格納領域です。
m_pDataBuf	測定データリストの先頭ポインタ領域です。
m_pTimeBuf	時刻情報データリストの先頭ポインタ領域です。
m_cur	カレントインデックス番号の格納領域です。
m_max	有効要素個数の格納領域です。
m_num	リスト要素個数の格納領域です。

データメンバ操作

getDataInfo	測定データを取得します。
getDateTime	時刻情報データを取得します。

プライベートメンバ

なし。

関数メンバ

CDAQMXDataBuffer::CDAQMXDataBuffer

構文

```
CDAQMXDataBuffer(void);
CDAQMXDataBuffer(CDAQMXChInfo & cMXChInfo);
virtual ~CDAQMXDataBuffer(void);
```

引数

cMXChInfo　　チャネル情報データを指定します。

説明

オブジェクトを構築，消滅します。
構築時，データメンバを初期化します。引数にチャネル情報データが指定された場合，データメンバに複写します。
消滅時，データメンバの各種情報を消滅させます。

参照

```
getClassMXChInfo initialize setChInfo
CDAQMXChInfo::initialize
```

CDAQMXDataBuffer::create

構文

```
int create(int num);
```

引数

num　　データ個数を指定します。

説明

指定されたデータ個数分のリストを構築します。
構築後，リストの先頭をカレントインデックス番号にします。
既に構築されている場合，指定されたデータ個数分にリストを拡張します。
指定されたデータ個数が有効要素個数になります。
構築，拡張した分のリストの要素にデータは存在しません。
構築できなかった場合，初期化されます。

戻り値

エラー番号を返します。
エラー：
Not Data　　データがありません。指定が負の数です。
Exception　　例外が発生しました。領域を構築できませんでした。

参照

```
initialize
```

CDAQMXDataBuffer::currentDataInfo

構文

```
CDAQMXDataInfo * currentDataInfo(void);
```

説明

カレントインデックス番号の示す測定データを取得します。
存在しない場合，NULLを返します。

戻り値

オブジェクトへのポインタを返します。

CDAQMXDataBuffer::currentDateTime

構文

```
CDAQMXDateTime * currentDateTime(void);
```

説明

カレントインデックス番号の示す時刻情報データを取得します。
存在しない場合，NULLを返します。

戻り値

オブジェクトへのクラスのポインタを返します。

CDAQMXDataBuffer::getClassMXChInfo

構文

```
CDAQMXChInfo & getClassMXChInfo(void);
```

説明

データメンバからチャネル情報データの格納領域を取得します。

戻り値

オブジェクトへの参照を返します。

CDAQMXDataBuffer::getDataInfo

構文

```
CDAQMXDataInfo * getDataInfo(int index);
```

引数

index　　格納位置順序番号を指定します。

説明

データメンバから指定された位置の測定データを取得します。
存在しない場合，NULLを返します。

戻り値

測定データへのポインタを返します。

CDAQMXDataBuffer::getDataNum

構文

```
int getDataNum(void);
```

説明

有効要素個数とカレントインデックス番号の差分から，残りのデータ個数を取得します。

戻り値

残りのデータ個数を返します。

CDAQMXDataBuffer::getDateTime

構文

```
CDAQMXDateTime * getDateTime(int index);
```

引数

index　　格納位置順序番号を指定します。

説明

データメンバから指定された位置の時刻情報データを取得します。
存在しない場合，NULLを返します。

戻り値

時刻情報データへのポインタを返します。

CDAQMXDataBuffer::initialize

構文

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

説明

データメンバを初期化します。
測定データと時刻情報データの格納領域を消滅させます。
チャネル情報データの格納領域は初期化しません。

CDAQMXDataBuffer::isCurrent

構文

```
int isCurrent(void);
```

説明

現在位置(カレントインデックス番号)のデータが有効か否かをチェックします。

戻り値

有効無効値を返します。

参照

currentDataInfo

CDAQMXDataBuffer::next

構文

```
int next(void);
```

説明

カレントインデックス番号をインクリメントします。
有効要素個数を超えた場合，格納位置が存在しないものとし，負の数とします。

戻り値

カレントインデックス番号を返します。

CDAQMXDataBuffer::setChInfo

構文

```
void setChInfo(CDAQMXChInfo & cMXChInfo);
```

引数

cMXChInfo　　チャネル情報データを指定します。

説明

指定されたチャネル情報データをデータメンバに複写します。

CDAQMXDataBuffer::setDataInfo

構文

```
int setDataInfo(int index, CDAQMXDataInfo & cMXDataInfo);
```

引数

index　　格納位置順序番号を指定します。
cMXDataInfo　　測定データを指定します。

説明

指定された測定データをデータメンバの指定位置領域に複写します。
リストの指定位置の要素にデータが存在しない場合，データを構築します。
チャネル情報データへの参照は，複写されず，データメンバのチャネル情報データ領域になります。

戻り値

エラー番号を返します。
エラー：
Not Data　　データがありません。格納位置順序番号が範囲外です。
Exception　　例外が発生しました。領域を構築できませんでした。

参照

```
getClassMXChInfo
CDAQMXDataInfo::CDAQMXDataInfo
CDAQMXDataInfo::setClassMXChInfo
```

CDAQMXDataBuffer::setDateTime

構文

```
int setDateTime(int index, CDAQMXDateTime & cMXDateTime);
```

引数

index	格納位置順序番号を指定します。
cMXDateTime	時刻情報データを指定します。

説明

指定された時刻情報データをデータメンバの指定位置に複写します。
リストの指定位置の要素にデータが存在しない場合，データを構築します。

戻り値

エラー番号を返します。
エラー：

Not Data	データがありません。格納位置順序番号が範囲外です。
Exception	例外が発生しました。領域を構築できませんでした。

参照

`CDAQMXDateTime::CDAQMXDateTime`

CDAQMXDOListクラス

CDAQMXList
CDAQMXDOList

本クラスは，コマンドDOの出力であるDOデータを管理するクラスです。
あらかじめ送信するDOデータを作成，保持することができます。データ識別子で識別します。

パブリックメンバ

構築・消滅

CDAQMXDOList	オブジェクトを構築します。
~CDAQMXDOList	オブジェクトを消滅します。

データメンバ操作

add	DOデータを追加します。
change	DOデータを変更します。
copyData	DOデータを複写します。
getClassMXDOData	DOデータを取得します。
initCurrent	現在のDOデータを初期化します。
getCurrent	現在のDOデータを取得します。

●オーバライドしたメンバ

データメンバ操作

create	DOデータを作成します。
copy	DOデータを複写します。

●継承するメンバ

CDAQMXList参照
del
getMaxNo
getNum
initialize
isData

プロテクトメンバ

データメンバ

m_list	リストの先頭ポインタです。
m_num	リストの要素個数の格納領域です。

データメンバ操作

addData	データをリストに追加します。
delData	データをリストから削除します。
getData	データをリストから取得します。

プライベートメンバ

なし。

関数メンバ

CDAQMXDOList::add

構文

```
int add(CDAQMXDOData * pcMXDOData);
```

引数

pcMXDOData　データをポインタで指定します。

説明

指定されたデータをリストに追加し，データ識別子を生成します。
追加できなかった場合，負の値を返します。

参照

addData

CDAQMXDOList::CDAQMXDOList

構文

```
CDAQMXDOList(void);
virtual ~CDAQMXDOList(void);
```

説明

オブジェクトを構築，消滅します。
構築時，データメンバを初期化します。
消滅時，リスト内のデータを削除します。

参照

initCurrent
CDAQMXList::CDAQMXList

CDAQMXDOList::change

構文

```
void change(int idDO, int doNo, int bValid, int bONOFF =
DAQMX_VALID_OFF);
```

引数

idDO	DOデータ識別子を指定します。
doNo	DOデータ番号を指定します。
bValid	有効/無効を有効無効値で指定します。
bONOFF	ON/OFFを有効無効値で指定します。

説明

指定されたDOデータ識別子のDOデータを変更します。
データ識別子に定数値の「全データ識別子指定」をした場合，リストにあるすべてのデータを処理します。
DOデータ番号に定数値の「全DO番号指定」をした場合，データ内のすべてを処理します。

参照

```
getClassMXDOData
CDAQMXDOData::setDO
```

CDAQMXList::copy

構文

```
virtual void copy(int idxNo, int idxSrc);
```

引数

idxNo	複写先のデータ識別子を指定します。
idxSrc	複写元のデータ識別子を指定します。

説明

複写元から複写先へデータ識別子の示すデータの内容を複写します。
オーバライドしないと，何もしません。

CDAQMXDOList::copyData

構文

```
void copyData(int idDO, CDAQMXDOData * pcMXDOData);
```

引数

idDO　　DOデータ識別子を指定します。
pcMXDOData　データをポインタで指定します。

説明

指定されたデータ識別子のデータにポインタで指定されたデータを複写します。
データ識別子に定数値の「全データ識別子指定」をした場合，リストにあるすべてのデータを処理します。

参照

```
getClassMXDOData
```

CDAQMXDOList::create

構文

```
virtual int create(void);
```

説明

データを作成し，リストに追加します。

戻り値

データ識別子を返します。

参照

```
add
CDAQMXDOData::CDAQMXDOData
```

CDAQMXDOList::getClassMXDOData

構文

```
CDAQMXDOData * getClassMXDOData(int idDO);
```

引数

idDO　　DOデータ識別子を指定します。

説明

指定されたデータ識別子のデータを取得します。
データ識別子に定数値の「カレントデータ指定」をした場合，データメンバから現在のデータ領域を取得します。
存在しない場合，NULLを返します。

戻り値

データへのポインタを返します。

参照

```
getCurrent getData
```

CDAQMXDOList::getCurrent

構文

```
CDAQMXDOData & getCurrent(void);
```

説明

データメンバから現在のデータ領域を取得します。

戻り値

データへの参照を返します。

CDAQMXDOList::initCurrent

構文

```
void initCurrent(void);
```

説明

データメンバの現在のデータ領域を初期化します。

参照

```
getCurrent
CDAQMXDOData::initialize
```

CDAQMXItemConfigクラス

CDAQCongig
CDAQMXItemConfig

本クラスは，設定項目で設定内容にアクセスする機能を提供する設定データのクラスです。
設定項目によるアクセスは，各内容の領域を読み出し，または，書き込みします。未使用や不定，不正な内容でも，原則そのまま処理されます。内容の整合性をチェックしません。
内容の領域を文字列で表す機能をサポートします。種類を定数で定義されている場合，各種類を示す文字列で表します。CFステータス種類，参照アラームのように複数の表記を表す場合，「,」で区切って表します。
項目名称の文字列は，定数の項を参照してください。
関数メンバの戻り値において，存在しない場合とは，主に内容の領域が存在しない場合です。
各項目値と小数点位置から浮動小数の値で取得する機能をサポートします。

パブリックメンバ

構築・消滅

CDAQMXItemConfig	オブジェクトを構築します。
~CDAQMXItemConfig	オブジェクトを消滅します。

設定項目操作

readItem	設定項目を読み出します。
writeItem	設定項目を書き込みます。

データメンバ操作

getHisterisys	ヒステリシスを取得します。
getDoubleHisterisys	ヒステリシスを浮動小数で取得します。
getDoubleAlarmON	アラームON値を浮動小数で取得します。
getDoubleAlarmOFF	アラームOFF値を浮動小数で取得します。
getDoubleSpanMin	スパン最小値を浮動小数で取得します。
getDoubleSpanMax	スパン最大値を浮動小数で取得します。
getDoubleScaleMin	スケール最小値を浮動小数で取得します。
getDoubleScaleMax	スケール最大値を浮動小数で取得します。
getDoublePresetValue	選択値が「指定値」の場合の値を浮動小数で取得します。

ユーティリティ

toItemName	設定項目文字列を取得します。
getItemNo	設定項目番号を取得します。
getMaxLenItemName	設定項目文字列の最大長を取得します。

●オーバライドしたメンバ

ユーティリティ

isObject	オブジェクトをチェックします。

●継承するメンバ

CDAQMXConfig参照

```
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
setScalling setSKIP setSTRAIN setTC setTempUnit setVOLT
```

プロテクトメンバ

●継承するメンバ

CDAQMXConfig参照

```
m_cMXSysInfo m_cMXStatus m_cMXNetInfo m_cMXChConfigData
```

プライベートメンバ

なし。

関数メンバ

CDAQMXItemConfig::CDAQMXItemConfig

構文

```
CDAQMXItemConfig(MXConfigData * pMXConfigData = NULL);
virtual ~CDAQMXItemConfig(void);
```

引数

pMXConfigData	設定データを指定します。

説明

オブジェクトを構築，消滅します。

構築時，データメンバに指定された値を設定します。指定がない場合，データメンバを初期化します。

参照

```
CDAQMXConfig::CDAQMXConfig
```

CDAQMXItemConfig::getDoubleAlarmOFF

構文

```
double getDoubleAlarmOFF(int chNo, int levelNo);
```

引数

chNo　　チャネル番号を指定します。
levelNo　　アラームレベルを指定します。

説明

指定されたチャネル番号，アラームレベルのアラーム停止のしきい値(Off値)を浮動小数値で取得します。
存在しない場合，0.0を返します。

戻り値

アラーム停止のしきい値(Off値)を浮動小数値で返します。

参照

```
getClassMXChConfig
CDAQMXChConfig::getAlarmValueOFF
CDAQMXChConfig::getPoint
CDAQMXDataInfo::toDoubleValue
```

CDAQMXItemConfig::getDoubleAlarmON

構文

```
double getDoubleAlarmON(int chNo, int levelNo);
```

引数

chNo　　チャネル番号を指定します。
levelNo　　アラームレベルを指定します。

説明

指定されたチャネル番号，アラームレベルのアラーム発生のしきい値(On値)を浮動小数値で取得します。
存在しない場合，0.0を返します。

戻り値

アラーム発生のしきい値(On値)を浮動小数値で返します。

参照

```
getClassMXChConfig
CDAQMXChConfig::getAlarmValueON
CDAQMXChConfig::getPoint
CDAQMXDataInfo::toDoubleValue
```

CDAQMXItemConfig::getDoubleHisterisys

構文

```
double getDoubleHisterisys(int chNo, int levelNo);
```

引数

chNo　　チャネル番号を指定します。
levelNo　　アラームレベルを指定します。

説明

指定されたチャネル番号，アラームレベルのヒステリシスを浮動小数値で取得します。
存在しない場合，0.0を返します。

戻り値

ヒステリシスを浮動小数値で返します。

参照

```
getClassMXChConfig
getHisterisys
CDAQMXChConfig::getPoint
CDAQMXDataInfo::toDoubleValue
```

CDAQMXItemConfig::getDoublePresetValue

構文

```
double getDoublePresetValue(int outputNo);
```

引数

outputNo　　出力チャネルデータ番号を指定します。

説明

指定された出力チャネルデータ番号の選択値が「指定値」の場合の値を浮動小数値で取得します。
存在しない場合，0.0を返します。

戻り値

選択値が「指定値」の場合の値を浮動小数値で返します。

参照

```
getClassMXOutputData
getSpanPoint
CDAQMXOutputData::getPresetValue
CDAQMXDataInfo::toDoubleValue
```

CDAQMXItemConfig::getDoubleScaleMax

構文

```
double getDoubleScaleMax(int chNo);
```

引数

chNo　　チャネル番号を指定します。

説明

指定されたチャネル番号のスケール最大値の値を浮動小数値で取得します。
存在しない場合，0.0を返します。

戻り値

スケール最大値を浮動小数値で返します。

参照

```
getClassMXChConfig
CDAQMXChConfig::getPoint
CDAQMXChConfig::getScaleMax
CDAQMXDataInfo::toDoubleValue
```

CDAQMXItemConfig::getDoubleScaleMin

構文

```
double getDoubleScaleMin(int chNo);
```

引数

chNo　　チャネル番号を指定します。

説明

指定されたチャネル番号のスケール最小値の値を浮動小数値で取得します。
存在しない場合，0.0を返します。

戻り値

スケール最小値を浮動小数値で返します。

参照

```
getClassMXChConfig
CDAQMXChConfig::getPoint
CDAQMXChConfig::getScaleMin
CDAQMXDataInfo::toDoubleValue
```

CDAQMXItemConfig::getDoubleSpanMax

構文

```
double getDoubleSpanMax(int chNo);
```

引数

chNo　　チャネル番号を指定します。

説明

指定されたチャネル番号のスパン最大値の値を浮動小数値で取得します。
存在しない場合，0.0を返します。

戻り値

スパン最大値を浮動小数値で返します。

参照

```
getClassMXChConfig getSpanPoint
CDAQMXChConfig::getSpanMax
CDAQMXDataInfo::toDoubleValue
```

CDAQMXItemConfig::getDoubleSpanMin

構文

```
double getDoubleSpanMin(int chNo);
```

引数

chNo　　チャネル番号を指定します。

説明

指定されたチャネル番号のスパン最小値の値を浮動小数値で取得します。
存在しない場合，0.0を返します。

戻り値

スパン最小値を浮動小数値で返します。

参照

```
getClassMXChConfig getSpanPoint
CDAQMXChConfig::getSpanMin
CDAQMXDataInfo::toDoubleValue
```

CDAQMXItemConfig::getHisterisys

構文

```
int getHisterisys(int chNo, int levelNo);
```

引数

chNo　　チャネル番号を指定します。
levelNo　　アラームレベルを指定します。

説明

指定されたチャネル番号，アラームレベルのヒステリシスを取得します。
存在しない場合，0を返します。

戻り値

ヒステリシスを返します。

参照

```
getClassMXChConfig
CDAQMXChConfig::getAlarmValueOFF
CDAQMXChConfig::getAlarmValueON
```

CDAQMXItemConfig::getMaxLenItemName

構文

```
static int getMaxLenItemName(void);
```

説明

設定項目の項目名称の文字列の最大長を取得します。
戻り値には，終端は含まれません。

戻り値

文字列の長さを返します。

CDAQMXItemConfig::isObject

構文

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

引数

classname　　クラス名を文字列で指定します。

説明

指定されたクラス名を継承しているかをチェックします。
引数が省略された場合，本クラスであるかをチェックします。
本クラスを継承したクラスが，自分自身のクラスをチェックするためには，オーバーライドする必要があります。
クラスを継承している場合，「有効値」(真)を返します。それ以外は，「無効値」(偽)を返します。
本クラスと異なる場合，親クラスをチェックします。

戻り値

有効無効値を返します。

参照

CDAQMXConfig::isObject

CDAQMXItemConfig::readItem

構文

```
int readItem(int itemNo, char * strItem, int lenItem);
```

引数

itemNo　　設定項目を指定します。
strItem　　文字列を格納する領域を指定します。
lenItem　　文字列を格納する領域のバイト数を指定します。

説明

指定された設定項目の内容を文字列で取得します。
指定された格納先に文字列を格納します。
格納される文字列は，原則ascii文字列です。
存在しない場合，0を返します。

戻り値

実際の文字列の長さを返します。

参照

getClassMXBalanceData getClassMXChConfig getClassMXNetInfo
getClassMXOutputData getClassMXStatus getClassMXSysInfo

CDAQMXItemConfig::toItemName

構文

```
static int toItemName(int itemNo, char * strItem, int
lenItem);
```

引数

itemNo	設定項目を指定します。
strItem	文字列を格納する領域を指定します。
lenItem	文字列を格納する領域のバイト数を指定します。

説明

指定された設定項目の項目名称を文字列で取得します。
指定された格納先に文字列を格納します。
領域に格納する文字列には，終端も含まれます。戻り値には，終端は含まれません。
存在しない場合，空文字列を格納し，0を返します。
格納される文字列は，原則ascii文字列です。

戻り値

文字列の長さを返します。

CDAQMXItemConfig::toItemNo

構文

```
static int toItemNo(const char * strItem);
```

引数

strItem	設定項目の項目名称を文字列で指定します。

説明

文字列から設定項目を取得します。
存在しない場合，「不明」を返します。
指定する文字列は，原則ascii文字列です。

戻り値

設定項目を返します。

CDAQMXItemConfig::writeItem

構文

```
int writeItem(int itemNo, const char * strItem);
```

引数

itemNo　　設定項目を指定します。
strItem　　内容を文字列で指定します。

説明

指定された設定項目に指定された内容を設定します。
文字列の書式は，設定項目の読み出し関数メンバの出力に従います。
指定する文字列は，原則ascii文字列です。

戻り値

エラー番号を返します。
エラー：
Not Support　　設定項目，または，内容がサポートされていません。

参照

`getClassMXBalanceData getClassMXChConfig getClassMXNetInfo getClassMXOutputData getClassMXStatus getClassMXSysInfo`

CDAQMXListクラス

CDAQMXList

本クラスは，ユーザが作成するデータを管理するクラスです。
登録されたデータは，データ識別子で識別します。データ識別子は，リストのインデックス番号に相当します。
本クラスは，作成するデータを定義していません。データメンバ操作の関数メンバをオーバライドします。

パブリックメンバ

構築・消滅

CDAQMXList	オブジェクトを構築します。
~CDAQMXList	オブジェクトを消滅します。

データメンバ操作

initialize	データメンバを初期化します。
create	データを作成します。
del	データを削除します。
copy	データを複写します。

ユーティリティ

getNum	データ個数を取得します。
getMaxNo	データ識別子の最大値を取得します。
isData	データの存在をチェックします。

プロテクトメンバ

データメンバ

m_list	リストの先頭ポインタです。
m_num	リストの要素個数の格納領域です。

データメンバ操作

addData	データをリストに追加します。
delData	データをリストから削除します。
getData	データをリストから取得します。

プライベートメンバ

なし。

関数メンバ

CDAQMXList::addData

構文

```
int addData(void * pData);
```

引数

pData　　データをポインタで指定します。

説明

指定されたデータをリストに追加し，データ識別子を生成します。
追加できなかった場合，負の値を返します。

戻り値

データ識別子を返します。

CDAQMXList::CDAQMXList

構文

```
CDAQMXList(void);
virtual ~CDAQMXList(void);
```

説明

オブジェクトを構築，消滅します。
構築時，データメンバを初期化します。
消滅時，リスト内のデータを削除します。

参照

`initialize`

CDAQMXList::copy

構文

```
virtual void copy(int idxNo, int idxSrc);
```

引数

idxNo　　複写先のデータ識別子を指定します。
idxSrc　　複写元のデータ識別子を指定します。

説明

複写元から複写先へデータ識別子の示すデータの内容を複写します。
オーバライドしないと，何もしません。

CDAQMXList::create

構文

```
virtual int create(void);
```

説明

データを作成し，リストに追加します。
オーバライドしないと，データは作成されません。

戻り値

データ識別子を返します。

参照

addData

CDAQMXList::del

構文

```
virtual void del(int idxNo);
```

引数

idxNo　　データ識別子を指定します。

説明

指定されたデータ識別子のデータをリストから削除し，消滅させます。
定数値の「全データ識別子指定」をした場合，リストにあるすべてのデータを処理します。

参照

delData

CDAQMXList::delData

構文

```
void delData(int idxNo);
```

引数

idxNo　　データ識別子を指定します。

説明

指定されたデータ識別子のデータをリストから削除し，消滅させます。
定数値の「全データ識別子指定」をした場合，リストにあるすべてのデータを処理します。

参照

getData

CDAQMXList::getData

構文

```
void * getData(int idxNo);
```

引数

idxNo　　データ識別子を指定します。

説明

指定されたデータ識別子のデータを取得します。
存在しない場合，NULLを返します。

戻り値

データへのポインタを返します。

CDAQMXList::getMaxNo

構文

```
int getMaxNo(void);
```

説明

リスト内に存在するデータのデータ識別子の最大値を取得します。

戻り値

データ識別子を返します。

CDAQMXList::getNum

構文

```
int getNum(void);
```

説明

データ個数を取得します。
リスト内に存在するデータの個数を返します。

戻り値

データ個数を返します。

CDAQMXList::initialize

構文

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

説明

データメンバを初期化します。
リストにある全てのデータを消滅させします。

参照

```
delData
```

CDAQMXList::isData

構文

```
int isData(int idxNo);
```

引数

idxNo　　データ識別子を指定します。

説明

指定されたデータ識別子のデータがリストに存在するかをチェックします。
存在した場合，「有効値」を返します。それ以外は，「無効値」を返します。

戻り値

有効無効値を返します。

CDAQMXTransmitListクラス

CDAQMXList
　　CDAQMXTransmitList

本クラスは，伝送出力の指定である伝送出力データを管理するクラスです。
あらかじめ送信する伝送出力データを作成，保持することができます。データ識別子で識別します。

パブリックメンバ

構築・消滅

CDAQMXTransmitList	オブジェクトを構築します。
~CDAQMXTransmitList	オブジェクトを消滅します。

データメンバ操作

add	伝送出力データを追加します。
change	伝送出力データを変更します。
copyData	伝送出力データを複写します。
getClassMXTransmit	伝送出力データを取得します。
initCurrent	現在の伝送出力データを初期化します。
getCurrent	現在の伝送出力データを取得します。

●オーバライドしたメンバ

データメンバ操作

create	伝送出力データを作成します。
copy	伝送出力データを複写します。

●継承するメンバ

CDAQMXList参照

```
del
getMaxNo
getNum
initialize
isData
```

プロテクトメンバ

データメンバ

m_cCurrent	現在の伝送出力データを格納する領域です。

●継承するメンバ

CDAQMXList参照
m_list
m_num
addData
delData
getData

プライベートメンバ

なし。

関数メンバ

CDAQMXTransmitList::add

構文

```
int add(CDAQMXTransmit * pcMXTransmit);
```

引数

pcMXTransmit　データをポインタで指定します。

説明

指定されたデータをリストに追加し，データ識別子を生成します。
追加できなかった場合，負の値を返します。

戻り値

データ識別子を返します。

参照

addData

CDAQMXTransmitList::CDAQMXTransmitList

構文

```
CDAQMXTransmitList(void);
virtual ~CDAQMXTransmitList(void);
```

説明

オブジェクトを構築，消滅します。
構築時，データメンバを初期化します。
消滅時，リスト内のデータを削除します。

参照

initCurrent
CDAQMXList::CDAQMXList

CDAQMXTransmitList::change

構文

```
void change(int idTransmit, int aopwmNo, int iTransmit);
```

引数

idTransmit	伝送出力データ識別子を指定します。
aopwmNo	AO/PWMデータ番号を指定します。
iTransmit	伝送状態を指定します。

説明

指定された伝送出力データ識別子の伝送出力データを変更します。
データ識別子に定数値の「全データ識別子指定」をした場合，リストにあるすべてのデータを処理します。
伝送出力データ番号に定数値の「全伝送出力データ番号指定」をした場合，データ内のすべてを処理します。

参照

`getClassMXTransmit`
`CDAQMXTransmit::setTransmit`

CDAQMXTransmitList::copy

構文

```
virtual void copy(int idxNo, int idxSrc);
```

引数

idxNo	複写先のデータ識別子を指定します。
idxSrc	複写元のデータ識別子を指定します。

説明

複写元から複写先へデータ識別子の示すデータの内容を複写します。

参照

`copyData`
`getClassMXTransmit`

CDAQMXTransmitList::copyData

構文

```
void copyData(int idTransmit, CDAQMXTransmit * pcMXTransmit);
```

引数

idTransmit　　伝送出力データ識別子を指定します。
pcMXTransmit　データをポインタで指定します。

説明

指定されたデータ識別子のデータにポインタで指定されたデータを複写します。
データ識別子に定数値の「全データ識別子指定」をした場合，リストにあるすべてのデータを処理します。

参照

`getClassMXTransmit`

CDAQMXTransmitList::create

構文

```
virtual int create(void);
```

説明

データを作成し，リストに追加します。

戻り値

データ識別子を返します。

参照

`add`
`CDAQMXTransmit::CDAQMXTransmit`

CDAQMXTransmitList::getClassMXTransmit

構文

```
CDAQMXTransmit * getClassMXTransmit(int idTransmit);
```

引数

idTransmit　　伝送出力データ識別子を指定します。

説明

指定されたデータ識別子のデータを取得します。
データ識別子に定数値の「カレントデータ指定」をした場合，データメンバから現在のデータ領域を取得します。
存在しない場合，NULLを返します。

戻り値

データへのポインタを返します。

参照

`getCurrent`
`getData`

CDAQMXTransmitList::getCurrent

構文

```
CDAQMXTransmit & getCurrent(void);
```

説明

データメンバから現在のデータ領域を取得します。

戻り値

データへの参照を返します。

CDAQMXTransmitList::initCurrent

構文

```
void initCurrent(void);
```

説明

データメンバの現在のデータ領域を初期化します。

参照

```
getCurrent
CDAQMXTransmit::initialize
```

13.1 機能と関数の対応－MX100/Visual C－

拡張APIでサポートする機能と，Visual Cの関数郡の対応を示します。
状態遷移関数と取得関数の2種類あります。
状態遷移関数はMX100本体を制御します。データ取得機能で測定データを取得すると測定点が1点分だけ進みます(拡張APIの状態が遷移します)。
取得関数は項目値を返します。データ値取得すると現在の状態のデータ値を返します(拡張APIの状態は遷移しません)。

状態遷移関数

表の「FIFO」欄は，FIFO中に関数を実行したときの，FIFOの動作を示します。
停止：関数を実行するとFIFOを停止します。
継続：関数を実行してもFIFOを継続します。

通信機能

機能	FIFO	関数
MX100と通信接続	継続	openMX100
MX100との通信を切断	継続	closeMX100

FIFOの開始/停止

機能	FIFO	関数
FIFOを開始	継続	measStartMX100
FIFOを停止	継続	measStopMX100

制御機能

機能		FIFO	クラスと関数メンバ
日付時刻設定 現在時刻		停止	setDateTimeNowMX100
バックアップ 有効無効の設定		継続	switchBackupMX100
CFカードのフォーマット		停止	formatCFMX100
ユニット	システムの再構築	停止	reconstructMX100
	システムの初期化	停止	initSetValueMX100
	アラームリセット（アラームACK）	停止	ackAlarmMX100
7セグメントLEDの表示		継続	displaySegmentMX100
保持データの初期化	チャネル指定	継続	initDataChMX100
	FIFO指定	継続	initDataFIFOMX100

制御機能は，通信の最後に状態更新を行います。
各データの送信，設定機能については， データ操作機能を参照してください。
任意の時刻設定はできません。

設定機能

機能		FIFO	関数
設定データを一括設定(一括送信)	全設定データ	停止	sendConfigMX100
	基本設定データ	停止	sendConfigMX100
設定データを個別設定	システム構成データ	停止	sendConfigMX100
	チャネル設定データ	停止	sendConfigMX100
	初期バランスデータ	停止	sendConfigMX100
	出力チャネルデータ	停止	sendConfigMX100
初期バランスデータ	実行	停止	initBalanceMX100
	リセット	停止	clearBalanceMX100

設定データの設定機能は，保持しているデータを送信します。
任意の初期バランスデータを設定する場合，データ操作機能で初期バランスデータの送信機能を参照してください。

設定変更機能

設定機能は，設定送信して，状態更新を行います。
単独チャネル毎の設定なので，設定できなかった場合，原則エラーを返します。
データ値，または，測定値(倍精度浮動小数)での指定ができます。

レンジ設定

機能	FIFO	関数
スキップ	停止	setRangeMX100
直流電圧入力	停止	setRangeMX100
熱電対入力	停止	setRangeMX100
測温抵抗体	停止	setRangeMX100
ディジタル入力	停止	setRangeMX100
抵抗	停止	setRangeMX100
ひずみ	停止	setRangeMX100
AO	停止	setRangeMX100
PWM	停止	setRangeMX100
チャネル間差演算	停止	setChDELTAMX100
リモートRJC	停止	setChRRJCMX100
パルス	停止	setRangeMX100
通信	停止	setRangeMX100

チャネル設定

機能		FIFO	関数
単位名		停止	setChUnitMX100
タグ		停止	setChTagMX100
コメント		停止	setChCommentMX100
AI/DI/AO/PWM	スパン	停止	setSpanMX100 setDoubleSpanMX100
AI/DI	スケール	停止	setScaleMX100 setDoubleScaleMX100
	アラーム	停止	setAlarmMX100 setDoubleAlarmMX100 setAlarmValueMX100 setDoubleAlarmValueMX100
	ヒステリシス	停止	setHisterisysMX100 setDoubleHisterisysMX100
AI	フィルタ係数	停止	setFilterMX100
	基準接点補償(RJC)	停止	setRJCTypeMX100
	バーンアウト	停止	setBurnoutMX100
DO	非励磁	停止	setDeenergizeMX100
	保持	停止	setHoldMX100
	参照アラーム	停止	setRefAlarmMX100
チャネル種類	DO種類	停止	setChKindMX100
	AO種類	停止	setChKindMX100
	PWM種類	停止	setChKindMX100
PI	チャタリングフィルタ	停止	channelChatFilterMX100

モジュール設定

機能	FIFO	関数
周期種類	停止	setIntervalMX100
AD積分時間種類	停止	setIntegralMX100

ユニット設定

機能	FIFO	関数
ユニット番号	停止	setUnitNoMX100
温度単位種類	停止	setUnitTempMX100
CF書き込み種類	停止	setCFWriteModeMX100

出力チャネルデータ

機能	FIFO	関数
出力種類	停止	setOutputTypeMX100
選択値	停止	setChoiceMX100 setDoubleChoiceMX100
パルス周期倍率	停止	setPulseTimeMX100

データ操作機能

DOデータ

機能		FIFO	関数
作成		継続	createDOMX100
削除		継続	deleteDOMX100
部分変更	ユーザ指定	継続	changeDOMX100
	コピー	継続	copyDOMX100
送信	既存指定	継続	commandDOMX100
	変更指定	継続	switchDOMX100

AO/PWMデータ

機能		FIFO	関数
作成		継続	createAOPWMMX100
削除		継続	deleteAOPWMMX100
部分変更	出力データ値	継続	changeAOPWMMX100
	実出力値	継続	changeAOPWMValueMX100
	コピー	継続	copyAOPWMMX100
送信		継続	commandAOPWMMX100

初期バランスデータ

機能		FIFO	関数
作成		継続	createBalanceMX100
削除		継続	deleteBalanceMX100
部分変更	ユーザ指定	継続	changeBalanceMX100
	コピー	継続	copyBalanceMX100
送信		停止	commandBalanceMX100

伝送出力データ

機能		FIFO	関数
作成		継続	createTransmitMX100
削除		継続	deleteTransmitMX100
部分変更	ユーザ指定	継続	changeTransmitMX100
	コピー	継続	copyTransmitMX100
送信	既存指定	継続	commandTransmitMX100
	変更指定	継続	switchTransmitMX100

各データ識別子で操作します。
送信以外は，状態更新(通信)を行いません。

取得機能

機能			FIFO	関数
ステータスデータ			継続	updateStatusMX100
システム構成データ			継続	updateSystemMX100
設定データ			継続	updateConfigMX100
出力データ	DOデータ		継続	updateDODataMX100
	AO/PWMデータ 伝送出力データ		継続	updateAOPWMDataMX100
チャネル情報データ			継続	updateInfoChMX100
測定データ	チャネル指定	FIFO値	継続	measDataChMX100
		瞬時値	継続	measInstChMX100
	FIFO指定	FIFO値	継続	measDataFIFOMX100
		瞬時値	継続	measInstFIFOMX100
初期バランスデータ			継続	updateBalanceMX100
出力チャネルデータ			継続	updateOutputMX100

データ取得は，本API内部で一括取得が行われます。
収集によって，状態更新も行われます。
チャネル情報データや設定データ(システム構成データ，初期バランスデータ，出力チャネルデータを含む)は，内部で保持されていますが，ユーザが明示的に保持しているデータを更新できます。

設定項目

機能		FIFO	関数
設定データ	一括受信	継続	getItemAllMX100
	一括送信	停止	setItemAllMX100
設定項目	読み出し	継続	readItemMX100
	書き込み	継続	writeItemMX100
	初期化	継続	initItemMX100

設定項目の読み出し，書き込み，初期化は，保持している領域へのアクセスで，領域の整合性チェックをしません。また，状態更新(通信)を行いません。

取得関数

測定データ

データ名		関数
データ値		dataValueMX100
データステータス値		dataStatusMX100
アラーム（有無）		dataAlarmMX100
測定値	倍精度浮動小数	dataDoubleValueMX100
	文字列	dataStringValueMX100
時刻	秒数	dataTimeMX100
	ミリ秒	dataMilliSecMX100
	年	dataYearMX100
	月	dataMonthMX100
	日	dataDayMX100
	時	dataHourMX100
	分	dataMinuteMX100
	秒	dataSecondMX100
有効データ(有無)		dataValidMX100

チャネル情報データ

データ名	関数
FIFO番号	channelFIFONoMX100
FIFO内チャネル順序番号	channelFIFOIndexMX100
表示最小値	channelDisplayMinMX100
表示最大値	channelDisplayMaxMX100
実範囲最小値	channelRealMinMX100
実範囲最大値	channelRealMaxMX100

チャネル設定データ

データ名				関数
チャネルステータス(有無)				channelValidMX100
小数点位置				channelPointMX100
チャネル種類				channelKindMX100
レンジ種類				channelRangeMX100
スケール種類				channelScaleTypeMX100
単位名				toChannelUnitMX100 getChannelUnitMX100
タグ				toChannelTagMX100 getChannelTagMX100
コメント				toChannelCommentMX100 getChannelCommentMX100
AI/DI/AO/PWM				
	スパン	最小値	データ値	channelSpanMinMX100
			測定値	channelDoubleSpanMinMX100
		最大値	データ値	channelSpanMaxMX100
			測定値	channelDoubleSpanMaxMX100
AI/DI	スケール	最小値	データ値	channelScaleMinMX100
			測定値	channelDoubleScaleMinMX100
		最大値	データ値	channelScaleMaxMX100
			測定値	channelDoubleScaleMaxMX100
	アラーム種類			alarmTypeMX100
	アラーム値(ON値)		データ値	alarmValueONMX100
			測定値	alarmDoubleValueONMX100
	アラーム値(OFF値)		データ値	alarmValueOFFMX100
			測定値	alarmDoubleValueOFFMX100
	ヒステリシス		データ値	alarmHisterisysMX100
			測定値	alarmDoubleHisterisysMX100
AI	フィルタ係数			channelFilterMX100
	RJC種類			channelRJCTypeMX100
	RJC電圧値			channelRJCVoltMX100
	バーンアウト			channelBurnoutMX100
DO	非励磁			channelDeenergizeMX100
	保持			channelHoldMX100
	参照アラーム			channelRefAlarmMX100
チャネル間差演算/リモートRJC/AO/PWM				
	基準チャネル番号			channelRefChNoMX100
初期バランスデータ		有効無効値		channelBalanceValidMX100
		初期バランス値		channelBalanceValueMX100
出力チャネルデータ		出力種類		channelOutputTypeMX100
		アイドル時の選択値		channelIdleChoiceMX100
		エラー時の選択値		channelErrorChoiceMX100
		選択値が「指定値」の場合の値		
			データ値	channelPresetValueMX100
			測定値	channelDoublePresetValueMX100
		パルス周期倍率		channelPulseTimeMX100
PI	チャタリングフィルタ			channelChatFilterMX100

ネットワーク情報データ

データ名	関数
ホスト名	toNetHostMX100 getNetHostMX100
IPアドレス	netAddressMX100
ポート番号	netPortMX100
サブネットマスク	netSubmaskMX100
Gatewayアドレス	netGatewayMX100

システム構成データ

データ名		関数
モジュール	モジュール種類	moduleTypeMX100
	チャネル数	moduleChNumMX100
	周期種類	moduleIntervalMX100
	AD積分時間種類	moduleIntegralMX100
	有効無効値	moduleValidMX100
	起動時モジュール種類	moduleStandbyTypeMX100
	実際のモジュール種類	moduleRealTypeMX100
	端子種類	moduleTerminalMX100
	バージョン	moduleVersionMX100
	FIFO番号	moduleFIFONoMX100
	シリアル番号	toModuleSerialMX100 getModuleSerialMX100
ユニット	ユニット種類	unitTypeMX100
	スタイル	unitStyleMX100
	ユニット番号	unitNoMX100
	温度単位種類	unitTempMX100
	電源周波数	unitFrequencyMX100
	パート番号	toUnitPartNoMX100 getUnitPartNoMX100
	オプション	unitOptionMX100
	シリアル番号	toUnitSerialMX100 getUnitSreialMX100
	MACアドレス	unitMACMX100
	CF書き込み種類	unitCFWriteModeMX100

ステータスデータ

データ名		関数
ユニットステータス値		statusUnitMX100
FIFOの有効個数		statusFIFONumMX100
バックアップ(有無)		statusBackupMX100
FIFO	FIFOステータス値	statusFIFOMX100
	周期種類	statusFIFOIntervalMX100
CF	CFステータス種類	statusCFMX100
	サイズ	statusCFSizeMX100
	残容量	statusCFRemainMX100
ステータ返却時刻	秒数	statusTimeMX100
	ミリ秒	statusMilliSecMX100
	年	statusYearMX100
	月	statusMonthMX100
	日	statusDayMX100
	時	statusHourMX100
	分	statusMinuteMX100
	秒	statusSecondMX100

カレントデータ

データ名		関数
DOデータ	有効無効値	currentDOValidMX100
	ON/OFF状態	currentDOValueMX100
AO/PWMデータ	有効無効値	currentAOPWMValidMX100
	出力データ値	currentAOPWMValueMX100
	出力値	currentDoubleAOPWMValueMX100
初期バランスデータ	有効無効値	currentBalanceValidMX100
	初期バランス値	currentBalanceValueMX100
	初期バランス結果	currentBalanceResultMX100
伝送出力データ伝送状態		currentTransmitMX100

データ取得機能で取得された各データの状態です。
初期バランスデータの初期バランス結果は，設定機能による実行結果です。
DOデータ， AO/PWMデータなど，実際に出力されている出力状態をカレントデータとして取得できます。ただし，データを送信した直後は，設定した値が返却されて，実際の出力は次のタイミングになることがあります。
保持しているデータは，状態更新で取得したときの値です。取得関数を呼び出した時刻のデータではありません。

ユーザデータ

データ名		関数
DOデータ	有効無効値	userDOValidMX100
	ON/OFF状態	userDOValueMX100
AO/PWMデータ	有効無効値	userAOPWMValidMX100
	出力データ値	userAOPWMValueMX100
	出力値	userDoubleAOPWMValueMX100
初期バランスデータ	有効無効値	userBalanceValidMX100
	初期バランス値	userBalanceValueMX100
伝送出力データ	伝送状態	userTransmitMX100

ユーザがデータ操作機能で作成したデータの値を取得します。

ユーティリティ

機能/データ名		関数
残りデータ個数	チャネル単位で取得	dataNumChMX100
	FIFO単位で取得	dataNumFIFOMX100
エラー	MX固有エラーの取得	lastErrorMX100
	エラーメッセージ文字列を取得	toErrorMessageMX100 getErrorMessageMX100
	エラーメッセージ文字列の最大長を取得	errorMaxLengthMX100
	エラー検出した設定項目番号を取得	itemErrorMX100
FIFO情報から，チャネル番号に変換		channelNumberMX100
レンジ種類別の小数点位置を取得		rangePointMX100
測定値	倍精度浮動小数に変換	toDoubleValueMX100
	文字列に変換	toStringValueMX100
アラーム	アラーム種類の文字列を取得	toAlarmNameMX100 getAlarmNameMX100
	アラーム文字列の最大長を取得	alarmMaxLengthMX100
本APIのバージョン番号を取得		versionAPIMX100
本APIのリビジョン番号を取得		revisionAPIMX100
IPアドレスのパート分割を取得		addressPartMX100
AO/PWM	出力値を出力データ値に変換	toAOPWMValueMX100
	出力データ値を出力値に変換	toRealValueMX100
設定項目	設定項目番号から設定項目文字列を取得	toItemNameMX100
	設定項目文字列から設定項目番号を取得	toItemNoMX100
	設定項目文字列の最大長を取得	itemMaxLengthMX100
スタイルバージョンに変換		toStyleVersionMX100

13.2 プログラムーMX100/Visual Cー

インクルードファイルのパスを追加

プロジェクトに，インクルードファイル(DAQMX100.h)のパスを追加します。追加方法は，ご使用の環境により異なります。

ソースファイルでの宣言

ソースファイルに宣言を記述します。

```
#include "DAQMX100.h"
```

Note

共通部のインクルードファイル(DAQHandler.h)とMX100部のインクルードファイル(DAQMX.h))は，上記インクルードファイルから参照されているので，宣言を記述する必要はありません。

ロードライブラリの記述

本APIの実行可能モジュール(.dll)がプロセスとリンクできるようにするため，下記の記述をします。

本APIの実行可能モジュール(.dll)をアドレス空間内にマップします(LoadLibrary)。次に，実行可能モジュール内のエクスポート関数のアドレスを取得(GetProcAddress)します。

関数ポインタのコールバック型は，関数名に接頭語「DLL」をつけてすべて大文字にしたものです。本APIのインクルードファイルで定義されています。

```
HMODULE pDll = LoadLibrary("DAQMX100");
DLLOPENMX100 openMX100 = (DLLOPENMX100)GetProcAddress(pDll,
"openMX100");
```

測定データの取得

プログラム例

```
////////////////////////////////////////////////////////////
// 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;
  DLLMEASDATACHMX100 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 = (DLLMEASDATACHMX100)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;
}
////////////////////////////////////////////////////////////
```

説明

全般

データ取得は，FIFOを開始することで可能になります。MX100のチャネル1のFIFOデータのうち，取得可能な分の測定データを一度に取得し，領域に格納します。その中から，現在状態(先頭の計測点)の測定値データ(1点)を取得し，終了します。

通信接続

```
comm = openMX100("192.168.1.12", &rc);
```

ＭＸ１００のＩＰアドレスを指定しています。通信用ポートは，通信用定数DAQMX_COMMPORT(MX100の通信ポート番号)を指定したことになります。

FIFO開始

```
rc = measStartMX100(comm);
```

MX100でFIFOを開始します。

チャネル1の測定データの取得

```
rc = measDataChMX100(comm, 1);
```

MX100から，チャネル1の取得可能な分の測定データを一度に取得し，領域に格納します。先頭の計測点を現在状態とします。

測定値の取得

```
value = dataValueMX100(comm, 1);
```

測定データを格納している領域から，チャネル1の現在状態の測定値を取得します。

FIFO停止

```
rc = measStopMX100(comm);
```

FIFOを停止します。

通信切断

```
rc = closeMX100(comm);
```

通信を切断します。

参考

サンプルプログラムでは，measDataChMX100を一度だけ実行して終了しています。measDataChMX100を繰り返して実行すると，実行されるごとに，計測点をひとつ進めて現在状態とします。格納している計測点の最後まで到達したら，続く取得可能な分のデータを取得します。

設定データの読み出しと書き込み

プログラム例

```
////////////////////////////////////////////////////////////
// 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;
  //laod
  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;
}
////////////////////////////////////////////////////////////////
```

説明

全般

全設定項目の読み出しと書き込みのプログラム例です。下記の4つを実行します。

・MX100から設定データを一括受信
・設定データ領域の設定データを1項目ずつ取得
・設定データを1項目ずつ設定データ領域に書き込む
・MX100に設定データを一括送信

先頭番号から最終番号まで，1項目ずつ取得と書き込みをしています。
文字列領域はサイズに余裕を持って用意してください。
項目番号と項目文字列の組を保存，ロードすることで設定データをバックアップすることも可能になります。
設定項目番号については，6.3節を参照してください。

通信接続

```
comm = openMX100("192.168.1.12", &rc);
```

ＭＸ１００のＩＰアドレスを指定しています。通信用ポートは，通信用定数DAQMX_COMMPORT(MX100の通信ポート番号)を指定したことになります。

設定データの一括受信

```
rc = daqmx100.getItemAll();
```

MX100の設定データの全項目を一括受信し，設定データ領域に格納します。

設定データを1項目ずつ取得

```
rc = readItemMX100(comm, i, strItem, BUFSIZ, &realLen);
```

設定データ領域から項目番号「i」の内容を取得します。

設定データを1項目ずつ書き込む

```
rc = writeItemMX100(comm, i, strItem);
```

設定データ領域の項目番号「i」に，strItemの内容を書き込みます。

設定データの一括送信

```
rc = setItemAllMX100(comm);
```

設定データの全項目をMX100に一括送信します。

通信切断

```
rc = closeMX100(comm);
```

通信を切断します。

14.1 機能と関数の対応ーMX100/Visual Basicー

拡張APIでサポートする機能と，Visual Basicの関数郡の対応を示します。
状態遷移関数と取得関数の2種類あります。
状態遷移関数はMX100本体を制御します。データ取得機能で測定データを取得すると測定点が1点分だけ進みます(拡張APIの状態が遷移します)。
取得関数は項目値を返します。取得関数を使用した場合，データ値取得を使用すると拡張APIが保持している現在の状態のデータ値を返します(拡張APIの状態は遷移しません)。

状態遷移関数

表の「FIFO」欄は，FIFO中に関数を実行したときの，FIFOの動作を示します。
停止：関数を実行するとFIFOを停止します。
継続：関数を実行してもFIFOを継続します。

通信機能

機能	FIFO	関数
MX100と通信接続	継続	openMX100
MX100との通信を切断	継続	closeMX100

FIFOの開始/停止

機能	FIFO	関数
FIFOを開始	継続	measStartMX100
FIFOを停止	停止	measStopMX100

制御機能

機能		FIFO	関数
日付時刻設定 現在時刻		停止	setDateTimeNowMX100
バックアップ 有効無効の設定		継続	switchBackupMX100
CFカードのフォーマット		停止	formatCFMX100
ユニット	システムの再構築	停止	reconstructMX100
	システムの初期化	停止	initSetValueMX100
	アラームリセット（アラームACK）	停止	ackAlarmMX100
7セグメントLEDの表示		継続	displaySegmentMX100
保持データの初期化	チャネル指定	継続	initDataChMX100
	FIFO指定	継続	initDataFIFOMX100

制御機能は，通信の最後に状態更新を行います。
各データの送信，設定機能については， データ操作機能を参照してください。
任意の時刻設定はできません。

設定機能

機能		FIFO	関数
設定データを一括設定(一括送信)	全設定データ	停止	sendConfigMX100
	基本設定データ	停止	sendConfigMX100
設定データを個別設定	システム構成データ	停止	sendConfigMX100
	チャネル設定データ	停止	sendConfigMX100
	初期バランスデータ	停止	sendConfigMX100
	出力チャネルデータ	停止	sendConfigMX100
初期バランスデータ	実行	停止	initBalanceMX100
	リセット	停止	clearBalanceMX100

設定データの設定機能は，保持しているデータを送信します。
任意の初期バランスデータを設定する場合，データ操作機能で初期バランスデータの送信機能を参照してください。

設定変更機能

設定機能は，設定送信して，状態更新を行います。
単独チャネル毎の設定なので，設定できなかった場合，原則エラーを返します。
データ値，または，測定値(倍精度浮動小数)での指定ができます。

レンジ設定

機能	FIFO	関数
スキップ	停止	setRangeMX100
直流電圧入力	停止	setRangeMX100
熱電対入力	停止	setRangeMX100
測温抵抗体	停止	setRangeMX100
ディジタル入力	停止	setRangeMX100
抵抗	停止	setRangeMX100
ひずみ	停止	setRangeMX100
AO	停止	setRangeMX100
PWM	停止	setRangeMX100
チャネル間差演算	停止	setChDELTAMX100
リモートRRJC	停止	setChRRJCMX100
パルス	停止	setRangeMX100
通信	停止	setRangeMX100

チャネル設定

機能		FIFO	関数
単位名		停止	setChUnitMX100
タグ		停止	setChTagMX100
コメント		停止	setChCommentMX100
AI/DI/AO/PWM	スパン	停止	setSpanMX100 setDoubleSpanMX100
AI/DI	スケール	停止	setScaleMX100 setDoubleScaleMX100
	アラーム	停止	setAlarmMX100 setDoubleAlarmMX100 setAlarmValueMX100 setDoubleAlarmValueMX100
	ヒステリシス	停止	setHisterisysMX100 setDoubleHisterisysMX100
AI	フィルタ係数	停止	setFilterMX100
	基準接点補償(RJC)	停止	setRJCTypeMX100
	バーンアウト種類	停止	setBurnoutMX100
DO	非励磁	停止	setDeenergizeMX100
	保持	停止	setHoldMX100
	参照アラーム	停止	setRefAlarmMX100
チャネル種類	DO種類	停止	setChKindMX100
	AO種類	停止	setChKindMX100
	PWM種類	停止	setChKindMX100
PI	チャタリングフィルタ	停止	channelChatFilterMX1000

モジュール設定

機能	FIFO	関数
周期種類	停止	setIntervalMX100
AD積分時間種類	停止	setIntegralMX100

ユニット設定

機能	FIFO	関数
ユニット番号	停止	setUnitNoMX100
温度単位種類	停止	setUnitTempMX100
CF書き込み種類	停止	setCFWriteModeMX100

出力チャネルデータ

機能	FIFO	関数
出力種類	停止	setOutputTypeMX100
選択値	停止	setChoiceMX100 setDoubleChoiceMX100
パルス周期倍率	停止	setPulseTimeMX100

データ操作機能

DOデータ

機能		FIFO	関数
作成		継続	createDOMX100
削除		継続	deleteDOMX100
部分変更	ユーザ指定	継続	changeDOMX100
	コピー	継続	copyDOMX100
送信	既存指定	継続	commandDOMX100
	変更指定	継続	switchDOMX100

AO/PWMデータ

機能		FIFO	関数
作成		継続	createAOPWMMX100
削除		継続	deleteAOPWMMX100
部分変更	出力データ値	継続	changeAOPWMMX100
	実出力値	継続	changeAOPWMValueMX100
	コピー	継続	copyAOPWMMX100
送信		継続	commandAOPWMMX100

初期バランスデータ

機能		FIFO	関数
作成		継続	createBalanceMX100
削除		継続	deleteBalanceMX100
部分変更	ユーザ指定	継続	changeBalanceMX100
	コピー	継続	copyBalanceMX100
送信		停止	commandBalanceMX100

伝送出力データ

機能		FIFO	関数
作成		継続	createTransmitMX100
削除		継続	deleteTransmitMX100
部分変更	ユーザ指定	継続	changeTransmitMX100
	コピー	継続	copyTransmitMX100
送信	既存指定	継続	commandTransmitMX100
	変更指定	継続	switchTransmitMX100

各データ識別子で操作します。
送信以外は，状態更新(通信)を行いません。

取得機能

機能			FIFO	関数
ステータスデータ			継続	updateStatusMX100
システム構成データ			継続	updateSystemMX100
設定データ			継続	updateConfigMX100
出力データ	DOデータ		継続	updateDODataMX100
	AO/PWMデータ 伝送出力データ		継続	updateAOPWMDataMX100
チャネル情報データ			継続	updateInfoChMX100
測定データ	チャネル指定	FIFO値	継続	measDataChMX100
		瞬時値	継続	measInstChMX100
	FIFO指定	FIFO値	継続	measDataFIFOMX100
		瞬時値	継続	measInstFIFOMX100
初期バランスデータ			継続	updateBalanceMX100
出力チャネルデータ			継続	updateOutputMX100

データ取得は，本API内部で一括取得が行われます。
収集によって，状態更新も行われます。
チャネル情報データや設定データ(システム構成データ，初期バランスデータ，出力チャネルデータを含む)は，内部で保持されていますが，ユーザが明示的に保持しているデータを更新できます。

設定項目

機能		FIFO	関数
設定データ	一括受信	継続	getItemAllMX100
	一括送信	停止	setItemAllMX100
設定項目	読み出し	継続	readItemMX100
	書き込み	継続	writeItemMX100
	初期化	継続	initItemMX100

設定項目の読み出し，書き込み，初期化は，保持している領域へのアクセスで，領域の整合性チェックをしません。また，状態更新(通信)を行いません。

取得関数

測定データ

データ名		関数
データ値		dataValueMX100
データステータス値		dataStatusMX100
アラーム（有無）		dataAlarmMX100
測定値	倍精度浮動小数	dataDoubleValueMX100
	文字列	dataStringValueMX100
時刻	秒数	dataTimeMX100
	ミリ秒	dataMilliSecMX100
	年	dataYearMX100
	月	dataMonthMX100
	日	dataDayMX100
	時	dataHourMX100
	分	dataMinuteMX100
	秒	dataSecondMX100
有効データ(有無)		dataValidMX100

チャネル情報データ

データ名	関数
FIFO番号	channelFIFONoMX100
FIFO内チャネル順序番号	channelFIFOIndexMX100
表示最小値	channelDisplayMinMX100
表示最大値	channelDisplayMaxMX100
実範囲最小値	channelRealMinMX100
実範囲最大値	channelRealMaxMX100

チャネル設定データ

データ名				関数
チャネルステータス(有無)				channelValidMX100
小数点位置				channelPointMX100
チャネル種類				channelKindMX100
レンジ種類				channelRangeMX100
スケール種類				channelScaleTypeMX100
単位名				toChannelUnitMX100 getChannelUnitMX100
タグ				toChannelTagMX100 getChannelTagMX100
コメント				toChannelCommentMX100 getChannelCommentMX100
AI/DI/AO/PWM				
	スパン	最小値	データ値	channelSpanMinMX100
			測定値	channelDoubleSpanMinMX100
		最大値	データ値	channelSpanMaxMX100
			測定値	channelDoubleSpanMaxMX100
AI/DI	スケール	最小値	データ値	channelScaleMinMX100
			測定値	channelDoubleScaleMinMX100
		最大値	データ値	channelScaleMaxMX100
			測定値	channelDoubleScaleMaxMX100
	アラーム種類			alarmTypeMX100
	アラーム値(ON値)		データ値	alarmValueONMX100
			測定値	alarmDoubleValueONMX100
	アラーム値(OFF値)		データ値	alarmValueOFFMX100
			測定値	alarmDoubleValueOFFMX100
	ヒステリシス		データ値	alarmHisterisysMX100
			測定値	alarmDoubleHisterisysMX100
AI	フィルタ			channelFilterMX100
	RJC種類			channelRJCTypeMX100
	RJC電圧値			channelRJCVoltMX100
	バーンアウト			channelBurnoutMX100
DO	非励磁			channelDeenergizeMX100
	保持			channelHoldMX100
	参照アラーム			channelRefAlarmMX100
チャネル間差演算/リモートRJC/AO/PWM				
	基準チャネル番号			channelRefChNoMX100
初期バランスデータ		有効無効値		channelBalanceValidMX100
		初期バランス値		channelBalanceValueMX100
出力チャネルデータ		出力種類		channelOutputTypeMX100
		アイドル時の選択値		channelIdleChoiceMX100
		エラー時の選択値		channelErrorChoiceMX100
		選択値が「指定値」の場合の値		
			データ値	channelPresetValueMX100
			測定値	channelDoublePresetValueMX100
		パルス周期倍率		channelPulseTimeMX100
PI	チャタリングフィルタ			channelChatFilterMX100

ネットワーク情報データ

データ名	関数
ホスト名	toNetHostMX100
IPアドレス	netAddressMX100
ポート番号	netPortMX100
サブネットマスク	netSubmaskMX100
Gatewayアドレス	netGatewayMX100

システム構成データ

データ名		関数
モジュール	モジュール種類	moduleTypeMX100
	チャネル数	moduleChNumMX100
	周期種類	moduleIntervalMX100
	AD積分時間種類	moduleIntegralMX100
	有効無効値	moduleValidMX100
	起動時モジュール種類	moduleStandbyTypeMX100
	実際のモジュール種類	moduleRealTypeMX100
	端子種類	moduleTerminalMX100
	バージョン	moduleVersionMX100
	FIFO番号	moduleFIFONoMX100
	シリアル番号	toModuleSerialMX100
ユニット	ユニット種類	unitTypeMX100
	スタイル	unitStyleMX100
	ユニット番号	unitNoMX100
	温度単位種類	unitTempMX100
	電源周波数	unitFrequencyMX100
	パート番号	toUnitPartNoMX100
	オプション	unitOptionMX100
	シリアル番号	toUnitSerialMX100
	MACアドレス	unitMACMX100
	CF書き込み種類	unitCFWriteModeMX100

ステータスデータ

データ名		関数
ユニットステータス値		statusUnitMX100
FIFOの有効個数		statusFIFONumMX100
バックアップ(有無)		statusBackupMX100
FIFO	FIFOステータス値	statusFIFOMX100
	周期種類	statusFIFOIntervalMX100
CF	CFステータス種類	statusCFMX100
	サイズ	statusCFSizeMX100
	残容量	statusCFRemainMX100
ステータス返却時刻	秒数	statusTimeMX100
	ミリ秒	statusMilliSecMX100
	年	statusYearMX100
	月	statusMonthMX100
	日	statusDayMX100
	時	statusHourMX100
	分	statusMinuteMX100
	秒	statusSecondMX100

カレントデータ

データ名		関数
DOデータ	有効無効値	currentDOValidMX100
	ON/OFF状態	currentDOValueMX100
AO/PWMデータ	有効無効値	currentAOPWMValidMX100
	出力データ値	currentAOPWMValueMX100
	出力値	currentDoubleAOPWMValueMX100
初期バランスデータ	有効無効値	currentBalanceValidMX100
	初期バランス値	currentBalanceValueMX100
	初期バランス結果	currentBalanceResultMX100
伝送出力データ伝送状態		currentTransmitMX100

データ取得機能で取得された各データの状態です。
初期バランスデータの初期バランス結果は，設定機能による実行結果です。
DOデータ，AO/PWMデータなど，実際に出力されている出力状態をカレントデータとして取得できます。ただし，データを送信した直後は，設定した値が返却されて，実際の出力は次のタイミングになることがあります。
保持しているデータは，状態更新で取得したときの値です。取得関数を呼び出した時刻のデータではありません。

ユーザデータ

データ名		関数
DOデータ	有効無効値	userDOValidMX100
	ON/OFF状態	userDOValueMX100
AO/PWMデータ	有効無効値	userAOPWMValidMX100
	出力データ値	userAOPWMValueMX100
	出力値	userDoubleAOPWMValueMX100
初期バランスデータ	有効無効値	userBalanceValidMX100
	初期バランス値	userBalanceValueMX100
伝送出力データ	伝送状態	userTransmitMX100

ユーザがデータ操作機能で作成したデータの値を取得します。

ユーティリティ

機能/データ名		関数
残りデータ個数	チャネル単位で取得	dataNumChMX100
	FIFO単位で取得	dataNumFIFOMX100
エラー	MX固有エラーの取得	lastErrorMX100
	エラーメッセージ文字列を取得	toErrorMessageMX100
	エラーメッセージ文字列の最大長を取得	errorMaxLengthMX100
	エラー検出した設定項目番号を取得	itemErrorMX100
FIFO情報から，チャネル番号に変換		channelNumberMX100
レンジ種類別の小数点位置を取得		rangePointMX100
測定値	倍精度浮動小数に変換	toDoubleValueMX100
	文字列に変換	toStringValueMX100
アラーム	アラーム種類の文字列を取得	toAlarmNameMX100
	アラーム文字列の最大長を取得	alarmMaxLengthMX100
本APIのバージョン番号を取得		versionAPIMX100
本APIのリビジョン番号を取得		revisionAPIMX100
IPアドレスのパート分割を取得		addressPartMX100
AO/PWM	出力値を出力データ値に変換	toAOPWMValueMX100
	出力データ値を出力値に変換	toRealValueMX100
設定項目	設定項目番号から設定項目文字列を取得	toItemNameMX100
	設定項目文字列から設定項目番号を取得	toItemNoMX100
	設定項目文字列の最大長を取得	itemMaxLengthMX100
スタイルバージョンに変換		toStyleVersionMX100

14.2 プログラムーMX100/Visual Basicー

型，関数，定数の宣言

Visual Basic用の型，関数，定数を使用するためには，あらかじめ宣言をしておく必要があります。次の宣言記述方法があります。

全宣言の記述

プロジェクトにVisual Basic用標準モジュールファイル(DAQMX100.bas)を追加すると，すべての型，関数，定数を宣言したことになります。

宣言の選択記述

Visual Studioに付属しているAPIビューアで，任意の型，関数，定数の宣言記述をコピーできます。この機能を使用するためには，APIビューアで，APIビューア用テキストファイル(DAQMX100.txt)を読み込んでください。
APIビューアの使用方法については，Visual Studioの取扱説明書をご覧ください。

宣言の直接記述

記述例を示します。

```
Public Declare Function openMX100 Lib "DAQMX100" (ByVal
strAddress As String, ByRef errorCode As Long) As Long
```

測定データの取得

プログラム例

```
Attribute VB_Name = "Module1"
Public Sub 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 Sub
```

説明

全般

データ取得は，FIFOを開始することで可能になります。MX100のチャネル1のFIFOデータのうち，取得可能な分の測定データを一度に取得し，領域に格納します。その中から，現在状態(先頭の計測点)の測定値データ(1点)を取得し，終了します。

通信接続

```
comm = openMX100("192.168.1.12", rc)
```

ＭＸ１００のＩＰアドレスを指定しています。通信用ポートは，通信用定数DAQMX_COMMPORT(MX100の通信ポート番号)を指定したことになります。

FIFO開始

```
rc = measStartMX100(comm)
```

MX100でFIFOを開始します。

チャネル1の測定データの取得

```
rc = measDataChMX100(comm, 1)
```

MX100から，チャネル1の取得可能な分の測定データを一度に取得し，領域に格納します。先頭の計測点を現在状態とします。

測定値の取得

```
value = dataValueMX100(comm, 1)
```

測定データを格納している領域から，チャネル1の現在状態の測定値を取得します。

FIFO停止

```
rc = measStopMX100(comm)
```

FIFOを停止します。

通信切断

```
rc = closeMX100(comm)
```

通信を切断します。

参考

サンプルプログラムでは，measDataChMX100を一度だけ実行して終了しています。measDataChMX100を繰り返して実行すると，実行されるごとに，計測点をひとつ進めて現在状態とします。格納している計測点の最後まで到達したら，続く取得可能な分のデータを取得します。

設定データの読み出しと書き込み

プログラム例

```
Attribute VB_Name = "Module1"
Public Sub 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 Sub
```

説明

全般

全設定項目の読み出しと書き込みのプログラム例です。下記の4つを実行します。

・MX100から設定データを一括受信
・設定データ領域の設定データを1項目ずつ取得
・設定データを1項目ずつ設定データ領域に書き込む
・MX100に設定データを一括送信

先頭番号から最終番号まで，1項目ずつ取得と書き込みをしています。
文字列領域はサイズに余裕を持って用意してください。
項目番号と項目文字列の組を保存，ロードすることで設定データをバックアップすることも可能になります。
設定項目番号については，6.3節を参照してください。

通信接続

```
comm = openMX100("192.168.1.12", rc)
```

ＭＸ１００のＩＰアドレスを指定しています。通信用ポートは，通信用定数DAQMX_COMMPORT(MX100の通信ポート番号)を指定したことになります。

設定データの一括受信

```
rc = getItemAllMX100(comm)
```

MX100の設定データの全項目を一括受信し，設定データ領域に格納します。

設定データを1項目ずつ取得

```
rc = readItemMX100(comm, i, strItem, lenItem, realLen)
```

設定データ領域から項目番号「i」の内容を取得します。

設定データを1項目ずつ書き込む

```
rc = writeItemMX100(comm, i, strItem)
```

設定データ領域の項目番号「i」に，strItemの内容を書き込みます。

設定データの一括送信

```
rc = setItemAllMX100(comm)
```

設定データの全項目をMX100に一括送信します。

通信切断

```
rc = closeMX100(comm)
```

通信を切断します。

15.1 機能と関数の対応ーMX100/Visual Basic.NETー

拡張APIでサポートする機能と，Visual Basicの関数郡の対応を示します。
状態遷移関数と取得関数の2種類あります。
状態遷移関数はMX100本体を制御します。データ取得機能で測定データを取得すると測定点が1点分だけ進みます(拡張APIの状態が遷移します)。
取得関数は項目値を返します。取得関数を使用した場合，データ値取得を使用すると拡張APIが保持している現在の状態のデータ値を返します(拡張APIの状態は遷移しません)。

状態遷移関数

表の「FIFO」欄は，FIFO中に関数を実行したときの，FIFOの動作を示します。
停止：関数を実行するとFIFOを停止します。
継続：関数を実行してもFIFOを継続します。

通信機能

機能	FIFO	関数
MX100と通信接続	継続	openMX100
MX100との通信を切断	継続	closeMX100

FIFOの開始/停止

機能	FIFO	関数
FIFOを開始	継続	measStartMX100
FIFOを停止	停止	measStopMX100

制御機能

機能		FIFO	クラスと関数メンバ
日付時刻設定 現在時刻		停止	setDateTimeNowMX100
バックアップ 有効無効の設定		継続	switchBackupMX100
CFカードのフォーマット		停止	formatCFMX100
ユニット	システムの再構築	停止	reconstructMX100
	システムの初期化	停止	initSetValueMX100
	アラームリセット（アラームACK）	停止	ackAlarmMX100
7セグメントLEDの表示		継続	displaySegmentMX100
保持データの初期化	チャネル指定	継続	initDataChMX100
	FIFO指定	継続	initDataFIFOMX100

制御機能は，通信の最後に状態更新を行います。
各データの送信，設定機能については， データ操作機能を参照してください。
任意の時刻設定はできません。

設定機能

機能		FIFO	関数
設定データを一括設定(一括送信)	全設定データ	停止	sendConfigMX100
	基本設定データ	停止	sendConfigMX100
設定データを個別設定	システム構成データ	停止	sendConfigMX100
	チャネル設定データ	停止	sendConfigMX100
	初期バランスデータ	停止	sendConfigMX100
	出力チャネルデータ	停止	sendConfigMX100
初期バランスデータ	実行	停止	initBalanceMX100
	リセット	停止	clearBalanceMX100

設定データの設定機能は，保持しているデータを送信します。
任意の初期バランスデータを設定する場合，データ操作機能で初期バランスデータの送信機能を参照してください。

設定変更機能

設定機能は，設定送信して，状態更新を行います。
単独チャネル毎の設定なので，設定できなかった場合，原則エラーを返します。
データ値，または，測定値(倍精度浮動小数)での指定ができます。

レンジ設定

機能	FIFO	関数
スキップ	停止	setRangeMX
直流電圧入力	停止	setRangeMX
熱電対入力	停止	setRangeMX
測温抵抗体	停止	setRangeMX
ディジタル入力	停止	setRangeMX
抵抗	停止	setRangeMX
ひずみ	停止	setRangeMX
AO	停止	setRangeMX
PWM	停止	setRangeMX
チャネル間差演算	停止	setChDELTAMX100
リモートRJC	停止	setChRRJCMX100
パルス	停止	setRangeMX100
通信	停止	setRangeMX100

チャネル設定

機能		FIFO	関数
単位名		停止	setChUnitMX100
タグ		停止	setChTagMX100
コメント		停止	setChCommentMX100
AI/DI/AO/PWM	スパン	停止	setSpanMX100 setDoubleSpanMX100
AI/DI	スケール	停止	setScaleMX100 setDoubleScaleMX100
	アラーム	停止	setAlarmMX100 setDoubleAlarmMX100 setAlarmValueMX100 setDoubleAlarmValueMX100
	ヒステリシス	停止	setHisterisysMX100 setDoubleHisterisysMX100
AI	フィルタ係数	停止	setFilterMX100
	基準接点補償(RJC)	停止	setRJCTypeMX100
	バーンアウト	停止	setBurnoutMX100
DO	非励磁	停止	setDeenergizeMX100
	保持	停止	setHoldMX100
	参照アラーム	停止	setRefAlarmMX100
チャネル種類	DO種類	停止	setChKindMX100
	AO種類	停止	setChKindMX100
	PWM種類	停止	setChKindMX100
PI	チャタリングフィルタ	停止	channelChatFilterMX100

モジュール設定

機能	FIFO	関数
周期種類	停止	setIntervalMX100
A/D積分時間種類	停止	setIntegralMX100

ユニット設定

機能	FIFO	関数
ユニット番号	停止	setUnitNoMX100
温度単位種類	停止	setUnitTempMX100
CF書き込み種類	停止	setCFWriteModeMX100

出力チャネルデータ

機能	FIFO	関数
出力種類	停止	setOutputTypeMX100
選択値	停止	setChoiceMX100 setDoubleChoiceMX100
パルス周期倍率	停止	setPulseTimeMX100

データ操作機能

DOデータ

機能		FIFO	関数
作成		継続	createDOMX100
削除		継続	deleteDOMX100
部分変更	ユーザ指定	継続	changeDOMX100
	コピー	継続	copyDOMX100
送信	既存指定	継続	commandDOMX100
	変更指定	継続	switchDOMX100

AO/PWMデータ

機能		FIFO	関数
作成		継続	createAOPWMMX100
削除		継続	deleteAOPWMMX100
部分変更	出力データ値	継続	changeAOPWMMX100
	実出力値	継続	changeAOPWMValueMX100
	コピー	継続	copyAOPWMMX100
送信		継続	commandAOPWMMX100

初期バランスデータ

機能		FIFO	関数
作成		継続	createBalanceMX100
削除		継続	deleteBalanceMX100
部分変更	ユーザ指定	継続	changeBalanceMX100
	コピー	継続	copyBalanceMX100
送信		停止	commandBalanceMX100

伝送出力データ

機能		FIFO	関数
作成		継続	createTransmitMX100
削除		継続	deleteTransmitMX100
部分変更	ユーザ指定	継続	changeTransmitMX100
	コピー	継続	copyTransmitMX100
送信	既存指定	継続	commandTransmitMX100
	変更指定	継続	switchTransmitMX100

各データ識別子で操作します。
送信以外は，状態更新(通信)を行いません。

取得機能

機能			FIFO	関数
ステータスデータ			継続	updateStatusMX100
システム構成データ			継続	updateSystemMX100
設定データ			継続	updateConfigMX100
出力データ	DOデータ		継続	updateDODataMX100
	AO/PWMデータ 伝送出力データ		継続	updateAOPWMDataMX100
チャネル情報データ			継続	updateInfoChMX100
測定データ	チャネル指定	FIFO値	継続	measDataChMX100
		瞬時値	継続	measInstChMX100
	FIFO指定	FIFO値	継続	measDataFIFOMX100
		瞬時値	継続	measInstFIFOMX100
初期バランスデータ			継続	updateBalanceMX100
出力チャネルデータ			継続	updateOutputMX100

データ取得は，本API内部で一括取得が行われます。
収集によって，状態更新も行われます。
チャネル情報データや設定データ(システム構成データ，初期バランスデータ，出力チャネルデータを含む)は，内部で保持されていますが，ユーザが明示的に保持しているデータを更新できます。

設定項目

機能		FIFO	関数
設定データ	一括受信	継続	getItemAllMX100
	一括送信	停止	setItemAllMX100
設定項目	読み出し	継続	readItemMX100
	書き込み	継続	writeItemMX100
	初期化	継続	initItemMX100

設定項目の読み出し，書き込み，初期化は，保持している領域へのアクセスで，領域の整合性チェックをしません。また，状態更新(通信)を行いません。

取得関数

測定データ

データ名		関数
データ値		dataValueMX100
データステータス値		dataStatusMX100
アラーム（有無）		dataAlarmMX100
測定値	倍精度浮動小数	dataDoubleValueMX100
	文字列	dataStringValueMX100
時刻	秒数	dataTimeMX100
	ミリ秒	dataMilliSecMX100
	年	dataYearMX100
	月	dataMonthMX100
	日	dataDayMX100
	時	dataHourMX100
	分	dataMinuteMX100
	秒	dataSecondMX100
有効データ(有無)		dataValidMX100

チャネル情報データ

データ名	関数
FIFO番号	channelFIFONoMX100
FIFO内チャネル順序番号	channelFIFOIndexMX100
表示最小値	channelDisplayMinMX100
表示最大値	channelDisplayMaxMX100
実範囲最小値	channelRealMinMX100
実範囲最大値	channelRealMaxMX100

チャネル設定データ

データ名				関数
チャネルステータス(有無)				channelValidMX100
小数点位置				channelPointMX100
チャネル種類				channelKindMX100
レンジ種類				channelRangeMX100
スケール種類				channelScaleTypeMX100
単位名				toChannelUnitMX100 getChannelUnitMX100
タグ				toChannelTagMX100 getChannelTagMX100
コメント				toChannelCommentMX100 getChannelCommentMX100
AI/DI/AO/PWM				
	スパン	最小値	データ値	channelSpanMinMX100
			測定値	channelDoubleSpanMinMX100
		最大値	データ値	channelSpanMaxMX100
			測定値	channelDoubleSpanMaxMX100
AI/DI	スケール	最小値	データ値	channelScaleMinMX100
			測定値	channelDoubleScaleMinMX100
		最大値	データ値	channelScaleMaxMX100
			測定値	channelDoubleScaleMaxMX100
	アラーム種類			alarmTypeMX100
	アラーム値(ON値)		データ値	alarmValueONMX100
			測定値	alarmDoubleValueONMX100
	アラーム値(OFF値)		データ値	alarmValueOFFMX100
			測定値	alarmDoubleValueOFFMX100
	ヒステリシス		データ値	alarmHisterisysMX100
			測定値	alarmDoubleHisterisysMX100
AI	フィルタ			channelFilterMX100
	RJC種類			channelRJCTypeMX100
	RJC電圧値			channelRJCVoltMX100
	バーンアウト			channelBurnoutMX100
DO	非励磁			channelDeenergizeMX100
	保持			channelHoldMX100
	参照アラーム			channelRefAlarmMX100
チャネル間差演算/リモートRJC/AO/PWM				
	基準チャネル番号			channelRefChNoMX100
初期バランスデータ		有効無効値		channelBalanceValidMX100
		初期バランス値		channelBalanceValueMX100
出力チャネルデータ		出力種類		channelOutputTypeMX100
		アイドル時の選択値		channelIdleChoiceMX100
		エラー時の選択値		channelErrorChoiceMX100
		選択値が「指定値」の場合の値		
			データ値	channelPresetValueMX100
			測定値	channelDoublePresetValueMX100
		パルス周期倍率		channelPulseTimeMX100
PI	チャタリングフィルタ			channelChatFilterMX100

ネットワーク情報データ

データ名	関数
ホスト名	toNetHostMX100
IPアドレス	netAddressMX100
ポート番号	netPortMX100
サブネットマスク	netSubmaskMX100
Gatewayアドレス	netGatewayMX100

システム構成データ

データ名		関数
モジュール	モジュール種類	moduleTypeMX100
	チャネル数	moduleChNumMX100
	周期種類	moduleIntervalMX100
	AD積分時間種類	moduleIntegralMX100
	有効無効値	moduleValidMX100
	起動時モジュール種類	moduleStandbyTypeMX100
	実際のモジュール種類	moduleRealTypeMX100
	端子種類	moduleTerminalMX100
	バージョン	moduleVersionMX100
	FIFO番号	moduleFIFONoMX100
	シリアル番号	toModuleSerialMX100
ユニット	ユニット種類	unitTypeMX100
	スタイル	unitStyleMX100
	ユニット番号	unitNoMX100
	温度単位種類	unitTempMX100
	電源周波数	unitFrequencyMX100
	パート番号	toUnitPartNoMX100
	オプション	unitOptionMX100
	シリアル番号	toUnitSerialMX100
	MACアドレス	unitMACMX100
	CF書き込み種類	unitCFWriteModeMX100

ステータスデータ

データ名		関数
ユニットステータス値		statusUnitMX100
FIFOの有効個数		statusFIFONumMX100
バックアップ(有無)		statusBackupMX100
FIFO	FIFOステータス値	statusFIFOMX100
	周期種類	statusFIFOIntervalMX100
CF	CFステータス種類	statusCFMX100
	サイズ	statusCFSizeMX100
	残容量	statusCFRemainMX100
ステータス返却時刻	秒数	statusTimeMX100
	ミリ秒	statusMilliSecMX100
	年	statusYearMX100
	月	statusMonthMX100
	日	statusDayMX100
	時	statusHourMX100
	分	statusMinuteMX100
	秒	statusSecondMX100

カレントデータ

データ名		関数
DOデータ	有効無効値	currentDOValidMX100
	ON/OFF状態	currentDOValueMX100
AO/PWMデータ	有効無効値	currentAOPWMValidMX100
	出力データ値	currentAOPWMValueMX100
	出力値	currentDoubleAOPWMValueMX100
初期バランスデータ	有効無効値	currentBalanceValidMX100
	初期バランス値	currentBalanceValueMX100
	初期バランス結果	currentBalanceResultMX100
伝送出力データ伝送状態		currentTransmitMX100

データ取得機能で取得された各データの状態です。
初期バランスデータの初期バランス結果は，設定機能による実行結果です。
DOデータ，AO/PWMデータなど，実際に出力されている出力状態をカレントデータとして取得できます。ただし，データを送信した直後は，設定した値が返却されて，実際の出力は次のタイミングになることがあります。
保持しているデータは，状態更新で取得したときの値です。取得関数を呼び出した時刻のデータではありません。

ユーザデータ

データ名		関数
DOデータ	有効無効値	userDOValidMX100
	ON/OFF状態	userDOValueMX100
AO/PWMデータ	有効無効値	userAOPWMValidMX100
	出力データ値	userAOPWMValueMX100
	出力値	userDoubleAOPWMValueMX100
初期バランスデータ	有効無効値	userBalanceValidMX100
	初期バランス値	userBalanceValueMX100
伝送出力データ	伝送状態	userTransmitMX100

ユーザがデータ操作機能で作成したデータの値を取得します。

ユーティリティ

機能/データ名		関数
残りデータ個数	チャネル単位で取得	dataNumChMX100
	FIFO単位で取得	dataNumFIFOMX100
エラー	MX固有エラーの取得	lastErrorMX100
	エラーメッセージ文字列を取得	toErrorMessageMX100
	エラーメッセージ文字列の最大長を取得	errorMaxLengthMX100
	エラー検出した設定項目番号を取得	itemErrorMX100
FIFO情報から，チャネル番号に変換		channelNumberMX100
レンジ種類別の小数点位置を取得		rangePointMX100
測定値	倍精度浮動小数に変換	toDoubleValueMX100
	文字列に変換	toStringValueMX100
アラーム	アラーム種類の文字列を取得	toAlarmNameMX100
	アラーム文字列の最大長を取得	alarmMaxLengthMX100
本APIのバージョン番号を取得		versionAPIMX100
本APIのリビジョン番号を取得		revisionAPIMX100
IPアドレスのパート分割を取得		addressPartMX100
AO/PWM	出力値を出力データ値に変換	toAOPWMValueMX100
	出力データ値を出力値に変換	toRealValueMX100
設定項目	設定項目番号から設定項目文字列を取得	toItemNameMX100
	設定項目文字列から設定項目番号を取得	toItemNoMX100
	設定項目文字列の最大長を取得	itemMaxLengthMX100
スタイルバージョンに変換		toStyleVersionMX100

15.2 プログラムーMX100/Visual Basic.NETー

関数，定数の宣言

Visual Basic.NET用の関数，定数を使用するためには，あらかじめ宣言をしておく必要があります。次の宣言記述方法があります。

宣言の記述

プロジェクトにVisual Basic.NET用モジュールファイル(DAQMX100.vb)を追加すると，すべての関数，定数を宣言したことになります。

測定データの取得

プログラム例

```
Module Module1
    Public Sub 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 Sub
End Module
```

説明

全般

データ取得は，FIFOを開始することで可能になります。MX100のチャネル1のFIFOデータのうち，取得可能な分の測定データを一度に取得し，領域に格納します。その中から，現在状態(先頭の計測点)の測定値データ(1点)を取得し，終了します。

通信接続

`comm = openMX100("192.168.1.12", rc)`

ＭＸ１００のＩＰアドレスを指定しています。通信用ポートは，通信用定数DAQMX_COMMPORT(MX100の通信ポート番号)を指定したことになります。

FIFO開始

`rc = measStartMX100(comm)`

MX100でFIFOを開始します。

チャネル1の測定データの取得

`rc = measDataChMX100(comm, 1)`

MX100から，チャネル1の取得可能な分の測定データを一度に取得し，領域に格納します。先頭の計測点を現在状態とします。

測定値の取得

`value = dataValueMX100(comm, 1)`

測定データを格納している領域から，チャネル1の現在状態の測定値を取得します。

FIFO停止

```
rc = measStopMX100(comm)
```

FIFOを停止します。

通信切断

```
rc = closeMX100(comm)
```

通信を切断します。

参考

サンプルプログラムでは，measDataChMX100を一度だけ実行して終了しています。measDataChMX100を繰り返して実行すると，実行されるごとに，計測点をひとつ進めて現在状態とします。格納している計測点の最後まで到達したら，続く取得可能な分のデータを取得します。

設定データの読み出しと書き込み

プログラム例

```
Module Module1
    Public Sub 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 Sub
End Module
```

説明

全般

全設定項目の読み出しと書き込みのプログラム例です。下記の4つを実行します。

・MX100から設定データを一括受信

・設定データ領域の設定データを1項目ずつ取得

・設定データを1項目ずつ設定データ領域に書き込む

・MX100に設定データを一括送信

先頭番号から最終番号まで，1項目ずつ取得と書き込みをしています。

文字列領域はサイズに余裕を持って用意してください。

項目番号と項目文字列の組を保存，ロードすることで設定データをバックアップすることも可能になります。

設定項目番号については，6.3節を参照してください。

通信接続

```
comm = openMX100("192.168.1.12", rc)
```

ＭＸ１００のＩＰアドレスを指定しています。通信用ポートは，通信用定数DAQMX_COMMPORT(MX100の通信ポート番号)を指定したことになります。

設定データの一括受信

```
rc = getItemAllMX100(comm)
```

MX100の設定データの全項目を一括受信し，設定データ領域に格納します。

設定データを1項目ずつ取得

```
rc = readItemMX100(comm, i, strItem, lenItem, realLen)
```

設定データ領域から項目番号「i」の内容を取得します。

設定データを1項目ずつ書き込む

```
rc = writeItemMX100(comm, i, strItem)
```

設定データ領域の項目番号「i」に，strItemの内容を書き込みます。

設定データの一括送信

```
rc = setItemAllMX100(comm)
```

設定データの全項目をMX100に一括送信します。

通信切断

```
rc = closeMX100(comm)
```

通信を切断します。

16.1 機能と関数の対応ーMX100/C#ー

拡張APIでサポートする機能と，Visual Basicの関数郡の対応を示します。
状態遷移関数と取得関数の2種類あります。
状態遷移関数はMX100本体を制御します。データ取得機能で測定データを取得すると測定点が1点分だけ進みます(拡張APIの状態が遷移します)。
取得関数は項目値を返します。取得関数を使用した場合，データ値取得を使用すると拡張APIが保持している現在の状態のデータ値を返します(拡張APIの状態は遷移しません)。

状態遷移関数

表の「FIFO」欄は，FIFO中に関数を実行したときの，FIFOの動作を示します。
停止：関数を実行するとFIFOを停止します。
継続：関数を実行してもFIFOを継続します。

通信機能

機能	FIFO	関数
MX100と通信接続	継続	DAQMX100:: open
MX100との通信を切断	継続	DAQMX100:: close

FIFOの開始/停止

機能	FIFO	関数
FIFOを開始	継続	DAQMX100. measStartMX100
FIFOを停止	停止	DAQMX100. measStopMX100

制御機能

機能		FIFO	関数
日付時刻設定 現在時刻		停止	DAQMX100. setDateTimeNowMX100
バックアップ 有効無効の設定		継続	DAQMX100. switchBackupMX100
CFカードのフォーマット		停止	DAQMX100. formatCFMX100
ユニット	システムの再構築	停止	DAQMX100. reconstructMX100
	システムの初期化	停止	DAQMX100. initSetValueMX100
	アラームリセット（アラームACK）	停止	DAQMX100. ackAlarmMX100
7セグメントLEDの表示		継続	DAQMX100. displaySegmentMX100
保持データの初期化	チャネル指定	継続	DAQMX100. initDataChMX100
	FIFO指定	継続	DAQMX100. initDataFIFOMX100

制御機能は，通信の最後に状態更新を行います。
各データの送信，設定機能については， データ操作機能を参照してください。

設定機能

機能		FIFO	関数
設定データを一括設定(一括送信)	全設定データ	停止	DAQMX100. sendConfigMX100
	基本設定データ	停止	DAQMX100. sendConfigMX100
設定データを個別設定	システム構成データ	停止	DAQMX100. sendConfigMX100
	チャネル設定データ	停止	DAQMX100. sendConfigMX100
	初期バランスデータ	停止	DAQMX100. sendConfigMX100
	出力チャネルデータ	停止	DAQMX100. sendConfigMX100
初期バランスデータ	実行	停止	DAQMX100. initBalanceMX100
	リセット	停止	DAQMX100. clearBalanceMX100

設定データの設定機能は，保持しているデータを送信します。
任意の初期バランスデータを設定する場合，データ操作機能で初期バランスデータの送信機能を参照してください。

設定変更機能

設定機能は，設定送信して，状態更新を行います。
単独チャネル毎の設定なので，設定できなかった場合，原則エラーを返します。
データ値，または，測定値(倍精度浮動小数)での指定ができます。

レンジ設定

機能	FIFO	関数
スキップ	停止	DAQMX100. setRangMX100
直流電圧入力	停止	DAQMX100. setRangMX100
熱電対入力	停止	DAQMX100. setRangMX100
測温抵抗体	停止	DAQMX100. setRangMX100
ディジタル入力	停止	DAQMX100. setRangMX100
抵抗	停止	DAQMX100. setRangMX100
ひずみ	停止	DAQMX100. setRangMX100
AO	停止	DAQMX100. setRangMX100
PWM	停止	DAQMX100. setRangMX100
チャネル間差演算	停止	DAQMX100. setChDELTAMX100
リモートRRJC	停止	DAQMX100. setChRRJCMX100
パルス	停止	DAQMX100. setRangeMX100
通信	停止	DAQMX100. setRangeMX100

チャネル設定

機能		FIFO	関数
単位名		停止	DAQMX100. setChUnitMX100
タグ		停止	DAQMX100. setChTagMX100
コメント		停止	DAQMX100. setChCommentMX100
AI/DI/AO/PWM	スパン	停止	DAQMX100. setSpanMX100
			DAQMX100. setDoubleSpanMX100
AI/DI	スケール	停止	DAQMX100. setScaleMX100
			DAQMX100. setDoubleScaleMX100
	アラーム	停止	DAQMX100. setAlarmMX100
			DAQMX100. setDoubleAlarmMX100
			DAQMX100. setAlarmValueMX100
			DAQMX100. setDoubleAlarmValueMX100
	ヒステリシス	停止	DAQMX100. setHisterisysMX100
			DAQMX100. setDoubleHisterisysMX100
AI	フィルタ係数	停止	DAQMX100. setFilterMX100
	基準接点補償(RJC)	停止	DAQMX100. setRJCTypeMX100
	バーンアウト	停止	DAQMX100. setBurnoutMX100
DO	非励磁	停止	DAQMX100. setDeenergizeMX100
	保持	停止	DAQMX100. setHoldMX100
	参照アラーム	停止	DAQMX100. setRefAlarmMX100
チャネル種類	DO種類	停止	DAQMX100. setChKindMX100
	AO種類	停止	DAQMX100. setChKindMX100
	PWM種類	停止	DAQMX100. setChKindMX100
PI	チャタリングフィルタ	停止	DAQMX100. setChatFilterMX100

モジュール設定

機能	FIFO	関数
周期種類	停止	DAQMX100. setIntervalMX100
A/D積分時間種類	停止	DAQMX100. setIntegralMX100

ユニット設定

機能	FIFO	関数
ユニット番号	停止	DAQMX100. setUnitNoMX100
温度単位種類	停止	DAQMX100. setUnitTempMX100
CF書き込み種類	停止	DAQMX100. setCFWriteModeMX100

出力チャネルデータ

機能	FIFO	クラスと関数メンバ
出力種類	停止	DAQMX100. setOutputTypeMX100
選択値	停止	DAQMX100. setChoiceMX100 DAQMX100. setDoubleChoiceMX100
パルス周期倍率	停止	DAQMX100. setPulseTimeMX100

データ操作機能

DOデータ

機能		FIFO	関数
作成		継続	DAQMX100. createDOMX100
削除		継続	DAQMX100. deleteDOMX100
部分変更	ユーザ指定	継続	DAQMX100. changeDOMX100
	コピー	継続	DAQMX100. copyDOMX100
送信	既存指定	継続	DAQMX100. commandDOMX100
	変更指定	継続	DAQMX100. switchDOMX100

AO/PWMデータ

機能		FIFO	関数
作成		継続	DAQMX100. createAOPWMMX100
削除		継続	DAQMX100. deleteAOPWMMX100
部分変更	出力データ値	継続	DAQMX100. changeAOPWMMX100
	実出力値	継続	DAQMX100. changeAOPWMValueMX100
	コピー	継続	DAQMX100. copyAOPWMMX100
送信		継続	DAQMX100. commandAOPWMMX100

初期バランスデータ

機能		FIFO	関数
作成		継続	DAQMX100. createBalanceMX100
削除		継続	DAQMX100. deleteBalanceMX100
部分変更	ユーザ指定	継続	DAQMX100. changeBalanceMX100
	コピー	継続	DAQMX100. copyBalanceMX100
送信		停止	DAQMX100. commandBalanceMX100

伝送出力データ

機能		FIFO	関数
作成		継続	DAQMX100. createTransmitMX100
削除		継続	DAQMX100. deleteTransmitMX100
部分変更	ユーザ指定	継続	DAQMX100. changeTransmitMX100
	コピー	継続	DAQMX100. copyTransmitMX100
送信	既存指定	継続	DAQMX100. commandTransmitMX100
	変更指定	継続	DAQMX100. switchTransmitMX100

各データ識別子で操作します。
送信以外は，状態更新(通信)を行いません。

取得機能

機能		FIFO	関数
ステータス		継続	DAQMX100. updateStatusMX100
システム構成データ		継続	DAQMX100. updateSystemMX100
設定データ		継続	DAQMX100. updateConfigMX100
出力データ	DOデータ	継続	DAQMX100. updateDODataMX100
	AO/PWMデータ 伝送出力データ	継続	DAQMX100. updateAOPWMDataMX100
チャネル情報データ		継続	DAQMX100. updateInfoChMX100
測定データ	FIFO値	継続	DAQMX100. measDataChMX100
(チャネル指定)	瞬時値	継続	DAQMX100. measInstChMX100
測定データ	FIFO値	継続	DAQMX100. measDataFIFOMX100
(FIFO指定)	瞬時値	継続	DAQMX100. measInstFIFOMX100
初期バランスデータ		継続	DAQMX100. updateBalanceMX100
出力チャネルデータ		継続	DAQMX100. updateOutputMX100

データ取得は，本API内部で一括取得が行われます。
収集によって，状態更新も行われます。
チャネル情報データや設定データ(システム構成データ，初期バランスデータ，出力チャネルデータを含む)は，内部で保持されていますが，ユーザが明示的に保持しているデータを更新できます。

設定項目

機能		FIFO	関数
設定データ	一括受信	継続	DAQMX100. getItemAllMX100
	一括送信	停止	DAQMX100. setItemAllMX100
設定項目	読み出し	継続	DAQMX100. readItemMX100
	書き込み	継続	DAQMX100. writeItemMX100
	初期化	継続	DAQMX100. initItemMX100

設定項目の読み出し，書き込み，初期化は，保持している領域へのアクセスで，領域の整合性チェックをしません。また，状態更新(通信)を行いません。

取得関数

測定データ

データ名		関数
データ値		DAQMX100. dataValueMX100
データステータス値		DAQMX100. dataStatusMX100
アラーム（有無）		DAQMX100. dataAlarmMX100
測定値	倍精度浮動小数	DAQMX100. dataDoubleValueMX100
	文字列	DAQMX100. dataStringValueMX100
時刻	秒数	DAQMX100. dataTimeMX100
	ミリ秒	DAQMX100. dataMilliSecMX100
	年	DAQMX100. dataYearMX100
	月	DAQMX100. dataMonthMX100
	日	DAQMX100. dataDayMX100
	時	DAQMX100. dataHourMX100
	分	DAQMX100. dataMinuteMX100
	秒	DAQMX100. dataSecondMX100
有効データ(有無)		DAQMX100. dataValidMX100

チャネル情報データ

データ名	関数
FIFO番号	DAQMX100. channelFIFONoMX100
FIFO内チャネル順序番号	DAQMX100. channelFIFOIndexMX100
表示最小値	DAQMX100. channelDisplayMinMX100
表示最大値	DAQMX100. channelDisplayMaxMX100
実範囲最小値	DAQMX100. channelRealMinMX100
実範囲最大値	DAQMX100. channelRealMaxMX100

チャネル設定データ

データ名				関数
チャネルステータス(有無)				DAQMX100. channelValidMX100
小数点位置				DAQMX100. channelPointMX100
チャネル種類				DAQMX100. channelKindMX100
レンジ種類				DAQMX100. channelRangeMX100
スケール種類				DAQMX100. channelScaleTypeMX100
単位名				DAQMX100. toChannelUnitMX100
タグ				DAQMX100. toChannelTagMX100
コメント				DAQMX100. toChannelCommentMX100
AI/DI/AO/PWM				
	スパン	最小値	データ値	DAQMX100. channelSpanMinMX100
			測定値	DAQMX100. channelDoubleSpanMinMX100
		最大値	データ値	DAQMX100. channelSpanMaxMX100
			測定値	DAQMX100. channelDoubleSpanMaxMX100
AI/DI	スケール	最小値	データ値	DAQMX100. channelScaleMinMX100
			測定値	DAQMX100. channelDoubleScaleMinMX100
		最大値	データ値	DAQMX100. channelScaleMaxMX100
			測定値	DAQMX100. channelDoubleScaleMaxMX100
	アラーム種類			DAQMX100. alarmTypeMX100
	アラーム値(ON値)		データ値	DAQMX100. alarmValueONMX100
			測定値	DAQMX100. alarmDoubleValueONMX100
	アラーム値(OFF値)		データ値	DAQMX100. alarmValueOFFMX100
			測定値	DAQMX100. alarmDoubleValueOFFMX100
	ヒステリシス		データ値	DAQMX100. alarmHisterisysMX100
			測定値	DAQMX100. alarmDoubleHisterisysMX100
AI	フィルタ			DAQMX100. channelFilterMX100
	RJC種類			DAQMX100. channelRJCTypeMX100
	RJC電圧値			DAQMX100. channelRJCVoltMX100
	バーンアウト			DAQMX100. channelBurnoutMX100
DO	非励磁			DAQMX100. channelDeenergizeMX100
	保持			DAQMX100. channelHoldMX100
	参照アラーム			DAQMX100. channelRefAlarmMX100
チャネル間差演算/リモートRJC/AO/PWM				
	基準チャネル番号			DAQMX100. channelRefChNoMX100
初期バランスデータ		有効無効値		DAQMX100. channelBalanceValidMX100
		初期バランス値		DAQMX100. channelBalanceValueMX100
出力チャネルデータ		出力種類		DAQMX100. channelOutputTypeMX100
		アイドル時の選択値		DAQMX100. channelIdleChoiceMX100
		エラー時の選択値		DAQMX100. channelErrorChoiceMX100
		選択値が「指定値」の場合の値		
			データ値	DAQMX100. channelPresetValueMX100
			測定値	DAQMX100. channelDoublePresetValueMX100
		パルス周期倍率		DAQMX100. channelPulseTimeMX100
PI	チャタリングフィルタ			DAQMX100. channelChatFilterMX100

ネットワーク情報データ

データ名	関数
ホスト名	DAQMX100. toNetHostMX100
IPアドレス	DAQMX100. netAddressMX100
ポート番号	DAQMX100. netPortMX100
サブネットマスク	DAQMX100. netSubmaskMX100
Gatewayアドレス	DAQMX100. netGatewayMX100

システム構成データ

データ名		関数
モジュール	モジュール種類	DAQMX100. moduleTypeMX100
	チャネル数	DAQMX100. moduleChNumMX100
	周期種類	DAQMX100. moduleIntervalMX100
	AD積分時間種類	DAQMX100. moduleIntegralMX100
	有効無効値	DAQMX100. moduleValidMX100
	起動時モジュール種類	DAQMX100. moduleStandbyTypeMX100
	実際のモジュール種類	DAQMX100. moduleRealTypeMX100
	端子種類	DAQMX100. moduleTerminalMX100
	バージョン	DAQMX100. moduleVersionMX100
	FIFO番号	DAQMX100. moduleFIFONoMX100
	シリアル番号	DAQMX100. toModuleSerialMX100
ユニット	ユニット種類	DAQMX100. unitTypeMX100
	スタイル	DAQMX100. unitStyleMX100
	ユニット番号	DAQMX100. unitNoMX100
	温度単位種類	DAQMX100. unitTempMX100
	電源周波数	DAQMX100. unitFrequencyMX100
	パート番号	DAQMX100. toUnitPartNoMX100
	オプション	DAQMX100. unitOptionMX100
	シリアル番号	DAQMX100. toUnitSerialMX100
	MACアドレス	DAQMX100. unitMACMX100
	CF書き込み種類	DAQMX100. unitCFWriteModeMX100

ステータスデータ

データ名		関数
ユニットステータス値		DAQMX100. statusUnitMX100
FIFOの有効個数		DAQMX100. statusFIFONumMX100
バックアップ(有無)		DAQMX100. statusBackupMX100
FIFO	FIFOステータス値	DAQMX100. statusFIFOMX100
	周期種類	DAQMX100. statusFIFOIntervalMX100
CF	CFステータス種類	DAQMX100. statusCFMX100
	サイズ	DAQMX100. statusCFSizeMX100
	残容量	DAQMX100. statusCFRemainMX100
ステータス返却時刻	秒数	DAQMX100. statusTimeMX100
	ミリ秒	DAQMX100. statusMilliSecMX100
	年	DAQMX100. statusYearMX100
	月	DAQMX100. statusMonthMX100
	日	DAQMX100. statusDayMX100
	時	DAQMX100. statusHourMX100
	分	DAQMX100. statusMinuteMX100
	秒	DAQMX100. statusSecondMX100

カレントデータ

データ名		関数
DOデータ	有効無効値	DAQMX100. currentDOValidMX100
	ON/OFF状態	DAQMX100. currentDOValueMX100
AO/PWMデータ	有効無効値	DAQMX100. currentAOPWMValidMX100
	出力データ値	DAQMX100. currentAOPWMValueMX100
	出力値	DAQMX100. currentDoubleAOPWMValueMX100
初期バランスデータ	有効無効値	DAQMX100. currentBalanceValidMX100
	初期バランス値	DAQMX100. currentBalanceValueMX100
	初期バランス結果	DAQMX100. currentBalanceResultMX100
伝送出力データ伝送状態		DAQMX100. currentTransmitMX100

データ取得機能で取得された各データの状態です。
初期バランスデータの初期バランス結果は，設定機能による実行結果です。
DOデータ， AO/PWMデータなど，実際に出力されている出力状態をカレントデータとして取得できます。ただし，データを送信した直後は，設定した値が返却されて，実際の出力は次のタイミングになることがあります。
保持しているデータは，状態更新で取得したときの値です。取得関数を呼び出した時刻のデータではありません。

ユーザデータ

データ名		関数
DOデータ	有効無効値	DAQMX100. userDOValidMX100
	ON/OFF状態	DAQMX100. userDOValueMX100
AO/PWMデータ	有効無効値	DAQMX100. userAOPWMValidMX100
	出力データ値	DAQMX100. userAOPWMValueMX100
	出力値	DAQMX100. userDoubleAOPWMValueMX100
初期バランスデータ	有効無効値	DAQMX100. userBalanceValidMX100
	初期バランス値	DAQMX100. userBalanceValueMX100
伝送出力データ	伝送状態	DAQMX100. userTransmitMX100

ユーザがデータ操作機能で作成したデータの値を取得します。

ユーティリティ

機能/データ名		関数
残りデータ個数	チャネル単位で取得	DAQMX100. dataNumChMX100
	FIFO単位で取得	DAQMX100. dataNumFIFOMX100
エラー	MX固有エラーの取得	DAQMX100. lastErrorMX100
	エラーメッセージ文字列を取得	DAQMX100. toErrorMessageMX100
	エラーメッセージ文字列の最大長を取得	DAQMX100. errorMaxLengthMX100
	エラー検出した設定項目番号を取得	DAQMX100. itemErrorMX100
FIFO情報から，チャネル番号に変換		DAQMX100. channelNumberMX100
レンジ種類別の小数点位置を取得		DAQMX100. rangePointMX100
測定値	倍精度浮動小数に変換	DAQMX100. toDoubleValueMX100
	文字列に変換	DAQMX100. toStringValueMX100
アラーム	アラーム種類の文字列を取得	DAQMX100. toAlarmNameMX100
	アラーム文字列の最大長を取得	DAQMX100. alarmMaxLengthMX100
本APIのバージョン番号を取得		DAQMX100. versionAPIMX100
本APIのリビジョン番号を取得		DAQMX100. revisionAPIMX100
IPアドレスのパート分割を取得		DAQMX100. addressPartMX100
AO/PWM	出力値を出力データ値に変換	DAQMX100. toAOPWMValueMX100
	出力データ値を出力値に変換	DAQMX100. toRealValueMX100
設定項目	設定項目番号から設定項目文字列を取得	DAQMX100. toItemNameMX100
	設定項目文字列から設定項目番号を取得	DAQMX100. toItemNoMX100
	設定項目文字列の最大長を取得	DAQMX100. itemMaxLengthMX100
スタイルバージョンに変換		DAQMX100. toStyleVersionMX100

16.2 プログラムーMX100/C#ー

関数，定数の宣言

C#用の関数，定数を使用するためには，あらかじめ宣言をしておく必要があります。次の宣言記述方法があります。

宣言の記述

プロジェクトにC#用モジュールファイル(DAQMX100.cs)を追加すると，すべての関数，定数を宣言したことになります。

測定データの取得

プログラム例

```
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);
          }
  }
}
```

説明

全般

データ取得は，FIFOを開始することで可能になります。MX100のチャネル1のFIFOデータのうち，取得可能な分の測定データを一度に取得し，領域に格納します。その中から，現在状態(先頭の計測点)の測定値データ(1点)を取得し，終了します。

通信接続

```
int comm = DAQMX100.openMX100(enc.GetBytes(address), out rc);
```

ＭＸ１００のＩＰアドレスを指定しています。通信用ポートは，通信用定数DAQMX_COMMPORT(MX100の通信ポート番号)を指定したことになります。

FIFO開始

```
rc = DAQMX100.measStartMX100(comm);
```

MX100でFIFOを開始します。

チャネル1の測定データの取得

```
rc = DAQMX100.measDataChMX100(comm, 1);
```

MX100から，チャネル1の取得可能な分の測定データを一度に取得し，領域に格納します。先頭の計測点を現在状態とします。

測定値の取得

```
int val = DAQMX100.dataValueMX100(comm, 1);
```

測定データを格納している領域から，チャネル1の現在状態の測定値を取得します。

FIFO停止

```
rc = DAQMX100.measStopMX100(comm);
```

FIFOを停止します。

通信切断

```
rc = DAQMX100.closeMX100(comm);
```

通信を切断します。

参考

サンプルプログラムでは，measDataChMX100を一度だけ実行して終了しています。measDataChMX100を繰り返して実行すると，実行されるごとに，計測点をひとつ進めて現在状態とします。格納している計測点の最後まで到達したら，続く取得可能な分のデータを取得します。

設定データの読み出しと書き込み

プログラム例

```
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);
          }
  }
}
```

説明

全般

全設定項目の読み出しと書き込みのプログラム例です。下記の4つを実行します。
・MX100から設定データを一括受信
・設定データ領域の設定データを1項目ずつ取得
・設定データを1項目ずつ設定データ領域に書き込む
・MX100に設定データを一括送信
先頭番号から最終番号まで，1項目ずつ取得と書き込みをしています。
設定項目を使用して，設定データの一括処理が可能になります。
文字列領域はサイズに余裕を持って用意してください。
項目番号と項目文字列の組を保存，ロードすることで設定データをバックアップすることも可能になります。
設定項目番号については，6.3節を参照してください。

通信接続

```
int comm = DAQMX100.openMX100(enc.GetBytes(address), out rc);
```

ＭＸ１００のＩＰアドレスを指定しています。通信用ポートは，通信用定数DAQMX_COMMPORT(MX100の通信ポート番号)を指定したことになります。

設定データの一括受信

```
rc = DAQMX100.getItemAllMX100(comm);
```

MX100の設定データの全項目を一括受信し，設定データ領域に格納します。

設定データを1項目ずつ取得

```
rc = DAQMX100.readItemMX100(comm, i, strItem, lenItem, out
realLen);
```

設定データ領域から項目番号「i」の内容を取得します。

設定データを1項目ずつ書き込む

```
rc = DAQMX100.writeItemMX100(comm, i, strItem);
```

設定データ領域の項目番号「i」に，strItemの内容を書き込みます。

設定データの一括送信

```
rc = DAQMX100.setItemAllMX100(comm);
```

設定データの全項目をMX100に一括送信します。

通信切断

```
rc = DAQMX100.closeMX100(comm);
```

通信を切断します。

17.1 関数の詳細ーMX100(Visual C/Visual Basic/Visual Basic.NET/C#)ー状態遷移関数

ここでは，Visual C，Visual Basic，Visual Basic.NET，およびC#で使用するMX100用関数について説明しています。関数は，関数名のアルファベット順で並んでいます。

定数，型については第18章をご覧ください。
MX100の用語については付録1をご覧ください。

ほとんどの関数は戻り値として，エラー番号を返します。正常終了の場合は，エラー番号「0」を返します。

C#の場合，宣言をまとめたクラス(DAQMX100)のメンバになります。

ackAlarmMX100

構文

```
int ackAlarmMX100(DAQMX100 daqmx100);
```

宣言

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);
```

引数

daqmx100　　機器記述子を指定します。

説明

アラームリセットを実行します。

戻り値

エラー番号を返します。

エラー：

Not descriptor　機器記述子がありません。

参照

```
CDAQMX100::ackAlarm
```

changeAOPWMMX100

構文

```
int changeAOPWMMX100(DAQMX100 daqmx100, int idAOPWM, int
aopwmNo, int bValid, int iAOPWMValue);
```

宣言

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);
```

引数

daqmx100	機器記述子を指定します。
idAOPWM	AO/PWMデータ識別子を指定します。
aopwmNo	AO/PWMデータ番号を指定します。
bValid	有効/無効を有効無効値で指定します。
iAOPWMValue	出力データ値を指定します。

説明

指定されたAO/PWMデータ識別子のAO/PWMデータを変更します。

・ 出力データ値は，実際の出力値を変換した値を指定します。

戻り値

エラー番号を返します。

エラー：

Not descriptor　機器記述子がありません。

参照

```
CDAQMX100::getClassMXAOPWMList
CDAQMXAOPWMList::change
```

changeAOPWMValueMX100

構文

```
int changeAOPWMValueMX100(DAQMX100 daqmx100, int idAOPWM, int
aopwmNo, int bValid, double realValue);
```

宣言

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
Doouble) 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);
```

引数

daqmx100	機器記述子を指定します。
idAOPWM	AO/PWMデータ識別子を指定します。
aopwmNo	AO/PWMデータ番号を指定します。
bValid	有効/無効を有効無効値で指定します。
realValue	実際の出力値を指定します。

説明

指定されたAO/PWMデータ識別子のAO/PWMデータを変更します。

・ ユーザ指定の出力値の違いを除けば，changeAOPWMMX100関数と同じです。

・ ユーザ指定の出力値は，小数点位置を含む浮動小数値を指定します。

戻り値

エラー番号を返します。

エラー：

Not descriptor　機器記述子がありません。

参照

```
CDAQMX100::changeAOPWMValue
```

changeBalanceMX100

構文

```
int changeBalanceMX100(DAQMX100 daqmx100, int idBalance, int
balanceNo, int bValid, int iValue);
```

宣言

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);
```

引数

daqmx100	機器記述子を指定します。
idBalance	初期バランスデータ識別子を指定します。
balanceNo	初期バランスデータ番号を指定します。
bValid	有効/無効を有効無効値で指定します。
iValue	初期バランス値を指定します。

説明

指定された初期バランスデータ識別子の初期バランスデータを変更します。

戻り値

エラー番号を返します。

エラー：

Not descriptor　機器記述子がありません。

参照

```
CDAQMX100::getClassMXBalanceList
CDAQMXBalanceList::change
```

changeDOMX100

構文

```
int changeDOMX100(DAQMX100 daqmx100, int idDO, int doNo, int
bValid, int bONOFF);
```

宣言

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);
```

引数

daqmx100	機器記述子を指定します。
idDO	DOデータ識別子を指定します。
doNo	DOデータ番号を指定します。
bValid	有効/無効を有効無効値で指定します。
bONOFF	ON/OFFを有効無効値で指定します。

説明

指定されたDOデータ識別子のDOデータを変更します。

戻り値

エラー番号を返します。

エラー：

Not descriptor　機器記述子がありません。

参照

```
CDAQMX100::getClassMXDOList
CDAQMXDOList::change
```

changeTransmitMX100

構文

```
int changeTransmitMX100(DAQMX100 daqmx100, int idTrans, int
aopwmNo, int iTransmit);
```

宣言

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);
```

引数

daqmx100	機器記述子を指定します。
idTrans	伝送出力データ識別子を指定します。
aopwmNo	AO/PWMデータ番号を指定します。
iTransmit	伝送状態を指定します。

説明

指定された伝送出力データ識別子の伝送出力データを変更します。

戻り値

エラー番号を返します。

エラー：

Not descriptor　機器記述子がありません。

参照

```
CDAQMX100::getClassMXTransmitList
CDAQMXTransmitList::change
```

clearBalanceMX100

構文

```
int clearBalanceMX100(DAQMX100 daqmx100);
```

宣言

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);
```

引数

daqmx100　　機器記述子を指定します。

説明

初期バランス値を初期化します。

戻り値

エラー番号を返します。

エラー：

Not descriptor　機器記述子がありません。

参照

```
CDAQMX100::clearBalance
```

closeMX100

構文

```
int closeMX100(DAQMX100 daqmx100);
```

宣言

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);
```

引数

daqmx100　　機器記述子を指定します。

説明

指定された機器記述子による通信を切断をします。

・ 通信を切断すると，機器記述子の値は無意味になります。

・ 切断後は，機器記述子の値は使用しないでください。

戻り値

エラー番号を返します。

エラー：

Not descriptor　機器記述子がありません。

参照

`CDAQMX100::close`

commandAOPWMMX100

構文

```
int commandAOPWMMX100(DAQMX100 daqmx100, int idAOPWM);
```

宣言

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);
```

引数

daqmx100	機器記述子を指定します。
idAOPWM	AO/PWMデータ識別子を指定します。

説明

指定されたAO/PWMデータ識別子のAO/PWMデータを送信します。

戻り値

エラー番号を返します。

エラー：

Not descriptor　機器記述子がありません。

参照

CDAQMX100::commandAOPWM

commandBalanceMX100

構文

```
int commandBalanceMX100(DAQMX100 daqmx100, int idBalance);
```

宣言

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);
```

引数

daqmx100　　機器記述子を指定します。

idBalance　　初期バランスデータ識別子を指定します。

説明

指定された初期バランスデータ識別子の初期バランスデータを送信します。

戻り値

エラー番号を返します。

エラー：

Not descriptor　機器記述子がありません。

参照

```
CDAQMX100::reloadBalance
```

commandDOMX100

構文

```
int commandDOMX100(DAQMX100 daqmx100, int idDO);
```

宣言

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);
```

引数

daqmx100	機器記述子を指定します。
idDO	DOデータ識別子を指定します。

説明

指定されたDOデータ識別子のDOデータを送信します。

戻り値

エラー番号を返します。

エラー：

Not descriptor　機器記述子がありません。

参照

`CDAQMX100::commandDO`

commandTransmitMX100

構文

```
int commandTransmitMX100(DAQMX100 daqmx100, int idTrans);
```

宣言

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);
```

引数

daqmx100	機器記述子を指定します。
idTrans	伝送出力データ識別子を指定します。

説明

指定された伝送出力データ識別子の伝送出力データを送信します。

戻り値

エラー番号を返します。

エラー：

Not descriptor	機器記述子がありません。

参照

```
CDAQMX100::commandTransmit
```

copyAOPWMMX100

構文

```
int copyAOPWMMX100(DAQMX100 daqmx100, int idAOPWM, int
idAOPWMSrc);
```

宣言

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);
```

引数

daqmx100	機器記述子を指定します。
idAOPWM	コピー先のAO/PWMデータ識別子を指定します。
idAOPWMSrc	コピー元のAO/PWMデータ識別子を指定します。

説明

指定されたAO/PWMデータ識別子のコピー元からコピー先へAO/PWMデータを複写します。

・ コピー元に定数値の「カレントデータ指定」を指定すると，保持している機器の状態データをコピー元にします。

戻り値

エラー番号を返します。

エラー：

Not descriptor　機器記述子がありません。

参照

```
CDAQMX100::getClassMXAOPWMList
CDAQMXAOPWMList::copy
```

copyBalanceMX100

構文

```
int copyBalanceMX100(DAQMX100 daqmx100, int idBalance, int
idBalanceSrc);
```

宣言

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);
```

引数

daqmx100	機器記述子を指定します。
idBalance	コピー先の初期バランスデータ識別子を指定します。
idBalanceSrc	コピー元の初期バランスデータ識別子を指定します。

説明

指定された初期バランスデータ識別子のコピー元からコピー先へ初期バランスデータを複写します。

・ コピー元に定数値の「カレントデータ指定」を指定すると，保持している機器の状態データをコピー元にします。

戻り値

エラー番号を返します。

エラー：

Not descriptor　機器記述子がありません。

参照

```
CDAQMX100::getClassMXBalanceList
CDAQMXBalanceList::copy
```

copyDOMX100

構文

```
int copyDOMX100(DAQMX100 daqmx100, int idDO, int idDOSrc);
```

宣言

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);
```

引数

daqmx100	機器記述子を指定します。
idDO	コピー先のDOデータ識別子を指定します。
idDOSrc	コピー元のDOデータ識別子を指定します。

説明

指定されたDOデータ識別子のコピー元からコピー先へDOデータを複写します。

・ コピー元に定数値の「カレントデータ指定」を指定すると，保持している機器の状態データをコピー元にします。

戻り値

エラー番号を返します。

エラー：

Not descriptor　機器記述子がありません。

参照

```
CDAQMX100::getClassMXDOList
CDAQMXDOList::copy
```

copyTransmitMX100

構文

```
int copyTransmitMX100(DAQMX100 daqmx100, int idTrans, int
idTransSrc);
```

宣言

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);
```

引数

daqmx100	機器記述子を指定します。
idTrans	コピー先の伝送出力データ識別子を指定します。
idTransSrc	コピー元の伝送出力データ識別子を指定します。

説明

指定された伝送出力データ識別子のコピー元からコピー先へ伝送出力データを複写します。

・ コピー元に定数値の「カレントデータ指定」を指定すると，保持している機器の状態データをコピー元にします。

戻り値

エラー番号を返します。

エラー：

Not descriptor　機器記述子がありません。

参照

```
CDAQMX100::getClassMXTransmitList
CDAQMXTransmitList::copy
```

createAOPWMMX100

構文

```
int createAOPWMMX100(DAQMX100 daqmx100, int * errCode);
```

宣言

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);
```

引数

daqmx100	機器記述子を指定します。
errorCode	エラー番号の返却先を指定します。

説明

新規にAO/PWMデータを作成します。

- AOPWMデータ識別子を戻り値として返却します。
- 失敗した場合，負の数を返します。
- 返却先が指定されていれば，エラー番号を格納します。

戻り値

データ識別子を返します。

エラー：

Not descriptor	機器記述子がありません。
Not data	データ作成に失敗しました。

参照

```
CDAQMX100::getClassMXAOPWMList
CDAQMXAOPWMList::create
```

createBalanceMX100

構文

```
int createBalanceMX100(DAQMX100 daqmx100, int * errCode);
```

宣言

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);
```

引数

daqmx100	機器記述子を指定します。
errorCode	エラー番号の返却先を指定します。

説明

新規に初期バランスデータを作成します。

- 初期バランスデータ識別子を戻り値として返却します。
- 失敗した場合，負の数を返します。
- 返却先が指定されていれば，エラー番号を格納します。

戻り値

データ識別子を返します。

エラー：

Not descriptor	機器記述子がありません。
Not data	データ作成に失敗しました。

参照

```
CDAQMX100::getClassMXBalanceList
CDAQMXBalanceList::create
```

createDOMX100

構文

```
int createDOMX100(DAQMX100 daqmx100, int * errorCode);
```

宣言

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);
```

引数

daqmx100	機器記述子を指定します。
errorCode	エラー番号の返却先を指定します。

説明

新規にDOデータを作成します。

- DOデータ識別子を戻り値として返却します。
- 失敗した場合，負の数を返します。
- 返却先が指定されていれば，エラー番号を格納します。

戻り値

データ識別子を返します。

エラー：

Not descriptor	機器記述子がありません。
Not data	データ作成に失敗しました。

参照

```
CDAQMX100::getClassMXDOList
CDAQMXDOList::create
```

createTransmitMX100

構文

```
int createTransmitMX100(DAQMX100 daqmx100, int * errCode);
```

宣言

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);
```

引数

daqmx100	機器記述子を指定します。
errorCode	エラー番号の返却先を指定します。

説明

新規に伝送出力データを作成します。

- 伝送出力データ識別子を戻り値として返却します。
- 失敗した場合，負の数を返します。
- 返却先が指定されていれば，エラー番号を格納します。

戻り値

データ識別子を返します。

エラー：

Not descriptor	機器記述子がありません。
Not data	データ作成に失敗しました。

参照

```
CDAQMX100::getClassMXTransmitList
CDAQMXTransmitList::create
```

deleteAOPWMMX100

構文

```
int deleteAOPWMMX100(DAQMX100 daqmx100, int idAOPWM);
```

宣言

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);
```

引数

daqmx100	機器記述子を指定します。
idAOPWM	AO/PWMデータ識別子を指定します。

説明

指定されたAO/PWMデータ識別子のAO/PWMデータを削除します。

・ データ識別子に，定数値の「全データ識別子指定」を指定すると，リスト全体を初期化します。

戻り値

エラー番号を返します。

エラー：

Not descriptor　機器記述子がありません。

参照

```
CDAQMX100::getClassMXAOPWMList
CDAQMXAOPWMList::del
CDAQMXAOPWMList::initialize
```

deleteBalanceMX100

構文

```
int deleteBalanceMX100(DAQMX100 daqmx100, int idBalance);
```

宣言

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);
```

引数

daqmx100	機器記述子を指定します。
idBalance	初期バランスデータ識別子を指定します。

説明

指定された初期バランスデータ識別子の初期バランスデータを削除します。

・ データ識別子に，定数値の「全データ識別子指定」を指定すると，リスト全体を初期化します。

戻り値

エラー番号を返します。

エラー：

Not descriptor　機器記述子がありません。

参照

```
CDAQMX100::getClassMXBalanceList
CDAQMXBalanceList::del
CDAQMXBalanceList::initialize
```

deleteDOMX100

構文

```
int deleteDOMX100(DAQMX100 daqmx100, int idDO);
```

宣言

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);
```

引数

daqmx100	機器記述子を指定します。
idDO	DOデータ識別子を指定します。

説明

指定されたDOデータ識別子のDOデータを削除します。

・ データ識別子に，定数値の「全データ識別子指定」を指定すると，リスト全体を初期化します。

戻り値

エラー番号を返します。

エラー：

Not descriptor　機器記述子がありません。

参照

```
CDAQMX100::getClassMXDOList
CDAQMXDOList::del
CDAQMXDOList::initialize
```

deleteTransmitMX100

構文

```
int deleteTransmitMX100(DAQMX100 daqmx100, int idTrans);
```

宣言

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);
```

引数

daqmx100	機器記述子を指定します。
idTrans	伝送出力データ識別子を指定します。

説明

指定された伝送出力データ識別子の伝送出力データを削除します。

・ データ識別子に，定数値の「全データ識別子指定」を指定すると，リスト全体を初期化します。

戻り値

エラー番号を返します。

エラー：

Not descriptor　機器記述子がありません。

参照

```
CDAQMX100::getClassMXTransmitList
CDAQMXTransmitList::del
CDAQMXTransmitList::initialize
```

displaySegmentMX100

構文

```
int displaySegmentMX100(DAQMX100 daqmx100, int dispPattern0,
int dispPattern1, int dispType, int dispTime);
```

宣言

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);
```

引数

daqmx100	機器記述子を指定します。
dispPattern0	セグメント番号0の表示パターンを指定します。
dispPattern1	セグメント番号1の表示パターンを指定します。
dispType	表示形式を指定します。
dispTime	表示時間を指定します。

説明

7セグメントLEDの表示を設定します。

・ 変更前の表示パターンを返却しません。

戻り値

エラー番号を返します。

エラー：

Not descriptor　機器記述子がありません。

参照

`CDAQMX100::displaySegmentMX100`

formatCFMX100

構文

```
int formatCFMX100(DAQMX100 daqmx100);
```

宣言

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);
```

引数

daqmx100　　機器記述子を指定します。

説明

CF(Compact Flash)をフォーマットします。

戻り値

エラー番号を返します。

エラー：

Not descriptor　機器記述子がありません。

参照

```
CDAQMX100::formatCF
```

getItemAllMX100

構文

```
int getItemAllMX100(DAQMX100 daqmx100);
```

宣言

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);
```

引数

daqmx100　　機器記述子を指定します。

説明

設定データを一括で取得します。

戻り値

エラー番号を返します。

エラー：

Not descriptor　機器記述子がありません。

参照

```
CDAQMX100::getItemAll
```

initBalanceMX100

構文

```
int initBalanceMX100(DAQMX100 daqmx100);
```

宣言

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);
```

引数

daqmx100　　機器記述子を指定します。

説明

初期バランスを実行します。

戻り値

エラー番号を返します。

エラー：

Not descriptor　機器記述子がありません。

参照

```
CDAQMX100::initBalance
```

initDataChMX100

構文

```
int initDataChMX100(DAQMX100 daqmx100, int chNo);
```

宣言

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);
```

引数

daqmx100	機器記述子を指定します。
chNo	チャネル番号を指定します。

説明

指定されたチャネル番号の保持している測定データを初期化します。

- 測定データの取得が，FIFOの先頭からになります。
- チャネル番号に，定数値の「全チャネル番号指定」を指定すると，全チャネルを処理します。

戻り値

エラー番号を返します。

エラー：

Not descriptor　機器記述子がありません。

参照

CDAQMX100::initDataCh

initDataFIFOMX100

構文

```
int initDataFIFOMX100(DAQMX100 daqmx100, int fifoNo);
```

宣言

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);
```

引数

daqmx100	機器記述子を指定します。
fifoNo	FIFO番号を指定します。

説明

指定されたFIFO番号の保持している測定データを初期化します。

・ 測定データの取得が，FIFOの先頭からになります。

・ FIFO番号に，定数値の「全FIFO番号指定」を指定すると，全FIFOを処理します。

戻り値

エラー番号を返します。

エラー：

Not descriptor　機器記述子がありません。

参照

`CDAQMX100::initDataFIFO`

initItemMX100

構文

```
int initItemMX100(DAQMX100 daqmx100);
```

宣言

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);
```

引数

daqmx100　　機器記述子を指定します。

説明

保持している設定データを初期化します。

・保持している領域に上書きします。整合性のチェックは行われません。

・各取得関数の結果が正しくなくなることがあります。

戻り値

エラー番号を返します。

エラー：

Not descriptor　機器記述子がありません。

参照

```
CDAQMX100::getClassMXItemConfig
CDAQMXItemConfig::initialize
```

initSetValueMX100

構文

```
int initSetValueMX100(DAQMX100 daqmx100);
```

宣言

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);
```

引数

daqmx100　　機器記述子を指定します。

説明

設定を初期化します。

戻り値

エラー番号を返します。

エラー：

Not descriptor　機器記述子がありません。

参照

```
CDAQMX100::initSetValue
```

measDataChMX100

構文

```
int measDataChMX100(DAQMX100 daqmx100, int chNo);
```

宣言

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);
```

引数

daqmx100	機器記述子を指定します。
chNo	チャネル番号を指定します。

説明

指定されたチャネル番号の測定データを取得します。

- チャネル番号に，定数値の「全チャネル番号指定」を指定すると，全チャネルを処理します。
- 計測点を一点だけ進めます。
- FIFO指定や瞬時値指定の取得と混在して使用すると，データ順序が変わります。
- まず，取得可能な分のデータを一度に取得して保持し，先頭のデータを現在状態のデータとします。次に，この関数が呼び出されるごとに保持しているデータの計測点をひとつ進めて現在状態のデータとします。保持しているデータの最後まで到達したら，再度，取得可能な分のデータ取得から繰り返します。
- 通信でデータ取得をした場合に，他の状態更新を行います。

戻り値

エラー番号を返します。

エラー：

Not descriptor　機器記述子がありません。

参照

```
CDAQMX100::measDataCh
```

measDataFIFOMX100

構文

```
int measDataFIFOMX100(DAQMX100 daqmx100, int fifoNo);
```

宣言

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);
```

引数

daqmx100	機器記述子を指定します。
fifoNo	FIFO番号を指定します。

説明

指定されたFIFO番号の測定データを取得します。

- FIFO番号に，定数値の「全FIFO番号指定」を指定すると，全FIFOを処理します。
- 計測点を一点だけ進めます。
- FIFO内のチャネルは，同じ計測点のデータになります。
- チャネル指定や瞬時値指定の取得と混在して使用すると，データ順序が変わります。
- まず，取得可能な分のデータを一度に取得して保持し，先頭のデータを現在状態のデータとします。次に，この関数が呼び出されるごとに保持しているデータの計測点をひとつ進めて現在状態のデータとします。保持しているデータの最後まで到達したら，再度，取得可能な分のデータ取得から繰り返します。
- 通信でデータ取得をした場合に，他の状態更新を行います。

戻り値

エラー番号を返します。

エラー：

Not descriptor　機器記述子がありません。

参照

```
CDAQMX100::measDataFIFO
```

measInstChMX100

構文

```
int measInstChMX100(DAQMX100 daqmx100, int chNo);
```

宣言

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);
```

引数

daqmx100	機器記述子を指定します。
chNo	チャネル番号を指定します。

説明

指定されたチャネル番号の瞬時値を取得します。

- チャネル番号に，定数値の「全チャネル番号指定」を指定すると，全チャネルを処理します。
- 他の状態更新を行います。

戻り値

エラー番号を返します。

エラー：

Not descriptor　機器記述子がありません。

参照

CDAQMX100::measInstCh

measInstFIFOMX100

構文

```
int measInstFIFOMX100(DAQMX100 daqmx100, int fifoNo);
```

宣言

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);
```

引数

daqmx100	機器記述子を指定します。
fifoNo	FIFO番号を指定します。

説明

指定されたFIFO番号の瞬時値を取得します。

・ FIFO番号に，定数値の「全FIFO番号指定」を指定すると，全FIFOを処理します。

・ 他の状態更新を行います。

戻り値

エラー番号を返します。

エラー：

Not descriptor　機器記述子がありません。

参照

```
CDAQMX100::measInstFIFO
```

measStartMX100

構文

```
int measStartMX100(DAQMX100 daqmx100);
```

宣言

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);
```

引数

daqmx100　　機器記述子を指定します。

説明

データ収集を開始します。

・ FIFOを開始します。

・ 既に開始している場合，FIFOは継続します。

戻り値

エラー番号を返します。

エラー：

Not descriptor　機器記述子がありません。

参照

`CDAQMX100::measStart`

measStopMX100

構文

```
int measStopMX100(DAQMX100 daqmx100);
```

宣言

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);
```

引数

daqmx100　　機器記述子を指定します。

説明

データ収集を停止します。

・FIFOを停止し，収集したデータを破棄します。

戻り値

エラー番号を返します。

エラー：

Not descriptor　機器記述子がありません。

参照

```
CDAQMX100::measStop
```

openMX100

構文

```
DAQMX100 openMX100(const char * strAddress, int * errorCode);
```

宣言

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);
```

引数

strAddress　　IPアドレスを文字列で指定します。
errorCode　　エラー番号の返却先を指定します。

説明

引数で指定されたアドレスの機器と通信接続をします。

- 機器記述子を作成し，戻り値として返却します。
- 返却先が指定されていれば，エラー番号を格納します。
- 保持するデータを初期化します。設定データ，チャネル情報データなど，機器の状態を取得して保持します。
- 指定する文字列は，原則ascii文字列です。
- 失敗した場合，Visual CではNULLを，Visual Basic，Visual Basic.NET，C#では0を返します。

戻り値

機器記述子を返します。
エラー：
Creating descriptor is failure　　機器記述子の作成に失敗しました。

参照

`CDAQMX100::open`

readItemMX100

構文

```
int readItemMX100(DAQMX100 daqmx100, int itemNo, char *
strItem, int lenItem, int * realLen);
```

宣言

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);
```

引数

daqmx100	機器記述子を指定します。
itemNo	設定項目番号を指定します。
strItem	文字列を格納する領域を指定します。
lenItem	文字列を格納する領域のバイト数を指定します。
realLen	実際の文字列の長さの返却先を指定します。

説明

指定された設定項目の内容を文字列で指定された領域に格納します。

- 領域に格納する文字列には，終端(NULL)も含まれます。
- 返却先が指定されていれば，実際の文字列の長さを返します。終端は含まれません。
- 格納される文字列は，原則ascii文字列です。

戻り値

エラー番号を返します。

エラー：

Not descriptor	機器記述子がありません。
Not Support	サポートしていない設定項目です。
Not Data	文字列の格納領域が不足しています。

参照

```
CDAQMX100::getClassMXItemConfig
CDAQMXItemConfig::readItem
```

reconstructMX100

構文

```
int reconstructMX100(DAQMX100 daqmx100);
```

宣言

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);
```

引数

daqmx100　　機器記述子を指定します。

説明

システムを再構築します。

戻り値

エラー番号を返します。

エラー：

Not descriptor　機器記述子がありません。

参照

```
CDAQMX100::reconstruct
```

sendConfigMX100

構文

```
int sendConfigMX100(DAQMX100 daqmx100);
```

宣言

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);
```

引数

daqmx100　　機器記述子を指定します。

説明

保持している設定データを送信します。

戻り値

エラー番号を返します。

エラー：

Not descriptor　機器記述子がありません。

参照

`CDAQMX100::sendConfig`

setAlarmMX100

構文

```
int setAlarmMX100(DAQMX100 daqmx100, int chNo, int levelNo,
int iAlarmType, int value);
```

宣言

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);
```

引数

daqmx100	機器記述子を指定します。
chNo	チャネル番号を指定します。
levelNo	アラームレベルを指定します。
iAlarmType	アラーム種類を指定します。
value	アラーム値を指定します。

説明

指定されたチャネル番号のアラームレベルにアラームを設定します。

- 設定されているレンジ種類に従って設定されます。
- ヒステリシスは，0になります。
- 指定するアラーム値は，小数点位置を除いた整数値を指定します。

戻り値

エラー番号を返します。

エラー：

Not descriptor　機器記述子がありません。

参照

`CDAQMX100::setAlarm`

setAlarmValueMX100

構文

```
int setAlarmValueMX100(DAQMX100 daqmx100, int chNo, int
levelNo, int iAlarmType, int valueON, int valueOFF);
```

宣言

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);
```

引数

daqmx100	機器記述子を指定します。
chNo	チャネル番号を指定します。
levelNo	アラームレベルを指定します。
iAlarmType	アラーム種類を指定します。
valueON	アラーム発生のしきい値(On値)を指定します。
valueOFF	アラーム停止のしきい値(Off値)を指定します。

説明

指定されたチャネル番号のアラームレベルにアラームを設定します。

- 設定されているレンジ種類に従って設定されます。
- ヒステリシスは，アラーム発生と停止のしきい値で指定します。
- 指定するアラーム値は，小数点位置を除いた整数値を指定します。

戻り値

エラー番号を返します。

エラー：

Not descriptor　機器記述子がありません。

参照

`CDAQMX100::setAlarm`

setBurnoutMX100

構文

```
int setBurnoutMX100(DAQMX100 daqmx100, int chNo, int
iBurnout);
```

宣言

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);
```

引数

daqmx100	機器記述子を指定します。
chNo	チャネル番号を指定します。
iBurnout	バーンアウト種類を指定します。

説明

指定されたチャネル番号にバーンアウト種類を設定します。

戻り値

エラー番号を返します。

エラー：

Not descriptor　機器記述子がありません。

参照

`CDAQMX100::setBurnout`

setCFWriteModeMX100

構文

```
int setCFWriteModeMX100(DAQMX100 daqmx100, int iCFWriteMode);
```

宣言

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);
```

引数

daqmx100　　機器記述子を指定します。
iCFWriteMode　CF書き込み種類を指定します。

説明

CF書き込み種類を設定します。

戻り値

エラー番号を返します。
エラー：
Not descriptor　機器記述子がありません。

参照

`CDAQMX100::setCFWriteMode`

setChatFilterMX100

構文

```
int setChatFilterMX100(DAQMX100 daqmx100, int chNo, int
bChatFilter);
```

宣言

Visual Basic

```
Public Declare Function setChatFilterMX100 Lib "DAQMX100"
(ByVal daqmx100 As Long, ByVal chNo As Long, ByVal bChatFilter
As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function setChatFilterMX100 Lib "DAQMX100"
(ByVal daqmx100 As Integer, ByVal chNo As Integer, ByVal
bChatFilter As Integer) As Integer
```

C#

```
[DllImport("DAQMX100.dll", CharSet=CharSet.Auto,
EntryPoint="setChatFilterMX100")]
public static extern int setChatFilterMX100(int daqmx100, int
chNo, int iFilter);
```

引数

daqmx100	機器記述子を指定します。
chNo	チャネル番号を指定します。
bChatFilter	チャタリングフィルタを有効無効値で指定します。

説明

指定されたチャネル番号にチャタリングフィルタを設定します。

戻り値

エラー番号を返します。

エラー：

Not descriptor　機器記述子がありません。

参照

`CDAQMX100::setChatFilter`

setChCommentMX100

構文

```
int setChCommentMX100(DAQMX100 daqmx100, int chNo, const char
* strComment);
```

宣言

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);
```

引数

daqmx100	機器記述子を指定します。
chNo	チャネル番号を指定します。
strTag	コメントを指定します。

説明

指定されたチャネル番号にコメントを設定します。

・ 指定する文字列は，原則ascii文字列です。

戻り値

エラー番号を返します。

エラー：

Not descriptor　機器記述子がありません。

参照

CDAQMX100::setChComment

setChDELTAMX100

構文

```
int setChDELTAMX100(DAQMX100 daqmx100, int chNo, int refChNo,
int iRange);
```

宣言

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);
```

引数

daqmx100	機器記述子を指定します。
chNo	チャネル番号を指定します。
refChNo	基準チャネルをチャネル番号で指定します。
iRange	自チャネルのレンジ種類を指定します。

説明

指定されたチャネル番号にチャネル間差演算を設定します。

- レンジ種類に参照レンジを指定すると，自チャネルの測定レンジを基準チャネルと同じレンジにします。
- スパン，スケール，アラームは既定値に設定されます。

戻り値

エラー番号を返します。

エラー：

Not descriptor　機器記述子がありません。

参照

CDAQMX100::setChDELTA

setChKindMX100

構文

```
int setChKindMX100(DAQMX100 daqmx100, int chNo, int iKind, int
refChNo);
```

宣言

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);
```

引数

daqmx100	機器記述子を指定します。
chNo	チャネル番号を指定します。
iKind	チャネル種類を指定します。
refChNo	参照チャネルをチャネル番号で指定します。

説明

指定されたチャネル番号にチャネル種類を設定します。

・ レンジ，スパン，スケール，アラームなど各設定項目は既定値に初期化されます。

・ 参照チャネルの指定は，チャネル種類が「チャネル間差」，「リモートRJC」，「AO」，「PWM」で有効です。

戻り値

エラー番号を返します。

エラー：

Not descriptor　機器記述子がありません。

参照

CDAQMX100::setChKind

setChoiceMX100

構文

```
int setChoiceMX100(DAQMX100 daqmx100, int outputNo, int
idleChoice, int errorChoice, int presetValue);
```

宣言

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);
```

引数

daqmx100	機器記述子を指定します。
outputNo	出力チャネル（AO/PWMデータ番号）を指定します。
idleChoice	アイドル時の選択値を指定します。
errorChoice	エラー時の選択値を指定します。
presetValue	選択値が「指定値」の場合の出力値を指定します。

説明

指定された出力チャネルにアイドル時とエラー時の出力を設定します。

・ ユーザ指定の出力値は，小数点位置を除いた整数値を指定します。

戻り値

エラー番号を返します。

エラー：

Not descriptor　機器記述子がありません。

参照

CDAQMX100::setChoice

setChRRJCMX100

構文

```
int setChRRJCMX100(DAQMX100 daqmx100, int chNo, int refChNo);
```

宣言

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);
```

引数

daqmx100	機器記述子を指定します。
chNo	チャネル番号を指定します。
refChNo	基準チャネルをチャネル番号で指定します。

説明

指定されたチャネル番号にリモートRJCを設定します。

・ スパン，アラームは既定値に設定されます。

戻り値

エラー番号を返します。

エラー：

Not descriptor　機器記述子がありません。

参照

CDAQMX100::setChRRJC

setChTagMX100

構文

```
int setChTagMX100(DAQMX100 daqmx100, int chNo, const char *
strTag);
```

宣言

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);
```

引数

daqmx100	機器記述子を指定します。
chNo	チャネル番号を指定します。
strTag	タグを指定します。

説明

指定されたチャネル番号にタグを設定します。

・ 指定する文字列は，原則ascii文字列です。

戻り値

エラー番号を返します。

エラー：

Not descriptor　機器記述子がありません。

参照

`CDAQMX100::setChTag`

setChUnitMX100

構文

```
int setChUnitMX100(DAQMX100 daqmx100, int chNo, const char *
strUnit);
```

宣言

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);
```

引数

daqmx100	機器記述子を指定します。
chNo	チャネル番号を指定します。
strUnit	単位名を指定します。

説明

指定されたチャネル番号に単位名を設定します。

・ 指定する文字列は，原則ascii文字列です。

戻り値

エラー番号を返します。

エラー：

Not descriptor　機器記述子がありません。

参照

CDAQMX100::setChUnit

setDateTimeNowMX100

構文

```
int setDateTimeNowMX100(DAQMX100 daqmx100);
```

宣言

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);
```

引数

daqmx100　　機器記述子を指定します。

説明

PCの現在の日付時刻を設定します。

・ ミリ秒は無視されます。

・ 本関数は，応答に1秒以上の時間がかかることがあります。

戻り値

エラー番号を返します。

エラー：

Not descriptor　機器記述子がありません。

参照

`CDAQMX100::setDateTime`

setDeenergizeMX100

構文

```
int setDeenergizeMX100(DAQMX100 daqmx100, int doNo, int
bDeenergize);
```

宣言

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);
```

引数

daqmx100	機器記述子を指定します。
doNo	DOデータ番号を指定します。
bDeenergize	非励磁を有効無効値で指定します。

説明

指定されたDOデータ番号のDOチャネルに非励磁を設定します。

戻り値

エラー番号を返します。

エラー：

Not descriptor	機器記述子がありません。

参照

`CDAQMX100::setDeenergize`

setDoubleAlarmMX100

構文

```
int setDoubleAlarmMX100(DAQMX100 daqmx100, int chNo, int
levelNo, int iAlarmType, double value);
```

宣言

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);
```

引数

daqmx100	機器記述子を指定します。
chNo	チャネル番号を指定します。
levelNo	アラームレベルを指定します。
iAlarmType	アラーム種類を指定します。
value	アラーム値を指定します。

説明

指定されたチャネル番号のアラームレベルにアラームを設定します。

- 指定するアラーム値の違いを除けば，setAlarmMX100関数と同じです。
- 指定するアラーム値は，小数点位置を含む浮動小数値を指定します。

戻り値

エラー番号を返します。

エラー：

Not descriptor　機器記述子がありません。

参照

CDAQMX100::setAlarm

setDoubleAlarmValueMX100

構文

```
int setDoubleAlarmValueMX100(DAQMX100 daqmx100, int chNo, int
levelNo, int iAlarmType, double valueON, double valueOFF)
```

宣言

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);
```

引数

daqmx100	機器記述子を指定します。
chNo	チャネル番号を指定します。
levelNo	アラームレベルを指定します。
iAlarmType	アラーム種類を指定します。
valueON	アラーム発生のしきい値(On値)を指定します。
valueOFF	アラーム停止のしきい値(Off値)を指定します。

説明

指定されたチャネル番号のアラームレベルにアラームを設定します。

・ 指定するしきい値の違いを除けば，setAlarmValueMX100関数と同じです。

・ 指定するしきい値は，小数点位置を含む浮動小数値を指定します。

戻り値

エラー番号を返します。

エラー：

Not descriptor　機器記述子がありません。

参照

```
CDAQMX100::setAlarm
```

setDoubleChoiceMX100

構文

```
int setDoubleChoiceMX100(DAQMX100 daqmx100, int outputNo, int
idleChoice, int errorChoice, double presetValue);
```

宣言

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);
```

引数

daqmx100	機器記述子を指定します。
outputNo	出力チャネル（AO/PWMデータ番号）を指定します。
idleChoice	アイドル時の選択値を指定します。
errorChoice	エラー時の選択値を指定します。
presetValue	選択値が「指定値」の場合の出力値を指定します。

説明

指定された出力チャネルにアイドル時とエラー時の出力を設定します。

- ユーザ指定の出力値の違いを除けば，setChoiceMX100関数と同じです。
- ユーザ指定の出力値は，小数点位置を含む浮動小数値を指定します。

戻り値

エラー番号を返します。

エラー：

Not descriptor　機器記述子がありません。

参照

`CDAQMX100::setChoice`

setDoubleHisterisysMX100

構文

```
int setDoubleHisterisysMX100(DAQMX100 daqmx100, int chNo, int
levelNo, double histerisys);
```

宣言

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);
```

引数

daqmx100	機器記述子を指定します。
chNo	チャネル番号を指定します。
levelNo	アラームレベルを指定します。
histerisys	ヒステリシスを指定します。

説明

指定されたチャネル番号のアラームレベルにヒステリシスを設定します。

- 指定するヒステリシスの違いを除けば，setHisterisysMX100関数と同じです。
- 指定するヒステリシスは，小数点位置を含む浮動小数値を指定します。

戻り値

エラー番号を返します。

エラー：

Not descriptor　機器記述子がありません。

参照

CDAQMX100::setHisterisys

setDoubleScaleMX100

構文

```
int setDoubleScaleMX100(DAQMX100 daqmx100, int chNo, double
scaleMin, double scaleMax, int scalePoint);
```

宣言

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);
```

引数

daqmx100	機器記述子を指定します。
chNo	チャネル番号を指定します。
scaleMin	スケール最小値を指定します。
scaleMax	スケール最大値を指定します。
scalePoint	スケール時の小数点位置を指定します。

説明

指定されたチャネル番号にスケールを設定します。

・ 指定するスケールの値の違いを除けば，setScaleMX100関数と同じです。

・ 指定するスケールの値は，小数点位置を含む浮動小数値を指定します。

戻り値

エラー番号を返します。

エラー：

Not descriptor　機器記述子がありません。

参照

CDAQMX100::setScale

setDoubleSpanMX100

構文

```
int setDoubleSpanMX100(DAQMX100 daqmx100, int chNo, double
spanMin, double spanMax);
```

宣言

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);
```

引数

daqmx100	機器記述子を指定します。
chNo	チャネル番号を指定します。
spanMin	スパン最小値を指定します。
spanMax	スパン最大値を指定します。

説明

指定されたチャネル番号にスパンを設定します。

・ 指定するスパンの値の違いを除けば，setSpanMX100関数と同じです。

・ 指定するスパンの値は，小数点位置を含む浮動小数値を指定します。

戻り値

エラー番号を返します。

エラー：

Not descriptor　機器記述子がありません。

参照

CDAQMX100::setSpan

setFilterMX100

構文

```
int setFilterMX100(DAQMX100 daqmx100, int chNo, int iFilter);
```

宣言

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);
```

引数

daqmx100	機器記述子を指定します。
chNo	チャネル番号を指定します。
iFilter	フィルタ係数を指定します。

説明

指定されたチャネル番号にフィルタ係数を設定します。

戻り値

エラー番号を返します。

エラー：

Not descriptor　機器記述子がありません。

参照

`CDAQMX100::setFilter`

setHisterisysMX100

構文

```
int setHisterisysMX100(DAQMX100 daqmx100, int chNo, int
levelNo, int histerisys);
```

宣言

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);
```

引数

daqmx100	機器記述子を指定します。
chNo	チャネル番号を指定します。
levelNo	アラームレベルを指定します。
histerisys	ヒステリシスを指定します。

説明

指定されたチャネル番号のアラームレベルにヒステリシスを設定します。

・ 指定するヒステリシスは，小数点位置を除いた整数値を指定します。

戻り値

エラー番号を返します。

エラー：

Not descriptor　機器記述子がありません。

参照

CDAQMX100::setHisterisys

setHoldMX100

構文

```
int setHoldMX100(DAQMX100 daqmx100, int doNo, int bHold);
```

宣言

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);
```

引数

daqmx100	機器記述子を指定します。
doNo	DOデータ番号を指定します。
bHold	保持を有効無効値で指定します。

説明

指定されたDOデータ番号のDOチャネルに保持を設定します。

戻り値

エラー番号を返します。

エラー：

Not descriptor　機器記述子がありません。

参照

`CDAQMX100::setHold`

setIntegralMX100

構文

```
int setIntegralMX100(DAQMX100 daqmx100, int moduleNo, int
iIntegral);
```

宣言

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);
```

引数

daqmx100	機器記述子を指定します。
moduleNo	モジュール番号を指定します。
iIntegral	A/D積分時間種類を指定します。

説明

指定されたモジュール番号のモジュールにA/D積分時間種類を設定します。

戻り値

エラー番号を返します。

エラー：

Not descriptor	機器記述子がありません。

参照

`CDAQMX100::setIntegral`

setIntervalMX100

構文

```
int setIntervalMX100(DAQMX100 daqmx100, int moduleNo, int
iInterval);
```

宣言

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);
```

引数

daqmx100	機器記述子を指定します。
moduleNo	モジュール番号を指定します。
iInterval	周期種類を指定します。

説明

指定されたモジュール番号のモジュールに周期種類を設定します。

戻り値

エラー番号を返します。

エラー：

Not descriptor　機器記述子がありません。

参照

`CDAQMX100::setInterval`

setItemAllMX100

構文

```
int setItemAllMX100(DAQMX100 daqmx100);
```

宣言

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);
```

引数

daqmx100　　機器記述子を指定します。

説明

設定データを一括送信します。

戻り値

エラー番号を返します。

エラー：

Not descriptor　機器記述子がありません。

参照

```
CDAQMX100::setItemAll
```

setOutputTypeMX100

構文

```
int setOutputTypeMX100(DAQMX100 daqmx100, int outputNo, int
iOutput);
```

宣言

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);
```

引数

daqmx100	機器記述子を指定します。
outputNo	出力チャネル（AO/PWMデータ番号）を指定します。
iOutput	出力種類を指定します。

説明

指定された出力チャネルに出力種類を設定します。

- 指定されたチャネルの各設定項目は既定値に初期化されます。
- データ操作で作成されたデータは変更されません。

戻り値

エラー番号を返します。

エラー：

Not descriptor　機器記述子がありません。

参照

`CDAQMX100::setOutputType`

setPulseTimeMX100

構文

```
int setPulseTimeMX100(DAQMX100 daqmx100, int outputNo, int
pulseTime);
```

宣言

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);
```

引数

daqmx100	機器記述子を指定します。
outputNo	出力チャネル（PWMデータ番号）を指定します。
pulseTime	パルス周期倍率を指定します。

説明

指定された出力チャネルにパルス周期倍率を設定します。

戻り値

エラー番号を返します。

エラー：

Not descriptor　機器記述子がありません。

参照

`CDAQMX100::setPulseTime`

setRangeMX100

構文

```
int setRangeMX100(DAQMX100 daqmx100, int chNo, int iRange);
```

宣言

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);
```

引数

daqmx100	機器記述子を指定します。
chNo	チャネル番号を指定します。
iRange	レンジ種類を指定します。

説明

指定されたチャネル番号にレンジを設定します。

- 指定されたチャネルの各設定項目は既定値に初期化されます。
- データ操作で作成されたデータは変更されません。

戻り値

エラー番号を返します。

エラー：

Not descriptor　機器記述子がありません。

参照

`CDAQMX100::setRange`

setRefAlarmMX100

構文

```
int setRefAlarmMX100(DAQMX100 daqmx100, int doNo, int refChNo,
int levelNo, int bValid);
```

宣言

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);
```

引数

daqmx100	機器記述子を指定します。
doNo	DOデータ番号を指定します。
refChNo	参照チャネルをチャネル番号で指定します。
levelNo	アラームレベルを指定します。
bValid	有効無効値を指定します。

説明

指定されたDOデータ番号のDOチャネルに参照アラームを設定します。

・ 参照アラームは，チャネル番号とアラームレベルで指定します。

戻り値

エラー番号を返します。

エラー：

Not descriptor　機器記述子がありません。

参照

CDAQMX100::setRefAlarm

setRJCTypeMX100

構文

```
int setRJCTypeMX100(DAQMX100 daqmx100, int chNo, int iRJCType,
int volt);
```

宣言

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);
```

引数

daqmx100	機器記述子を指定します。
chNo	チャネル番号を指定します。
iRJCType	RJC種類を指定します。
volt	RJC電圧値を指定します。

説明

指定されたチャネル番号にRJC関連を設定します。

戻り値

エラー番号を返します。

エラー：

Not descriptor　機器記述子がありません。

参照

CDAQMX100::setRJCType

setScaleMX100

構文

```
int setScaleMX100(DAQMX100 daqmx100, int chNo, int scaleMin,
int scaleMax, int scalePoint);
```

宣言

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 Intger, ByVal chNo As Intger, ByVal
scaleMin As Intger, ByVal scaleMax As Intger, ByVal scalePoint
As Intger) As Intger
```

C#

```
[DllImport("DAQMX100.dll", CharSet=CharSet.Auto,
EntryPoint="setScaleMX100")]
public static extern int setScaleMX100(int daqmx100, int chNo,
int scaleMin, int scaleMax, int scalePoint);
```

引数

daqmx100	機器記述子を指定します。
chNo	チャネル番号を指定します。
scaleMin	スケール最小値を指定します。
scaleMax	スケール最大値を指定します。
scalePoint	スケール時の小数点位置を指定します。

説明

指定されたチャネル番号にスケールを設定します。

- 設定されているレンジ種類に従って設定されます。
- スケールの値が範囲外の場合，可能な値に丸め込みます。
- 最小値と最大値が等しい場合，スケールは「スケールなし」になります。
- 指定するスケールの値は，小数点位置を除いた整数値を指定します。

戻り値

エラー番号を返します。

エラー：

Not descriptor　機器記述子がありません。

参照

CDAQMX100::setScale

setSpanMX100

構文

```
int setSpanMX100(DAQMX100 daqmx100, int chNo, int spanMin, int
spanMax);
```

宣言

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);
```

引数

daqmx100	機器記述子を指定します。
chNo	チャネル番号を指定します。
spanMin	スパン最小値を指定します。
spanMax	スパン最大値を指定します。

説明

指定されたチャネル番号にスパンを設定します。

- 設定されているレンジ種類に従って設定されます。
- スパンの値が範囲外の場合，可能な値に丸め込みます。
- 最小値と最大値が等しい場合，既定値が設定されます。
- 指定するスパンの値は，小数点位置を除いた整数値を指定します。

戻り値

エラー番号を返します。

エラー：

Not descriptor　機器記述子がありません。

参照

CDAQMX100::setSpan

setUnitNoMX100

構文

```
int setUnitNoMX100(DAQMX100 daqmx100, int unitNo);
```

宣言

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);
```

引数

daqmx100	機器記述子を指定します。
unitNo	ユニット番号を指定します。

説明

ユニット番号を設定します。

戻り値

エラー番号を返します。

エラー：

Not descriptor	機器記述子がありません。

参照

```
CDAQMX100::setUnitNo
```

setUnitTempMX100

構文

```
int setUnitTempMX100(DAQMX100 daqmx100, int iTempUnit);
```

宣言

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);
```

引数

daqmx100	機器記述子を指定します。
iTempUnit	温度単位種類を指定します。

説明

温度単位種類を設定します。

戻り値

エラー番号を返します。

エラー：

Not descriptor　機器記述子がありません。

参照

```
CDAQMX100::setUnitTemp
```

switchBackupMX100

構文

```
int switchBackupMX100(DAQMX100 daqmx100, int bBackup);
```

宣言

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);
```

引数

daqmx100	機器記述子を指定します。
bBackup	有効無効値を指定します。

説明

バックアップを設定します。

戻り値

エラー番号を返します。

エラー：

Not descriptor	機器記述子がありません。

参照

```
CDAQMX100::switchBackup
```

switchDOMX100

構文

```
int switchDOMX100(DAQMX100 daqmx100, int idDO, int bONOFF);
```

宣言

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);
```

引数

daqmx100	機器記述子を指定します。
idDO	DOデータ識別子を指定します。
bONOFF	ON/OFFを有効無効値で指定します。

説明

指定されたDOデータ識別子のDOデータを送信します。

・ DOデータの有効チャネルを指定されたON/OFF値に変更して送信します。

戻り値

エラー番号を返します。

エラー：

Not descriptor　機器記述子がありません。

参照

`CDAQMX100::switchDO`

switchTransmitMX100

構文

```
int switchTransmitMX100(DAQMX100 daqmx100, int idTrans, int
iTransmit);
```

宣言

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);
```

引数

daqmx100	機器記述子を指定します。
idTrans	伝送出力データ識別子を指定します。
iTransmit	伝送状態を指定します。

説明

指定された伝送出力データ識別子の伝送出力データを送信します。

・ 伝送出力データの全チャネルを指定された伝送状態に変更して送信します。

戻り値

エラー番号を返します。

エラー：

Not descriptor	機器記述子がありません。

参照

CDAQMX100::switchTransmit

updateAOPWMDataMX100

構文

```
int updateAOPWMDataMX100(DAQMX100 daqmx100);
```

宣言

Visual Basic

```
Public Declare Function updateAOPWMDataMX100 Lib "DAQMX100"
(ByVal daqmx100 As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function updateAOPWMDataMX100 Lib
"DAQMX100" (ByVal daqmx100 As Integer) As Integer
```

C#

```
[DllImport("DAQMX100.dll", CharSet=CharSet.Auto,
EntryPoint="updateAOPWMDataMX100")]
public static extern int updateAOPWMDataMX100(int daqmx100);
```

引数

daqmx100　　機器記述子を指定します。

説明

保持しているAO/PWMデータと伝送出力データを更新します。

戻り値

エラー番号を返します。

エラー：

Not descriptor　機器記述子がありません。

参照

CDAQMX100::updateAOPWMData

updateBalanceMX100

構文

```
int updateBalanceMX100(DAQMX100 daqmx100);
```

宣言

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);
```

引数

daqmx100　　機器記述子を指定します。

説明

保持している初期バランスデータを更新します。

戻り値

エラー番号を返します。

エラー：

Not descriptor　機器記述子がありません。

参照

CDAQMX100::updateBalance

updateConfigMX100

構文

```
int updateConfigMX100(DAQMX100 daqmx100);
```

宣言

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);
```

引数

daqmx100　　機器記述子を指定します。

説明

保持している設定データを更新します。

戻り値

エラー番号を返します。

エラー：

Not descriptor　機器記述子がありません。

参照

`CDAQMX100::updateConfig`

updateDODataMX100

構文

```
int updateDODataMX100(DAQMX100 daqmx100);
```

宣言

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);
```

引数

daqmx100　　機器記述子を指定します。

説明

保持しているDOデータを更新します。

戻り値

エラー番号を返します。

エラー：

Not descriptor　機器記述子がありません。

参照

```
CDAQMX100::updateDOData
```

updateInfoChMX100

構文

```
int updateInfoChMX100(DAQMX100 daqmx100, int chNo);
```

宣言

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);
```

引数

daqmx100	機器記述子を指定します。
chNo	チャネル番号を指定します。

説明

指定されたチャネル番号のチャネル情報データを更新します。

・ チャネル番号に，定数値の「全チャネル番号指定」を指定すると，全チャネルを処理します。

戻り値

エラー番号を返します。

エラー：

Not descriptor　機器記述子がありません。

参照

`CDAQMX100::updateInfoCh`

updateOutputMX100

構文

```
int updateOutputMX100(DAQMX100 daqmx100);
```

宣言

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);
```

引数

daqmx100　　機器記述子を指定します。

説明

保持している出力チャネルデータを更新します。

戻り値

エラー番号を返します。

エラー：

Not descriptor　機器記述子がありません。

参照

CDAQMX100::updateOutput

updateStatusMX100

構文

```
int updateStatusMX100(DAQMX100 daqmx100);
```

宣言

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);
```

引数

daqmx100　　機器記述子を指定します。

説明

保持しているステータスデータを更新します。

戻り値

エラー番号を返します。

エラー：

Not descriptor　機器記述子がありません。

参照

CDAQMX100::updateStatus

updateSystemMX100

構文

```
int updateSystemMX100(DAQMX100 daqmx100);
```

宣言

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);
```

引数

daqmx100　　機器記述子を指定します。

説明

保持しているシステム構成データを更新します。

戻り値

エラー番号を返します。

エラー：

Not descriptor　機器記述子がありません。

参照

CDAQMX100::updateSystem

writeItemMX100

構文

```
int writeItemMX100(DAQMX100 daqmx100, int itemNo, const char *
strItem);
```

宣言

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);
```

引数

daqmx100	機器記述子を指定します。
itemNo	設定項目番号を指定します。
strItem	項目内容を文字列で指定します。

説明

指定された設定項目に指定された文字列の内容を書き込みます。

- 保持している領域に上書きします。整合性のチェックは行われません。
- 各取得関数の結果が正しくなくなることがあります。
- 指定する文字列は，原則ascii文字列です。

戻り値

エラー番号を返します。

エラー：

Not descriptor　機器記述子がありません。

参照

```
CDAQMX100::getClassMXItemConfig
CDAQMXItemConfig::writeItem
```

17.2 関数の詳細ーMX100(Visual C/Visual Basic/Visual Basic.NET/C#)ー取得関数

ここでは，Visual C，Visual Basic，Visual Basic.NET，およびC#で使用するMX100用関数について説明しています。関数は，関数名のアルファベット順で並んでいます。

定数，型については第18章をご覧ください。
MX100の用語については付録1をご覧ください。

ほとんどの関数は戻り値として，エラー番号を返します。正常終了の場合は，エラー番号「0」を返します。

値が存在しない場合とは，引数の指定が間違っていたり，範囲外だったり，領域が存在しない，取得に失敗した場合です。

C#の場合，宣言をまとめたクラス(DAQMX100)のメンバになります。

addressPartMX100

構文

```
int addressPartMX100(unsigned int address, int index);
```

宣言

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);
```

引数

address	IPアドレスを指定します。
index	パート位置を指定します。

説明

指定されたIPアドレスをパート位置で分割したバイト値を取得します。

- 指定されたパート位置のバイト値を返します。
- パート位置は，バイト単位のインデックス値(0から)で指定します。範囲は，0から3です。
- 存在しない場合，0を返します。

戻り値

バイト値を返します。

参照

`CDAQMXNetInfo::getPart`

alarmDoubleHisterisysMX100

構文

```
double alarmDoubleHisterisysMX100(DAQMX100 daqmx100, int chNo,
int levelNo);
```

宣言

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);
```

引数

daqmx100	機器記述子を指定します。
chNo	チャネル番号を指定します。
levelNo	アラームレベルを指定します。

説明

保持している現在のチャネル設定データから，指定されたチャネル番号，アラームレベルのヒステリシスを取得します。

- 戻り値は，小数点位置を含む浮動小数値です。
- 存在しない場合，0.0を返します。

戻り値

ヒステリシスを返します。

参照

```
CDAQMX100::getClassMXItemConfig
CDAQMXItemConfig::getDoubleHisterisys
```

alarmDoubleValueOFFMX100

構文

```
double alarmDoubleValueOFFMX100(DAQMX100 daqmx100, int chNo,
int levelNo);
```

宣言

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);
```

引数

daqmx100	機器記述子を指定します。
chNo	チャネル番号を指定します。
levelNo	アラームレベルを指定します。

説明

保持している現在のチャネル設定データから，指定されたチャネル番号，アラームレベルのアラーム値(OFF値)を取得します。

- 戻り値は，小数点位置を含む浮動小数値です。
- 存在しない場合，0.0を返します。

戻り値

アラーム値(OFF値)を返します。

参照

```
CDAQMX100::getClassMXItemConfig
CDAQMXItemConfig::getDoubleAlarmOFF
```

alarmDoubleValueONMX100

構文

```
double alarmDoubleValueONMX100(DAQMX100DAQMX100 daqmx100, int
chNo, int levelNo);
```

宣言

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);
```

引数

daqmx100	機器記述子を指定します。
chNo	チャネル番号を指定します。
levelNo	アラームレベルを指定します。

説明

保持している現在のチャネル設定データから，指定されたチャネル番号，アラームレベルのアラーム値(ON値)を取得します。

- 戻り値は，小数点位置を含む浮動小数値です。
- 存在しない場合，0.0を返します。

戻り値

アラーム値(ON値)を返します。

参照

```
CDAQMX100::getClassMXItemConfig
CDAQMXItemConfig::getDoubleAlarmON
```

alarmHisterisysMX100

構文

```
int alarmHisterisysMX100(DAQMX100 daqmx100, int chNo, int
levelNo);
```

宣言

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);
```

引数

daqmx100	機器記述子を指定します。
chNo	チャネル番号を指定します。
levelNo	アラームレベルを指定します。

説明

保持している現在のチャネル設定データから，指定されたチャネル番号，アラームレベルのヒステリシスを取得します。

- 戻り値は，小数点位置を除く整数値です。
- 存在しない場合，0を返します。

戻り値

ヒステリシスを返します。

参照

```
CDAQMX100::getClassMXItemConfig
CDAQMXItemConfig::getHisterisys
```

alarmMaxLengthMX100

構文

```
int alarmMaxLengthMX100(void);
```

宣言

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();
```

説明

アラーム種類の文字列の最大長を取得します。

・ 戻り値に終端は含まれません。

戻り値

文字列の長さを返します。

参照

```
CDAQMXDataInfo::getMaxLenAlarmName
```

alarmTypeMX100

構文

```
int alarmTypeMX100(DAQMX100 daqmx100, int chNo, int levelNo);
```

宣言

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);
```

引数

daqmx100	機器記述子を指定します。
chNo	チャネル番号を指定します。
levelNo	アラームレベルを指定します。

説明

保持している現在のチャネル設定データから，指定されたチャネル番号，アラームレベルのアラーム種類を取得します。

・ 存在しない場合，「アラームなし」を返します。

戻り値

アラーム種類を返します。

参照

```
CDAQMX100::getClassMXItemConfig
CDAQMXChConfig::getAlarmType
CDAQMXItemConfig::getClassMXChConfig
```

alarmValueOFFMX100

構文

```
int alarmValueOFFMX100(DAQMX100 daqmx100, int chNo, int
levelNo);
```

宣言

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);
```

引数

daqmx100	機器記述子を指定します。
chNo	チャネル番号を指定します。
levelNo	アラームレベルを指定します。

説明

保持している現在のチャネル設定データから，指定されたチャネル番号，アラームレベルのアラーム値(OFF値)を取得します。

- 戻り値は，小数点位置を除く整数値です。
- 存在しない場合，0を返します。

戻り値

アラーム値(OFF値)を返します。

参照

```
CDAQMX100::getClassMXItemConfig
CDAQMXChConfig::getAlarmValueOFF
CDAQMXItemConfig::getClassMXChConfig
```

alarmValueONMX100

構文

```
int alarmValueONMX100(DAQMX100 daqmx100, int chNo, int
levelNo);
```

宣言

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);
```

引数

daqmx100	機器記述子を指定します。
chNo	チャネル番号を指定します。
levelNo	アラームレベルを指定します。

説明

保持している現在のチャネル設定データから，指定されたチャネル番号，アラームレベルのアラーム値(ON値)を取得します。

- 戻り値は，小数点位置を除く整数値です。
- 存在しない場合，0を返します。

戻り値

アラーム値(ON値)を返します。

参照

```
CDAQMX100::getClassMXItemConfig
CDAQMXChConfig::getAlarmValueON
CDAQMXItemConfig::getClassMXChConfig
```

channelBalanceValidMX100

構文

```
int channelBalanceValidMX100(DAQMX100 daqmx100, int
balanceNo);
```

宣言

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);
```

引数

daqmx100	機器記述子を指定します。
balanceNo	初期バランスデータ番号を指定します。

説明

保持している現在のチャネル設定データから，指定された初期バランスデータ番号の有効/無効を取得します。

・ 存在しない場合，「無効値」を返します。

戻り値

有効無効値を返します。

参照

```
CDAQMX100::getClassMXItemConfig
CDAQMXBalanceData::getBalanceValid
CDAQMXItemConfig::getClassMXBalanceData
```

channelBalanceValueMX100

構文

```
int channelBalanceValueMX100(DAQMX100 daqmx100, int
balanceNo);
```

宣言

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);
```

引数

daqmx100	機器記述子を指定します。
balanceNo	初期バランスデータ番号を指定します。

説明

保持している現在のチャネル設定データから，指定された初期バランスデータ番号の初期バランス値を取得します。

・ 存在しない場合，0を返します。

戻り値

初期バランス値を返します。

参照

```
CDAQMX100::getClassMXItemConfig
CDAQMXBalanceData::getBalanceValue
CDAQMXItemConfig::getClassMXBalanceData
```

channelBurnoutMX100

構文

```
int channelBurnoutMX100(DAQMX100 daqmx100, int chNo);
```

宣言

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);
```

引数

daqmx100	機器記述子を指定します。
chNo	チャネル番号を指定します。

説明

保持している現在のチャネル設定データから，指定されたチャネル番号のバーンアウト種類を取得します。

・ 存在しない場合，「検出なし」を返します。

戻り値

バーンアウト種類を返します。

参照

```
CDAQMX100::getClassMXItemConfig
CDAQMXChConfig::getBurnout
CDAQMXItemConfig::getClassMXChConfig
```

channelChatFilterMX100

構文

```
int channelChatFilterMX100(DAQMX100 daqmx100, int chNo);
```

宣言

Visual Basic

```
Public Declare Function channelChatFilterMX100 Lib "DAQMX100"
(ByVal daqmx100 As Long, ByVal chNo As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function channelChatFilterMX100 Lib
"DAQMX100" (ByVal daqmx100 As Integer, ByVal chNo As Integer)
As Integer
```

C#

```
[DllImport("DAQMX100.dll", CharSet=CharSet.Auto,
EntryPoint="channelChatFilterMX100")]
public static extern int channelChatFilterMX100(int daqmx100,
int chNo);
```

引数

daqmx100	機器記述子を指定します。
chNo	チャネル番号を指定します。

説明

保持している現在のチャネル設定データから，指定されたチャネル番号のチャタリングフィルタの値を取得します。

戻り値

有効無効値を返します。
存在しない場合，「無効値」を返します。

参照

```
CDAQMX100::getClassMXItemConfig
CDAQMXChConfig::isChatFilter
CDAQMXItemConfig::getClassMXChConfig
```

channelDeenergizeMX100

構文

```
int channelDeenergizeMX100(DAQMX100 daqmx100, int doNo);
```

宣言

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);
```

引数

daqmx100	機器記述子を指定します。
doNo	DOデータ番号を指定します。

説明

保持している現在のチャネル設定データから，指定されたDOデータ番号の非励磁を有効無効値で取得します。

・ 存在しない場合，「無効値」を返します。

戻り値

有効無効値を返します。

参照

```
CDAQMX100::getClassMXItemConfig
CDAQMXChConfig::isDeenergize
CDAQMXItemConfig::getClassMXChConfig
```

channelDisplayMaxMX100

構文

```
double channelDisplayMaxMX100(DAQMX100 daqmx100, int chNo);
```

宣言

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);
```

引数

daqmx100	機器記述子を指定します。
chNo	チャネル番号を指定します。

説明

保持している現在のチャネル情報データから，指定されたチャネル番号の表示最大値を取得します。

・ 存在しない場合，0.0を返します。

戻り値

表示最大値を返します。

参照

```
CDAQMX100::getClassMXDataBuffer
CDAQMXChInfo::getDisplayMax
CDAQMXDataBuffer::getClassMXChInfo
```

channelDisplayMinMX100

構文

```
double channelDisplayMinMX100(DAQMX100 daqmx100, int chNo);
```

宣言

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);
```

引数

daqmx100　　機器記述子を指定します。

chNo　　チャネル番号を指定します。

説明

保持している現在のチャネル情報データから，指定されたチャネル番号の表示最小値を取得します。

・ 存在しない場合，0.0を返します。

戻り値

表示最小値を返します。

参照

```
CDAQMX100::getClassMXDataBuffer
CDAQMXChInfo::getDisplayMin
CDAQMXDataBuffer::getClassMXChInfo
```

channelDoublePresetValueMX100

構文

```
double channelDoublePresetValueMX100(DAQMX100 daqmx100, int
outputNo);
```

宣言

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);
```

引数

daqmx100	機器記述子を指定します。
outputNo	出力チャネルデータ番号を指定します。

説明

保持している現在のチャネル設定データから，指定された出力チャネルデータ番号のユーザ指定の出力値を取得します。

- 戻り値は，小数点位置を含む浮動小数値です。
- 存在しない場合，0.0を返します。

戻り値

ユーザ指定の出力値を返します。

参照

```
CDAQMX100::getClassMXItemConfig
CDAQMXItemConfig::getDoublePresetValue
```

channelDoubleScaleMaxMX100

構文

```
double channelDoubleScaleMaxMX100(DAQMX100 daqmx100, int
chNo);
```

宣言

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);
```

引数

daqmx100	機器記述子を指定します。
chNo	チャネル番号を指定します。

説明

保持している現在のチャネル設定データから，指定されたチャネル番号のスケール最大値を取得します。

- 戻り値は，小数点位置を含む浮動小数値です。
- 存在しない場合，0.0を返します。

戻り値

スケール最大値を返します。

参照

```
CDAQMX100::getClassMXItemConfig
CDAQMXItemConfig::getDoubleScaleMax
```

channelDoubleScaleMinMX100

構文

```
double channelDoubleScaleMinMX100(DAQMX100 daqmx100, int
chNo);
```

宣言

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);
```

引数

daqmx100	機器記述子を指定します。
chNo	チャネル番号を指定します。

説明

保持している現在のチャネル設定データから，指定されたチャネル番号のスケール最小値を取得します。

- 戻り値は，小数点位置を含む浮動小数値です。
- 存在しない場合，0.0を返します。

戻り値

スケール最小値を返します。

参照

```
CDAQMX100::getClassMXItemConfig
CDAQMXItemConfig::getDoubleScaleMin
```

channelDoubleSpanMaxMX100

構文

```
double channelDoubleSpanMaxMX100(DAQMX100 daqmx100, int chNo);
```

宣言

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);
```

引数

daqmx100	機器記述子を指定します。
chNo	チャネル番号を指定します。

説明

保持している現在のチャネル設定データから，指定されたチャネル番号のスパン最大値を取得します。

- 戻り値は，小数点位置を含む浮動小数値です。
- 存在しない場合，0.0を返します。

戻り値

スパン最大値を返します。

参照

```
CDAQMX100::getClassMXItemConfig
CDAQMXItemConfig::getDoubleSpanMax
```

channelDoubleSpanMinMX100

構文

```
double channelDoubleSpanMinMX100(DAQMX100 daqmx100, int chNo);
```

宣言

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);
```

引数

daqmx100	機器記述子を指定します。
chNo	チャネル番号を指定します。

説明

保持している現在のチャネル設定データから，指定されたチャネル番号のスパン最小値を取得します。

- 戻り値は，小数点位置を含む浮動小数値です。
- 存在しない場合，0.0を返します。

戻り値

スパン最小値を返します。

参照

```
CDAQMX100::getClassMXItemConfig
CDAQMXItemConfig::getDoubleSpanMin
```

channelErrorChoiceMX100

構文

```
int channelErrorChoiceMX100(DAQMX100 daqmx100, int outputNo);
```

宣言

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);
```

引数

daqmx100	機器記述子を指定します。
outputNo	出力チャネルデータ番号を指定します。

説明

保持している現在のチャネル設定データから，指定された出力チャネルデータ番号のエラー時選択値を取得します。

・ 存在しない場合，「前回値」を返します。

戻り値

選択値を返します。

参照

```
CDAQMX100::getClassMXItemConfig
CDAQMXItemConfig::getClassMXOutputData
CDAQMXOutputData::getErrorChoice
```

channelFIFOIndexMX100

構文

```
int channelFIFOIndexMX100(DAQMX100 daqmx100, int chNo);
```

宣言

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);
```

引数

daqmx100	機器記述子を指定します。
chNo	チャネル番号を指定します。

説明

保持している現在のチャネル情報データから，指定されたチャネル番号のFIFO内チャネル順序番号を取得します。

・ 存在しない場合，負の値を返します。

戻り値

FIFO内チャネル順序番号を返します。

参照

```
CDAQMX100::getClassMXDataBuffer
CDAQMXChInfo::getFIFOIndex
CDAQMXDataBuffer::getClassMXChInfo
```

channelFIFONoMX100

構文

```
int channelFIFONoMX100(DAQMX100 daqmx100, int chNo);
```

宣言

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);
```

引数

daqmx100	機器記述子を指定します。
chNo	チャネル番号を指定します。

説明

保持している現在のチャネル情報データから，指定されたチャネル番号のFIFO番号を取得します。

・存在しない場合，負の値を返します。

戻り値

FIFO番号を返します。

参照

```
CDAQMX100::getClassMXDataBuffer
CDAQMXChInfo::getFIFONo
CDAQMXDataBuffer::getClassMXChInfo
```

channelFilterMX100

構文

```
int channelFilterMX100(DAQMX100 daqmx100, int chNo);
```

宣言

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);
```

引数

daqmx100	機器記述子を指定します。
chNo	チャネル番号を指定します。

説明

保持している現在のチャネル設定データから，指定されたチャネル番号のフィルタ時定数を取得します。

・ 存在しない場合，「時定数0」を返します。

戻り値

フィルタ時定数を返します。

参照

```
CDAQMX100::getClassMXItemConfig
CDAQMXChConfig::getFilter
CDAQMXItemConfig::getClassMXChConfig
```

channelHoldMX100

構文

```
int channelHoldMX100(DAQMX100 daqmx100, int doNo);
```

宣言

Visual Basic

```
q 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);
```

引数

daqmx100	機器記述子を指定します。
doNo	DOデータ番号を指定します。

説明

保持している現在のチャネル設定データから，指定されたDOデータ番号の保持を有効無効値で取得します。

・存在しない場合，「無効値」を返します。

戻り値

有効無効値を返します。

参照

```
CDAQMX100::getClassMXItemConfig
CDAQMXChConfig::isHold
CDAQMXItemConfig::getClassMXChConfig
```

channelIdleChoiceMX100

構文

```
int channelIdleChoiceMX100(DAQMX100 daqmx100, int outputNo);
```

宣言

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);
```

引数

daqmx100	機器記述子を指定します。
outputNo	出力チャネルデータ番号を指定します。

説明

保持している現在のチャネル設定データから，指定された出力チャネルデータ番号のアイドル時選択値を取得します。

・ 存在しない場合，「前回値」を返します。

戻り値

選択値を返します。

参照

```
CDAQMX100::getClassMXItemConfig
CDAQMXItemConfig::getClassMXOutputData
CDAQMXOutputData::getIdleChoice
```

channelKindMX100

構文

```
int channelKindMX100(DAQMX100 daqmx100, int chNo);
```

宣言

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);
```

引数

daqmx100	機器記述子を指定します。
chNo	チャネル番号を指定します。

説明

保持している現在のチャネル設定データから，指定されたチャネル番号のチャネル種類を取得します。

・存在しない場合，「未使用」を返します。

戻り値

チャネル種類を返します。

参照

```
CDAQMX100::getClassMXItemConfig
CDAQMXChConfig::getKind
CDAQMXItemConfig::getClassMXChConfig
```

channelNumberMX100

構文

```
int channelNumberMX100(DAQMX100 daqmx100, int fifoNo, int
fifoIndex);
```

宣言

Visual Basic

```
Public Declare Function channelNumberMX100 Lib "DAQMX100"
(ByVal daqmx100 As Long, ByVal fifoNo As Long, ByVal
fifioIndex As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function channelNumberMX100 Lib "DAQMX100"
(ByVal daqmx100 As Integer, ByVal fifoNo As Integer, ByVal
fifioIndex As Integer) As Integer
```

C#

```
[DllImport("DAQMX100.dll", CharSet=CharSet.Auto,
EntryPoint="channelNumberMX100")]
public static extern int channelNumberMX100(int daqmx100, int
fifoNo, int fifioIndex);
```

引数

daqmx100	機器記述子を指定します。
fifoNo	FIFO番号を指定します。
fifoIndex	FIFO内チャネル順序番号を指定します。

説明

指定されたFIFO番号とFIFO内チャネル順序番号から，チャネル番号を取得します。

・ 存在しない場合，0を返します。

戻り値

チャネル番号を返します。

参照

CDAQMX100::toChNo

channelOutputTypeMX100

構文

```
int channelOutputTypeMX100(DAQMX100 daqmx100, int outputNo);
```

宣言

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);
```

引数

daqmx100	機器記述子を指定します。
outputNo	出力チャネルデータ番号を指定します。

説明

保持している現在のチャネル設定データから，指定された出力チャネルデータ番号の出力種類を取得します。

・ 存在しない場合，「出力なし」を返します。

戻り値

出力種類を返します。

参照

```
CDAQMX100::getClassMXItemConfig
CDAQMXItemConfig::getClassMXOutputData
CDAQMXOutputData::getOutputType
```

channelPointMX100

構文

```
int channelPointMX100(DAQMX100 daqmx100, int chNo);
```

宣言

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);
```

引数

daqmx100	機器記述子を指定します。
chNo	チャネル番号を指定します。

説明

保持している現在のチャネル設定データから，指定されたチャネル番号の小数点位置を取得します。

・ 存在しない場合，0を返します。

戻り値

小数点位置を返します。

参照

```
CDAQMX100::getClassMXItemConfig
CDAQMXChConfig::getPoint
CDAQMXItemConfig::getClassMXChConfig
```

channelPresetValueMX100

構文

```
int channelPresetValueMX100(DAQMX100 daqmx100, int outputNo);
```

宣言

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);
```

引数

daqmx100　　機器記述子を指定します。

outputNo　　出力チャネルデータ番号を指定します。

説明

保持している現在のチャネル設定データから，指定された出力チャネルデータ番号のユーザ指定の出力値を取得します。

・戻り値は，小数点位置を除く整数値です。

・存在しない場合，0を返します。

戻り値

ユーザ指定の出力値を返します。

参照

```
CDAQMX100::getClassMXItemConfig
CDAQMXItemConfig::getClassMXOutputData
CDAQMXOutputData::getPresetValue
```

channelPulseTimeMX100

構文

```
int channelPulseTimeMX100(DAQMX100 daqmx100, int outputNo);
```

宣言

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);
```

引数

daqmx100	機器記述子を指定します。
outputNo	出力チャネルデータ番号を指定します。

説明

保持している現在のチャネル設定データから，指定された出力チャネルデータ番号のパルス周期倍率を取得します。

・ 存在しない場合，1(最小値)を返します。

戻り値

パルス周期倍率を返します。

参照

```
CDAQMX100::getClassMXItemConfig
CDAQMXItemConfig::getClassMXOutputData
CDAQMXOutputData::getPulseTime
```

channelRangeMX100

構文

```
int channelRangeMX100(DAQMX100 daqmx100, int chNo);
```

宣言

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);
```

引数

daqmx100	機器記述子を指定します。
chNo	チャネル番号を指定します。

説明

保持している現在のチャネル設定データから，指定されたチャネル番号のレンジ種類を取得します。

・ スキップ(未使用)のレンジ種類は，・存在しない場合と同じ扱いになります。チャネルステータスを参照してください。

・ 存在しない場合，0(「20mV」)を返します。

戻り値

レンジ種類を返します。

参照

```
CDAQMX100::getClassMXItemConfig
CDAQMXChConfig::getRange
CDAQMXItemConfig::getClassMXChConfig
```

channelRealMaxMX100

構文

```
double channelRealMaxMX100(DAQMX100 daqmx100, int chNo);
```

宣言

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);
```

引数

daqmx100	機器記述子を指定します。
chNo	チャネル番号を指定します。

説明

保持している現在のチャネル情報データから，指定されたチャネル番号の実範囲最大値を取得します。

・ 存在しない場合，0.0を返します。

戻り値

実範囲最大値を返します。

参照

```
CDAQMX100::getClassMXDataBuffer
CDAQMXChInfo::getRealMax
CDAQMXDataBuffer::getClassMXChInfo
```

channelRealMinMX100

構文

```
double channelRealMinMX100(DAQMX100 daqmx100, int chNo);
```

宣言

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);
```

引数

daqmx100	機器記述子を指定します。
chNo	チャネル番号を指定します。

説明

保持している現在のチャネル情報データから，指定されたチャネル番号の実範囲最小値を取得します。

・ 存在しない場合，0.0を返します。

戻り値

実範囲最小値を返します。

参照

```
CDAQMX100::getClassMXDataBuffer
CDAQMXChInfo::getRealMin
CDAQMXDataBuffer::getClassMXChInfo
```

channelRefAlarmMX100

構文

```
int channelRefAlarmMX100(DAQMX100 daqmx100, int doNo, int
refChNo, int levelNo);
```

宣言

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);
```

引数

daqmx100	機器記述子を指定します。
doNo	DOデータ番号を指定します。
refChNo	参照チャネルをチャネル番号で指定します。
levelNo	アラームレベルを指定します。

説明

保持している現在のチャネル設定データから，指定されたDOデータ番号の参照アラームを有効無効値で取得します。

- 参照アラームは，チャネル番号とアラームレベルで指定します。
- 存在しない場合，「無効値」を返します。

戻り値

有効無効値を返します。

参照

```
CDAQMX100::getClassMXItemConfig
CDAQMXChConfig::isRefAlarm
CDAQMXItemConfig::getClassMXChConfig
```

channelRefChNoMX100

構文

```
int channelRefChNoMX100(DAQMX100 daqmx100, int chNo);
```

宣言

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);
```

引数

daqmx100	機器記述子を指定します。
chNo	チャネル番号を指定します。

説明

保持している現在のチャネル設定データから，指定されたチャネル番号の基準チャネル番号を取得します。

・ 存在しない場合，定数値の「未定義参照チャネル番号」を返します。

戻り値

基準チャネル番号を返します。

参照

```
CDAQMX100::getClassMXItemConfig
CDAQMXChConfig::getRefChNo
CDAQMXItemConfig::getClassMXChConfig
```

channelRJCTypeMX100

構文

```
int channelRJCTypeMX100(DAQMX100 daqmx100, int chNo);
```

宣言

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);
```

引数

daqmx100	機器記述子を指定します。
chNo	チャネル番号を指定します。

説明

保持している現在のチャネル設定データから，指定されたチャネル番号のRJC種類を取得します。

・ 存在しない場合，「内部」を返します。

戻り値

RJC種類を返します。

参照

```
CDAQMX100::getClassMXItemConfig
CDAQMXChConfig::getRJCType
CDAQMXItemConfig::getClassMXChConfig
```

channelRJCVoltMX100

構文

```
int channelRJCVoltMX100(DAQMX100 daqmx100, int chNo);
```

宣言

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);
```

引数

daqmx100	機器記述子を指定します。
chNo	チャネル番号を指定します。

説明

保持している現在のチャネル設定データから，指定されたチャネル番号のRJC電圧値を取得します。

・ 存在しない場合，0を返します。

戻り値

RJC電圧値を返します。

参照

```
CDAQMX100::getClassMXItemConfig
CDAQMXChConfig::getRJCVolt
CDAQMXItemConfig::getClassMXChConfig
```

channelScaleMaxMX100

構文

```
int channelScaleMaxMX100(DAQMX100 daqmx100, int chNo);
```

宣言

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);
```

引数

daqmx100	機器記述子を指定します。
chNo	チャネル番号を指定します。

説明

保持している現在のチャネル設定データから，指定されたチャネル番号のスケール最大値を取得します。

- 戻り値は，小数点位置を除く整数値です。
- 存在しない場合，0を返します。

戻り値

スケール最大値を返します。

参照

```
CDAQMX100::getClassMXItemConfig
CDAQMXChConfig::getScaleMax
CDAQMXItemConfig::getClassMXChConfig
```

channelScaleMinMX100

構文

```
int channelScaleMinMX100(DAQMX100 daqmx100, int chNo);
```

宣言

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);
```

引数

daqmx100	機器記述子を指定します。
chNo	チャネル番号を指定します。

説明

保持している現在のチャネル設定データから，指定されたチャネル番号のスケール最小値を取得します。

- 戻り値は，小数点位置を除く整数値です。
- 存在しない場合，0を返します。

戻り値

スケール最小値を返します。

参照

```
CDAQMX100::getClassMXItemConfig
CDAQMXChConfig::getScaleMin
CDAQMXItemConfig::getClassMXChConfig
```

channelScaleTypeMX100

構文

```
int channelScaleTypeMX100(DAQMX100 daqmx100, int chNo);
```

宣言

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);
```

引数

daqmx100	機器記述子を指定します。
chNo	チャネル番号を指定します。

説明

保持している現在のチャネル設定データから，指定されたチャネル番号のスケール種類を取得します。

・ 存在しない場合，「スケールなし」を返します。

戻り値

スケール種類を返します。

参照

```
CDAQMX100::getClassMXItemConfig
CDAQMXChConfig::getScale
CDAQMXItemConfig::getClassMXChConfig
```

channelSpanMaxMX100

構文

```
int channelSpanMaxMX100(DAQMX100 daqmx100, int chNo);
```

宣言

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);
```

引数

daqmx100	機器記述子を指定します。
chNo	チャネル番号を指定します。

説明

保持している現在のチャネル設定データから，指定されたチャネル番号のスパン最大値を取得します。

- 戻り値は，小数点位置を除く整数値です。
- 存在しない場合，0を返します。

戻り値

スパン最大値を返します。

参照

```
CDAQMX100::getClassMXItemConfig
CDAQMXChConfig::getSpanMax
CDAQMXItemConfig::getClassMXChConfig
```

channelSpanMinMX100

構文

```
int channelSpanMinMX100(DAQMX100 daqmx100, int chNo);
```

宣言

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);
```

引数

daqmx100	機器記述子を指定します。
chNo	チャネル番号を指定します。

説明

保持している現在のチャネル設定データから，指定されたチャネル番号のスパン最小値を取得します。

- 戻り値は，小数点位置を除く整数値です。
- 存在しない場合，0を返します。

戻り値

スパン最小値を返します。

参照

```
CDAQMX100::getClassMXItemConfig
CDAQMXChConfig::getSpanMin
CDAQMXItemConfig::getClassMXChConfig
```

channelValidMX100

構文

```
int channelValidMX100(DAQMX100 daqmx100, int chNo);
```

宣言

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);
```

引数

daqmx100	機器記述子を指定します。
chNo	チャネル番号を指定します。

説明

保持している現在のチャネル設定データから，指定されたチャネル番号のチャネルステータスを有効無効値で取得します。

・存在しない場合，「無効値」を返します。

戻り値

有効無効値を返します。

参照

```
CDAQMX100::getClassMXItemConfig
CDAQMXChConfig::isValid
CDAQMXItemConfig::getClassMXChConfig
```

currentAOPWMValidMX100

構文

```
int currentAOPWMValidMX100(DAQMX100 daqmx100, int aopwmNo);
```

宣言

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);
```

引数

daqmx100	機器記述子を指定します。
aopwmNo	AO/PWMデータ番号を指定します。

説明

保持している現在のAO/PWMデータから，指定されたAO/PWMデータ番号の有効/無効を有効無効値で取得します。

・ 存在しない場合，「無効値」を返します。

戻り値

有効無効値を返します。

参照

```
CDAQMX100::getClassMXAOPWMList
CDAQMXAOPWMData::getAOPWMValid
CDAQMXAOPWMList::getCurrent
```

currentAOPWMValueMX100

構文

```
int currentAOPWMValueMX100(DAQMX100 daqmx100, int aopwmNo);
```

宣言

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);
```

引数

daqmx100	機器記述子を指定します。
aopwmNo	AO/PWMデータ番号を指定します。

説明

保持している現在のAO/PWMデータから，指定されたAO/PWMデータ番号の出力データ値を取得します。

・ 存在しない場合，0を返します。

戻り値

出力データ値を返します。

参照

```
CDAQMX100::getClassMXAOPWMList
CDAQMXAOPWMData::getAOPWMValue
CDAQMXAOPWMList::getCurrent
```

currentBalanceResultMX100

構文

```
int currentBalanceResultMX100(DAQMX100 daqmx100, int
balanceNo);
```

宣言

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);
```

引数

daqmx100	機器記述子を指定します。
balanceNo	初期バランスデータ番号を指定します。

説明

保持している現在の初期バランスデータから，指定された初期バランスデータ番号の初期バランス結果を取得します。

- 最後に実行された初期バランスの設定機能による結果を返します。
- 存在しない場合，「指定なし」を返します。

戻り値

初期バランス結果を返します。

参照

```
CDAQMX100::getClassMXBalanceList
CDAQMXBalanceList::getCurrent
CDAQMXBalanceResult::getResult
```

currentBalanceValidMX100

構文

```
int currentBalanceValidMX100(DAQMX100 daqmx100, int
balanceNo);
```

宣言

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);
```

引数

daqmx100	機器記述子を指定します。
balanceNo	初期バランスデータ番号を指定します。

説明

保持している現在の初期バランスデータから，指定された初期バランスデータ番号の有効/無効を有効無効値で取得します。

・存在しない場合，「無効値」を返します。

戻り値

有効無効値を返します。

参照

```
CDAQMX100::getClassMXBalanceList
CDAQMXBalanceList::getCurrent
CDAQMXBalanceResult::getBalanceValid
```

currentBalanceValueMX100

構文

```
int currentBalanceValueMX100(DAQMX100 daqmx100, int
balanceNo);
```

宣言

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);
```

引数

daqmx100	機器記述子を指定します。
balanceNo	初期バランスデータ番号を指定します。

説明

保持している現在の初期バランスデータから，指定された初期バランスデータ番号の初期バランス値を取得します。

・ 存在しない場合，0を返します。

戻り値

初期バランス値を返します。

参照

```
CDAQMX100::getClassMXBalanceList
CDAQMXBalanceList::getCurrent
CDAQMXBalanceResult::getBalanceValue
```

currentDoubleAOPWMValueMX100

構文

```
double currentDoubleAOPWMValueMX100(DAQMX100 daqmx100, int
aopwmNo);
```

宣言

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);
```

引数

daqmx100	機器記述子を指定します。
aopwmNo	AO/PWMデータ番号を指定します。

説明

保持している現在のAO/PWMデータから，指定されたAO/PWMデータ番号の出力データ値を実際の出力値で取得します。

・ 存在しない場合，0.0を返します。

戻り値

実際の出力値を返します。

参照

`CDAQMX100::currentDoubleAOPWMValue`

currentDOValidMX100

構文

```
int currentDOValidMX100(DAQMX100 daqmx100, int doNo);
```

宣言

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);
```

引数

daqmx100	機器記述子を指定します。
doNo	DOデータ番号を指定します。

説明

保持している現在のDOデータから，指定されたDOデータ番号の有効/無効を有効無効値で取得します。

・ 存在しない場合，「無効値」を返します。

戻り値

有効無効値を返します。

参照

```
CDAQMX100::getClassMXDOList
CDAQMXDOData::getDOValid
CDAQMXDOList::getCurrent
```

currentDOValueMX100

構文

```
int currentDOValueMX100(DAQMX100 daqmx100, int doNo);
```

宣言

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);
```

引数

daqmx100	機器記述子を指定します。
doNo	DOデータ番号を指定します。

説明

保持している現在のDOデータから，指定されたDOデータ番号のON/OFFを有効無効値で取得します。

・ 存在しない場合，「無効値」を返します。

戻り値

有効無効値を返します。

参照

```
CDAQMX100::getClassMXDOList
CDAQMXDOData::getDOONOFF
CDAQMXDOList::getCurrent
```

currentTransmitMX100

構文

```
int currentTransmitMX100(DAQMX100 daqmx100, int aopwmNo);
```

宣言

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);
```

引数

daqmx100	機器記述子を指定します。
aopwmNo	AO/PWMデータ番号を指定します。

説明

保持している現在の伝送出力データから，指定されたAO/PWMデータ番号の伝送状態を取得します。

・ 存在しない場合，「指定なし(不明)」を返します。

戻り値

伝送状態を返します。

参照

```
CDAQMX100::getClassMXTransmitList
CDAQMXTransmit::getTransmit
CDAQMXTransmitList::getCurrent
```

dataAlarmMX100

構文

```
int dataAlarmMX100(DAQMX100 daqmx100, int chNo, int levelNo);
```

宣言

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);
```

引数

daqmx100	機器記述子を指定します。
chNo	チャネル番号を指定します。
levelNo	アラームレベルを指定します。

説明

保持している現在の測定データから，指定されたチャネル番号のアラームレベルに対応するアラームの有無を有効無効値で取得します。

・存在しない場合，「無効値」を返します。

戻り値

有効無効値を返します。

参照

```
CDAQMX100::getClassMXDataBuffer
CDAQMXDataBuffer::currentDataInfo
CDAQMXDataInfo::isAlarm
```

dataDayMX100

構文

```
int dataDayMX100(DAQMX100 daqmx100, int chNo);
```

宣言

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);
```

引数

daqmx100	機器記述子を指定します。
chNo	チャネル番号を指定します。

説明

保持している現在の時刻情報データから，指定されたチャネル番号の日を取得します。

- 日は1から31の数値です。
- 存在しない場合，0を返します。

戻り値

日の値を返します。

参照

```
CDAQMX100::getClassMXDataBuffer
CDAQMXDataBuffer::currentDateTime
CDAQMXDateTime::toLocalDateTime
```

dataDoubleValueMX100

構文

```
double dataDoubleValueMX100(DAQMX100 daqmx100, int chNo);
```

宣言

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);
```

引数

daqmx100	機器記述子を指定します。
chNo	チャネル番号を指定します。

説明

保持している現在の測定データから，指定されたチャネル番号の測定値を取得します。

・ 存在しない場合，0.0を返します。

戻り値

測定値を倍精度浮動小数で返します。

参照

```
CDAQMX100::getClassMXDataBuffer
CDAQMXDataBuffer::currentDataInfo
CDAQMXDataInfo::getDoubleValue
```

dataHourMX100

構文

```
int dataHourMX100(DAQMX100 daqmx100, int chNo);
```

宣言

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);
```

引数

daqmx100	機器記述子を指定します。
chNo	チャネル番号を指定します。

説明

保持している現在の時刻情報データから，指定されたチャネル番号の時を取得します。

- 時は0から23の数値です。
- 存在しない場合，0を返します。

戻り値

時の値を返します。

参照

```
CDAQMX100::getClassMXDataBuffer
CDAQMXDataBuffer::currentDateTime
CDAQMXDateTime::toLocalDateTime
```

dataMilliSecMX100

構文

```
int dataMilliSecMX100(DAQMX100 daqmx100, int chNo);
```

宣言

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);
```

引数

daqmx100	機器記述子を指定します。
chNo	チャネル番号を指定します。

説明

保持している現在の時刻情報データから，指定されたチャネル番号のミリ秒を取得します。

・ 存在しない場合，0を返します。

戻り値

ミリ秒の値を返します。

参照

```
CDAQMX100::getClassMXDataBuffer
CDAQMXDataBuffer::currentDateTime
CDAQMXDateTime::getMilliSecond
```

dataMinuteMX100

構文

```
int dataMinuteMX100(DAQMX100 daqmx100, int chNo);
```

宣言

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);
```

引数

daqmx100	機器記述子を指定します。
chNo	チャネル番号を指定します。

説明

保持している現在の時刻情報データから，指定されたチャネル番号の分を取得します。

- 分は0から59の数値です。
- 存在しない場合，0を返します。

戻り値

分の値を返します。

参照

```
CDAQMX100::getClassMXDataBuffer
CDAQMXDataBuffer::currentDateTime
CDAQMXDateTime::toLocalDateTime
```

dataMonthMX100

構文

```
int dataMonthMX100(DAQMX100 daqmx100, int chNo);
```

宣言

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);
```

引数

daqmx100	機器記述子を指定します。
chNo	チャネル番号を指定します。

説明

保持している現在の時刻情報データから，指定されたチャネル番号の月を取得します。

・ 月は1から12の数値です。
・ 存在しない場合，0を返します。

戻り値

月の値を返します。

参照

```
CDAQMX100::getClassMXDataBuffer
CDAQMXDataBuffer::currentDateTime
CDAQMXDateTime::toLocalDateTime
```

dataNumChMX100

構文

```
int dataNumChMX100(DAQMX100 daqmx100, int chNo);
```

宣言

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);
```

引数

daqmx100	機器記述子を指定します。
chNo	チャネル番号を指定します。

説明

データ収集機能で取得保持しているデータの内，指定されたチャネル番号の現在状態のデータより残りのデータ個数を取得します。

・ 存在しない場合，0を返します。

戻り値

残りのデータ個数を返します。

参照

```
CDAQMX100::getClassMXDataBuffer
CDAQMXDataBuffer::getDataNum
```

dataNumFIFOMX100

構文

```
int dataNumFIFOMX100(DAQMX100 daqmx100, int fifoNo);
```

宣言

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);
```

引数

daqmx100	機器記述子を指定します。
fifoNo	FIFO番号を指定します。

説明

データ収集機能で取得保持しているデータの内，指定されたFIFO番号の現在状態のデータより残りのデータ個数を取得します。

- FIFO内チャネルの中で最小値を返します。
- 存在しない場合，0を返します。

戻り値

残りのデータ個数を返します。

参照

`CDAQMX100::getDataNum`

dataSecondMX100

構文

```
int dataSecondMX100(DAQMX100 daqmx100, int chNo);
```

宣言

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);
```

引数

daqmx100	機器記述子を指定します。
chNo	チャネル番号を指定します。

説明

保持している現在の時刻情報データから，指定されたチャネル番号の秒を取得します。

- 秒は0から59の数値です。
- 存在しない場合，0を返します。

戻り値

秒の値を返します。

参照

```
CDAQMX100::getClassMXDataBuffer
CDAQMXDataBuffer::currentDateTime
CDAQMXDateTime::toLocalDateTime
```

dataStatusMX100

構文

```
int dataStatusMX100(DAQMX100 daqmx100, int chNo);
```

宣言

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);
```

引数

daqmx100	機器記述子を指定します。
chNo	チャネル番号を指定します。

説明

保持している現在の測定データから，指定されたチャネル番号のデータステータス値を取得します。

・存在しない場合，「不明状態」を返します。

戻り値

データステータス値を返します。

参照

```
CDAQMX100::getClassMXDataBuffer
CDAQMXDataBuffer::currentDataInfo
CDAQMXDataInfo::getStatus
```

dataStringValueMX100

構文

```
int dataStringValueMX100(DAQMX100 daqmx100, int chNo, char *
strValue, int lenValue);
```

宣言

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);
```

引数

daqmx100	機器記述子を指定します。
chNo	チャネル番号を指定します。
strValue	文字列を格納する領域を指定します。
lenValue	文字列を格納する領域のバイト数を指定します。

説明

保持している現在の測定データから，指定されたチャネル番号の測定値を取得します。

- 文字列に変換して，指定された領域に格納します。
- 領域に格納する文字列には，終端(NULL)も含まれます。
- 実際の文字列の長さが戻り値になります。戻り値に終端は含まれません。
- 存在しない場合，0を返します。
- 格納される文字列は，原則ascii文字列です。

戻り値

文字列の長さを返します。

参照

```
CDAQMX100::getClassMXDataBuffer
CDAQMXDataBuffer::currentDataInfo
CDAQMXDataInfo::getStringValue
```

dataTimeMX100

構文

```
int dataTimeMX100(DAQMX100 daqmx100, int chNo);
```

宣言

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);
```

引数

daqmx100	機器記述子を指定します。
chNo	チャネル番号を指定します。

説明

保持している現在の時刻情報データから，指定されたチャネル番号の秒数を取得します。

・ ここで，秒数は基準日時(1970年1月1日)からの時間です。
・ 存在しない場合，0を返します。

戻り値

秒数を返します。

参照

```
CDAQMX100::getClassMXDataBuffer
CDAQMXDataBuffer::currentDateTime
CDAQMXDateTime::getTime
```

dataValidMX100

構文

```
int dataValidMX100(DAQMX100 daqmx100, int chNo);
```

宣言

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);
```

引数

daqmx100	機器記述子を指定します。
chNo	チャネル番号を指定します。

説明

保持している現在の測定データから，指定されたチャネル番号の測定データの有無を有効無効値で取得します。

・存在しない場合，「無効値」を返します。

戻り値

有効無効値を返します。

参照

```
CDAQMX100::getClassMXDataBuffer
CDAQMXDataBuffer::isCurrent
```

dataValueMX100

構文

```
int dataValueMX100(DAQMX100 daqmx100, int chNo);
```

宣言

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);
```

引数

daqmx100	機器記述子を指定します。
chNo	チャネル番号を指定します。

説明

保持している現在の測定データから，指定されたチャネル番号のデータ値を取得します。

・ 存在しない場合，0を返します。

戻り値

データ値を返します。

参照

```
CDAQMX100::getClassMXDataBuffer
CDAQMXDataBuffer::currentDataInfo
CDAQMXDataInfo::getValue
```

dataYearMX100

構文

```
int dataYearMX100(DAQMX100 daqmx100, int chNo);
```

宣言

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);
```

引数

daqmx100	機器記述子を指定します。
chNo	チャネル番号を指定します。

説明

保持している現在の時刻情報データから，指定されたチャネル番号の年を取得します。

- 年は4桁の数値です。
- 存在しない場合，0を返します。

戻り値

年の値を返します。

参照

```
CDAQMX100::getClassMXDataBuffer
CDAQMXDataBuffer::currentDateTime
CDAQMXDateTime::toLocalDateTime
```

errorMaxLengthMX100

構文

```
int errorMaxLengthMX100(void);
```

宣言

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();
```

説明

エラーメッセージ文字列の最大長を取得します。

・ 戻り値に終端は含まれません。

戻り値

文字列の長さを返します。

参照

```
CDAQMX100::getMaxLenErrorMessage
```

getAlarmNameMX100　　[Visual Cのみ]

構文

```
const char * getAlarmNameMX100(int iAlarmType);
```

引数

iAlarmType　　アラーム種類を指定します。

説明

指定されたアラーム種類に対応する文字列を取得します。

・ 存在しない場合，「アラームなし」に対応する文字列へのポインタを返します。

戻り値

文字列へのポインタを返します。

参照

`CDAQMXDataInfo::getAlarmName`

getChannelCommentMX100 [Visual Cのみ]

構文

```
const char * getChannelCommentMX100(DAQMX100 daqmx100, int
chNo);
```

引数

daqmx100　機器記述子を指定します。
chNo　チャネル番号を指定します。

説明

保持している現在のチャネル設定データから，指定されたチャネル番号のコメントを取得します。

・ 存在しない場合，NULLを返します。

戻り値

文字列へのポインタを返します。

参照

CDAQMX100::getClassMXItemConfig
CDAQMXChConfig::getComment
CDAQMXItemConfig::getClassMXChConfig

getChannelTagMX100 [Visual Cのみ]

構文

```
const char * getChannelTagMX100(DAQMX100 daqmx100, int chNo);
```

引数

daqmx100　機器記述子を指定します。
chNo　チャネル番号を指定します。

説明

保持している現在のチャネル設定データから，指定されたチャネル番号のタグを取得します。

・ 存在しない場合，NULLを返します。

戻り値

文字列へのポインタを返します。

参照

`CDAQMX100::getClassMXItemConfig`
`CDAQMXChConfig::getTag`
`CDAQMXItemConfig::getClassMXChConfig`

getChannelUnitMX100 [Visual Cのみ]

構文

```
const char * getChannelUnitMX100(DAQMX100 daqmx100, int chNo);
```

引数

daqmx100	機器記述子を指定します。
chNo	チャネル番号を指定します。

説明

保持している現在のチャネル設定データから，指定されたチャネル番号の単位名を取得します。

・ 存在しない場合，NULLを返します。

戻り値

文字列へのポインタを返します。

参照

CDAQMX100::getClassMXItemConfig
CDAQMXChConfig::getUnit
CDAQMXItemConfig::getClassMXChConfig

getErrorMessageMX100 [Visual Cのみ]

構文

```
const char * getErrorMessageMX100(int errorCode);
```

引数

errorCode エラー番号を指定します。

説明

指定されたエラー番号に対応するエラーメッセージ文字列を取得します。

・ 存在しない場合，文字列「Unknown」へのポインタを返します。

戻り値

文字列へのポインタを返します。

参照

`CDAQMX100::getErrorMessage`

getModuleSerialMX100 [Visual Cのみ]

構文

```
const char * getModuleSerialMX100(DAQMX100 daqmx100, int
moduleNo);
```

引数

daqmx100　機器記述子を指定します。
moduleNo　モジュール番号を指定します。

説明

保持している現在のシステム構成データから，指定されたモジュール番号のシリアル番号を取得します。

・ 存在しない場合，NULLを返します。

戻り値

文字列へのポインタを返します。

参照

```
CDAQMX100::getClassMXItemConfig
CDAQMXItemConfig::getClassMXSysInfo
CDAQMXSysInfo::getModuleSerial
```

getNetHostMX100 [Visual Cのみ]

構文

```
const char * getNetHostMX100(DAQMX100 daqmx100);
```

引数

daqmx100　　機器記述子を指定します。

説明

保持している現在のネットワーク情報データから，ホスト名を取得します。

・ 存在しない場合，NULLを返します。

戻り値

文字列へのポインタを返します。

参照

```
CDAQMX100::getClassMXItemConfig
CDAQMXItemConfig::getClassMXNetInfo
CDAQMXNetInfo::getHost
```

getUnitPartNoMX100 [Visual Cのみ]

構文

```
const char * getUnitPartNoMX100(DAQMX100 daqmx100);
```

引数

daqmx100　機器記述子を指定します。

説明

保持している現在のシステム構成データから，パート番号を取得します。

・ 存在しない場合，NULLを返します。

戻り値

文字列へのポインタを返します。

参照

```
CDAQMX100::getClassMXItemConfig
CDAQMXItemConfig::getClassMXSysInfo
CDAQMXSysInfo::getPartNo
```

getUnitSerialMX100 [Visual Cのみ]

構文

```
const char * getUnitSerialMX100(DAQMX100 daqmx100);
```

引数

daqmx100　　機器記述子を指定します。

説明

保持している現在のシステム構成データから，ユニットのシリアル番号を取得します。

・ 存在しない場合，NULLを返します。

戻り値

文字列へのポインタを返します。

参照

CDAQMX100::getClassMXItemConfig
CDAQMXItemConfig::getClassMXSysInfo
CDAQMXSysInfo::getUnitSerial

itemErrorMX100

構文

```
int itemErrorMX100(DAQMX100 daqmx100);
```

宣言

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);
```

引数

daqmx100　　機器記述子を指定します。

説明

最後にエラー検出した設定項目番号を取得します。

・ 存在しない場合，「不明」を返します。

戻り値

設定項目番号を返します。

参照

CDAQMX100::getItemError

itemMaxLengthMX100

構文

```
int itemMaxLengthMX100(void);
```

宣言

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();
```

説明

設定項目番号に対応する名称の文字列の最大長を取得します。

・ 戻り値に終端は含まれません。

戻り値

文字列の長さを返します。

参照

CDAQMXItemConfig::getMaxLenItemName

lastErrorMX100

構文

```
int lastErrorMX100(DAQMX100 daqmx100);
```

宣言

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);
```

引数

daqmx100　　機器記述子を指定します。

説明

最後に通信で受信したMX固有エラーを取得します。

・ 存在しない場合，0を返します。

戻り値

MX固有エラーを返します。

参照

CDAQMX100::getLastError

moduleChNumMX100

構文

```
int moduleChNumMX100(DAQMX100 daqmx100, int moduleNo);
```

宣言

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);
```

引数

daqmx100	機器記述子を指定します。
moduleNo	モジュール番号を指定します。

説明

保持している現在のシステム構成データから，指定されたモジュール番号のチャネル数を取得します。

・ 存在しない場合，「0」を返します。

戻り値

チャネル数を返します。

参照

```
CDAQMX100::getClassMXItemConfig
CDAQMXItemConfig::getClassMXSysInfo
CDAQMXSysInfo::getChNum
```

moduleFIFONoMX100

構文

```
int moduleFIFONoMX100(DAQMX100 daqmx100, int moduleNo);
```

宣言

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);
```

引数

daqmx100　　機器記述子を指定します。

moduleNo　　モジュール番号を指定します。

説明

保持している現在のシステム構成データから，指定されたモジュール番号のFIFO番号を取得します。

・ 存在しない場合，負の数を返します。

戻り値

FIFO番号を返します。

参照

```
CDAQMX100::getClassMXItemConfig
CDAQMXItemConfig::getClassMXSysInfo
CDAQMXSysInfo::getFIFONo
```

moduleIntegralMX100

構文

```
int moduleIntegralMX100(DAQMX100 daqmx100, int moduleNo);
```

宣言

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);
```

引数

daqmx100	機器記述子を指定します。
moduleNo	モジュール番号を指定します。

説明

保持している現在のシステム構成データから，指定されたモジュール番号のA/D積分時間種類を取得します。

・ 存在しない場合，「自動」を返します。

戻り値

A/D積分時間種類を返します。

参照

```
CDAQMX100::getClassMXItemConfig
CDAQMXItemConfig::getClassMXSysInfo
CDAQMXSysInfo::getIntegral
```

moduleIntervalMX100

構文

```
int moduleIntervalMX100(DAQMX100 daqmx100, int moduleNo);
```

宣言

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);
```

引数

daqmx100	機器記述子を指定します。
moduleNo	モジュール番号を指定します。

説明

保持している現在のシステム構成データから，指定されたモジュール番号の周期種類を取得します。

・ 存在しない場合，0を返します。

戻り値

周期種類を返します。

参照

```
CDAQMX100::getClassMXItemConfig
CDAQMXItemConfig::getClassMXSysInfo
CDAQMXSysInfo::getInterval
```

moduleRealTypeMX100

構文

```
int moduleRealTypeMX100(DAQMX100 daqmx100, int moduleNo);
```

宣言

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);
```

引数

daqmx100　　機器記述子を指定します。

moduleNo　　モジュール番号を指定します。

説明

保持している現在のシステム構成データから，指定されたモジュール番号の実際のモジュール種類を取得します。

・ 存在しない場合，「モジュールなし」を返します。

戻り値

モジュール種類を返します。

参照

```
CDAQMX100::getClassMXItemConfig
CDAQMXItemConfig::getClassMXSysInfo
CDAQMXSysInfo::getRealType
```

moduleStandbyTypeMX100

構文

```
int moduleStandbyTypeMX100(DAQMX100 daqmx100, int moduleNo);
```

宣言

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);
```

引数

daqmx100	機器記述子を指定します。
moduleNo	モジュール番号を指定します。

説明

保持している現在のシステム構成データから，指定されたモジュール番号の起動時モジュール種類を取得します。

・ 存在しない場合，「モジュールなし」を返します。

戻り値

モジュール種類を返します。

参照

```
CDAQMX100::getClassMXItemConfig
CDAQMXItemConfig::getClassMXSysInfo
CDAQMXSysInfo::getStandbyType
```

moduleTerminalMX100

構文

```
int moduleTerminalMX100(DAQMX100 daqmx100, int moduleNo);
```

宣言

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);
```

引数

daqmx100	機器記述子を指定します。
moduleNo	モジュール番号を指定します。

説明

保持している現在のシステム構成データから，指定されたモジュール番号の端子種類を取得します。

・ 存在しない場合，「ねじ」を返します。

戻り値

端子種類を返します。

参照

```
CDAQMX100::getClassMXItemConfig
CDAQMXItemConfig::getClassMXSysInfo
CDAQMXSysInfo::getTerminalType
```

moduleTypeMX100

構文

```
int moduleTypeMX100(DAQMX100 daqmx100, int moduleNo);
```

宣言

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);
```

引数

daqmx100	機器記述子を指定します。
moduleNo	モジュール番号を指定します。

説明

保持している現在のシステム構成データから，指定されたモジュール番号のモジュール種類を取得します。

・ 存在しない場合，「モジュールなし」を返します。

戻り値

モジュール種類を返します。

参照

```
CDAQMX100::getClassMXItemConfig
CDAQMXItemConfig::getClassMXSysInfo
CDAQMXSysInfo::getModuleType
```

moduleValidMX100

構文

```
int moduleValidMX100(DAQMX100 daqmx100, int moduleNo);
```

宣言

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);
```

引数

daqmx100	機器記述子を指定します。
moduleNo	モジュール番号を指定します。

説明

保持している現在のシステム構成データから，指定されたモジュール番号のモジュールの有無を有効無効値で取得します。

・ 存在しない場合，「無効値」を返します。

戻り値

有効無効値を返します。

参照

```
CDAQMX100::getClassMXItemConfig
CDAQMXItemConfig::getClassMXSysInfo
CDAQMXSysInfo::isModuleValid
```

moduleVersionMX100

構文

```
int moduleVersionMX100(DAQMX100 daqmx100, int moduleNo);
```

宣言

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);
```

引数

daqmx100	機器記述子を指定します。
moduleNo	モジュール番号を指定します。

説明

保持している現在のシステム構成データから，指定されたモジュール番号のモジュールのバージョンを取得します。

・ 存在しない場合，0を返します。

戻り値

バージョンを返します。

参照

```
CDAQMX100::getClassMXItemConfig
CDAQMXItemConfig::getClassMXSysInfo
CDAQMXSysInfo::getModuleVersion
```

netAddressMX100

構文

```
unsigned int netAddressMX100(DAQMX100 daqmx100);
```

宣言

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);
```

引数

daqmx100　　機器記述子を指定します。

説明

保持している現在のネットワーク情報データから，IPアドレスを取得します。

・ 存在しない場合，0を返します。

戻り値

IPアドレスを返します。

参照

```
CDAQMX100::getClassMXItemConfig
CDAQMXItemConfig::getClassMXNetInfo
CDAQMXNetInfo::getAddress
```

netGatewayMX100

構文

```
unsigned int netGatewayMX100(DAQMX100 daqmx100);
```

宣言

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);
```

引数

daqmx100　　機器記述子を指定します。

説明

保持している現在のネットワーク情報データから，Gatewayアドレスを取得します。

・ 存在しない場合，0を返します。

戻り値

Gatewayアドレスを返します。

参照

```
CDAQMX100::getClassMXItemConfig
CDAQMXItemConfig::getClassMXNetInfo
CDAQMXNetInfo::getGateway
```

netPortMX100

構文

```
unsigned int netPortMX100(DAQMX100 daqmx100);
```

宣言

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);
```

引数

daqmx100　　機器記述子を指定します。

説明

保持している現在のネットワーク情報データから，ポート番号を取得します。

・ 存在しない場合，0を返します。

戻り値

ポート番号を返します。

参照

```
CDAQMX100::getClassMXItemConfig
CDAQMXItemConfig::getClassMXNetInfo
CDAQMXNetInfo::getPort
```

netSubmaskMX100

構文

```
unsigned int netSubmaskMX100(DAQMX100 daqmx100);
```

宣言

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);
```

引数

daqmx100　　機器記述子を指定します。

説明

保持している現在のネットワーク情報データから，サブネットマスクを取得します。

・ 存在しない場合，0を返します。

戻り値

サブネットマスクを返します。

参照

```
CDAQMX100::getClassMXItemConfig
CDAQMXItemConfig::getClassMXNetInfo
CDAQMXNetInfo::getSubMask
```

rangePointMX100

構文

```
int rangePointMX100(DAQMX100 daqmx100, int iRange);
```

宣言

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);
```

引数

daqmx100	機器記述子を指定します。
iRange	レンジ種類を指定します。

説明

指定されたレンジ種類の小数点位置を取得します。

- レンジ種類でデジタル入力レンジの場合，デジタル入力の詳細レンジを指定します。モジュール種類を意識しない指定はできません。
- 存在しない場合，0を返します。

戻り値

小数点位置を返します。

参照

```
CDAQMX100::getClassMXItemConfig
CDAQMXItemConfig::getRangePoint
```

revisionAPIMX100

構文

```
const int revisionAPIMX100(void);
```

宣言

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();
```

説明

本APIのリビジョン番号を取得します。

戻り値

リビジョン番号を返します。

参照

CDAQMX100::getRevisionAPIMX

statusBackupMX100

構文

```
int statusBackupMX100(DAQMX100 daqmx100);
```

宣言

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);
```

引数

daqmx100　　機器記述子を指定します。

説明

保持している現在のステータスデータから，バックアップの指定を有効無効値で取得します。

・ 存在しない場合，「無効値」を返します。

戻り値

有効無効値を返します。

参照

```
CDAQMX100::getClassMXItemConfig
CDAQMXItemConfig::getClassMXStatus
CDAQMXStatus::isBackup
```

statusCFMX100

構文

```
int statusCFMX100(DAQMX100 daqmx100);
```

宣言

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);
```

引数

daqmx100　　機器記述子を指定します。

説明

保持している現在のステータスデータから，CFステータス種類を取得します。

・ 存在しない場合，「全オフ」を返します。

戻り値

CFステータス種類を返します。

参照

```
CDAQMX100::getClassMXItemConfig
CDAQMXItemConfig::getClassMXStatus
CDAQMXStatus::getCFStatus
```

statusCFRemainMX100

構文

```
int statusCFRemainMX100(DAQMX100 daqmx100);
```

宣言

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);
```

引数

daqmx100　　機器記述子を指定します。

説明

保持している現在のステータスデータから，CFの残容量を取得します。

- 単位はKBです。
- 存在しない場合，0を返します。

戻り値

残容量を返します。

参照

```
CDAQMX100::getClassMXItemConfig
CDAQMXItemConfig::getClassMXStatus
CDAQMXStatus::getCFRemain
```

statusCFSizeMX100

構文

```
int statusCFSizeMX100(DAQMX100 daqmx100);
```

宣言

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);
```

引数

daqmx100　　機器記述子を指定します。

説明

保持している現在のステータスデータから，CFのサイズを取得します。

・ 単位はKBです。

・ 存在しない場合，0を返します。

戻り値

サイズを返します。

参照

```
CDAQMX100::getClassMXItemConfig
CDAQMXItemConfig::getClassMXStatus
CDAQMXStatus::getCFSize
```

statusDayMX100

構文

```
int statusDayMX100(DAQMX100 daqmx100);
```

宣言

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);
```

引数

daqmx100　　機器記述子を指定します。

説明

保持している現在のステータスデータから，日を取得します。

- 基準日時からの秒数を変換して返します。
- 日は1から31の数値です。
- 存在しない場合，0を返します。

戻り値

日の値を返します。

参照

```
statusTimeMX100
CDAQMXDateTime::toLocalDateTime
```

statusFIFOIntervalMX100

構文

```
int statusFIFOIntervalMX100(DAQMX100 daqmx100, int fifoNo);
```

宣言

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);
```

引数

daqmx100	機器記述子を指定します。
fifoNo	FIFO番号を指定します。

説明

保持している現在のステータスデータから，指定されたFIFO番号の周期種類を取得します。

・ 存在しない場合，0を返します。

戻り値

周期種類を返します。

参照

```
CDAQMX100::getClassMXItemConfig
CDAQMXItemConfig::getClassMXStatus
CDAQMXStatus::getInterval
```

statusFIFOMX100

構文

```
int statusFIFOMX100(DAQMX100 daqmx100, int fifoNo);
```

宣言

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);
```

引数

daqmx100	機器記述子を指定します。
fifoNo	FIFO番号を指定します。

説明

保持している現在のステータスデータから，指定されたFIFO番号のFIFOステータス値を取得します。

・ 存在しない場合，「不明」を返します。

戻り値

FIFOステータス値を返します。

参照

```
CDAQMX100::getClassMXItemConfig
CDAQMXItemConfig::getClassMXStatus
CDAQMXStatus::getFIFOStatus
```

statusFIFONumMX100

構文

```
int statusFIFONumMX100(DAQMX100 daqmx100);
```

宣言

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);
```

引数

daqmx100　　機器記述子を指定します。

説明

保持している現在のステータスデータから，FIFOの有効個数を取得します。

・ 存在しない場合，0を返します。

戻り値

FIFOの有効個数を返します。

参照

```
CDAQMX100::getClassMXItemConfig
CDAQMXItemConfig::getClassMXStatus
CDAQMXStatus::getFIFONum
```

statusHourMX100

構文

```
int statusHourMX100(DAQMX100 daqmx100);
```

宣言

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);
```

引数

daqmx100　　機器記述子を指定します。

説明

保持している現在のステータスデータから，時を取得します。

- 基準日時からの秒数を変換して返します。
- 時は0から23の数値です。
- 存在しない場合，0を返します。

戻り値

時の値を返します。

参照

```
statusTimeMX100
CDAQMXDateTime::toLocalDateTime
```

statusMilliSecMX100

構文

```
int statusMilliSecMX100(DAQMX100 daqmx100);
```

宣言

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);
```

引数

daqmx100　　機器記述子を指定します。

説明

保持している現在のステータスデータから，ミリ秒を取得します。

・ 存在しない場合，0を返します。

戻り値

ミリ秒の値を返します。

参照

```
CDAQMX100::getClassMXItemConfig
CDAQMXItemConfig::getClassMXStatus
CDAQMXStatus::getMilliSecond
```

statusMinuteMX100

構文

```
int statusMinuteMX100(DAQMX100 daqmx100);
```

宣言

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);
```

引数

daqmx100　　機器記述子を指定します。

説明

保持している現在のステータスデータから，分を取得します。

- 基準日時からの秒数を変換して返します。
- 分は0から59の数値です。
- 存在しない場合，0を返します。

戻り値

分の値を返します。

参照

`statusTimeMX100`
`CDAQMXDateTime::toLocalDateTime`

statusMonthMX100

構文

```
int statusMonthMX100(DAQMX100 daqmx100);
```

宣言

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);
```

引数

daqmx100　　機器記述子を指定します。

説明

保持している現在のステータスデータから，月を取得します。

- 基準日時からの秒数を変換して返します。
- 月は1から12の数値です。
- 存在しない場合，0を返します。

戻り値

月の値を返します。

参照

```
statusTimeMX100
CDAQMXDateTime::toLocalDateTime
```

statusSecondMX100

構文

```
int statusSecondMX100(DAQMX100 daqmx100);
```

宣言

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);
```

引数

daqmx100　　機器記述子を指定します。

説明

保持している現在のステータスデータから，秒を取得します。

- 基準日時からの秒数を変換して返します。
- 秒は0から59の数値です。
- 存在しない場合，0を返します。

戻り値

秒の値を返します。

参照

```
statusTimeMX100
CDAQMXDateTime::toLocalDateTime
```

statusTimeMX100

構文

```
int statusTimeMX100(DAQMX100 daqmx100);
```

宣言

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);
```

引数

daqmx100　　機器記述子を指定します。

説明

保持している現在のステータスデータから，秒数を取得します。

・ ここで，秒数は基準日時(1970年1月1日)からの時間です。

・ 存在しない場合，0を返します。

戻り値

秒数を返します。

参照

```
CDAQMX100::getClassMXItemConfig
CDAQMXItemConfig::getClassMXStatus
CDAQMXStatus::getTime
```

statusUnitMX100

構文

```
int statusUnitMX100(DAQMX100 daqmx100);
```

宣言

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);
```

引数

daqmx100　　機器記述子を指定します。

説明

保持している現在のステータスデータから，ユニットステータス値を取得します。

・ 存在しない場合，「不明」を返します。

戻り値

ユニットステータス値を返します。

参照

```
CDAQMX100::getClassMXItemConfig
CDAQMXItemConfig::getClassMXStatus
CDAQMXStatus::getUnitStatus
```

statusYearMX100

構文

```
int statusYearMX100(DAQMX100 daqmx100);
```

宣言

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);
```

引数

daqmx100　　機器記述子を指定します。

説明

保持している現在のステータスデータから，年を取得します。

- 基準日時からの秒数を変換して返します。
- 年は4桁の数値です。
- 存在しない場合，0を返します。

戻り値

年の値を返します。

参照

`statusTimeMX100`
`CDAQMXDateTime::toLocalDateTime`

toAlarmNameMX100

構文

```
int toAlarmNameMX100(int iAlarmType, char * strAlarm, int
lenAlarm);
```

宣言

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);
```

引数

iAlarmType	アラーム種類を指定します。
strAlarm	文字列を格納する領域を指定します。
lenAlarm	文字列を格納する領域のバイト数を指定します。

説明

指定されたアラーム種類に対応する文字列を，指定された領域に格納します。

- 領域に格納する文字列には，終端(NULL)も含まれます。
- 実際の文字列の長さが戻り値になります。戻り値に終端は含まれません。
- 格納される文字列は，原則ascii文字列です。

戻り値

文字列の長さを返します。

参照

getAlarmNameMX100

toAOPWMValueMX100

構文

```
int toAOPWMValueMX100(double realValue, int iRangeAOPWM);
```

宣言

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);
```

引数

realValue　　実際の出力値を指定します。
iRangeAOPWM　レンジ種類を指定します。

説明

実際の出力値を，指定されたレンジ種類に従って，AO/PWMデータの出力データ値に変換します。

・ 有効なレンジ種類は，AOレンジとPWMレンジです。
・ 存在しない場合，0を返します。

戻り値

出力データ値を返します。

参照

`CDAQMXAOPWMData::toAOPWMValue`

toChannelCommentMX100

構文

```
int toChannelCommentMX100(DAQMX100 daqmx100, int chNo, char *
strComment, int lenComment);
```

宣言

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);
```

引数

daqmx100	機器記述子を指定します。
chNo	チャネル番号を指定します。
strComment	文字列を格納する領域を指定します。
lenComment	文字列を格納する領域のバイト数を指定します。

説明

保持している現在のチャネル設定データから，指定されたチャネル番号のコメントを取得します。

- 指定された格納先に文字列を格納します。
- 領域に格納する文字列には，終端(NULL)も含まれます。
- 実際の文字列の長さが戻り値になります。戻り値に終端は含まれません。
- 存在しない場合，0を返します。
- 格納される文字列は，原則ascii文字列です。

戻り値

実際の文字列の長さを返します。

参照

getChannelCommentMX100

toChannelTagMX100

構文

```
int toChannelTagMX100(DAQMX100 daqmx100, int chNo, char *
strTag, int lenTag);
```

宣言

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);
```

引数

daqmx100	機器記述子を指定します。
chNo	チャネル番号を指定します。
strTag	文字列を格納する領域を指定します。
lenTag	文字列を格納する領域のバイト数を指定します。

説明

保持している現在のチャネル設定データから，指定されたチャネル番号のタグを取得します。

- 指定された格納先に文字列を格納します。
- 領域に格納する文字列には，終端(NULL)も含まれます。
- 実際の文字列の長さが戻り値になります。戻り値に終端は含まれません。
- 存在しない場合，0を返します。
- 格納される文字列は，原則ascii文字列です。

戻り値

実際の文字列の長さを返します。

参照

```
getChannelTagMX100
```

toChannelUnitMX100

構文

```
int toChannelUnitMX100(DAQMX100 daqmx100, int chNo, char *
strUnit, int lenUnit);
```

宣言

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);
```

引数

daqmx100	機器記述子を指定します。
chNo	チャネル番号を指定します。
strUnit	文字列を格納する領域を指定します。
lenUnit	文字列を格納する領域のバイト数を指定します。

説明

保持している現在のチャネル設定データから，指定されたチャネル番号の単位名を取得します。

- 指定された格納先に文字列を格納します。
- 領域に格納する文字列には，終端(NULL)も含まれます。
- 実際の文字列の長さが戻り値になります。戻り値に終端は含まれません。
- 存在しない場合，0を返します。
- 格納される文字列は，原則ascii文字列です。

戻り値

文字列の長さを返します。

参照

`getChannelUnitMX100`

toDoubleValueMX100

構文

```
double toDoubleValueMX100(int dataValue, int point);
```

宣言

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);
```

引数

dataValue	データ値を指定します。
point	小数点位置を指定します。

説明

指定されたデータ値と小数点位置から測定値を生成します。

戻り値

測定値を倍精度浮動小数で返します。

参照

```
CDAQMXDataInfo::toDoubleValue
```

toErrorMessageMX100

構文

```
int toErrorMessageMX100(int errorCode, char * errStr, int
errLen);
```

宣言

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);
```

引数

errorCode	エラー番号を指定します。
errStr	文字列を格納する領域を指定します。
errLen	文字列を格納する領域のバイト数を指定します。

説明

エラー番号に対応するエラーメッセージ文字列を，指定された領域に格納します。

- 領域に格納する文字列には，終端(NULL)も含まれます。
- 実際の文字列の長さが戻り値になります。戻り値に終端は含まれません。
- 格納される文字列は，原則ascii文字列です。

戻り値

文字列の長さを返します。

参照

getErrorMessageMX100

toItemNameMX100

構文

```
int toItemNameMX100(int itemNo, char * strItem, int lenItem);
```

宣言

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);
```

引数

itemNo	設定項目番号を指定します。
strItem	文字列を格納する領域を指定します。
lenItem	文字列を格納する領域のバイト数を指定します。

説明

指定された設定項目番号に対応する名称の文字列を，指定された領域に格納します。

- 領域に格納する文字列には，終端(NULL)も含まれます。
- 実際の文字列の長さが戻り値になります。戻り値に終端は含まれません。
- 存在しない場合，0を返します。
- 格納される文字列は，原則ascii文字列です。

戻り値

文字列の長さを返します。

参照

```
CDAQMXItemConfig::toItemName
```

toItemNoMX100

構文

```
int toItemNoMX100(const char * strItem);
```

宣言

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);
```

引数

strItem 設定項目番号に対応する名称の文字列を指定します。

説明

指定された文字列に対応する設定項目番号を取得します。

- 大文字小文字は区別されます。
- 指定する文字列は，原則ascii文字列です。
- 存在しない場合，「不明」を返します。

戻り値

設定項目番号を返します。

参照

CDAQMXItemConfig::toItemNo

toModuleSerialMX100

構文

```
int toModuleSerialMX100(DAQMX100 daqmx100, int moduleNo, char
* strSerial, int lenSerial);
```

宣言

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);
```

引数

daqmx100	機器記述子を指定します。
moduleNo	モジュール番号を指定します。
strSerial	文字列を格納する領域を指定します。
lenSerial	文字列を格納する領域のバイト数を指定します。

説明

保持している現在のシステム構成データから，指定されたモジュール番号のシリアル番号を取得します。

- 指定された格納先に文字列を格納します。
- 領域に格納する文字列には，終端(NULL)も含まれます。
- 実際の文字列の長さが戻り値になります。戻り値に終端は含まれません。
- 存在しない場合，0を返します。
- 格納される文字列は，原則ascii文字列です。

戻り値

文字列の長さを返します。

参照

getModuleSerialMX100

toNetHostMX100

構文

```
int toNetHostMX100(DAQMX100 daqmx100, char * strHost, int
lenHost);
```

宣言

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);
```

引数

daqmx100	機器記述子を指定します。
strHost	文字列を格納する領域を指定します。
lenHost	文字列を格納する領域のバイト数を指定します。

説明

保持している現在のネットワーク情報データから，ホスト名を取得します。

- 指定された格納先に文字列を格納します。
- 領域に格納する文字列には，終端(NULL)も含まれます。
- 実際の文字列の長さが戻り値になります。戻り値に終端は含まれません。
- 存在しない場合，0を返します。
- 格納される文字列は，原則ascii文字列です。

戻り値

文字列の長さを返します。

参照

getNetHostMX100

toRealValueMX100

構文

```
double toRealValueMX100(int iAOPWMValue, int iRangeAOPWM);
```

宣言

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);
```

引数

iAOPWMValue	出力データ値を指定します。
iRangeAOPWM	レンジ種類を指定します。

説明

AO/PWMデータの出力データ値を，指定されたレンジ種類に従って，実際の出力値に変換します。

- 有効なレンジ種類は，AOレンジとPWMレンジです。
- 存在しない場合，0.0を返します。

戻り値

実際の出力値を返します。

参照

`CDAQMXAOPWMData::toRealValue`

toStringValueMX100

構文

```
int toStringValueMX100(int dataValue, int point, char *
strValue, int lenValue);
```

宣言

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);
```

引数

dataValue	データ値を指定します。
point	小数点位置を指定します。
strValue	文字列を格納する領域を指定します。
lenValue	文字列を格納する領域のバイト数を指定します。

説明

指定されたデータ値と小数点位置から測定値を生成します。

- 生成された測定値を文字列に変換して，指定された領域に格納します。
- 領域に格納する文字列には，終端(NULL)も含まれます。
- 実際の文字列の長さが戻り値になります。戻り値に終端は含まれません。
- 格納される文字列は，原則ascii文字列です。

戻り値

文字列の長さを返します。

参照

```
CDAQMXDataInfo::toStringValue
```

toStyleVersionMX100

構文

```
int toStyleVersionMX100(int style)
```

宣言

Visual Basic

```
Public Declare Function toStyleVersionMX100 Lib "DAQMX100"
(ByVal style As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function toStyleVersionMX100 Lib
"DAQMX100" (ByVal style As Integer) As Integer
```

C#

```
[DllImport("DAQMX100.dll", CharSet=CharSet.Auto,
EntryPoint="toStyleVersionMX100")]
public static extern int toStyleVersionMX100(int style);
```

引数

style　　スタイルを指定します。

説明

指定されたスタイルからスタイルバージョンを取得します。

戻り値

スタイルバージョンを返します。

参照

```
CDAQMXSysInfo::toStyleVersion
```

toUnitPartNoMX100

構文

```
int toUnitPartNoMX100(DAQMX100 daqmx100, char * strPartNo, int
lenPartNo);
```

宣言

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);
```

引数

daqmx100	機器記述子を指定します。
strPartNo	文字列を格納する領域を指定します。
lenPartNo	文字列を格納する領域のバイト数を指定します。

説明

保持している現在のシステム構成データから，パート番号を取得します。

- 指定された格納先に文字列を格納します。
- 領域に格納する文字列には，終端(NULL)も含まれます。
- 実際の文字列の長さが戻り値になります。戻り値に終端は含まれません。
- 存在しない場合，0を返します。
- 格納される文字列は，原則ascii文字列です。

戻り値

文字列の長さを返します。

参照

getUnitPartNoMX100

toUnitSerialMX100

構文

```
int toUnitSerialMX100(DAQMX100 daqmx100, char * strSerial, int
lenSerial);
```

宣言

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);
```

引数

daqmx100	機器記述子を指定します。
strSerial	文字列を格納する領域を指定します。
lenSerial	文字列を格納する領域のバイト数を指定します。

説明

保持している現在のシステム構成データから，ユニットのシリアル番号を取得します。

- 指定された格納先に文字列を格納します。
- 領域に格納する文字列には，終端(NULL)も含まれます。
- 実際の文字列の長さが戻り値になります。戻り値に終端は含まれません。
- 存在しない場合，0を返します。
- 格納される文字列は，原則ascii文字列です。

戻り値

文字列の長さを返します。

参照

getUnitSerialMX100

unitCFWriteModeMX100

構文

```
int unitCFWriteModeMX100(DAQMX100 daqmx100);
```

宣言

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);
```

引数

daqmx100　　機器記述子を指定します。

説明

保持している現在のシステム構成データから，CF書き込み種類を取得します。

・ 存在しない場合，「上書きなし」を返します。

戻り値

CF書き込み種類を返します。

参照

```
CDAQMX100::getClassMXItemConfig
CDAQMXItemConfig::getClassMXSysInfo
CDAQMXSysInfo::getCFWriteMode
```

unitFrequencyMX100

構文

```
int unitFrequencyMX100(DAQMX100 daqmx100);
```

宣言

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);
```

引数

daqmx100　　機器記述子を指定します。

説明

保持している現在のシステム構成データから，電源周波数を取得します。

・ 存在しない場合，0を返します。

戻り値

電源周波数を返します。

参照

```
CDAQMX100::getClassMXItemConfig
CDAQMXItemConfig::getClassMXSysInfo
CDAQMXSysInfo::getFrequency
```

unitMACMX100

構文

```
int unitMACMX100(DAQMX100 daqmx100, int index);
```

宣言

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);
```

引数

daqmx100	機器記述子を指定します。
index	バイト位置を指定します。

説明

保持している現在のシステム構成データから，MACアドレスをバイト位置で取得します。

- 指定されたバイト位置のバイト値を返します。
- バイト位置は，個数値の「MACアドレスの要素数」内をインデックス値(0から)で指定します。
- 存在しない場合，0を返します。

戻り値

バイト値を返します。

参照

```
CDAQMX100::getClassMXItemConfig
CDAQMXItemConfig::getClassMXSysInfo
CDAQMXSysInfo::getMAC
```

unitNoMX100

構文

```
int unitNoMX100(DAQMX100 daqmx100);
```

宣言

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);
```

引数

daqmx100　　機器記述子を指定します。

説明

保持している現在のシステム構成データから，ユニット番号を取得します。

・ 存在しない場合，0を返します。

戻り値

ユニット番号を返します。

参照

```
CDAQMX100::getClassMXItemConfig
CDAQMXItemConfig::getClassMXSysInfo
CDAQMXSysInfo::getUnitNo
```

unitOptionMX100

構文

```
int unitOptionMX100(DAQMX100 daqmx100);
```

宣言

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);
```

引数

daqmx100　　機器記述子を指定します。

説明

保持している現在のシステム構成データから，オプションを取得します。

・ 存在しない場合，「オプションなし」を返します。

戻り値

オプションを返します。

参照

```
CDAQMX100::getClassMXItemConfig
CDAQMXItemConfig::getClassMXSysInfo
CDAQMXSysInfo::getOption
```

unitStyleMX100

構文

```
int unitStyleMX100(DAQMX100 daqmx100);
```

宣言

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);
```

引数

daqmx100　　機器記述子を指定します。

説明

保持している現在のシステム構成データから，スタイルを取得します。

・ 存在しない場合，0を返します。

戻り値

スタイルを返します。

参照

```
CDAQMX100::getClassMXItemConfig
CDAQMXItemConfig::getClassMXSysInfo
CDAQMXSysInfo::getStyle
```

unitTempMX100

構文

```
int unitTempMX100(DAQMX100 daqmx100);
```

宣言

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);
```

引数

daqmx100　　機器記述子を指定します。

説明

保持している現在のシステム構成データから，温度単位種類を取得します。

・ 存在しない場合，「℃」を返します。

戻り値

温度単位種類を返します。

参照

```
CDAQMX100::getClassMXItemConfig
CDAQMXItemConfig::getClassMXSysInfo
CDAQMXSysInfo::getTempUnit
```

unitTypeMX100

構文

```
int unitTypeMX100(DAQMX100 daqmx100);
```

宣言

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);
```

引数

daqmx100　　機器記述子を指定します。

説明

保持している現在のシステム構成データから，ユニット種類を取得します。

・ 存在しない場合，「不明」を返します。

戻り値

ユニット種類を返します。

参照

```
CDAQMX100::getClassMXItemConfig
CDAQMXItemConfig::getClassMXSysInfo
CDAQMXSysInfo::getUnitType
```

userAOPWMValidMX100

構文

```
int userAOPWMValidMX100(DAQMX100 daqmx100, int idAOPWM, int
aopwmNo);
```

宣言

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);
```

引数

daqmx100	機器記述子を指定します。
idAOPWM	AO/PWMデータ識別子を指定します。
aopwmNo	AO/PWMデータ番号を指定します。

説明

指定されたAO/PWMデータ識別子のAO/PWMデータから，指定されたAO/PWMデータ番号の有効/無効を有効無効値で取得します。

・ 存在しない場合，「無効値」を返します。

戻り値

有効無効値を返します。

参照

```
CDAQMX100::getClassMXAOPWMList
CDAQMXAOPWMData::getAOPWMValid
CDAQMXAOPWMList::getClassMXAOPWMData
```

userAOPWMValueMX100

構文

```
int userAOPWMValueMX100(DAQMX100 daqmx100, int idAOPWM, int
aopwmNo);
```

宣言

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);
```

引数

daqmx100	機器記述子を指定します。
idAOPWM	AO/PWMデータ識別子を指定します。
aopwmNo	AO/PWMデータ番号を指定します。

説明

指定されたAO/PWMデータ識別子のAO/PWMデータから，指定されたAO/PWMデータ番号の出力データ値を取得します。

・ 存在しない場合，0を返します。

戻り値

出力データ値を返します。

参照

```
CDAQMX100::getClassMXAOPWMList
CDAQMXAOPWMData::getAOPWMValue
CDAQMXAOPWMList::getClassMXAOPWMData
```

userBalanceValidMX100

構文

```
int userBalanceValidMX100(DAQMX100 daqmx100, int idBalance,
int balanceNo);
```

宣言

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);
```

引数

daqmx100	機器記述子を指定します。
idBalance	初期バランスデータ識別子を指定します。
balanceNo	初期バランスデータ番号を指定します。

説明

指定された初期バランスデータ識別子の初期バランスデータから，指定された初期バランスデータ番号の有効/無効を有効無効値で取得します。

・ 存在しない場合，「無効値」を返します。

戻り値

有効無効値を返します。

参照

```
CDAQMX100::getClassMXBalanceList
CDAQMXBalanceData::getBalanceValid
CDAQMXBalanceList::getClassMXBalanceData
```

userBalanceValueMX100

構文

```
int userBalanceValueMX100(DAQMX100 daqmx100, int idBalance,
int balanceNo);
```

宣言

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);
```

引数

daqmx100	機器記述子を指定します。
idBalance	初期バランスデータ識別子を指定します。
balanceNo	初期バランスデータ番号を指定します。

説明

指定された初期バランスデータ識別子の初期バランスデータから，指定された初期バランスデータ番号の初期バランス値を取得します。

・ 存在しない場合，0を返します。

戻り値

初期バランス値を返します。

参照

```
CDAQMX100::getClassMXBalanceList
CDAQMXBalanceData::getBalanceValue
CDAQMXBalanceList::getClassMXBalanceData
```

userDoubleAOPWMValueMX100

構文

```
double userDoubleAOPWMValueMX100(DAQMX100 daqmx100, int
idAOPWM, int aopwmNo);
```

宣言

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);
```

引数

daqmx100	機器記述子を指定します。
idAOPWM	AO/PWMデータ識別子を指定します。
aopwmNo	AO/PWMデータ番号を指定します。

説明

指定されたAO/PWMデータ識別子のAO/PWMデータから，指定されたAO/PWMデータ番号の出力データ値を実際の出力値で取得します。

・ 存在しない場合，0.0を返します。

戻り値

実際の出力値を返します。

参照

`CDAQMX100::userDoubleAOPWMValue`

userDOValidMX100

構文

```
int userDOValidMX100(DAQMX100a daqmx100, int idDO, int doNo);
```

宣言

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);
```

引数

daqmx100	機器記述子を指定します。
idDO	DOデータ識別子を指定します。
doNo	DOデータ番号を指定します。

説明

指定されたDOデータ識別子のDOデータから，指定されたDOデータ番号の有効/無効を有効無効値で取得します。

・存在しない場合，「無効値」を返します。

戻り値

有効無効値を返します。

参照

```
CDAQMX100::getClassMXDOList
CDAQMXDOData::getDOValid
CDAQMXDOList::getClassMXDOData
```

userDOValueMX100

構文

```
int userDOValueMX100(DAQMX100 daqmx100, int idDO, int doNo);
```

宣言

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);
```

引数

daqmx100	機器記述子を指定します。
idDO	DOデータ識別子を指定します。
doNo	DOデータ番号を指定します。

説明

指定されたDOデータ識別子のDOデータから，指定されたDOデータ番号のON/OFFを有効無効値で取得します。

・存在しない場合，「無効値」を返します。

戻り値

有効無効値を返します。

参照

```
CDAQMX100::getClassMXDOList
CDAQMXDOData::getDOONOFF
CDAQMXDOList::getClassMXDOData
```

userTransmitMX100

構文

```
int userTransmitMX100(DAQMX100 daqmx100, int idTrans, int
aopwmNo);
```

宣言

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);
```

引数

daqmx100	機器記述子を指定します。
idTrans	伝送出力データ識別子を指定します。
aopwmNo	AO/PWMデータ番号を指定します。

説明

指定された伝送出力データ識別子の伝送出力データから，指定されたAO/PWMデータ番号の伝送状態を取得します。

・ 存在しない場合，「指定なし(不明)」を返します。

戻り値

伝送状態を返します。

参照

```
CDAQMX100::getClassMXTransmitList
CDAQMXTransmit::getTransmit
CDAQMXTransmitList::getClassMXTransmit
```

versionAPIMX100

構文

```
const int versionAPIMX100(void);
```

宣言

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();
```

説明

本APIのバージョン番号を取得します。

戻り値

バージョン番号を返します。

参照

`CDAQMX100::getVersionAPI`

18.1 MX100の定数の概要

本拡張APIでは，以下の種類の定数を用意しています。

下記のデータ型が装備されています。Visual C/Visual C++では，6章の定数を継承します。また，拡張API用に定数値とレンジ種類の定数が追加されています。18.2節を参照してください。Visual Basic，Visual Basic.NET，C#の定数は18.2節に記載します。

種類	説明	ページ
個数値	モジュール数など	6-3，18-4
最大値	タグ文字列最大長など	6-3，18-4
定数値	瞬時値指定用データ番号など	6-4,18-3,18-5
有効無効値	有効(ON)設定，無効(OFF)設定	6-4，18-5
データステータス値	測定データの状態	6-4，18-5
アラーム種類	上限アラームなど	6-5，18-6
チャネル種類	ユニバーサル入力，ディジタル入力など	6-5，18-6
スケール種類	スケールなしまたは線形スケール	6-6，18-7
モジュール種類	ユニバーサル入力4CHなど	6-6，18-7
チャネル数	4または10	6-6，18-7
周期種類	10ms～60000ms	6-7，18-8
フィルタ時定数	入力フィルタ時定数	6-7，18-8
RJC種類	内蔵RJCまたは外部RJC	6-7，18-8
バーンアウト種類	Off/Up/Down	6-7，18-8
ユニット種類	MX100	6-7，18-8
端子種類	ねじ端子または押し締め端子	6-8，18-9
A/D積分時間種類	自動，50Hz，または60Hz	6-8，18-9
温度単位種類	℃	6-8，18-9
CF書き込み種類	CFへのデータ書き込み方式	6-8，18-9
CFステータス種類	CFの状態	6-8，18-9
ユニットステータス値	ユニットの状態	6-8，18-9
FIFOステータス値	FIFOの状態	6-9，18-10
表示形式値	7セグメントLEDの表示形式	6-9，18-10
出力種類	出力レンジの種類	6-9，18-10
選択値	出力値の選択	6-9，18-10
伝送状態	伝送出力の状態	6-9，18-10
初期バランス結果	初期バランスの実行結果	6-9，18-10
オプション	オプションの有無	6-10，18-11

種類	説明	ページ
レンジ種類		
参照レンジ	チャネル間差演算チャネルの測定レンジとして基準チャネルの測定レンジを参照	6-10，18-11
スキップ	未使用	18-3，18-11
直流電圧レンジ	20mVレンジなど	6-10，18-12
熱電対レンジ	Type Rなど	6-11，18-12
測温抵抗体(1mA)レンジ	Pt100など	6-12，18-13
測温抵抗体(2mA)レンジ	Pt100など	6-14，18-15
測温抵抗体(その他)レンジ	Pt500，Pt1000	6-15，18-16
抵抗レンジ	20Ω，200Ω，または2kΩ(0.25mA)	6-15，18-16
ディジタル入力(DI)レンジ	Levelまたは接点入力	18-3，18-11
ディジタル入力(DI)詳細レンジ	「ユニバーサル入力4CHモジュールの接点入力」など詳細レンジ	6-16，18-17
ひずみレンジ	2000μひずみ，20000μひずみ，または200000μひずみ	6-16，18-17
AOレンジ	V出力，またはmA出力	6-16，18-17
PWMレンジ	PWM出力分解能　1msまたは10ms	6-16，18-17
通信レンジ	CAN Bus入力	6-16，18-18
パルスレンジ	パルス入力	6-16，18-18

18.2 MX100の定数

定数のニーモニックと意味を説明しています。MX100の機能の詳細については，それぞれのユーザーズマニュアルを参照してください。

Visual C/Visual C++の定数

Visual C/Visual C++では，6章の定数を継承しています。また，以下の定数を追加しています。

定数値

ニーモニック	内容
DAQMX_LIST_ALL	全データ識別子指定。
DAQMX_LIST_CURRENT	コピー時のカレントデータ指定。

6.2節の定数値も参照してください。

レンジ種類

本拡張APIで特別に定義されたレンジを既存のレンジと区別するためのビットが定義されています。論理演算で区別処理が可能です。

ニーモニック	内容
DAQMX_RANGETYPE_DI	デジタル入力の特別レンジ種類
DAQMX_RANGETYPE_SKIP	その他の特別レンジ種類

6.2節のレンジ種類を参照してください。

ディジタル入力(DI)レンジ

デジタル入力レンジはデジタル入力の詳細レンジを使用します。モジュールを意識しないで指定する場合は，以下を使用します。

ニーモニック	内容
DAQMX_RANGE_DI_LEVEL	0：2.4V未満，1：2.4V以上
DAQMX_RANGE_DI_CONTACT	0：open, 1：close

スキップ

特別にレンジ設定に以下の定義を指定することができます。

ニーモニック	内容
DAQMX_RANGE_SKIP	スキップ(未使用)

Visual Basic，Visual Basic.NET，C#の定数

定数のニーモニックと意味を説明しています。MX100の機能の詳細については，それぞれのユーザーズマニュアルを参照してください。
C#では，DAQMX100クラスの定数データになります。各定数の前に「DAQMX100.」をつけてご使用ください(例　DAQMX100.DAQMX_COMMPORT)。

個数値

ニーモニック	内容
DAQMX100_NUMMODULE	モジュール数です。
DAQMX100_NUMCHANNEL	チャネル数です。
DAQMX100_NUMDO	DOデータ数です。
DAQMX100_NUMFIFO	FIFO数です。
DAQMX100_NUMALARM	アラーム数です。R3.01から個数が「4」になりました。
DAQMX100_NUMSEGMENT	7セグメントLED数です。
DAQMX100_NUMMACADDR	MACアドレスの要素数(バイト数)です。
DAQMX100_NUMAOPWM	AO/PWMデータ個数です。
DAQMX100_NUMBALANCE	初期バランスデータ個数です。
DAQMX100_NUMOUTPUT	出力チャネルデータ個数です。

最大値

ニーモニック	内容
DAQMX100_MAXHOSTNAMELEN	ホスト名文字列最大長です。
DAQMX100_MAXUNITLEN	単位名文字列最大長です。
DAQMX100_MAXTAGLEN	タグ文字列最大長です。
DAQMX100_MAXCOMMENTLEN	コメント文字列最大長です。
DAQMX100_MAXSERIALLEN	MX100のシリアル番号文字列最大長です。
DAQMX100_MAXPARTNOLEN	パート番号(ファームウエアの部品番号)文字列最大長です。
DAQMX100_MAXDECIMALPOINT	小数点位置の最大値です。
DAQMX100_MAXDISPTIME	7セグメントLED表示時間の最大値です。
DAQMX100_MAXPULSETIME	パルス周期倍率の最大値です。

文字列の最大長は，終端(NULL)を含みません。

定数値

ニーモニック	内容
DAQMX100_INSTANTANEOUS	瞬時値取得を指定するときのデータ番号。
DAQMX100_REFCHNO_ALL	全参照チャネル番号指定。
DAQMX100_LEVELNO_ALL	全アラームレベル番号指定。
DAQMX100_DONO_ALL	全DO番号指定。
DAQMX100_SEGMENTNO_ALL	7セグメントLEDの全セグメント番号指定。
DAQMX100_CHNO_ALL	全チャネル番号指定。
DAQMX100_MODULENO_ALL	全モジュール番号指定。
DAQMX100_FIFONO_ALL	全FIFO番号指定。
DAQMX100_AOPWMNO_ALL	全AO/PWMデータ番号指定。
DAQMX100_BALANCENO_ALL	全初期バランス番号指定。
DAQMX100_OUTPUTNO_ALL	全出力データ番号指定。
DAQMX100_REFCHNO_NONE	未定義参照チャネル番号。
DAQMX100_LIST_ALL	全データ識別子指定。
DAQMX100_LIST_CURRENT	コピー時のカレントデータ指定。

有効無効値

ニーモニック	内容
DAQMX100_VALID_OFF	無効(OFF)値
DAQMX100_VALID_ON	有効(ON)値

データステータス値

ニーモニック	内容
DAQMX100_DATA_UNKNOWN	不明状態です。
DAQMX100_DATA_NORMAL	正常状態です。
DAQMX100_DATA_PLUSOVER	プラスオーバ状態です。
DAQMX100_DATA_MINUSOVER	マイナスオーバ状態です。
DAQMX100_DATA_SKIP	スキップ(未使用)状態です。
DAQMX100_DATA_ILLEGAL	不明な不正データ状態です。
DAQMX100_DATA_NODATA	データなし状態です。
DAQMX100_DATA_LACK	データ抜け状態です。
DAQMX100_DATA_INVALID	不正状態です。

アラーム種類

□はスペースです。

ニーモニック	内容	文字列
DAQMX100_ALARM_NONE	アラームなし	□□
DAQMX100_ALARM_UPPER	上限アラーム	H□
DAQMX100_ALARM_LOWER	下限アラーム	L□
DAQMX100_ALARM_UPDIFF	差上限アラーム	dH
DAQMX100_ALARM_LOWDIFF	差下限アラーム	dL

チャネル種類

ニーモニック	内容
DAQMX100_CHKIND_NONE	未使用
DAQMX100_CHKIND_AI	AI*
DAQMX100_CHKIND_AIDIFF	AI*(チャネル間差演算指定)
DAQMX100_CHKIND_AIRJC	AI*(リモートRJC適用チャネル)
DAQMX100_CHKIND_DI	DI*
DAQMX100_CHKIND_DIDIFF	DI*(チャネル間差演算指定)
DAQMX100_CHKIND_DO	DO*(アラーム出力指定)
DAQMX100_CHKIND_DOCOM	DO*(コマンドDO指定)
DAQMX100_CHKIND_DOFAIL	DO*(システムFail出力指定)
DAQMX100_CHKIND_DOERR	DO*(システムError出力指定)
DAQMX100_CHKIND_AO	AO*(伝送出力)
DAQMX100_CHKIND_AOCOM	AO*(コマンドAO)
DAQMX100_CHKIND_PWM	PWM*(伝送出力)
DAQMX100_CHKIND_PWMCOM	PWM*(コマンドPWM)
DAQMX100_CHKIND_PI	パルス入力
DAQMX100_CHKIND_PIDIFF	パルス入力(チャネル間差演算指定)
DAQMX100_CHKIND_CI	CAN Bus入力
DAQMX100_CHKIND_CIDIFF	CAN Bus入力(チャネル間差演算指定)

* AI : Analog Input，直流電圧入力，TC入力など
AO : Analog Output, アナログ出力
DI : Digital Input，ディジタル入力
DO : Digital Output，ディジタル出力
PWM : Pulse Width Modulation，PWM出力

入力チャネルの場合，レンジ設定のレンジ種類によりチャネル種類は確定します。
出力チャネルの場合，チャネル設定でチャネル種類を設定します。

スケール種類

ニーモニック	内容
DAQMX100_SCALE_NONE	スケールなし
DAQMX100_SCALE_LINER	線形スケール

モジュール種類

ニーモニック	内容
DAQMX100_MODULE_NONE	なし
DAQMX100_MODULE_MX110UNVH04	4ch高速ユニバーサル入力モジュール
DAQMX100_MODULE_MX110UNVM10	10ch中速ユニバーサル入力モジュール
DAQMX100_MODULE_MX115D05H10	10ch高速ディジタル入力モジュール
DAQMX100_MODULE_MX125MKCM10	10ch中速ディジタル出力モジュール
DAQMX100_MODULE_MX110V4RM06	6ch中速4線式RTD抵抗入力モジュール
DAQMX100_MODULE_MX112NDIM04	4chひずみ入力モジュール(NDIS)
DAQMX100_MODULE_MX112B35M04	4chひずみ入力モジュール(350Ω)
DAQMX100_MODULE_MX112B12M04	4chひずみ入力モジュール(20Ω)
DAQMX100_MODULE_MX115D24H10	10ch高速ディジタル入力モジュール(DC24V)
DAQMX100_MODULE_MX120VAOM08	8ch中速アナログ出力モジュール
DAQMX100_MODULE_MX120PWMM08	8ch中速PWM出力モジュール
DAQMX100_MODULE_HIDDEN	複数スロット幅を使用するモジュールが，モジュールを装着したスロット以外に占有するスロット(仮想モジュール部)
DAQMX100_MODULE_MX114PLSM10	10chパルス入力モジュール
DAQMX100_MODULE_MX110VTDL30	30ch中速DCV/TC/DI入力モジュール
DAQMX100_MODULE_MX118CANM10	CAN Busモジュール 10ch*
DAQMX100_MODULE_MX118CANM20	CAN Busモジュール 20ch*
DAQMX100_MODULE_MX118CANM30	CAN Busモジュール 30ch*
DAQMX100_MODULE_MX118CANSUB	CAN Busモジュールが，モジュールを装着したスロット以外に占有するスロット(仮想CAN Busモジュール部)
DAQMX100_MODULE_MX118CANMERR	CAN Busモジュールの位置エラー
DAQMX100_MODULE_MX118CANSERR	CAN Busモジュールが，モジュールを装着したスロット以外に占有するスロット(仮想CAN Busモジュール部)のエラー。

チャネル数

ニーモニック	内容
DAQMX100_CHNUM_0	0
DAQMX100_CHNUM_4	4
DAQMX100_CHNUM_6	6
DAQMX100_CHNUM_8	8
DAQMX100_CHNUM_10	10
DAQMX100_CHNUM_30	30

周期種類

ニーモニック	内容
DAQMX100_INTERVAL_10	10msec
DAQMX100_INTERVAL_50	50msec
DAQMX100_INTERVAL_100	100msec
DAQMX100_INTERVAL_200	200msec
DAQMX100_INTERVAL_500	500msec
DAQMX100_INTERVAL_1000	1000msec
DAQMX100_INTERVAL_2000	2000msec
DAQMX100_INTERVAL_5000	5000msec
DAQMX100_INTERVAL_10000	10000msec
DAQMX100_INTERVAL_20000	20000msec
DAQMX100_INTERVAL_30000	30000msec
DAQMX100_INTERVAL_60000	60000msec

フィルタ係数

ニーモニック	内容
DAQMX100_FILTER_0	係数0
DAQMX100_FILTER_5	係数5
DAQMX100_FILTER_10	係数10
DAQMX100_FILTER_20	係数20
DAQMX100_FILTER_25	係数25
DAQMX100_FILTER_40	係数40
DAQMX100_FILTER_50	係数50
DAQMX100_FILTER_100	係数100

RJC種類

ニーモニック	内容
DAQMX100_RJC_INTERNAL	MX100のRJC機能
DAQMX100_RJC_EXTERNAL	外部のRJC機能

バーンアウト種類

ニーモニック	内容
DAQMX100_BURNOUT_OFF	バーンアウト検出機能なし
DAQMX100_BURNOUT_UP	バーンアウト検出時，＋レンジオーバの表示
DAQMX100_BURNOUT_DOWN	バーンアウト検出時，－レンジオーバの表示

ユニット種類論理

OR演算で合成されます。

ニーモニック	内容
DAQMX100_UNITTYPE_NONE	不明
DAQMX100_UNITTYPE_MX100	MX100

端子種類

ニーモニック	内容
DAQMX100_TERMINAL_SCREW	ねじ端子
DAQMX100_TERMINAL_CLAMP	押し締め端子
DAQMX100_TERMINAL_NDIS	NDIS
DAQMX100_TERMINAL_DSUB	D-SUB 9ピン

A/D積分時間種類

ニーモニック	内容
DAQMX100_INTEGRAL_AUTO	自動(50Hz/60HzをMX100が自動設定)
DAQMX100_INTEGRAL_50HZ	50Hz
DAQMX100_INTEGRAL_60HZ	60Hz

温度単位種類

ニーモニック	内容
DAQMX100_TEMPUNIT_C	℃

CF書き込み種類

ニーモニック	内容
DAQMX100_CFWRITEMODE_ONCE	上書きなし(空き容量がなくなると書き込みを停止します)
DAQMX100_CFWRITEMODE_FIFO	繰り返し(古いデータから順に上書きされます)

CFステータス種類

論理OR演算で合成されます。

ニーモニック	内容
DAQMX100_CFSTATUS_NONE	全OFF
DAQMX100_CFSTATUS_EXIST	存在の有無
DAQMX100_CFSTATUS_USE	CFカードを使用可能です。
DAQMX100_CFSTATUS_FORMAT	CFカードをフォーマット中です。

ユニットステータス値

ニーモニック	内容
DAQMX100_UNITSTAT_NONE	不明
DAQMX100_UNITSTAT_INIT	初期化中
DAQMX100_UNITSTAT_STOP	停止中
DAQMX100_UNITSTAT_RUN	測定中
DAQMX100_UNITSTAT_BACKUP	測定中(バックアップ中)

FIFOステータス値

ニーモニック	内容
DAQMX100_FIFOSTAT_NONE	不明
DAQMX100_FIFOSTAT_INIT	初期化中
DAQMX100_FIFOSTAT_STOP	停止中
DAQMX100_FIFOSTAT_RUN	測定中
DAQMX100_FIFOSTAT_BACKUP	測定中(バックアップ中)

表示形式値

ニーモニック	内容
DAQMX100_DISPTYPE_NONE	未定義
DAQMX100_DISPTYPE_ON	点灯
DAQMX100_DISPTYPE_BLINK	点滅表示

出力種類

ニーモニック	内容	設定範囲
DAQMX100_OUTPUT_NONE	出力なし	
DAQMX100_OUTPUT_AO_10V	V出力	−11.000～11.000V
DAQMX100_OUTPUT_AO_20MA	mA出力	0～22.000 mA
DAQMX100_OUTPUT_PWM_1MS	PWM出力 分解能 1ms	0～100.000 %
DAQMX100_OUTPUT_PWM_10MS	PWM出力 分解能 10ms	0～100.000 %

出力のレンジ種類と対応します。

選択値

ニーモニック	内容
DAQMX100_CHOICE_PREV	前回値
DAQMX100_CHOICE_PRESET	指定値

伝送状態

ニーモニック	内容
DAQMX100_TRANSMIT_NONE	指定なし(不明)
DAQMX100_TRANSMIT_RUN	出力開始(出力中)
DAQMX100_TRANSMIT_STOP	出力停止

初期バランス結果

ニーモニック	内容
DAQMX100_BALANCE_NONE	指定なし
DAQMX100_BALANCE_DONE	正常終了
DAQMX100_BALANCE_NG	範囲外
DAQMX100_BALANCE_ERROR	エラー

オプション

ニーモニック	内容
DAQMX100_OPTION_NONE	オプションなし
DAQMX100_OPTION_DS	Dual Save(/DSオプション)

レンジ種類

本拡張APIで特別に定義されたレンジを既存のレンジと区別するためのビットが定義されています。論理演算で区別処理が可能です。

ニーモニック	内容
DAQMX100_RANGETYPE_DI	デジタル入力の特別レンジ種類
DAQMX100_RANGETYPE_SKIP	その他の特別レンジ種類

参照レンジ

差演算チャネルの測定レンジとしてこの定数を指定すると，差演算チャネルの測定レンジが，基準チャネルの測定レンジと同じレンジに設定されます。
参照レンジは，差演算などで，参照する基準チャネルと同じレンジに設定したいときの指定に用います。

ニーモニック	内容
DAQMX100_RANGE_REFERENCE	基準チャネルの測定レンジ

ディジタル入力(DI)レンジ

デジタル入力レンジはデジタル入力の詳細レンジを使用します。モジュールを意識しないで指定する場合は，以下を使用します。

ニーモニック	内容
DAQMX_RANGE_DI_LEVEL	0：2.4V未満，1：2.4V以上
DAQMX_RANGE_DI_CONTACT	0：open, 1：close

スキップ

特別にレンジ設定に以下の定数を指定することができます。

ニーモニック	内容
DAQMX100_RANGE_SKIP	スキップ(未使用)

直流電圧レンジ

ニーモニック	内容	設定範囲
DAQMX100_RANGE_VOLT_20MV	20mV	−20.000～20.000 mV
DAQMX100_RANGE_VOLT_60MV	60mV	−60.00～60.00 mV
DAQMX100_RANGE_VOLT_200MV	200mV	−200.00～200.00 mV
DAQMX100_RANGE_VOLT_2V	2V	−2.0000～2.0000 V
DAQMX100_RANGE_VOLT_6V	6V	−6.000～6.000 V
DAQMX100_RANGE_VOLT_20V	20V	−20.000～20.000 V
DAQMX100_RANGE_VOLT_100V	100V	−100.00～100.00 V
DAQMX100_RANGE_VOLT_60MVH	60mV：高分解能	0.000～60.000 mV
DAQMX100_RANGE_VOLT_1V	1V	−10000～1.0000 V
DAQMX100_RANGE_VOLT_6VH	6V：高分解能	0.0000～6.0000 V

熱電対レンジ

ニーモニック	内容	設定範囲
DAQMX100_RANGE_TC_R	R	0.0～1760.0°C
DAQMX100_RANGE_TC_S	S	0.0～1760.0°C
DAQMX100_RANGE_TC_B	B	0.0～1820.0°C
DAQMX100_RANGE_TC_K	K	−200.0～1370.0°C
DAQMX100_RANGE_TC_E	E	−200.0～800.0°C
DAQMX100_RANGE_TC_J	J	−200.0～1100.0°C
DAQMX100_RANGE_TC_T	T	−200.0～400.0°C
DAQMX100_RANGE_TC_N	N	0.0～1300.0°C
DAQMX100_RANGE_TC_W	W	0.0～2315.0°C
DAQMX100_RANGE_TC_L	L	−200.0～900.0°C
DAQMX100_RANGE_TC_U	U	−200.0～400.0°C
DAQMX100_RANGE_TC_KP	KpAu7Fe	0.0～300.0K
DAQMX100_RANGE_TC_PL	PLATINEL	0.0～1400.0°C
DAQMX100_RANGE_TC_PR	PR40−20	0.0～1900.0°C
DAQMX100_RANGE_TC_NNM	NiNiMo	0.0～1310.0°C
DAQMX100_RANGE_TC_WR	WRe3−25	0.0～2400.0°C
DAQMX100_RANGE_TC_WWR	W/WRe26	0.0～2400.0°C
DAQMX100_RANGE_TC_AWG	Type−N (AWG14)	0.0～1300.0°C
DAQMX100_RANGE_TC_XK	XK	−200.0 ～ 600.0°C

測温抵抗体(1mA)レンジ

ニーモニック	内容	設定範囲
DAQMX100_RANGE_RTD_1MAPT	Pt100	−200.0～600.0°C
DAQMX100_RANGE_RTD_1MAJPT	JPt100	−200.0～550.0°C
DAQMX100_RANGE_RTD_1MAPTH	Pt100:高分解能	−140.00～150.00°C
DAQMX100_RANGE_RTD_1MAJPTH	JPt100:高分解能	−140.00～150.00°C
DAQMX100_RANGE_RTD_1MANIS	Ni100:SAMA	−200.0～250.0°C
DAQMX100_RANGE_RTD_1MANID	Ni100:DIN	−60.0～180.0°C
DAQMX100_RANGE_RTD_1MANI120	Ni120	−70.0～200.0°C
DAQMX100_RANGE_RTD_1MAPT50	Pt50	−200.0～550.0°C
DAQMX100_RANGE_RTD_1MACU10GE	Cu10:GE	−200.0～300.0°C
DAQMX100_RANGE_RTD_1MACU10LN	Cu10:L&N	−200.0～300.0°C
DAQMX100_RANGE_RTD_1MACU10WEED	Cu10:WEED	−200.0～300.0°C
DAQMX100_RANGE_RTD_1MACU10BAILEY	Cu10:BAILEY	−200.0～300.0°C
DAQMX100_RANGE_RTD_1MAJ263B	J263*B	0.0～300.0K
DAQMX100_RANGE_RTD_1MACU10A392	Cu10 at 20°C α=0.00392	−200.0 300.0°C
DAQMX100_RANGE_RTD_1MACU10A393	Cu10 at 20°C α=0.00393	−200.0～300.0°C
DAQMX100_RANGE_RTD_1MACU25	Cu25 at 0°C α=0.00425	−200.0～300.0°C
DAQMX100_RANGE_RTD_1MACU53	Cu53 at 0°C α=0.00426035	−50.0～150.0°C
DAQMX100_RANGE_RTD_1MACU100	Cu100 at 0°C α=0.00425	−50.0～150.0°C
DAQMX100_RANGE_RTD_1MAPT25	Pt25	−200.0～550.0°C
DAQMX100_RANGE_RTD_1MACU10GEH	Cu10:GE :高分解能	−200.0～300.0°C

ニーモニック	内容	設定範囲
DAQMX100_RANGE_RTD_1MACU10LNH	Cu10:L&N :高分解能	−500.0～500.0°C
DAQMX100_RANGE_RTD_1MACU10WEEDH	Cu10:WEED :高分解能	−500.0～500.0°C
DAQMX100_RANGE_RTD_1MACU10BAILEYH	Cu10:BAILEY :高分解能	−500.0～500.0°C
DAQMX100_RANGE_RTD_1MAPTN	Pt100 :高耐ノイズ	−800.0～800.0°C
DAQMX100_RANGE_RTD_1MAJPTN	Jpt100 :高耐ノイズ	−750.0～750.0°C
DAQMX100_RANGE_RTD_1MAPTG	Pt100G	−200.0 ～ 600.0°C
DAQMX100_RANGE_RTD_1MACU100G	Cu100G	−200.0 ～ 200.0°C
DAQMX100_RANGE_RTD_1MACU50G	Cu50G	−200.0 ～ 200.0°C
DAQMX100_RANGE_RTD_1MACU10G	Cu10G	−200.0 ～ 200.0°C

測温抵抗体(2mA)レンジ

ニーモニック	内容	設定範囲
DAQMX100_RANGE_RTD_2MAPT	Pt100	−200.0～250.0°C
DAQMX100_RANGE_RTD_2MAJPT	JPt100	−200.0～250.0°C
DAQMX100_RANGE_RTD_2MAPTH	Pt100 :高分解能	−140.00～150.00°C
DAQMX100_RANGE_RTD_2MAJPTH	JPt100 :高分解能	−140.00～150.00°C
DAQMX100_RANGE_RTD_2MAPT50	Pt50	−200.0～550.0°C
DAQMX100_RANGE_RTD_2MACU10GE	CU10:GE	−200.0～300.0°C
DAQMX100_RANGE_RTD_2MACU10LN	Cu10:L&N	−200.0～300.0°C
DAQMX100_RANGE_RTD_2MACU10WEED	Cu10:WEED	−200.0～300.0°C
DAQMX100_RANGE_RTD_2MACU10BAILEY	Cu10:BAILEY	−200.0～300.0°C
DAQMX100_RANGE_RTD_2MAJ263B	J263*B	0.0～300.0K
DAQMX100_RANGE_RTD_2MACU10A392	Cu10 at 20°C α=0.00392	−200.0～300.0°C
DAQMX100_RANGE_RTD_2MACU10A393	Cu10 at 20°C α=0.00393	−200.0～300.0°C
DAQMX100_RANGE_RTD_2MACU25	Cu25 at 0°C α=0.00425	−200.0～300.0°C
DAQMX100_RANGE_RTD_2MACU53	Cu53 at 0°C α=0.00426035	−50.0～150.0°C
DAQMX100_RANGE_RTD_2MACU100	Cu100 at 0°C α=0.00425	−50.0～150.0°C
DAQMX100_RANGE_RTD_2MAPT25	Pt25	−200.0～550.0°C
DAQMX100_RANGE_RTD_2MACU10GEH	CU10:GE :高分解能	−200.0～300.0°C
DAQMX100_RANGE_RTD_2MACU10LNH	Cu10:L&N :高分解能	−200.0～300.0°C
DAQMX100_RANGE_RTD_2MACU10WEEDH	Cu10:WEED :高分解能	−200.0～300.0°C

ニーモニック	内容	設定範囲
DAQMX100_RANGE_RTD_2MACU10BAILEYH	Cu10:BAILEY :高分解能	−200.0～300.0°C
DAQMX100_RANGE_RTD_2MAPTN	Pt100 :高耐ノイズ	−200.0～250.0°C
DAQMX100_RANGE_RTD_2MAJPTN	Jpt100 :高耐ノイズ	−200.0～250.0°C
DAQMX100_RANGE_RTD_2MACU100G	Cu100G	−200.0 ～ 200.0°C
DAQMX100_RANGE_RTD_2MACU50G	Cu50G	−200.0 ～ 200.0°C
DAQMX100_RANGE_RTD_2MACU10G	Cu10G	−200.0 ～ 200.0°C

測温抵抗体(その他)のレンジ

ニーモニック	内容	設定範囲
DAQMX100_RANGE_RTD_025MAPT500	0.25mA Pt500	−200.0～600.0 ℃
DAQMX100_RANGE_RTD_025MAPT1K	0.25mA Pt1000	−200.0～600.0 ℃

抵抗レンジ

ニーモニック	内容	設定範囲
DAQMX100_RANGE_RES_20	20Ω	0～20.000
DAQMX100_RANGE_RES_200	200Ω	0～200.00
DAQMX100_RANGE_RES_2K	2kΩ (0.25mA)	0～2000.0

ディジタル入力(DI)詳細レンジ

ニーモニック	内容
DAQMX100_RANGE_DI_LEVEL_AI	ユニバーサル入力モジュールのDI/Level
DAQMX100_RANGE_DI_CONTACT_AI4	4chユニバーサル入力モジュールのDI/接点入力
DAQMX100_RANGE_DI_CONTACT_AI10	10chユニバーサル入力モジュールのDI/接点入力
DAQMX100_RANGE_DI_CONTACT_AI30	30chDCV/TC/DI入力モジュールのDI/接点入力
DAQMX100_RANGE_DI_LEVEL_DI	ディジタル入力モジュールのDI/Level
DAQMX100_RANGE_DI_CONTACT_DI	ディジタル入力モジュールのDI/接点入力
DAQMX100_RANGE_DI_LEVEL_DI5V*	DI 5V ディジタル入力モジュール(5V用)のDI/接点入力
DAQMX100_RANGE_DI_LEVEL_DI24V	DI 24V ディジタル入力モジュール(24V用)のDI/接点入力

* DAQMX100_RANGE_DI_LEVEL_DIの別名称です。 24V用と区別するために定義されています。

ひずみレンジ

ニーモニック	内容	設定範囲
DAQMX100_RANGE_STRAIN_2K	2000μSTR	−2000.0～2000.0 ±563200000
DAQMX100_RANGE_STRAIN_20K	20000μSTR	−20000～20000 ±56320000
DAQMX100_RANGE_STRAIN_200K	200000μSTR	−200000～200000 ±5632000

AOレンジ

ニーモニック	内容	設定範囲
DAQMX100_RANGE_AO_10V	V出力	−10.000～10.000V
DAQMX100_RANGE_AO_20MA	mA出力	0.000～20.000mA

PWMレンジ

ニーモニック	内容	設定範囲
DAQMX100_RANGE_PWM_1MS	PWM出力 分解能 1ms	0～100.000 %
DAQMX100_RANGE_PWM_10MS	PWM出力 分解能 10ms	0～100.000 %

通信レンジ

ニーモニック	内容	設定範囲
DAQMX100_RANGE_COM_CAN	CAN Bus	−30000 ～ 30000

パルスレンジ

ニーモニック	内容	設定範囲
DAQMX100_RANGE_PI_LEVEL	パルス/Level	0 ～ 30000
DAQMX100_RANGE_PI_CONTACT	パルス/接点入力	0 ～ 30000

18.3 M100の設定項目番号

6.3節をご覧ください。

18.4 MX100の型

DAQMX100

本機能用の機器記述子を格納するための型です。
Visual BasicではLong型で扱います。 Visual C++/Visual Cでは R3.01より前はint型，R3.01からはvoid*型で扱います。Visual Basic.NETではInteger型で扱います。C#ではint型で扱います。

コールバック型

型	説明
コールバック型	関数名に接頭辞「DLL」を付加し，大文字で記述します。 例. openMX100関数のコールバック型：DLLOPENMX100

コールバック型は，Visual Cを使用するときに，実行可能モジュール(.dll)とリンクするために使用します。

19.1 DARWINのクラス

本拡張APIは，APIに以下のクラスを追加した構成になります。

■ CDAQHandler
　・ CDAQDARWIN
　　▲ CDAQDA100
　　♦ CDAQDA100Reader
▲ CDAQDARWINDataBuffer

■ ：MX100とDARWINに共通のクラスです。
・ ：DARWIN専用のクラスです。
▲ ：DA100(拡張API)用に追加されたクラスです。
♦ ：DA100Reader用(瞬時値データ読み込み)に追加されたクラスです。

CDAQDA100

拡張API用のハンドラクラスです。

CDAQDA100Reader

瞬時値データ読み込み用のハンドラクラスです。

CDAQDARWINDataBuffer

チャネルのデータクラスです。

Note

データ種類と取得方法

DARWINから取得するデータは，種別ごとにクラス化されています。設定データは，行単位で取得するため，クラス化されていません。

19.2 機能とクラス/関数メンバの対応ーDARWINー

本拡張APIでサポートする機能と，クラスの対応を示します。

Note

本拡張APIでは，DARWINシリーズ機器の共通機能の一部を提供しています。機種別の機能，セットアップモードの設定機能，A/D校正機能は実装されていません。DARWIN通信機能のコマンドを使用して，機能を追加することができます。

表中の「コマンド」とは，DARWIN通信機能のコマンドのことです。コマンドの詳細については，通信インターフェースユーザーズマニュアルをご覧ください。

状態遷移関数と取得関数の2種類あります。
状態遷移関数はDARWIN本体を制御します。
取得関数では現在の状態の項目値を取得します。取得関数を使用した場合，保持している現在の状態のデータ値を返します(拡張APIの状態は遷移しません)。

状態遷移関数

通信機能

機能	コマンド	クラスと関数メンバ
通信接続	ー	CDAQDA100:: open
通信切断	ー	CDAQDA100:: close
行単位送信	ー	CDAQDA100:: sendLine
行単位受信	ー	CDAQDA100:: receiveLine
バイト単位受信	ー	CDAQDA100:: receiveByte
トリガ送信	(ESC T)	CDAQDA100:: sendTrigger
ステータス状態更新	(ESC S)	CDAQDA100:: updateStatus
コマンド実行	ー	CDAQDA100:: runCommand

通信機能は，通信接続とステータス状態更新を除き，保持データの状態更新は行いません。

制御機能

機能	コマンド	クラスと関数メンバ
操作モード切替	DS	CDAQDA100:: switchMode
取得コード種類 (バイナリ/ASCIIコード)切替	ー	CDAQDA100:: switchCode
再構築	RS	CDAQDA100:: reconstruct
設定値の初期化	RC	CDAQDA100:: initSetValue
アラームリセット	AR	CDAQDA100:: ackAlarm
日付時刻設定	SD	CDAQDA100:: setDateTime
演算のスタート，ストップ	EX	CDAQDA100:: switchCompute
レポートのスタート，ストップ	DR	CDAQDA100:: switchReport
セットアップモード確定	XE	CDAQDA100:: establish

原則，処理の最後に状態更新を行います。
セットアップモード確定は，状態更新を行いません。

設定(運転モード)機能

機能		コマンド	クラスと関数メンバ
レンジ	スキップ(未使用)	SR	CDAQDA100:: setRange
	直流電圧入力	SR	CDAQDA100:: setRange
	熱電対入力	SR	CDAQDA100:: setRange
	測温抵抗体入力	SR	CDAQDA100:: setRange
	接点入力(DI)	SR	CDAQDA100:: setRange
	直流電流	SR	CDAQDA100:: setRange
	ひずみ	SR	CDAQDA100:: setRange
	パルス	SR	CDAQDA100:: setRange
	パワーモニタ	SR	CDAQDA100:: setRange
	チャネル間差演算	SR	CDAQDA100:: setChDELTA
	リモートRJC	SR	CDAQDA100:: setChRRJC
スケーリングの単位		SN	CDAQDA100:: setChUnit
アラーム		SA	CDAQDA100:: setChAlarm

チャネル単位の設定になります。
設定後，状態更新を行います。

データ取得機能

機能			コマンド	クラスと関数メンバ
測定データ		測定チャネル	TS，FM	CDAQDA100:: measInstCh
(瞬時値)		演算チャネル	TS，FM	CDAQDA100:: mathInstCh
チャネル		測定チャネル	TS，LF	CDAQDA100:: measInfoCh
情報データ		演算チャネル	TS，LF	CDAQDA100:: mathInfoCh
システム構成データ			TS，CF	CDAQDA100:: updateSystemConfig
レポートステータス			TS，RF	CDAQDA100:: updateReportStatus
設定データ				
宣言	運転モード	単一指定	TS，LF	CDAQDA100:: talkOperationChData
		範囲指定	TS，LF	CDAQDA100:: talkOperationData
	セットアップモード	単一指定	TS，LF	CDAQDA100:: talkSetupChData
		範囲指定	TS，LF	CDAQDA100:: talkSetupData
	校正モード	単一指定	TS，LF	CDAQDA100:: talkCalibrationChData
		範囲指定	TS，LF	CDAQDA100:: talkCalibrationData
行単位取得			ー	CDAQDA100:: getSetDataByLine

設定データは，保持しませんので，7.2節，7.3節と同じ手順で取得します。この場合，状態更新はされません。
チャネル情報データとシステム構成データは，内部で保持されていますが，ユーザが明示的に収集することができます。
レポートステータスは，内部で保持されていますが，ユーザが明示的に収集しない限り更新されません。

取得関数

測定データ

データ名		クラスと関数メンバ
データ値		CDAQDARWINDataInfo:: getValue
データステータス値		CDAQDARWINDataInfo:: getStatus
アラーム（有無）		CDAQDARWINDataBuffer:: isDataAlarm
測定値	倍精度浮動小数	CDAQDARWINDataInfo:: getDoubleValue
	文字列	CDAQDARWINDataInfo:: getStringValue
時刻	年	CDAQDARWINDateTime:: getFullYear
	月	CDAQDARWINDateTime:: getMonth
	日	CDAQDARWINDateTime:: getDay
	時	CDAQDARWINDateTime:: getHour
	分	CDAQDARWINDateTime:: getMinute
	秒	CDAQDARWINDateTime:: getSecond
アラーム種類		CDAQDARWINDataInfo:: getAlarm

「CDAQDA100::getClassDataBuffer」から「CDAQDARWINDataBuffer:: getClassDARWINDataInfo」と「CDAQDARWINDataBuffer:: getClassDARWINDateTime」でたどって取得します。

チャネル情報

データ名	クラスと関数メンバ
小数点位置	CDAQDARWINChInfo:: getPoint
チャネルステータス	CDAQDARWINChInfo:: getChStatus
単位名	CDAQDARWINChInfo:: getUnit

「CDAQDA100::getClassDataBuffer」から「CDAQDARWINDataBuffer:: getClassDARWINChInfo」でたどって取得します。

システム構成データ

データ名		クラスと関数メンバ
測定周期		CDAQDARWINSysInfo:: getInterval
ユニット	有無	CDAQDARWINSysInfo:: isExist
モジュール	内部コード	CDAQDA100:: getModuleCode
	モジュール名	CDAQDARWINSysInfo:: getModuleName

「CDAQDA100::getClassSysInfo」からたどって取得します。

状態データ

データ名	クラスと関数メンバ
ステータスバイト	CDAQDA100:: getByte
取得コード種類 (バイナリ/ASCII コード)	CDAQDA100:: getCode
レポートステータス	CDAQDA100:: getReport

ユーティリティ

機能/データ名		クラスと関数メンバ
測定値	倍精度浮動小数に変換	CDAQDARWINDataInfo:: toDoubleValue
	文字列に変換	CDAQDARWINDataInfo:: toStringValue
アラーム	アラーム種類文字列	CDAQDARWINDataInfo:: getAlarmName
	文字列の最大長を取得	CDAQDARWINDataInfo:: getMaxLenAlarmName
本APIのバージョン番号		CDAQDA100:: getVersionAPI
本APIのリビジョン番号		CDAQDA100:: getRevisionAPI
エラー	エラーメッセージ文字列	CDAQDA100:: getErrorMessage
	エラーメッセージ文字列の最大長	CDAQDA100:: getMaxLenErrorMessage

19.3 プログラムーDARWIN/Visual C++ー

インクルードファイルのパスを追加

プロジェクトに，インクルードファイル(DAQDA100.h)のパスを追加します。追加方法は，ご使用の環境により異なります。

ソースファイルでの宣言

ソースファイルに宣言を記述します。

```
#include "DAQDA100.h"
```

Note

共通部とDARWIN用のインクルードファイル(DAQHandler.h，DAQDARWIN.h)は，上記インクルードファイルから参照されているので，宣言を記述する必要はありません。

ライブラリの指定

プロジェクトにライブラリ(DAQDA100.lib，DAQDARWIN.lib，DAQHandler.lib)を追加します。追加方法は，ご使用の環境によりことなります。
すべてのクラスが使用可能になります。Visual C用の関数群も使用できます。

測定データの取得

プログラム例

```
////////////////////////////////////////////////////////////
// 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;
}
////////////////////////////////////////////////////////////
```

説明

全般

DARWINのサブユニット番号0，チャネル1の測定データの瞬時値を取得し，領域に格納します。測定値を読み出し，終了します。

通信接続

```
rc = daqda100.open("192.168.1.11");
```

DARWINのIPアドレスを指定しています。通信用ポートは，通信用定数の「DARWINの通信ポート番号」を指定したことになります。

チャネル1の測定データの取得

```
rc = daqda100.measInstCh(0, 1);
```

DARWINから，サブユニット番号0，チャネル1の測定データの瞬時値を取得し，領域に格納します。

測定値の読み出し

```
value = ((daqda100.getClassDataBuffer(0, 1))-
>getClassDARWINDataInfo()).getValue();
```

チャネル1の各種情報データから測定データをたどって，チャネル1の測定値を読み出します。

通信切断

```
rc = daqda100.close();
```

通信を切断します。

19.4 瞬時値データ読み込み用機能と関数/クラスメンバの対応

瞬時値データ読み込み機能でサポートする機能とクラスの対応を示します。

状態遷移関数と取得関数の2種類あります。
状態遷移関数はDARWINを制御します。状態遷移関数を使用した場合，データ取得機能で測定データを取得すると測定点が1点分だけ進みます(拡張APIの状態が遷移します)。
取得関数では現在の状態の項目値を取得します。取得関数を使用した場合，保持している現在の状態のデータ値を返します(拡張APIの状態は遷移しません)。

状態遷移関数

通信機能

機能	コマンド	クラスと関数メンバ
通信接続	—	CDAQDA100Reader:: open
通信切断	—	CDAQDA100Reader:: close

データ取得機能

機能		コマンド	クラスと関数メンバ
測定データ（瞬時値）	測定チャネル	EF	CDAQDA100Reader:: measInstCh
	演算チャネル	EF	CDAQDA100Reader:: mathInstCh
チャネル情報データ			
	測定チャネル	EL	CDAQDA100Reader:: measInfoCh
	演算チャネル	EL	CDAQDA100Reader:: mathInfoCh

チャネル情報データは，内部で保持されていますが，ユーザが明示的に収集することができます。

取得関数

測定データ

データ名		クラスと関数メンバ
データ値		CDAQDARWINDataInfo:: getValue
データステータス値		CDAQDARWINDataInfo:: getStatus
アラーム(有無)		CDAQDARWINDataBuffer:: isAlarm
測定値	倍精度浮動小数	CDAQDARWINDataInfo:: getDoubleValue
	文字列	CDAQDARWINDataInfo:: getStringValue
時刻	年	CDAQDARWINDateTime:: getFullYear
	月	CDAQDARWINDateTime:: getMonth
	日	CDAQDARWINDateTime:: getDay
	時	CDAQDARWINDateTime:: getHour
	分	CDAQDARWINDateTime:: getMinute
	秒	CDAQDARWINDateTime:: getSecond
	ミリ秒	CDAQDARWINDateTime:: getMilliSecond
アラーム種類		CDAQDARWINDataInfo:: getAlarm

「CDAQDA100Reader::getClassDataBuffer」から「CDAQDARWINDataBuffer::getClassDARWINDataInfo」と「CDAQDARWINDataBuffer::getClassDARWINDateTime」でたどって取得します。

チャネル情報データ

データ名	クラスと関数メンバ
小数点位置	CDAQDARWINChInfo:: getPoint
チャネルステータス	CDAQDARWINChInfo::getChStatus
単位名	CDAQDARWINChInfo:: getUnit

「CDAQDA100Reader::getClassDataBuffer」から「CDAQDARWINDataBuffer::getClassDARWINChInfo」でたどって取得します。

ユーティリティ

機能/データ名		クラスと関数メンバ
測定値	倍精度浮動小数に変換	CDAQDARWINDataInfo:: toDoubleValue
	文字列に変換	CDAQDARWINDataInfo:: toStringValue
アラーム	アラーム種類文字列を取得	CDAQDARWINDataInfo:: getAlarmName
	アラーム文字列の最大長を取得	CDAQDARWINDataInfo:: getMaxLenAlarmName
本APIのバージョン番号を取得		CDAQDA100Reader:: getVersionAPI
本APIのリビジョン番号を取得		CDAQDA100Reader:: getRevisionAPI
エラー	エラーメッセージ文字列を取得	CDAQDA100Reader:: getErrorMessage
	エラーメッセージ文字列の最大長を取得	CDAQDA100Reader:: getMaxLenErrorMessage

19.5 瞬時値データ読み込み用プログラムーDARWIN/Visual C++ー

インクルードファイルのパスを追加

プロジェクトに，インクルードファイル(DAQDA100Reader.h)のパスを追加します。追加方法は，ご使用の環境により異なります。

ソースファイルでの宣言

ソースファイルに宣言を記述します。

```
#include "DAQDA100Reader.h"
```

Note

共通部とDARWIN用のインクルードファイル(DAQHandler.h，DAQDARWIN.h，DAQDA100.h)は，上記インクルードファイルから参照されているので，宣言を記述する必要はありません。

ライブラリの指定

プロジェクトにライブラリ(DAQDA100.lib，DAQDARWIN.lib，DAQHandler.llib)を追加します。追加方法は，ご使用の環境により異なります。
すべてのクラスが使用可能になります。Visual C用の関数郡も使用できます。

測定データの取得

プログラム例

```
////////////////////////////////////////////////////////////
// DA100 sample for measurement
#include <stdio.h>
#include "DAQDA100Reader.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;
}
////////////////////////////////////////////////////////////
```

説明

全般

DARWINのサブユニット番号0，チャネル1の測定データの瞬時値を取得し，領域に格納します。測定値を読み出し，終了します。

通信接続

```
rc = daqda100.open("192.168.1.11");
```

DARWINのIPアドレスを指定しています。通信用ポートは，通信用定数の「瞬時値データ読み込み用ポート番号」を指定したことになります。

チャネル1の測定データの取得

```
rc = daqda100.measInstCh(0, 1);
```

DARWINから，サブユニット番号0，チャネル1の測定データの瞬時値を取得し，領域に格納します。

測定値の読み出し

```
value = ((daqda100.getClassDataBuffer(0, 1))-
>getClassDARWINDataInfo()).getValue();
```

チャネル1の各種情報データから測定データをたどって，チャネル1の測定値を読み出します。

通信切断

```
rc = daqda100.close();
```

通信を切断します。

19.6 DARWIN用クラス詳細

クラスは，クラス名のアルファベット順で並んでいます。

CDAQDA100クラス

- **CDAQHandler**
  - **CDAQDARWIN**
    - **CDAQDA100**

本クラスは，DARWINと通信を行い，取得したデータを保持するクラスです。
以下のデータを保持できます。

- ステータスバイト
- システム構成データ
- チャネル情報データ
- 時刻情報データ
- 測定データ

各機能を実行すると，必要に応じたデータを更新します。また，明示的にユーザが更新することも可能です。

パブリックメンバ

構築・消滅

CDAQDA100	オブジェクトを構築します。
~CDAQDA100	オブジェクトを消滅します。

通信機能

updateStatus	ステータスバイトを更新します。

制御機能

switchMode	操作モードを切り替えます。
switchCode	取得コード種類を切り替えます。
reconstruct	再構築を実行します。
initSetValue	運転モードの設定を初期化します。
ackAlarm	アラームリセットを実行します。
switchCompute	演算処理を切り替えます。
switchReport	レポート実行種類を切り替えます。

設定機能

setRange	レンジを設定します。
setChDELTA	チャネル間差演算を設定します。
setChRRJC	リモートRJCを設定します。
setChUnit	単位名を設定します。
setChAlarm	アラームを設定します。

データ取得機能

measInstCh	測定チャネルの測定データを取得します。
mathInstCh	演算チャネルの測定データを取得します。
measInfoCh	測定チャネルのチャネル情報データを取得します。
mathInfoCh	演算チャネルのチャネル情報データを取得します。
updateSystemConfig	システム構成データを更新します。
updateReportStatus	レポートステータスを更新します。
talkOperationChData	単独チャネル指定で運転モードの設定データを取得する宣言をします。
talkSetupChData	単独チャネル指定で基本設定モードの設定データを取得する宣言をします。
talkCalibrationChData	単独チャネル指定で校正モードの設定データを取得する宣言をします。

データメンバ操作

getClassSysInfo	システム構成データを取得します。
getClassDataBuffer	チャネルの各種情報を取得します。
getCode	取得コード種類を取得します。
getByte	ステータスバイトを取得します。
getReport	レポートステータスを取得します。

●オーバライドしたメンバ

通信機能

open	通信接続をします。

データ取得機能

getData	測定データを取得します。
getChannel	チャネル情報データを取得します。

制御機能

setDateTime	日付時刻を設定します。

ユーティリティ

isObject	オブジェクトをチェックします。

●継承するメンバ

CDAQHandler参照

```
close getErrorMessage getMaxLenErrorMessage getRevisionAPI
getVersionAPI receiveLine sendLine setTimeOut
```

CDAQDARWIN参照

```
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
```

プロテクトメンバ

データメンバ

m_code	取得コード種類の格納領域です。
m_statusByte	ステータスバイトの格納領域です。
m_reportStatus	レポートステータスの格納領域です。
m_cSysInfo	システム構成データの格納領域です。
m_cMeasData	測定チャネルの各種情報の格納領域です。
m_cMathData	演算チャネルの各種情報の格納領域です。

通信機能

updateAll	状態，情報データをすべて更新します。
updateRenew	状態を更新します。
updateChInfo	チャネル情報データをすべて更新します。

データ取得

getInstChBINARY	バイナリモードで測定データを取得します。
getInstChASCII	ASCIIモードで測定データを取得します。
getInfoCh	チャネル情報データを取得します。

データメンバ操作

measClear	測定データ取得のためのデータメンバを初期化します。

ユーティリティ

chNumMax	チャネル個数を取得します。
chNumMaxReport	レポートチャネルの個数を取得します。
getVersionDA100DLL	本DLLのバージョンを取得します。
getRevisionDA100DLL	本DLLのリビジョンを取得します。

●継承するメンバ

CDAQHandler参照

```
m_comm m_nRemainSize receive receiveRemain send
```

CDAQDARWIN参照

```
checkAck getVersionDLL getRevisionDLL startTalker
```

プライベートメンバ

なし。

関数メンバ(アルファベット順)

CDAQDA100::ackAlarm

構文

```
int ackAlarm(void);
```

説明

システム制御種類の「アラームリセット」を実行します。
実行に成功したら，状態を更新します。

戻り値

エラー番号を返します。

参照

```
initSystem updateRenew
```

CDAQDA100::CDAQDA100

構文

```
CDAQDA100(void);
CDAQDA100(const char * strAddress, unsigned int uiPort =
DAQDARWIN_COMMPORT, nt * errCode = NULL);
virtual ~CDAQDA100(void);
```

引数

strAddress	IPアドレスを文字列で指定します。
uiPort	ポート番号を指定します。
errCode	エラー番号の返却先を指定します。

説明

オブジェクトを構築，消滅します。
構築時，データメンバを初期化します。引数が指定されている場合，構築時に通信接続を行います。返却先が指定されていれば，通信接続時のエラー番号を返します。
消滅時，データメンバの領域を開放します。通信記述子が存在する場合，通信切断を行います。エラー番号は返却されません。

参照

`measClear open CDAQDARWIN::CDAQDARWIN`

CDAQDA100::chNumMax

構文

```
int chNumMax(int chType);
```

引数

chType	チャネルタイプを指定します。

説明

指定されたチャネルタイプ内のチャネル最大個数を取得します。
スタンドアロンモデルと拡張モデルをシステム構成データで識別して値を返します。

戻り値

チャネル最大個数を返します。

参照

`chNumMaxReport getClassSysInfo CDAQDARWINSysInfo::isExist`

CDAQDA100::chNumMaxReport

構文

```
virtual int chNumMaxReport(void);
```

説明

チャネルタイプがレポートの場合のチャネル最大個数を取得します。
DR130の場合，識別ができないため正しい値になりません。必要ならば，オーバライドしてください。

戻り値

チャネル最大個数を返します。

CDAQDA100::getByte

構文

```
int getByte(void);
```

説明

データメンバからステータスバイト領域の値を取得します。

戻り値

ステータスバイトを返します。

CDAQDA100::getChannel

構文

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

引数

chType　チャネルタイプを指定します。
chNo　チャネル番号を指定します。
cChInfo　チャネル情報データの返却先を指定します。

説明

チャネル単位で，チャネル情報データを取得するための関数です。
指定されたチャネルのチャネル情報データを取得します。

戻り値

エラー番号を返します。

参照

```
getClassDataBuffer measInstCh
CDAQDARWINDataBuffer::getClassDARWINChInfo
```

CDAQDA100::getClassDataBuffer

構文

```
CDAQDARWINDataBuffer * getClassDataBuffer(int chType, int
chNo);
```

引数

chType　　チャネルタイプを指定します。
chNo　　チャネル番号を指定します。

説明

データメンバから指定されたチャネルの各種情報のデータを取得します。
存在しない場合，NULLを返します。

戻り値

オブジェクトへのポインタを返します。

CDAQDA100::getClassSysInfo

構文

```
CDAQDARWINSysInfo & getClassSysInfo(void);
```

説明

データメンバからシステム構成データのオブジェクトを取得します。

戻り値

オブジェクトへの参照を返します。

CDAQDA100::getCode

構文

```
int getCode(void);
```

説明

データメンバから取得コード種類領域の値を取得します。

戻り値

取得コード種類を返します。

CDAQDA100::getData

構文

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

引数

chType	チャネルタイプを指定します。
chNo	チャネル番号を指定します。
cDateTime	時刻情報データの返却先を指定します。
cDataInfo	測定データの返却先を指定します。

説明

チャネル単位で，瞬時値を取得するための関数です。
バイナリコードで指定されたチャネルの測定データを取得します。

戻り値

エラー番号を返します。

参照

```
getClassDataBuffer measInstCh
CDAQDARWINDataBuffer::getClassDARWINDateTime
CDAQDARWINDataBuffer::getClassDARWINDataInfo
```

CDAQDA100::getInfoCh

構文

```
virtual int getInfoCh(int sChType, int sChNo, int eChType, int
eChNo);
```

引数

sChType	開始チャネルタイプを指定します。
sChNo	開始チャネル番号を指定します。
eChType	終了チャネルタイプを指定します。
eChNo	終了チャネル番号を指定します。

説明

指定されたチャネル範囲のチャネル情報データを取得します。
取得したデータをデータメンバに格納します。

戻り値

エラー番号を返します。

参照

```
talkChInfo getChInfo getClassDataBuffer
CDAQDARWINDataBuffer::setChInfo
```

CDAQDA100::getInstChASCII

構文

```
int getInstChASCII(int sChType, int sChNo, int eChType, int
eChNo);
```

引数

sChType	開始チャネルタイプを指定します。
sChNo	開始チャネル番号を指定します。
eChType	終了チャネルタイプを指定します。
eChNo	終了チャネル番号を指定します。

説明

ASCIIモードで測定データを取得します。
取得したデータをデータメンバに格納します。

戻り値

エラー番号を返します。

参照

```
getChDataByASCII getClassDataBuffer talkDataByASCII
CDAQDARWINDataBuffer::setDataInfo
CDAQDARWINDataBuffer::setDateTime
```

CDAQDA100::getInstChBINARY

構文

```
int getInstChBINARY(int sChType, int sChNo, int eChType, int
eChNo);
```

引数

sChType	開始チャネルタイプを指定します。
sChNo	開始チャネル番号を指定します。
eChType	終了チャネルタイプを指定します。
eChNo	終了チャネル番号を指定します。

説明

バイナリモードで測定データを取得します。
取得したデータをデータメンバに格納します。

戻り値

エラー番号を返します。

参照

```
getChDataByBinary getClassDataBuffer talkDataByBinary
CDAQDARWINDataBuffer::setDataInfo
CDAQDARWINDataBuffer::setDateTime
```

CDAQDA100::getReport

構文

```
int getReport(void);
```

説明

データメンバからレポートステータス領域の値を取得します。

戻り値

レポートステータスを返します。

CDAQDA100::getRevisionDA100DLL

構文

```
static const int getRevisionDA100DLL(void);
```

説明

本DLLのリビジョン番号を取得します。

戻り値

本DLLのリビジョン番号を返します。

CDAQDA100::getVersionDA100DLL

構文

```
static const int getVersionDA100DLL(void);
```

説明

本DLLのバージョン番号を取得します。

戻り値

本DLLのバージョン番号を返します。

CDAQDA100::initSetValue

構文

```
int initSetValue(void);
```

説明

システム制御種類の「運転モードの設定の初期化」を実行します。
実行に成功したら，状態を更新します。

戻り値

エラー番号を返します。

参照

```
initSystem updateAll
```

CDAQDA100::isObject

構文

```
virtual int isObject(const char * classname = "CDAQDA100");
```

引数

classname クラス名を文字列で指定します。

説明

指定されたクラス名を継承しているかをチェックします。
引数が省略された場合，本クラスであるかをチェックします。
本クラスを継承したクラスが，自分自身のクラスをチェックするためには，オーバーライドする必要があります。
クラスを継承している場合，「有効値」(真)を返します。それ以外は，「無効値」(偽)を返します。
本クラスと異なる場合，親クラスをチェックします。

戻り値

有効無効値を返します。

参照

`CDAQDARWIN::isObject`

CDAQDA100::mathInfoCh

構文

```
virtual int mathInfoCh(int chNo = DAQDA100_CHNO_ALL);
```

説明

指定された演算チャネルのチャネル情報データを取得します。
チャネル番号に定数値の「全チャネル番号指定」をすると，全演算チャネルを処理します。

戻り値

エラー番号を返します。

参照

`measInfoCh`

CDAQDA100::mathInstCh

構文

```
virtual int mathInstCh(int chNo = DAQDA100_CHNO_ALL);
```

引数

chNo　　チャネル番号を指定します。

説明

指定された演算チャネルの測定データを取得します。
チャネル番号に定数値の「全チャネル番号指定」をすると，全演算チャネルを処理します。

戻り値

エラー番号を返します。

参照

measInstCh

CDAQDA100::measClear

構文

```
void measClear(void);
```

説明

測定データ取得のためのデータメンバを初期化します。
チャネルの各種情報に既定値を設定します。

参照

getClassSysInfo
CDAQDARWINChInfo::setChType
CDAQDARWINChInfo::setChNo
CDAQDARWINDataBuffer::initialize
CDAQDARWINDataBuffer::getClassDARWINChInfo
CDAQDARWINSysInfo::initialize

CDAQDA100::measInfoCh

構文

```
virtual int measInfoCh(int chType = DAQDA100_CHTYPE_MEASALL,
int chNo = DAQDA100_CHNO_ALL);
```

引数

chType　　チャネルタイプを指定します。
chNo　　チャネル番号を指定します。

説明

指定されたチャネルのチャネル情報データを取得します。
取得したデータは，データメンバのチャネルの各種情報領域に格納します。
チャネルタイプに定数値の「全測定チャネルタイプ指定」をすると，全サブユニットを処理します。
チャネル番号に定数値の「全チャネル番号指定」をすると，チャネルタイプ内の全チャネルを処理します。
実行に成功したら，状態を更新します。

戻り値

エラー番号を返します。

参照

chNumMax getClassSysInfo getInfoCh updateRenew
CDAQDARWINSysInfo::isExist

CDAQDA100::measInstCh

構文

```
virtual int measInstCh(int chType = DAQDA100_CHTYPE_MEASALL,
int chNo = DAQDA100_CHNO_ALL);
```

引数

chType　　チャネルタイプを指定します。
chNo　　チャネル番号を指定します。

説明

指定されたチャネルの測定データを取得します。
取得したデータは，データメンバのチャネルの各種情報領域に格納します。
チャネルタイプに定数値の「全測定チャネルタイプ指定」をすると，全サブユニットを処理します。
チャネル番号に定数値の「全チャネル番号指定」をすると，チャネルタイプ内の全チャネルを処理します。
実行に成功したら，状態を更新します。

戻り値

エラー番号を返します。
エラー
Not support　　サポートしていない取得コード種類です。

参照

```
chNumMax getClassSysInfo getCode getInstChASCII
getInstChBINARY updateRenew CDAQDARWINSysInfo::isExist
```

CDAQDA100::open

構文

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

引数

strAddress　　IPアドレスを文字列で指定します。
uiPort　　ポート番号を指定します。

説明

引数で指定されたIPアドレスとポート番号の機器と通信接続をします。
ポート番号は省略可能で，省略時は通信用定数の「DARWINの通信ポート番号」になります。
測定データ取得のためのデータメンバを初期化し，接続に成功した場合，それらを取得して格納します。
通信タイムアウトを3分に設定します。

戻り値

エラー番号を返します。

参照

```
close measClear setTimeOut updateAll CDAQDARWIN::open
```

CDAQDA100::reconstruct

構文

```
int reconstruct(void);
```

説明

システム制御種類の「システム再構築」を実行します。
実行に成功したら，状態を更新します。

戻り値

エラー番号を返します。

参照

```
initSystem updateAll
```

CDAQDA100::setChAlarm

構文

```
int setChAlarm(int chType, int chNo, int levelNo, int
iAlarmType = DAQDARWIN_ALARM_NONE, int value = 0, int
relayType = 0, int relayNo = 0);
```

引数

chType	チャネルタイプを指定します。
chNo	チャネル番号を指定します。
levelNo	アラームレベルを指定します。
iAlarmType	アラーム種類を指定します。
value	アラーム値を指定します。
relayType	リレータイプを指定します。
relayNo	リレー番号を指定します。

説明

アラームを設定します。
チャネルタイプに定数値の「全測定チャネルタイプ指定」をすると，全サブユニットを処理します。
チャネル番号に定数値の「全チャネル番号指定」をすると，チャネルタイプ内の全チャネルを処理します。
アラームレベルに定数値の「全アラームレベル指定」をすると，チャネル内の全アラームレベルを処理します。
実行に成功したら，状態を更新します。

戻り値

エラー番号を返します。

参照

```
chNumMax getClassSysInfo measInfoCh setAlarm
CDAQDARWINSysInfo::isExist
```

CDAQDA100::setChDELTA

構文

```
int setChDELTA(int chType, int chNo, int refChNo, int spanMin
= 0, int spanMax = 0);
```

引数

chType	チャネルタイプを指定します。
chNo	チャネル番号を指定します。
refChNo	基準チャネルのチャネル番号を指定します。
spanMin	スパンのレフト値を指定します。
spanMax	スパンのライト値を指定します。

説明

指定された基準チャネルとの差演算を設定します。
レフト値とライト値が等しい場合，スパンは省略されたものとみなします。
チャネルタイプに定数値の「全測定チャネルタイプ指定」をすると，全サブユニットを処理します。
チャネル番号に定数値の「全チャネル番号指定」をすると，チャネルタイプ内の全チャネルを処理します。
実行に成功したら，状態を更新します。

戻り値

エラー番号を返します。

参照

chNumMax getClassSysInfo measInfoCh setDELTA
CDAQDARWINSysInfo::isExist

CDAQDA100::setChRRJC

構文

```
int setChRRJC(int chType, int chNo, int refChNo, int spanMin =
0, int spanMax = 0);
```

引数

chType	チャネルタイプを指定します。
chNo	チャネル番号を指定します。
refChNo	基準チャネルのチャネル番号を指定します。
spanMin	スパンのレフト値を指定します。
spanMax	スパンのライト値を指定します。

説明

指定された基準チャネルとのリモートRJCを設定します。
レフト値とライト値が等しい場合，スパンは省略されたものとみなします。
チャネルタイプに定数値の「全測定チャネルタイプ指定」をすると，全サブユニットを処理します。
チャネル番号に定数値の「全チャネル番号指定」をすると，チャネルタイプ内の全チャネルを処理します。
実行に成功したら，状態を更新します。

戻り値

エラー番号を返します。

参照

chNumMax getClassSysInfo measInfoCh setRRJC
CDAQDARWINSysInfo::isExist

CDAQDA100::setChUnit

構文

```
int setChUnit(int chType, int chNo, const char * strUnit);
```

引数

chType　　チャネルタイプを指定します。
chNo　　チャネル番号を指定します。
strUnit　　単位名を文字列で指定します。

説明

指定された単位名を設定します。
チャネルタイプに定数値の「全測定チャネルタイプ指定」をすると，全サブユニットを処理します。
チャネル番号に定数値の「全チャネル番号指定」をすると，チャネルタイプ内の全チャネルを処理します。
実行に成功したら，状態を更新します。

戻り値

エラー番号を返します。

参照

chNumMax getClassSysInfo measInfoCh setScallingUnit CDAQDARWINSysInfo::isExist

CDAQDA100::setDateTime

構文

```
virtual int setDateTime(CDAQDARWINDateTime * pcDARWINDateTime
= NULL);
```

引数

pcDARWINDateTime　時刻情報データを指定します。

説明

機器本体に時刻情報データを設定します。
指定がNULLの場合，PCの現在の日付時刻を設定します。
実行に成功したら，状態を更新します。

戻り値

エラー番号を返します。

参照

updateRenew CDAQDARWIN::setDateTime

CDAQDA100::setRange

構文

```
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);
```

引数

chType	チャネルタイプを指定します。
chNo	チャネル番号を指定します。
iRange	レンジ種類を指定します。
spanMin	スパンのレフト値を指定します。
spanMax	スパンのライト値を指定します。
scaleMin	スケールのレフト値を指定します。
scaleMax	スケールのライト値を指定します。
scalePoint	スケール時の小数点位置を指定します。
bFilter	フィルタのON/OFFを有効無効値で指定します。
iItem	パワー測定項目を指定します。
iWire	パワー接続方法を指定します。

説明

レンジを設定します。
レフト値とライト値が等しい場合，スパン，スケールは省略されたものとみなします。引数のフィルタのON/OFFは，パルスレンジの場合にのみ有効です。引数のパワー測定項目とパワー接続方法は，パワーモニタレンジの場合にのみ有効です。チャネルタイプに定数値の「全測定チャネルタイプ指定」をすると，全サブユニットを処理します。チャネル番号に定数値の「全チャネル番号指定」をすると，チャネルタイプ内の全チャネルを処理します。実行に成功したら，状態を更新します。

戻り値

エラー番号を返します。
エラー：

Not support	サポートしていないレンジ種類です。

参照

chNumMax getClassSysInfo measInfoCh setDI setMA setPOWER setPULSE setRTD setSKIP setSTRAIN setTC setVOLT CDAQDARWINSysInfo::isExist

CDAQDA100::switchCode

構文

```
int switchCode(int iCode);
```

引数

iCode　　取得コード種類を指定します。

説明

データメンバの取得コード種類領域に指定された値を格納します。
実行に成功したら，状態を更新します。

戻り値

エラー番号を返します。

参照

`updateRenew`

CDAQDA100::switchCompute

構文

```
int switchCompute(int iCompute);
```

引数

iCompute　　演算処理を指定します。

説明

指定された演算処理を実行します。
実行に成功したら，状態を更新します。

戻り値

エラー番号を返します。

参照

`compute updateRenew`

CDAQDA100::switchMode

構文

```
int switchMode(int iMode);
```

引数

iMode　　操作モードを指定します。

説明

指定された操作モードに切り替えます。
「運転モード」に切り替えた場合，チャネル情報データを更新します。
実行に成功したら，状態を更新します。

戻り値

エラー番号を返します。

参照

`transMode updateChInfo updateRenew`

CDAQDA100::switchReport

構文

```
int switchReport(int iReportRun);
```

引数

iReportRun　　レポート実行種類を指定します。

説明

指定されたレポート実行種類を実行します。
実行に成功したら，状態を更新します。

戻り値

エラー番号を返します。

参照

`reporting updateRenew`

CDAQDA100::talkCalibrationChData

構文

```
int talkCalibrationChData(int chType =
DAQDA100_CHTYPE_MEASALL, int chNo = DAQDA100_CHNO_ALL);
```

引数

chType　　チャネルタイプを指定します。
chNo　　チャネル番号を指定します。

説明

指定されたチャネルの校正モードの設定データを取得する宣言を実行します。
チャネルタイプに定数値の「全測定チャネルタイプ指定」をすると，全サブユニットを処理します。
チャネル番号に定数値の「全チャネル番号指定」をすると，チャネルタイプ内の全チャネルを処理します。

戻り値

エラー番号を返します。

参照

`talkCalibrationData`

CDAQDA100::talkOperationChData

構文

```
int talkOperationChData(int chType = DAQDA100_CHTYPE_MEASALL,
int chNo = DAQDA100_CHNO_ALL);
```

引数

chType　　チャネルタイプを指定します。
chNo　　チャネル番号を指定します。

説明

指定されたチャネルの運転モードの設定データを取得する宣言を実行します。
チャネルタイプに定数値の「全測定チャネルタイプ指定」をすると，全サブユニットを処理します。
チャネル番号に定数値の「全チャネル番号指定」をすると，チャネルタイプ内の全チャネルを処理します。

戻り値

エラー番号を返します。

参照

`talkOperationData`

CDAQDA100::talkSetupChData

構文

```
int talkSetupChData(int chType = DAQDA100_CHTYPE_MEASALL, int
chNo = DAQDA100_CHNO_ALL);
```

引数

chType　　チャネルタイプを指定します。
chNo　　チャネル番号を指定します。

説明

指定されたチャネルの基本設定モードの設定データを取得する宣言を実行します。
チャネルタイプに定数値の「全測定チャネルタイプ指定」をすると，全サブユニットを処理します。
チャネル番号に定数値の「全チャネル番号指定」をすると，チャネルタイプ内の全チャネルを処理します。

戻り値

エラー番号を返します。

参照

`talkSetupData`

CDAQDA100::updateAll

構文

```
int updateAll(void);
```

説明

データメンバの情報データをすべて更新します。
システム構成データ，チャネル情報データ，ステータスバイトを取得し格納します。

戻り値

エラー番号を返します。

参照

updateChInfo
updateRenew
updateSystemConfig

CDAQDA100::updateChInfo

構文

```
int updateChInfo(void);
```

説明

チャネル情報データをすべて更新します。

戻り値

エラー番号を返します。

参照

mathInfoCh
measInfoCh

CDAQDA100::updateRenew

構文

```
int updateRenew(void);
```

説明

データメンバの状態を更新します。
ステータスバイトを取得し格納します。

戻り値

エラー番号を返します。

参照

updateStatus

CDAQDA100::updateReportStatus

構文

```
int updateReportStatus(void);
```

説明

レポートステータスを取得します。
取得したデータをデータメンバのレポートステータス領域に格納します。

戻り値

エラー番号を返します。

参照

getReportStatus

CDAQDA100::updateStatus

構文

```
int updateStatus(void);
```

説明

ステータスバイトを取得します。
取得したステータスバイトをデータメンバのステータスバイト領域に格納します。

戻り値

エラー番号を返します。

参照

getStatusByte

CDAQDA100::updateSystemConfig

構文

```
int updateSystemConfig(void);
```

説明

システム構成データを取得します。
取得したデータをデータメンバのシステム構成データ領域に格納します。

戻り値

エラー番号を返します。

参照

getClassSysInfo
getSystemConfig

CDAQDA100Readerクラス

- CDAQHandler
  - CDAQDARWIN
    - CDAQDA100
      - CDAQDA100Reader

本クラスは，瞬時値データ読み込みのためにDARWINと通信を行い，取得したデータを保持するクラスです。
瞬時値データ読み込みのコマンドでチャネル情報データと測定データの取得機能をオーバライドしたクラスです。
以下のデータを保持できます。
・チャネル情報データ
・時刻情報データ
・測定データ
データ取得する以外の機能は正しく動作しません。
測定データの取得は，アラームデータがある場合のみをサポートしています。

パブリックメンバ

構築・消滅

CDAQDA100Reader	オブジェクトを構築します。
~CDAQDA100Reader	オブジェクトを消滅します。

●オーバライドしたメンバ

通信機能

open	通信接続をします。

データ取得機能

measInstCh	測定データを取得します。
measInfoCh	チャネル情報データを取得します。

ユーティリティ

isObject	オブジェクトをチェックします。

●継承するメンバ

CDAQHandler参照

```
close getErrorMessage getMaxLenErrorMessage getRevisionAPI
getVersionAPI receiveLine sendLine setTimeOut
```

CDAQDARWIN参照
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
CDAQDA100参照
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

プロテクトメンバ

データ取得

getInstCh　　測定データを取得します。

●オーバライドしたメンバ

データ取得

getInfoCh　　チャネル情報データを取得します。

●継承するメンバ

CDAQHandler参照
m_comm m_nRemainSize receive receiveRemain send

CDAQDARWIN参照
checkAck
getVersionDLL getRevisionDLL startTalker

CDAQDA100参照
chNumMax chNumMaxReport getInfoCh getInstChASCII
getInstChBINARY getRevisionDA100DLL
getVersionDA100DLL m_cMathData m_cMeasData
m_code m_cSysInfo m_reportStatus m_statusByte measClear
updateAll updateChInfo updateRenew

プライベートメンバ

なし。

関数メンバ(アルファベット順)

CDAQDA100Reader::CDAQDA100Reader

構文

```
CDAQDA100Reader(void);
CDAQDA100Reader(const char * strAddress, unsigned int uiPort =
DAQDA100READER_DATAPORT, int * errCode = NULL);
virtual ~CDAQDA100Reader(void);
```

引数

strAddress	IPアドレスを文字列で指定します。
uiPort	ポート番号を指定します。
errCode	エラー番号の返却先を指定します。

説明

オブジェクトを構築，消滅します。

構築時，データメンバを初期化します。引数が指定されている場合，構築時に通信接続を行います。返却先が指定されていれば，通信接続時のエラー番号を返します。

消滅時，データメンバの領域を開放します。通信記述子が存在する場合，通信切断を行います。エラー番号は返却されません。

参照

`open CDAQDA100::CDAQDA100`

CDAQDA100Reader::getInfoCh

構文

```
virtual int getInfoCh(int sChType, int sChNo, int eChType, int
eChNo);
```

引数

sChType	開始チャネルタイプを指定します。
sChNo	開始チャネル番号を指定します。
eChType	終了チャネルタイプを指定します。
eChNo	終了チャネル番号を指定します。

説明

指定されたチャネル範囲のチャネル情報データを取得します。
取得したデータをデータメンバに格納します。
「通信インターフェイス」のELコマンドを実行します。

戻り値

エラー番号を返します。

参照

```
getChInfo getClassDataBuffer send CDAQDARWINChInfo::toChName
CDAQDARWINDataBuffer::setChInfo
```

CDAQDA100Reader::getInstCh

構文

```
int getInstCh(int sChType, int sChNo, int eChType, int eChNo);
```

引数

sChType	開始チャネルタイプを指定します。
sChNo	開始チャネル番号を指定します。
eChType	終了チャネルタイプを指定します。
eChNo	終了チャネル番号を指定します。

説明

指定されたチャネル範囲の測定データを取得します。
取得したデータをデータメンバに格納します。
「通信インターフェイス」のEB，EFコマンドを実行します。

戻り値

エラー番号を返します。
エラー：
Not data　受信データが不足しています。

参照

```
getChDataByBinary getClassDataBuffer receive runCommand send
CDAQDARWINChInfo::toChName CDAQDARWINDataBuffer::setDataInfo
CDAQDARWINDataBuffer::setDateTime CDAQDARWINDateTime::setByte
```

CDAQDA100Reader::isObject

構文

```
virtual int isObject(const char * classname =
"CDAQDA100Reader");
```

引数

classname　　クラス名を文字列で指定します。

説明

指定されたクラス名を継承しているかをチェックします。
引数が省略された場合，本クラスであるかをチェックします。
本クラスを継承したクラスが，自分自身のクラスをチェックするためには，オーバーライドする必要があります。
クラスを継承している場合，「有効値」(真)を返します。それ以外は，「無効値」(偽)を返します。
本クラスと異なる場合，親クラスをチェックします。

戻り値

有効無効値を返します。

参照

CDAQDA100::isObject

CDAQDA100Reader::measInfoCh

構文

```
virtual int measInfoCh(int chType = DAQDA100_CHTYPE_MEASALL,
int chNo = DAQDA100_CHNO_ALL);
```

引数

chType　　チャネルタイプを指定します。
chNo　　チャネル番号を指定します。

説明

指定されたチャネルのチャネル情報データを取得します。
取得したデータは，データメンバのチャネルの各種情報領域に格納します。
チャネルタイプに定数値の「全測定チャネルタイプ指定」をすると，全サブユニットを処理します。
チャネル番号に定数値の「全チャネル番号指定」をすると，チャネルタイプ内の全チャネルを処理します。

戻り値

エラー番号を返します。

参照

getInfoCh

CDAQDA100Reader::measInstCh

構文

```
virtual int measInstCh(int chType = DAQDA100_CHTYPE_MEASALL,
int chNo = DAQDA100_CHNO_ALL);
```

引数

chType　　チャネルタイプを指定します。
chNo　　チャネル番号を指定します。

説明

指定されたチャネルの測定データを取得します。
取得したデータは，データメンバのチャネルの各種情報領域に格納します。
チャネルタイプに定数値の「全測定チャネルタイプ指定」をすると，全サブユニットを処理します。
チャネル番号に定数値の「全チャネル番号指定」をすると，チャネルタイプ内の全チャネルを処理します。

戻り値

エラー番号を返します。

参照

getInstCh

CDAQDA100Reader::open

構文

```
virtual int open(const char * strAddress, unsigned int uiPort
= DAQDA100READER_DATAPORT);
```

引数

strAddress　　IPアドレスを文字列で指定します。
uiPort　　ポート番号を指定します。

説明

引数で指定されたIPアドレスとポート番号の機器と通信接続をします。
ポート番号は省略可能で，省略時は通信用定数の「DARWINの瞬時値データ読み込み用ポート番号」になります。
測定データ取得のためのデータメンバを初期化し，接続に成功した場合，チャネル情報データを取得して格納します。
通信タイムアウトを3分に設定します。

戻り値

エラー番号を返します。

参照

close measClear setTimeOut updateChInfo CDAQDARWIN::open

CDAQDARWINDataBufferクラス

■ CDAQDARWINDataBuffer

本クラスは，DARWINのチャネルごとの各種情報をまとめたクラスです。
以下のデータを格納することができます。

・チャネル情報データ
・時刻情報データ
・測定データ

パブリックメンバ

構築・消滅

CDAQDARWINDataBuffer	オブジェクトを構築します。
~CDAQDARWINDataBuffer	オブジェクトを消滅します。

データメンバ操作

initialize	データメンバを初期化します。
getClassDARWINChInfo	チャネル情報データを取得します。
getClassDARWINDateTime	時刻情報データを取得します。
getClassDARWINDataInfo	測定データを取得します。
setChInfo	チャネル情報データを設定します。
setDateTime	時刻情報データを設定します。
setDataInfo	測定データを設定します。
isAlarm	アラームの有無を取得します。

プロテクトメンバ

データメンバ

m_cChInfo	チャネル情報データの格納領域です。
m_cTimeBuf	時刻情報データの格納領域です。
m_cDataBuf	測定データの格納領域です。

プライベートメンバ

なし。

関数メンバ(アルファベット順)

CDAQDARWINDataBuffer::CDAQDARWINDataBuffer

構文

```
CDAQDARWINDataBuffer(void);
virtual ~CDAQDARWINDataBuffer(void);
```

説明

オブジェクトを構築，消滅します。
構築時， データメンバを初期化します。

参照

```
initialize
```

CDAQDARWINDataBuffer::getClassDARWINChInfo

構文

```
CDAQDARWINChInfo & getClassDARWINChInfo(void);
```

説明

データメンバからチャネル情報データのオブジェクトを取得します。

戻り値

オブジェクトへの参照を返します。

CDAQDARWINDataBuffer::getClassDARWINDataInfo

構文

```
CDAQDARWINDataInfo & getClassDARWINDataInfo(void);
```

説明

データメンバから測定データのオブジェクトを取得します。

戻り値

オブジェクトへの参照を返します。

CDAQDARWINDataBuffer::getClassDARWINDateTime

構文

```
CDAQDARWINDateTime & getClassDARWINDateTime(void);
```

説明

データメンバから時刻情報データのオブジェクトを取得します。

戻り値

オブジェクトへの参照を返します。

CDAQDARWINDataBuffer::initialize

構文

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

説明

データメンバを初期化します。
初期値は，原則0です。
測定データにチャネル情報データとの関連を設定します。

参照

```
getClassDARWINChInfo getClassDARWINDataInfo
getClassDARWINDateTime CDAQDARWINChInfo::initialize
CDAQDARWINDataInfo::initialize
CDAQDARWINDataInfo::setClassDARWINChInfo
CDAQDARWINDateTime::initialize
```

CDAQDARWINDataBuffer::isAlarm

構文

```
int isAlarm(int levelNo);
```

引数

levelNo　　アラームレベルを指定します。

説明

指定されたアラームレベルのアラームの有無を取得します。
アラーム種類が「アラームなし」の場合，「無効値」を返します。

戻り値

有効無効値を返します。

参照

```
getClassDARWINDataInfo
CDAQDARWINDataInfo::getAlarm
```

CDAQDARWINDataBuffer::setChInfo

構文

```
void setChInfo(CDAQDARWINChInfo & cDARWINChInfo);
```

引数

cDARWINChInfo　　チャネル情報データを指定します。

説明

データメンバに指定されたデータを複写します。

CDAQDARWINDataBuffer::setDataInfo

構文

```
void setDataInfo(CDAQDARWINDataInfo & cDARWINDataInfo);
```

引数

cDARWINDataInfo　測定データを指定します。

説明

データメンバに指定されたデータを複写します。
チャネル情報データとの関連は，本クラス内のデータメンバになります。

参照

```
getClassDARWINChInfo getClassDARWINDataInfo
CDAQDARWINDataInfo::setClassDARWINChInfo
```

CDAQDARWINDataBuffer::setDateTime

構文

```
void setDateTime(CDAQDARWINDateTime & cDARWINDateTime);
```

引数

cDARWINDateTime　時刻情報データを指定します。

説明

データメンバに指定されたデータを複写します。

20.1 機能と関数の対応－DARWIN/Visual C－

本拡張APIでサポートする機能と，Visual Cの関数の対応を示します。

Note

本拡張APIでは，DARWINシリーズ機器の共通機能の一部を提供しています。機種別の機能，セットアップモードの設定機能，A/D校正機能は実装されていません。DARWIN通信機能のコマンドを使用して，機能を追加することができます。

表中の「コマンド」とは，DARWIN通信機能のコマンドのことです。コマンドの詳細については，通信インターフェースユーザーズマニュアルをご覧ください。

状態遷移関数と取得関数の2種類あります。
状態遷移関数はDARWIN本体を制御します。
取得関数では現在の状態の項目値を取得します。取得関数を使用した場合，保持している現在の状態のデータ値を返します(拡張APIの状態は遷移しません)。

状態遷移関数

通信機能

機能	コマンド	関数
DARWINと通信接続	－	openDA100
DARWINとの通信を切断	－	closeDA100
データを行単位で送信 特別にデータ受信を制御する場合に使用します。	－	sendLineDA100
データを行単位で受信 特別にデータ受信を制御する場合に使用します。	－	receiveLineDA100
バイト単位でデータを受信します。 特別にデータ受信を制御する場合に使用します。	－	receiveByteDA100
コマンドを送信し，応答を受信 機能コマンドを実装する場合に使用します。	－	runCommandDA100
ステータスバイトを取得 ステータスバイト出力コマンドを送信し，応答を受信します。	(ESC S)	updateStatusDA100
トリガコマンド(ESC T)を送信し，応答を受信 新たにトーカ機能を実装する場合に使用します。	(ESC T)	sendTriggerDA100

通信機能は，通信接続とステータス状態更新を除き，保持データの状態更新は行いません。

制御機能

機能	コマンド	関数
操作モード切替	DS	switchModeDA100
取得コード種類 (バイナリ/ASCIIコード)切替	ー	switchCodeDA100
再構築	RS	reconstructDA100
設定値の初期化	RC	initSetValueDA100
アラームリセット	AR	ackAlarmDA100
日付時刻設定(現在時刻)	SD	setDateTimeNowDA100
演算のスタート，ストップ	EX	switchComputeDA100
レポートのスタート，ストップ	DR	switchReportDA100
セットアップモード確定	XE	establishDA100

原則，処理の最後に状態更新を行います。
セットアップモード確定は，状態更新を行いません。

設定(運転モード)機能

機能		コマンド	関数
レンジ	スキップ(未使用)	SR	setRangeDA100
	直流電圧入力	SR	setRangeDA100
	熱電対入力	SR	SetRangeDA100
	測温抵抗体入力	SR	SetRangeDA100
	接点入力(DI)	SR	SetRangeDA100
	直流電流	SR	SetRangeDA100
	ひずみ	SR	SetRangeDA100
	パルス	SR	SetRangeDA100
	パワーモニタ	SR	SetRangeDA100
	チャネル間差演算	SR	setChDELTADA100
	リモートRJC	SR	setChRRJCDA100
単位名		SN	setChUnitDA100
アラーム		SA	setChAlarmDA100

チャネル単位の設定になります。
設定後，状態更新を行います。

データ取得機能

機能			コマンド	関数
測定データ		測定チャネル	TS，FM	measInstChDA100
(瞬時値)		演算チャネル	TS，FM	mathInstChDA100
チャネル		測定チャネル	TS，LF	measInfoChDA100
情報データ		演算チャネル	TS，LF	mathInfoChDA100
システム構成データ			TS，CF	updateSystemConfigDA100
レポートステータス			TS，RF	updateReportStatusDA100
設定データ				
宣言	運転モード	単一指定	TS，LF	talkOperationChDataDA100
		範囲指定	TS，LF	talkOperationDataDA100
	セットアップモード	単一指定	TS，LF	talkSetupChDataDA100
		範囲指定	TS，LF	talkSetupDataDA100
	校正モード	単一指定	TS，LF	talkCalibrationChDataDA100
		範囲指定	TS，LF	talkCalibrationDataDA100
行単位取得			ー	getSetDataByLineDA100

設定データは，保持しませんので，7.2節，7.3節と同じ手順で取得します。この場合，状態更新はされません。
チャネル情報データとシステム構成データは，内部で保持されていますが，ユーザが明示的に収集することができます。
レポートステータスは，内部で保持されていますが，ユーザが明示的に収集しない限り更新されません。

取得関数

測定データ

データ名		関数
データ値		dataValueDA100
データステータス値		dataStatusDA100
アラーム(有無)		dataAlarmDA100
測定値	倍精度浮動小数	dataDoubleValueDA100
	文字列	dataStringValueDA100
時刻	年	dataYearDA100
	月	dataMonthDA100
	日	dataDayDA100
	時	dataHourDA100
	分	dataMinuteDA100
	秒	dataSecondDA100
アラーム種類		alarmTypeDA100

チャネル情報

データ名	関数
小数点位置	channelPointDA100
チャネルステータス	channelStatusDA100
単位名	toChannelUnitDA100 getChannelUnitDA100

システム構成データ

データ名		関数
測定周期		unitIntervalDA100
ユニット	有無	unitValidDA100
モジュール	内部コード	moduleCodeDA100
	モジュール名	toModuleNameDA100 getModuleNameDA100

状態データ

データ名	関数
ステータスバイト	statusByteDA100
取得コード種類 (バイナリ/ASCII コード)	statusCodeDA100
レポートステータス	statusReportDA100

ユーティリティ

機能/データ名		関数
測定値	倍精度浮動小数に変換	toDoubleValueDA100
	文字列に変換	toStringValueDA100
アラーム	アラーム種類文字列を取得	toAlarmNameDA100 getAlarmNameDA100
	文字列の最大長を取得	alarmMaxLengthDA100
本APIのバージョン番号を取得		versionAPIDA100
本APIのリビジョン番号を取得		revisionAPIDA100
エラー	エラーメッセージ文字列を取得	toErrorMessageDA100 getErrorMessageDA100
	エラーメッセージ文字列の最大長を取得	errorMaxLengthDA100

20.2 プログラムーDARWIN/Visual Cー

インクルードファイルのパスを追加

プロジェクトに，インクルードファイル(DAQDA100.h)のパスを追加します。追加方法は，ご使用の環境により異なります。

ソースファイルでの宣言

ソースファイルに宣言を記述します。

```
#include "DAQDA100.h"
```

Note

共通部とDARWIN用のインクルードファイル(DAQHandler.h，DAQDARWIN.h)は，上記インクルードファイルから参照されているので，宣言を記述する必要はありません。

ロードライブラリの記述

本拡張APIの実行可能モジュール(.dll)がプロセスとリンクできるようにするため，下記の記述をします。

本拡張ＡＰＩの実行可能モジュール(.dll)をアドレス空間内にマップします(LoadLibrary)。次に，実行可能モジュール内のエクスポート関数のアドレスを取得(GetProcAddress)します。

関数ポインタのコールバック型は，関数名に接頭語「DLL」をつけてすべて大文字にしたものです。本拡張APIのインクルードファイルで定義されています。

```
HMODULE pDll = LoadLibrary("DAQDA100");
DLLOPENDA100 openDA100 = (DLLOPENDA100)GetProcAddress(pDll,
"openDA100");
```

測定データの取得

プログラム例

```
////////////////////////////////////////////////////////////
// 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;
  //laod
  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;
}
////////////////////////////////////////////////////////////
```

説明

全般

DARWINのサブユニット番号0，チャネル1の測定データの瞬時値を取得し，領域に格納します。測定値を読み出し，終了します。

通信接続

```
comm = openDA100("192.168.1.11", &rc);
```

DARWINのIPアドレスを指定しています。通信用ポートは，通信用定数の「DARWINの通信ポート番号」を指定したことになります。

チャネル1の測定データの取得

```
rc = measInstChDA100(comm, 0, 1);
```

DARWINから，サブユニット番号0，チャネル1の測定データの瞬時値を取得し，領域に格納します。

測定値の読み出し

```
value = dataValueDA100(comm, 0, 1);
```

測定データを格納している領域から，サブユニット番号0，チャネル1の測定値を読み出します。

通信切断

```
rc = closeDA100(comm);
```

通信を切断します。

20.3 瞬時値データ読み込み用機能と関数の対応 ―DARWIN/Visual C―

本拡張APIでサポートする機能と，Visual Cの関数の対応を示します。

Note

本拡張APIでは，DARWINシリーズ機器の共通機能の一部を提供しています。機種別の機能，セットアップモードの設定機能，A/D校正機能は実装されていません。DARWIN通信機能のコマンドを使用して，機能を追加することができます。

表中の「コマンド」とは，DARWIN通信機能のコマンドのことです。コマンドの詳細については，通信インターフェースユーザーズマニュアルをご覧ください。

状態遷移関数と取得関数の2種類あります。
状態遷移関数はDARWIN本体を制御します。
取得関数では現在の状態の項目値を取得します。取得関数を使用した場合，保持している現在の状態のデータ値を返します(拡張APIの状態は遷移しません)。

状態遷移関数

通信機能

機能	コマンド	関数
通信接続	-	openDA100Reader
通信切断	-	closeDA100Reader

データ取得機能

機能		コマンド	関数
測定データ	測定チャネル	EF	measInstChDA100Reader
(瞬時値)	演算チャネル	EF	mathInstChDA100Reader
チャネル	測定チャネル	EL	measInfoChDA100Reader
情報データ	演算チャネル	EL	mathInfoChDA100Reader

チャネル情報データは，内部で保持されていますが，ユーザが明示的に収集することができます。

取得関数

測定データ

データ名		関数
データ値		dataValueDA100Reader
データステータス値		dataStatusDA100Reader
アラーム(有無)		dataAlarmDA100Reader
測定値	倍精度浮動小数	dataDoubleValueDA100Reader
	文字列	dataStringValueDA100Reader
時刻	年	dataYearDA100Reader
	月	dataMonthDA100Reader
	日	dataDayDA100Reader
	時	dataHourDA100Reader
	分	dataMinuteDA100Reader
	秒	dataSecondDA100Reader
	ミリ秒	dataMilliSecDA100Reader
アラーム種類		alarmTypeDA100Reader

チャネル情報データ

データ名	関数
小数点位置	channelPointDA100Reader
チャネルステータス	channelStatusDA100Reader
単位名	toChannelUnitDA100Reader getChannelUnitDA100Reader

ユーティリティ

機能/データ名		関数
測定値	倍精度浮動小数に変換	toDoubleValueDA100Reader
	文字列に変換	toStringValueDA100Reader
アラーム	アラーム種類文字列を取得	toAlarmNameDA100Reader getAlarmNameDA100Reader
	アラーム文字列の最大長を取得	alarmMaxLengthDA100Reader
本APIのバージョン番号を取得		versionAPIDA100Reader
本APIのリビジョン番号を取得		revisionAPIDA100Reader
エラー	エラーメッセージ文字列を取得	toErrorMessageDA100Reader getErrorMessageDA100Reader
	エラーメッセージ文字列の最大長を取得	errorMaxLengthDA100Reader

20.4 瞬時値データ読み込み用プログラムーDARWIN/Visual C一

インクルードファイルのパスを追加

プロジェクトに，インクルードファイル(DAQDA100Reader.h)のパスを追加します。追加方法は，ご使用の環境により異なります。

ソースファイルでの宣言

ソースファイルに宣言を記述します。

```
#include "DAQDA100Reader.h"
```

Note

共通部とDARWIN用のインクルードファイル(DAQHandler.h，DAQDARWIN，DAQDA100)は，上記インクルードファイルから参照されているので，宣言を記述する必要はありません。

ロードライブラリの記述

本拡張APIの実行可能モジュール(.dll)がプロセスとリンクできるようにするため，下記の記述をします。

本拡張ＡＰＩの実行可能モジュール(.dll)をアドレス空間内にマップします(LoadLibrary)。次に，実行可能モジュール内のエクスポート関数のアドレスを取得(GetProcAddress)します。

関数ポインタのコールバック型は，関数名に接頭語「DLL」をつけてすべて大文字にしたものです。本拡張APIのインクルードファイルで定義されています。

```
HMODULE pDll = LoadLibrary("DAQDA100");
DLLOPENDA100READER openDA100Reader =
(DLLOPENDA100READER)GetProcAddress(pDll, "openDA100Reader");
```

測定データの取得

プログラム例

```
////////////////////////////////////////////////////////////
// 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
  DLLOPENDA100READER openDA100Reader;
  DLLCLOSEDA100READER closeDA100Reader;
  DLLMEASINSTCHDA100READER measInstChDA100Reader;
  DLLDATAVALUEDA100READER dataValueDA100Reader;
  //laod
  pDll = LoadLibrary("DAQDA100");
  //get address
  openDA100Reader = (DLLOPENDA100READER)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;
}
////////////////////////////////////////////////////////////
```

説明

全般

DARWINのサブユニット番号0，チャネル1の測定データの瞬時値を取得し，領域に格納します。測定値を読み出し，終了します。

通信接続

```
comm = openDA100Reader("192.168.1.11", &rc);
```

DARWINのIPアドレスを指定しています。通信用ポートは，通信用定数の「瞬時値データ読み込み用ポート番号」を指定したことになります。

チャネル1の測定データの取得

```
rc = measInstChDA100Reader(comm, 0, 1);
```

DARWINから，サブユニット番号0，チャネル1の測定データの瞬時値を取得し，領域に格納します。

測定値の読み出し

```
value = dataValueDA100Reader(comm, 0, 1);
```

測定データを格納している領域から，サブユニット番号0，チャネル1の測定値を読み出します。

通信切断

```
rc = closeDA100Reader(comm);
```

通信を切断します。

21.1 機能と関数の対応－DARWIN/Visual Basic－

本拡張APIでサポートする機能と，Visual Basicの関数の対応を示します。

Note

本拡張APIでは，DARWINシリーズ機器の共通機能の一部を提供しています。機種別の機能，セットアップモードの設定機能，A/D校正機能は実装されていません。DARWIN通信機能のコマンドを使用して，機能を追加することができます。

表中の「コマンド」とは，DARWIN通信機能のコマンドのことです。コマンドの詳細については，通信インターフェースユーザーズマニュアルをご覧ください。

状態遷移関数と取得関数の2種類あります。
状態遷移関数はDARWINを制御します。
取得関数では現在の状態の項目値を取得します。取得関数を使用した場合，保持している現在の状態のデータ値を返します(拡張APIの状態は遷移しません)。

状態遷移関数

通信機能

機能	コマンド	関数
DARWINと通信接続	－	openDA100
DARWINとの通信を切断	－	closeDA100
データを行単位で送信 特別にデータ受信を制御する場合に使用します。	－	sendLineDA100
データを行単位で受信 特別にデータ受信を制御する場合に使用します。	－	receiveLineDA100
バイト単位でデータを受信します。 特別にデータ受信を制御する場合に使用します。	－	receiveByteDA100
コマンドを送信し，応答を受信 機能コマンドを実装する場合に使用します。	－	runCommandDA100
ステータスバイトを取得 ステータスバイト出力コマンドを送信し，応答を受信します。	(ESC S)	updateStatusDA100
トリガコマンド(ESC T)を送信し，応答を受信 新たにトーカ機能を実装する場合に使用します。	(ESC T)	sendTriggerDA100

通信機能は，通信接続とステータス状態更新を除き，保持データの状態更新は行いません。

制御機能

機能	コマンド	関数
操作モード切替	DS	switchModeDA100
取得コード種類 (バイナリ/ASCIIコード)切替	ー	switchCodeDA100
再構築	RS	reconstructDA100
設定値の初期化	RC	initSetValueDA100
アラームリセット	AR	ackAlarmDA100
日付時刻設定(現在時刻)	SD	setDateTimeNowDA100
演算のスタート，ストップ	EX	switchComputeDA100
レポートのスタート，ストップ	DR	switchReportDA100
セットアップモード確定	XE	establishDA100

原則，処理の最後に状態更新を行います。
セットアップモード確定は，状態更新を行いません。

設定(運転モード)機能

機能		コマンド	関数
レンジ	スキップ(未使用)	SR	setRangeDA100
	直流電圧入力	SR	setRangeDA100
	熱電対入力	SR	SetRangeDA100
	測温抵抗体入力	SR	SetRangeDA100
	接点入力(DI)	SR	SetRangeDA100
	直流電流	SR	SetRangeDA100
	ひずみ	SR	SetRangeDA100
	パルス	SR	SetRangeDA100
	パワーモニタ	SR	SetRangeDA100
	チャネル間差演算	SR	setChDELTADA100
	リモートRJC	SR	setChRRJCDA100
単位名		SN	setChUnitDA100
アラーム		SA	setChAlarmDA100

チャネル単位の設定になります。
設定後，状態更新を行います。

データ取得機能

機能			コマンド	関数
測定データ		測定チャネル	TS，FM	measInstChDA100
(瞬時値)		演算チャネル	TS，FM	mathInstChDA100
チャネル		測定チャネル	TS，LF	measInfoChDA100
情報データ		演算チャネル	TS，LF	mathInfoChDA100
システム構成データ			TS，CF	updateSystemConfigDA100
レポートステータス			TS，RF	updateReportStatusDA100
設定データ				
宣言	運転モード	単一指定	TS，LF	talkOperationChDataDA100
		範囲指定	TS，LF	talkOperationDataDA100
	セットアップモード	単一指定	TS，LF	talkSetupChDataDA100
		範囲指定	TS，LF	talkSetupDataDA100
	校正モード	単一指定	TS，LF	talkCalibrationChDataDA100
		範囲指定	TS，LF	talkCalibrationDataDA100
行単位取得			ー	getSetDataByLineDA100

設定データは，保持しませんので，7.2節，7.3節と同じ手順で取得します。この場合，状態更新はされません。
チャネル情報データとシステム構成データは，内部で保持されていますが，ユーザが明示的に収集することができます。
レポートステータスは，内部で保持されていますが，ユーザが明示的に収集しない限り更新されません。

取得関数

測定データ

データ名		関数
データ値		dataValueDA100
データステータス値		dataStatusDA100
アラーム(有無)		dataAlarmDA100
測定値	倍精度浮動小数	dataDoubleValueDA100
	文字列	dataStringValueDA100
時刻	年	dataYearDA100
	月	dataMonthDA100
	日	dataDayDA100
	時	dataHourDA100
	分	dataMinuteDA100
	秒	dataSecondDA100
アラーム種類		alarmTypeDA100

チャネル情報

データ名	関数
小数点位置	channelPointDA100
チャネルステータス	channelStatusDA100
単位名	toChannelUnitDA100

システム構成データ

データ名		関数
測定周期		unitIntervalDA100
ユニット	有無	unitValidDA100
モジュール	内部コード	moduleCodeDA100
	モジュール名	toModuleNameDA100

状態データ

データ名	関数
ステータスバイト	statusByteDA100
取得コード種類 (バイナリ/ASCII コード)	statusCodeDA100
レポートステータス	statusReportDA100

ユーティリティ

機能/データ名		関数
測定値	倍精度浮動小数に変換	toDoubleValueDA100
	文字列に変換	toStringValueDA100
アラーム	アラーム種類文字列を取得	toAlarmNameDA100
	文字列の最大長を取得	alarmMaxLengthDA100
本APIのバージョン番号を取得		versionAPIDA100
本APIのリビジョン番号を取得		revisionAPIDA100
エラー	エラーメッセージ文字列を取得	toErrorMessageDA100
	エラーメッセージ文字列の最大長を取得	errorMaxLengthDA100

21.2 プログラムーDARWIN/Visual Basicー

型，関数，定数の宣言

Visual Basic用の型，関数，定数を使用するためには，あらかじめ宣言しておく必要があります。次の記述方法があります。

全宣言の記述

プロジェクトにVisual Basic用標準モジュールライブラリファイル(DAQDA100.bas)を追加すると，すべての型，関数，定数を宣言したことになります。

宣言の選択記述

Visual Studioに付属しているAPIビューアで，任意の型，関数，定数の宣言記述をコピーできます。この機能を使用するためには，APIビューアで，APIビューア用テキストファイル(DAQDA100.txt)を読み込んでください。
APIビューアの使用方法については，Visual Studioの取扱説明書をご覧ください。

宣言の直接記述

記述例を示します。

```
Public Declare Function openDA100 Lib "DAQDA100" (ByVal
strAddress As String, ByRef errorCode As Long) As Long
```

測定データの取得

プログラム例

```
Attribute VB_Name = "Module1"
Public Sub 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 Sub
```

説明

全般

DARWINのサブユニット番号0，チャネル1の測定データの瞬時値を取得し，領域に格納します。測定値を読み出し，終了します。

通信接続

```
comm = openDA100("192.168.1.11", rc)
```

DARWINのIPアドレスを指定しています。通信用ポートは，通信用定数の「DARWINの通信ポート番号」を指定したことになります。

チャネル1の測定データの取得

```
rc = measInstChDA100(comm, 0, 1)
```

DARWINから，サブユニット番号0，チャネル1の測定データの瞬時値を取得し，領域に格納します。

測定値の読み出し

```
value = dataValueDA100(comm, 0, 1)
```

測定データを格納している領域から，サブユニット番号0，チャネル1の測定値を読み出します。

通信切断

```
rc = closeDA100(comm)
```

通信を切断します。

21.3 瞬時値データ読み込み用機能と関数メンバの対応ーDARWIN/Visual Basicー

本拡張APIでサポートする機能と，Visual Basicの関数の対応を示します。

Note

本拡張APIでは，DARWINシリーズ機器の共通機能の一部を提供しています。機種別の機能，セットアップモードの設定機能，A/D校正機能は実装されていません。DARWIN通信機能のコマンドを使用して，機能を追加することができます。

表中の「コマンド」とは，DARWIN通信機能のコマンドのことです。コマンドの詳細については，通信インターフェースユーザーズマニュアルをご覧ください。

状態遷移関数と取得関数の2種類あります。
状態遷移関数はDARWINを制御します。
取得関数では現在の状態の項目値を取得します。取得関数を使用した場合，保持している現在の状態のデータ値を返します(拡張APIの状態は遷移しません)。

状態遷移関数

通信機能

機能	コマンド	関数
通信接続	ー	CDAQDA100Reader:: open
通信切断	ー	CDAQDA100Reader:: close

データ取得機能

機能		コマンド	関数
測定データ(瞬時値)	測定チャネル	EF	measInstChDA100Reader
	演算チャネル	EF	mathInstChDA100Reader
チャネル情報データ	測定チャネル	EL	measInfoChDA100Reader
	演算チャネル	EL	mathInfoChDA100Reader

チャネル情報データは，内部で保持されていますが，ユーザが明示的に収集することができます。

取得関数

測定データ

データ名		関数
データ値		dataValueDA100Reader
データステータス値		dataStatusDA100Reader
アラーム(有無)		dataAlarmDA100Reader
測定値	倍精度浮動小数	dataDoubleValueDA100Reader
	文字列	dataStringValueDA100Reade
時刻	年	dataYearDA100Reader
	月	dataMonthDA100Reader
	日	dataDayDA100Reader
	時	dataHourDA100Reader
	分	dataMinuteDA100Reader
	秒	dataSecondDA100Reader
	ミリ秒	dataMilliSecDA100Reader
アラーム種類		alarmTypeDA100Reader

チャネル情報データ

データ名	関数
小数点位置	channelPointDA100Reader
チャネルステータス	channelStatusDA100Reader
単位名	toChannelUnitDA100Reader

ユーティリティ

機能/データ名		関数
測定値	倍精度浮動小数に変換	toDoubleValueDA100Reader
	文字列に変換	toStringValueDA100Reader
アラーム	アラーム種類文字列を取得	toAlarmNameDA100Reader
	アラーム文字列の最大長を取得	alarmMaxLengthDA100Reader
本APIのバージョン番号を取得		versionAPIDA100Reader
本APIのリビジョン番号を取得		revisionAPIDA100Reader
エラー	エラーメッセージ文字列を取得	toErrorMessageDA100Reader
	エラーメッセージ文字列の最大長を取得	errorMaxLengthDA100Reader

21.4 瞬時値データ読み込み用プログラムーDARWIN/Visual Basicー

Visual Basic用の型，関数，定数を使用するためには，あらかじめ宣言しておく必要があります。次の記述方法があります。

全宣言の記述

プロジェクトにＶｉｓｕａｌ　Ｂａｓｉｃ用標準モジュールライブラリファイル(DAQDA100Reader.bas)を追加すると，すべての型，関数，定数を宣言したことになります。

宣言の選択記述

Visual Studioに付属しているAPIビューアで，任意の型，関数，定数の宣言記述をコピーできます。この機能を使用するためには，APIビューアで，APIビューア用テキストファイル(DAQDA100Reader.txt)を読み込んでください。
APIビューアの使用方法については，Visual Studioの取扱説明書をご覧ください。

宣言の直接記述

記述例を示します。

```
Public Declare Function openDA100Reader Lib "DAQDA100" (ByVal
strAddress As String, ByRef errorCode As Long) As Long
```

測定データの取得

プログラム例

```
Attribute VB_Name = "Module1"
Public Sub 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 Sub
```

説明

全般

DARWINのサブユニット番号0，チャネル1の測定データの瞬時値を取得し，領域に格納します。測定値を読み出し，終了します。

通信接続

```
comm = openDA100Reader("192.168.1.11", rc)
```

DARWINのIPアドレスを指定しています。通信用ポートは，通信用定数の「瞬時値データ読み込み用ポート番号」を指定したことになります。

チャネル1の測定データの取得

```
rc = measInstChDA100Reader(comm, 0, 1)
```

DARWINから，サブユニット番号0，チャネル1の測定データの瞬時値を取得し，領域に格納します。

測定値の読み出し

```
Value = dataValueDA100Reader(comm, 0, 1)
```

測定データを格納している領域から，サブユニット番号0，チャネル1の測定値を読み出します。

通信切断

```
rc = closeDA100Reader(comm)
```

通信を切断します。

22.1 機能と関数の対応－DARWIN/Visual Basic.NET－

本拡張APIでサポートする機能と，関数の対応を示します。

Note

本拡張APIでは，DARWINシリーズ機器の共通機能の一部を提供しています。機種別の機能，セットアップモードの設定機能，A/D校正機能は実装されていません。DARWIN通信機能のコマンドを使用して，機能を追加することができます。

表中の「コマンド」とは，DARWIN通信機能のコマンドのことです。コマンドの詳細については，通信インターフェースユーザーズマニュアルをご覧ください。

状態遷移関数と取得関数の2種類あります。
状態遷移関数はDARWINを制御します。
取得関数では現在の状態の項目値を取得します。取得関数を使用した場合，保持している現在の状態のデータ値を返します(拡張APIの状態は遷移しません)。

状態遷移関数

通信機能

機能	コマンド	関数
DARWINと通信接続	－	openDA100
DARWINとの通信を切断	－	closeDA100
データを行単位で送信 特別にデータ受信を制御する場合に使用します。	－	sendLineDA100
データを行単位で受信 特別にデータ受信を制御する場合に使用します。	－	receiveLineDA100
バイト単位でデータを受信します。 特別にデータ受信を制御する場合に使用します。	－	receiveByteDA100
コマンドを送信し，応答を受信 機能コマンドを実装する場合に使用します。	－	runCommandDA100
ステータスバイトを取得 ステータスバイト出力コマンドを送信し，応答を受信します。	(ESC S)	updateStatusDA100
トリガコマンド(ESC T)を送信し，応答を受信 新たにトーカ機能を実装する場合に使用します。	(ESC T)	sendTriggerDA100

通信機能は，通信接続とステータス状態更新を除き，保持データの状態更新は行いません。

制御機能

機能	コマンド	関数
操作モード切替	DS	switchModeDA100
取得コード種類 (バイナリ/ASCIIコード)切替	ー	switchCodeDA100
再構築	RS	reconstructDA100
設定値の初期化	RC	initSetValueDA100
アラームリセット	AR	ackAlarmDA100
日付時刻設定(現在時刻)	SD	setDateTimeNowDA100
演算のスタート，ストップ	EX	switchComputeDA100
レポートのスタート，ストップ	DR	switchReportDA100
セットアップモード確定	XE	establishDA100

原則，処理の最後に状態更新を行います。
セットアップモード確定は，状態更新を行いません。

設定(運転モード)機能

機能		コマンド	関数
レンジ	スキップ(未使用)	SR	setRangeDA100
	直流電圧入力	SR	setRangeDA100
	熱電対入力	SR	SetRangeDA100
	測温抵抗体入力	SR	SetRangeDA100
	接点入力(DI)	SR	SetRangeDA100
	直流電流	SR	SetRangeDA100
	ひずみ	SR	SetRangeDA100
	パルス	SR	SetRangeDA100
	パワーモニタ	SR	SetRangeDA100
	チャネル間差演算	SR	setChDELTADA100
	リモートRJC	SR	setChRRJCDA100
単位名		SN	setChUnitDA100
アラーム		SA	setChAlarmDA100

チャネル単位の設定になります。
設定後，状態更新を行います。

データ取得機能

機能			コマンド	関数
測定データ		測定チャネル	TS，FM	measInstChDA100
(瞬時値)		演算チャネル	TS，FM	mathInstChDA100
チャネル		測定チャネル	TS，LF	measInfoChDA100
情報データ		演算チャネル	TS，LF	mathInfoChDA100
システム構成データ			TS，CF	updateSystemConfigDA100
レポートステータス			TS，RF	updateReportStatusDA100
設定データ				
宣言	運転モード	単一指定	TS，LF	talkOperationChDataDA100
		範囲指定	TS，LF	talkOperationDataDA100
	セットアップモード	単一指定	TS，LF	talkSetupChDataDA100
		範囲指定	TS，LF	talkSetupDataDA100
	校正モード	単一指定	TS，LF	talkCalibrationChDataDA100
		範囲指定	TS，LF	talkCalibrationDataDA100
行単位取得			ー	getSetDataByLineDA100

設定データは，保持しませんので，7.2節，7.3節と同じ手順で取得します。この場合，状態更新はされません。
チャネル情報データとシステム構成データは，内部で保持されていますが，ユーザが明示的に収集することができます。
レポートステータスは，内部で保持されていますが，ユーザが明示的に収集しない限り更新されません。

取得関数

測定データ

データ名		関数
データ値		dataValueDA100
データステータス値		dataStatusDA100
アラーム(有無)		dataAlarmDA100
測定値	倍精度浮動小数	dataDoubleValueDA100
	文字列	dataStringValueDA100
時刻	年	dataYearDA100
	月	dataMonthDA100
	日	dataDayDA100
	時	dataHourDA100
	分	dataMinuteDA100
	秒	dataSecondDA100
アラーム種類		alarmTypeDA100

チャネル情報

データ名	関数
小数点位置	channelPointDA100
チャネルステータス	channelStatusDA100
単位名	toChannelUnitDA100 getChannelUnitDA100

システム構成データ

データ名		関数
測定周期		unitIntervalDA100
ユニット	有無	unitValidDA100
モジュール	内部コード	moduleCodeDA100
	モジュール名	toModuleNameDA100

状態データ

データ名	関数
ステータスバイト	statusByteDA100
取得コード種類 (バイナリ/ASCII コード)	statusCodeDA100
レポートステータス	statusReportDA100

ユーティリティ

機能/データ名		関数
測定値	倍精度浮動小数に変換	toDoubleValueDA100
	文字列に変換	toStringValueDA100
アラーム	アラーム種類文字列を取得	toAlarmNameDA100 getAlarmNameDA100
	文字列の最大長を取得	alarmMaxLengthDA100
本APIのバージョン番号を取得		versionAPIDA100
本APIのリビジョン番号を取得		revisionAPIDA100
エラー	エラーメッセージ文字列を取得	toErrorMessageDA100 getErrorMessageDA100
	エラーメッセージ文字列の最大長を取得	errorMaxLengthDA100

22.2 プログラムーDARWIN/Visual Basic.NETー

宣言の記述

プロジェクトにVisual Basic.NET用モジュールを追加すると，すべての関数，定数を宣言したことになります。

測定データの取得

プログラム例

```
Module Module1
    Public Sub 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 Sub
End Module
```

説明

全般

DARWINのサブユニット番号0，チャネル1の測定データの瞬時値を取得し，領域に格納します。測定値を読み出し，終了します。

通信接続

```
comm = openDA100("192.168.1.11", rc)
```

DARWINのIPアドレスを指定しています。通信用ポートは，通信用定数の「DARWINの通信ポート番号」を指定したことになります。

チャネル1の測定データの取得

```
rc = measInstChDA100(comm, 0, 1)
```

DARWINから，サブユニット番号0，チャネル1の測定データの瞬時値を取得し，領域に格納します。

測定値の取得

```
value = dataValueDA100(comm, 0, 1)
```

測定データを格納している領域から，サブユニット番号0，チャネル1の測定値を読み出します。

通信切断

```
rc = closeDA100(comm)
```

通信を切断します。

22.3 瞬時値データ読み込み用機能と関数の対応ーDARWIN/Visual Basic.NETー

本拡張APIでサポートする機能と，Visual Basic.NETの関数の対応を示します。

Note

本拡張APIでは，DARWINシリーズ機器の共通機能の一部を提供しています。機種別の機能，セットアップモードの設定機能，A/D校正機能は実装されていません。DARWIN通信機能のコマンドを使用して，機能を追加することができます。

表中の「コマンド」とは，DARWIN通信機能のコマンドのことです。コマンドの詳細については，通信インターフェースユーザーズマニュアルをご覧ください。

状態遷移関数と取得関数の2種類あります。
状態遷移関数はDARWIN本体を制御します。
取得関数では現在の状態の項目値を取得します。取得関数を使用した場合，保持している現在の状態のデータ値を返します(拡張APIの状態は遷移しません)。

状態遷移関数

通信機能

機能	コマンド	関数
通信接続	ー	openDA100Reader
通信切断	ー	closeDA100Reader

データ取得機能

機能		コマンド	関数
測定データ(瞬時値)	測定チャネル	EF	measInstChDA100Reader
	演算チャネル	EF	mathInstChDA100Reader
チャネル情報データ	測定チャネル	EL	measInfoChDA100Reader
	演算チャネル	EL	mathInfoChDA100Reader

チャネル情報データは，内部で保持されていますが，ユーザが明示的に収集することができます。

取得関数

測定データ

データ名		関数
データ値		dataValueDA100Reader
データステータス値		dataStatusDA100Reader
アラーム(有無)		dataAlarmDA100Reader
測定値	倍精度浮動小数	dataDoubleValueDA100Reader
	文字列	dataStringValueDA100Reade
時刻	年	dataYearDA100Reader
	月	dataMonthDA100Reader
	日	dataDayDA100Reader
	時	dataHourDA100Reader
	分	dataMinuteDA100Reader
	秒	dataSecondDA100Reader
	ミリ秒	dataMilliSecDA100Reader
アラーム種類		alarmTypeDA100Reader

チャネル情報データ

データ名	関数
小数点位置	channelPointDA100Reader
チャネルステータス	channelStatusDA100Reader
単位名	toChannelUnitDA100Reader

ユーティリティ

機能/データ名		関数
測定値	倍精度浮動小数に変換	toDoubleValueDA100Reader
	文字列に変換	toStringValueDA100Reader
アラーム	アラーム種類文字列を取得	toAlarmNameDA100Reader
	アラーム文字列の最大長を取得	alarmMaxLengthDA100Reader
本APIのバージョン番号を取得		versionAPIDA100Reader
本APIのリビジョン番号を取得		revisionAPIDA100Reader
エラー	エラーメッセージ文字列を取得	toErrorMessageDA100Reader
	エラーメッセージ文字列の最大長を取得	errorMaxLengthDA100Reader

22.4 瞬時値データ読み込み用プログラムーDARWIN/Visual Basic.NETー

宣言の記述

プロジェクトにVisual Basic.NET用モジュールを追加すると，すべての関数，定数を宣言したことになります。

測定データの取得

プログラム例

```
Module Module1
    Public Sub 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 Sub
End Module
```

説明

全般

DARWINのサブユニット番号0，チャネル1の測定データの瞬時値を取得し，領域に格納します。測定値を読み出し，終了します。

通信接続

```
comm = openDA100Reader("192.168.1.11", rc)
```

DARWINのIPアドレスを指定しています。通信用ポートは，通信用定数の「瞬時値データ読み込み用ポート番号」を指定したことになります。

チャネル1の測定データの取得

```
rc = measInstChDA100Reader(comm, 0, 1)
```

DARWINから，サブユニット番号0，チャネル1の測定データの瞬時値を取得し，領域に格納します。

測定値の取得

```
value = dataValueDA100Reader(comm, 0, 1)
```

測定データを格納している領域から，サブユニット番号0，チャネル1の測定値を読み出します。

通信切断

```
rc = closeDA100Reader(comm)
```

通信を切断します。

23.1 機能と関数の対応－DARWIN/C#－

本拡張APIでサポートする機能と，関数の対応を示します。

Note

本拡張APIでは，DARWINシリーズ機器の共通機能の一部を提供しています。機種別の機能，セットアップモードの設定機能，A/D校正機能は実装されていません。DARWIN通信機能のコマンドを使用して，機能を追加することができます。

表中の「コマンド」とは，DARWIN通信機能のコマンドのことです。コマンドの詳細については，通信インターフェースユーザーズマニュアルをご覧ください。

状態遷移関数と取得関数の2種類あります。
状態遷移関数はDARWINを制御します。
取得関数では現在の状態の項目値を取得します。取得関数を使用した場合，保持している現在の状態のデータ値を返します(拡張APIの状態は遷移しません)。

状態遷移関数

通信機能

機能	コマンド	関数
DARWINと通信接続	－	DAQDA100. openDA100
DARWINとの通信を切断	－	DAQDA100. closeDA100
データを行単位で送信 特別にデータ受信を制御する場合に使用します。	－	DAQDA100. sendLineDA100
データを行単位で受信 特別にデータ受信を制御する場合に使用します。	－	DAQDA100. receiveLineDA100
バイト単位でデータを受信します。 特別にデータ受信を制御する場合に使用します。	－	DAQDA100. receiveByteDA100
コマンドを送信し，応答を受信 機能コマンドを実装する場合に使用します。	－	DAQDA100. runCommandDA100
ステータスバイトを取得 ステータスバイト出力コマンドを送信し，応答を受信します。	(ESC S)	DAQDA100. updateStatusDA100
トリガコマンド(ESC T)を送信し，応答を受信 新たにトーカ機能を実装する場合に使用します。	(ESC T)	DAQDA100. sendTriggerDA100

通信機能は，通信接続とステータス状態更新を除き，保持データの状態更新は行いません。

制御機能

機能	コマンド	関数
操作モード切替	DS	DAQDA100. switchModeDA100
取得コード種類 (バイナリ/ASCIIコード)切替	ー	DAQDA100. switchCodeDA100
再構築	RS	DAQDA100. reconstructDA100
設定値の初期化	RC	DAQDA100. initSetValueDA100
アラームリセット	AR	DAQDA100. ackAlarmDA100
日付時刻設定(現在時刻)	SD	DAQDA100. setDateTimeNowDA100
演算のスタート，ストップ	EX	DAQDA100. switchComputeDA100
レポートのスタート，ストップ	DR	DAQDA100. switchReportDA100
セットアップモード確定	XE	DAQDA100. establishDA100

原則，処理の最後に状態更新を行います。
セットアップモード確定は，状態更新を行いません。

設定(運転モード)機能

機能		コマンド	関数
レンジ	スキップ(未使用)	SR	DAQDA100. setRangeDA100
	直流電圧入力	SR	DAQDA100. setRangeDA100
	熱電対入力	SR	DAQDA100. setRangeDA100
	測温抵抗体入力	SR	DAQDA100. setRangeDA100
	接点入力(DI)	SR	DAQDA100. setRangeDA100
	直流電流	SR	DAQDA100. setRangeDA100
	ひずみ	SR	DAQDA100. setRangeDA100
	パルス	SR	DAQDA100. setRangeDA100
	パワーモニタ	SR	DAQDA100. setRangeDA100
	チャネル間差演算	SR	DAQDA100. setChDELTADA100
	リモートRJC	SR	DAQDA100. setChRRJCDA100
単位名		SN	DAQDA100. setChUnitDA100
アラーム		SA	DAQDA100. setChAlarmDA100

チャネル単位の設定になります。
設定後，状態更新を行います。

データ取得機能一覧

機能			コマンド	関数
測定データ(瞬時値)	測定チャネル		TS，FM	DAQDA100. measInstChDA100
	演算チャネル		TS，FM	DAQDA100. mathInstChDA100
チャネル情報データ	測定チャネル		TS，LF	DAQDA100. measInfoChDA100
	演算チャネル		TS，LF	DAQDA100. mathInfoChDA100
システム構成データ			TS，CF	CDAQDA100. updateSystemConfigDA100
レポートステータス			TS，RF	CDAQDA100. updateReportStatusDA100
設定データ				
宣言	運転	単一指定	TS，LF	CDAQDA100. talkOperationChDataDA100
	モード	範囲指定	TS，LF	CDAQDA100. talkOperationDataDA100
	セットアップ	単一指定	TS，LF	CDAQDA100. talkSetupChDataDA100
	モード	範囲指定	TS，LF	CDAQDA100. talkSetupDataDA100
	校正	単一指定	TS，LF	CDAQDA100. talkCalibrationChDataDA100
	モード	範囲指定	TS，LF	CDAQDA100. talkCalibrationDataDA100
行単位取得			ー	CDAQDA100. getSetDataByLineDA100

設定データは，保持しませんので，7.2節，7.3節と同じ手順で取得します。この場合，状態更新はされません。
チャネル情報データとシステム構成データは，内部で保持されていますが，ユーザが明示的に収集することができます。
レポートステータスは，内部で保持されていますが，ユーザが明示的に収集しない限り更新されません。

取得関数

測定データ

データ名		関数
データ値		DAQDA100. dataValueDA100
データステータス値		DAQDA100. dataStatusDA100
アラーム(有無)		DAQDA100. dataAlarmDA100
測定値	倍精度浮動小数	DAQDA100. dataDoubleValueDA100
	文字列	DAQDA100. dataStringValueDA100
時刻	年	DAQDA100. dataYearDA100
	月	DAQDA100. dataMonthDA100
	日	DAQDA100. dataDayDA100
	時	DAQDA100. dataHourDA100
	分	DAQDA100. dataMinuteDA100
	秒	DAQDA100. dataSecondDA100
アラーム種類		DAQDA100. alarmTypeDA100

チャネル情報データ

データ名	関数
小数点位置	DAQDA100. channelPointDA100
チャネルステータス	DAQDA100. channelStatusDA100
単位名	DAQDA100. toChannelUnitDA100

システム構成データ

データ名		関数
測定周期		DAQDA100. unitIntervalDA100
ユニット	有無	DAQDA100. unitValidDA100
モジュール	内部コード	DAQDA100. moduleCodeDA100
	モジュール名	DAQDA100. toModuleNameDA100

状態データ

データ名	関数
ステータスバイト	DAQDA100. statusByteDA100
取得コード種類(バイナリ/ASCIIコード)	DAQDA100. statusCodeDA100
レポートステータス	DAQDA100. statusReportDA100

ユーティリティ

機能/データ名		クラスと関数メンバ
測定値	倍精度浮動小数に変換	DAQDA100. toDoubleValueDA100
	文字列に変換	DAQDA100. toStringValueDA100
アラーム	アラーム種類文字列	DAQDA100. toAlarmNameDA100
	アラーム文字列の最大長を取得	DAQDA100. alarmMaxLengthDA100
本APIのバージョン番号		DAQDA100. versionAPIDA100
本APIのリビジョン番号		DAQDA100. revisionAPIDA100
エラー	エラーメッセージ文字列	DAQDA100. toErrorMessageDA100
	エラーメッセージ文字列の最大長	DAQDA100. errorMaxLengthDA100

23.2 プログラムーDARWIN/C#ー

宣言の記述

プロジェクトにC#用クラスファイル(DAQDA100.cs)を追加すると，すべての関数，定数を利用できるようになります。

測定データの取得

プログラム例

```
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);
          }
  }
}
```

説明

全般

DARWINのサブユニット番号0，チャネル1の測定データの瞬時値を取得し，領域に格納します。測定値を読み出し，終了します。

通信接続

`int comm = DAQDA100.openDA100(enc.GetBytes(address), out rc);`

DARWINのIPアドレスを指定しています。通信用ポートは，通信用定数の「DARWINの通信ポート番号」を指定したことになります。

チャネル1の測定データの取得

`rc = DAQDA100.measInstChDA100(comm, 0, 1);`

DARWINから，サブユニット番号0，チャネル1の測定データの瞬時値を取得し，領域に格納します。

測定値の読み出し

```
int val = DAQDA100.dataValueDA100(comm, 0, 1);
```

測定データを格納している領域から，サブユニット番号0，チャネル1の測定値を読み出します。

通信切断

```
DAQDA100.closeDA100(comm);
```

通信を切断します。

23.3 瞬時値データ読み込み用機能と関数の対応ーDARWIN/C#ー

本拡張APIでサポートする機能と，関数の対応を示します。

Note

本拡張APIでは，DARWINシリーズ機器の共通機能の一部を提供しています。機種別の機能，セットアップモードの設定機能，A/D校正機能は実装されていません。DARWIN通信機能のコマンドを使用して，機能を追加することができます。

表中の「コマンド」とは，DARWIN通信機能のコマンドのことです。コマンドの詳細については，通信インターフェースユーザーズマニュアルをご覧ください。

状態遷移関数と取得関数の2種類あります。
状態遷移関数はDARWINを制御します。
取得関数では現在の状態の項目値を取得します。取得関数を使用した場合，保持している現在の状態のデータ値を返します(拡張APIの状態は遷移しません)。

状態遷移関数

通信機能

機能	コマンド	関数
通信接続	ー	DAQDA100Reader. openDA100Reader
通信切断	ー	DAQDA100Reader. closeDA100Reader

データ取得機能

機能		コマンド	関数
測定データ	測定チャネル	EF	DAQDA100Reader. measInstChDA100Reader
(瞬時値)	演算チャネル	EF	DAQDA100Reader. mathInstChDA100Reader
チャネル	測定チャネル	EL	DAQDA100Reader. measInfoChDA100Reader
情報データ	演算チャネル	EL	DAQDA100Reader. mathInfoChDA100Reader

チャネル情報データは，内部で保持されていますが，ユーザが明示的に収集することができます。

取得関数

測定データ

データ名		関数
データ値		DAQDA100Reader. dataValueDA100Reader
データステータス値		DAQDA100Reader. dataStatusDA100Reader
アラーム(有無)		DAQDA100Reader. dataAlarmDA100Reader
測定値	倍精度浮動小数	DAQDA100Reader. dataDoubleValueDA100Reader
	文字列	DAQDA100Reader. dataStringValueDA100Reader
時刻	年	DAQDA100Reader. dataYearDA100Reader
	月	DAQDA100Reader. dataMonthDA100Reader
	日	DAQDA100Reader. dataDayDA100Reader
	時	DAQDA100Reader. dataHourDA100Reader
	分	DAQDA100Reader. dataMinuteDA100Reader
	秒	DAQDA100Reader. dataSecondDA100Reader
	ミリ秒	DAQDA100Reader. dataMilliSecDA100Reader
アラーム種類		DAQDA100Reader. alarmTypeDA100Reader

チャネル情報データ

データ名	関数
小数点位置	DAQDA100Reader. channelPointDA100Reader
チャネルステータス	DAQDA100Reader. channelStatusDA100Reader
単位名	DAQDA100Reader. toChannelUnitDA100Reader

ユーティリティ

機能/データ名		関数
測定値	倍精度浮動小数に変換	DAQDA100Reader. toDoubleValueDA100Reader
	文字列に変換	DAQDA100Reader. toStringValueDA100Reader
アラーム	アラーム種類文字列を取得	DAQDA100Reader. toAlarmNameDA100Reader
	アラーム文字列の最大長を取得	DAQDA100Reader. alarmMaxLengthDA100Reader
本APIのバージョン番号を取得		DAQDA100Reader. versionAPIDA100Reader
本APIのリビジョン番号を取得		DAQDA100Reader. revisionAPIDA100Reader
エラー	エラーメッセージ文字列を取得	DAQDA100Reader. toErrorMessageDA100Reader
	エラーメッセージ文字列の最大長を取得	DAQDA100Reader. errorMaxLengthDA100Reader

23.4 瞬時値データ読み込み用プログラムーDARWIN/C#ー

宣言の記述

プロジェクトにC#用クラスファイル(DAQDA100Reader.cs)を追加すると，すべての関数，定数を利用できるようになります。

測定データの取得

プログラム例

```
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.openDA100Reader(enc.GetBytes(address), out rc);
             //get
             rc = DAQDA100Reader.measInstChDA100Reader(comm,
0, 1);
             int val =
DAQDA100Reader.dataValueDA100Reader(comm, 0, 1);
             //disconnect
             DAQDA100Reader.closeDA100Reader(comm);
          }
  }
}
```

説明

全般

DARWINのサブユニット番号0，チャネル1の測定データの瞬時値を取得し，領域に格納します。測定値を読み出し，終了します。

通信接続

```
int comm =
DAQDA100Reader.openDA100Reader(enc.GetBytes(address), out rc);
```

DARWINのIPアドレスを指定しています。通信用ポートは，通信用定数の「瞬時値データ読み込み用ポート番号」を指定したことになります。

チャネル1の測定データの取得

```
rc = DAQDA100Reader.measInstChDA100Reader(comm, 0, 1);
```

DARWINから，サブユニット番号0，チャネル1の測定データの瞬時値を取得し，領域に格納します。

測定値の読み出し

```
int val = DAQDA100Reader.dataValueDA100Reader(comm, 0, 1);
```

測定データを格納している領域から，サブユニット番号0，チャネル1の測定値を読み出します。

通信切断

```
DAQDA100Reader.closeDA100Reader(comm);
```

通信を切断します。

24.1 関数の詳細ーDARWIN(Visual C/Visual Basic/Visual Basic.NET/C#)ー状態遷移関数

ここでは，Visual C，Visual Basic，Visual Basic.NET，およびC#で使用するDARWIN用関数について説明しています。関数は，関数名のアルファベット順で並んでいます。

定数，型については第25章をご覧ください。
DARWINに関する用語については付録2をご覧ください。

ほとんどの関数は戻り値として，エラー番号を返します。正常終了の場合は，エラー番号「0」を返します。

ackAlarmDA100

構文

```
int ackAlarmDA100(DAQDA100 daqda100);
```

宣言

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);
```

引数

daqda100　　機器記述子を指定します。

説明

アラームリセットを実行します。

・ 処理の最後に状態更新を行います。

・ 本関数は「通信インターフェイス」のARコマンドを実行します。

戻り値

エラー番号を返します。

エラー：

Not descriptor　機器記述子がありません。

参照

`CDAQDA100::ackAlarm`

closeDA100

構文

```
int closeDA100(DAQDA100 daqda100);
```

宣言

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);
```

引数

daqda100　　機器記述子を指定します。

説明

指定された機器記述子による通信を切断をします。

・ 通信を切断すると， 機器記述子の値は無意味になります。

・ 切断後は， 機器記述子の値は使用しないでください。

戻り値

エラー番号を返します。

エラー：

Not descriptor　機器記述子がありません。

参照

`CDAQDA100::close`

establishDA100

構文

```
int establishDA100(DAQDA100 daqda100, int iSetup);
```

宣言

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);
```

引数

daqda100	機器記述子を指定します。
iSetup	セットアップ確定を指定します。

説明

セットアップモードの設定内容を確定します。

・ セットアップモードでのみ有効です。

・ 本関数は「通信インターフェイス」のXEコマンドを実行します。

戻り値

エラー番号を返します。

エラー：

Not descriptor　機器記述子がありません。

参照

```
CDAQDA100::establish
```

getSetDataByLineDA100

構文

```
int getSetDataByLineDA100(DAQDA100 daqda100, char * strLine,
int maxLine, int * lenLine, int * pFlag);
```

宣言

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);
```

引数

daqda100	機器記述子を指定します。
strLine	行単位の受信文字列を格納する領域を指定します。
maxLine	行単位の受信文字列を格納する領域のバイト数を指定します。
lenLine	実際に受信した文字列のバイト数の返却先を指定します。
pFlag	フラグの返却先を指定します。

説明

設定データを取得する宣言を実行した後，トーカ機能による出力を行単位で取得します。

- 改行を除いた受信文字列を格納します。
- 最終データを取得した場合，フラグにフラグステータスをセットします。 また，エラーで終了した場合もフラグステータスをセットします。
- データ取得を終了するまで，他関数で通信を行わないでください。 本関数でデータ取得中は，他の関数が正しく動作できません。

戻り値

エラー番号を返します。

エラー：

Not descriptor　機器記述子がありません。

参照

CDAQDA100::getSetDataByLine

initSetValueDA100

構文

```
int initSetValueDA100(DAQDA100 daqda100);
```

宣言

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);
```

引数

daqda100　　機器記述子を指定します。

説明

設定値の初期化を実行します。

・ 処理の最後に状態更新を行います。

・ 本関数は「通信インターフェイス」のRCコマンドを実行します。

戻り値

エラー番号を返します。

エラー：

Not descripto　機器記述子がありません。

参照

`CDAQDA100::initSetValue`

mathInfoChDA100

構文

```
int mathInfoChDA100(DAQDA100 daqda100, int chNo);
```

宣言

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);
```

引数

daqda100	機器記述子を指定します。
chNo	チャネル番号を指定します。

説明

指定された演算チャネルのチャネル情報データを取得します。

- チャネル番号に，定数値の「全チャネル番号指定」を指定すると，全演算チャネルを処理します。
- 測定チャネルと演算チャネルは，別々に指定してください。
- 処理の最後に状態更新を行います。
- 本関数は「通信インターフェイス」のTS,LFコマンドを実行します。

戻り値

エラー番号を返します。

エラー：

Not descriptor　機器記述子がありません。

参照

CDAQDA100::mathInfoCh

mathInstChDA100

構文

```
int mathInstChDA100(DAQDA100 daqda100, int chNo);
```

宣言

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);
```

引数

daqda100	機器記述子を指定します。
chNo	チャネル番号を指定します。

説明

指定された演算チャネルの測定データを取得します。

- チャネル番号に，定数値の「全チャネル番号指定」を指定すると，全演算チャネルを処理します。
- 測定チャネルと演算チャネルは，別々に指定してください。
- 処理の最後に状態更新を行います。
- 本関数は「通信インターフェイス」のTS,FMコマンドを実行します。

戻り値

エラー番号を返します。

エラー：

Not descriptor　機器記述子がありません。

参照

`CDAQDA100::mathInstCh`

measInfoChDA100

構文

```
int measInfoChDA100(DAQDA100 daqda100, int chType, int chNo);
```

宣言

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);
```

引数

daqda100	機器記述子を指定します。
chType	チャネルタイプを指定します。
chNo	チャネル番号を指定します。

説明

指定された測定チャネル(チャネルタイプ，チャネル番号で指定)のチャネル情報データを取得します。

- チャネルタイプに，定数値の「全測定チャネルタイプ指定」を指定すると，全サブユニットを処理します。
- チャネル番号に，定数値の「全チャネル番号指定」を指定すると，チャネルタイプ内の全チャネルを処理します。
- 測定チャネルと演算チャネルは，別々に指定してください。
- 処理の最後に状態更新を行います。
- 本関数は「通信インターフェイス」のTS,LFコマンドを実行します。

戻り値

エラー番号を返します。

エラー：

Not descriptor機器記述子がありません。

参照

CDAQDA100::measInfoCh

measInstChDA100

構文

```
int measInstChDA100(DAQDA100 daqda100, int chType, int chNo);
```

宣言

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);
```

引数

daqda100	機器記述子を指定します。
chType	チャネルタイプを指定します。
chNo	チャネル番号を指定します。

説明

指定された測定チャネル(チャネルタイプ，チャネル番号で指定)の測定データを取得します。

- チャネルタイプに，定数値の「全測定チャネルタイプ指定」を指定すると，全サブユニットを処理します。
- チャネル番号に，定数値の「全チャネル番号指定」を指定すると，チャネルタイプ内の全チャネルを処理します。
- 測定チャネルと演算チャネルは，別々に指定してください。
- 処理の最後に状態更新を行います。
- 本関数は「通信インターフェイス」のTS,FMコマンドを実行します。

戻り値

エラー番号を返します。

エラー：

Not descriptor　機器記述子がありません。

参照

CDAQDA100::measInstCh

openDA100

構文

```
DAQDA100 openDA100(const char * strAddress, int * errorCode);
```

宣言

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);
```

引数

strAddress	IPアドレスを文字列で指定します。
errorCode	エラー番号の返却先を指定します。

説明

引数で指定されたアドレスの機器と通信接続をします。

- 機器記述子を作成し，戻り値として返却します。
- 返却先が指定されていれば，エラー番号を格納します。
- ポート番号は固定で，通信用定数の「通信ポート番号」になります。
- 保持しているデータを初期化します。 システム構成データ，チャネル情報データを取得して保持します。
- 指定する文字列は，原則ascii文字列です。
- 失敗した場合，VCではNULLを，他では0を返します。

戻り値

機器記述子を返します。

エラー：

Creating descriptor is failure	機器記述子の作成に失敗しました。

参照

```
CDAQDA100::open
```

receiveByteDA100

構文

```
int receiveByteDA100(DAQDA100 daqda100, unsigned char *
byteData, int maxData, int * lenData) );
```

宣言

Visual Basic

```
Public Declare Function receiveByteDA100 Lib "DAQDA100" (ByVal
daqda100 As Long, ByRef byteData As Byte, ByVal maxData As
Long, ByRef lenData As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function receiveByteDA100 Lib "DAQDA100"
(ByVal daqda100 As Integer, ByRef byteData As Byte, ByVal
maxData As Integer, ByRef 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);
```

引数

daqda100	機器記述子を指定します。
byteData	受信バイトデータを格納する領域を指定します。
maxData	受信データのバイト数を指定します。
lenData	実際に受信したデータのバイト数の返却先を指定します。

説明

引数で指定された領域に，バイト数分になるまで受信データを格納します。

- 返却先が指定されていれば，実際に受信したデータのバイト数を返します。
- 複数バイトのデータがある場合，本関数を繰り返し使用します。
- データ取得を終了するまで，他の関数で通信を行わないでください。 本関数でデータ取得中は，他の関数が正しく動作できません。
- データ終了の判断は，ユーザが独自に行う必要があります。
- 機種特有のトーカ機能を実装する場合に，バイナリ出力を受信するのに使用します。

戻り値

エラー番号を返します。

エラー：

Not descriptor　機器記述子がありません。

参照

`CDAQDA100::receiveByte`

receiveLineDA100

構文

```
int receiveLineDA100(DAQDA100 daqda100, char * strLine, int
maxLine, int * lenLine);
```

宣言

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);
```

引数

daqda100	機器記述子を指定します。
strLine	受信文字列を格納する領域を指定します。
maxLine	受信文字列を格納する領域のバイト数を指定します。
lenLine	実際に受信した文字列のバイト数の返却先を指定します。

説明

引数で指定された受信文字列を格納する領域に，改行を検出するまで，または，バイト数分になるまで受信します。

- 格納する領域には，改行を除いた受信文字列を格納します。
- 返却先が指定されていれば，実際に受信し格納した文字列のバイト数を指定先に格納します。
- 複数行のデータがある場合，本関数を繰り返し使用します。
- データ取得を終了するまで，他の関数で通信を行わないでください。本関数でデータ取得中は，他の関数が正しく動作できません。
- データ終了の判断は，ユーザが独自に行う必要があります。
- 格納される文字列は，原則ascii文字列です。

戻り値

エラー番号を返します。

エラー：

Not descriptor　機器記述子がありません。

参照

```
CDAQDA100::receiveLine
```

reconstructDA100

構文

```
int reconstructDA100(DAQDA100 daqda100);
```

宣言

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);
```

引数

daqda100　　機器記述子を指定します。

説明

システム再構築を実行します。

・ 処理の最後に状態更新を行います。

・ 本関数は「通信インターフェイス」のRSコマンドを実行します。

戻り値

エラー番号を返します。

エラー：

Not descriptor　機器記述子がありません。

参照

`CDAQDA100::reconstruct`

runCommandDA100

構文

```
int runCommandDA100(DAQDA100 daqda100, const char * strCmd);
```

宣言

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);
```

引数

daqda100	機器記述子を指定します。
strCmd	送信するコマンドメッセージを指定します。

説明

指定されたコマンドメッセージとターミネータを送信し，応答を受信します。

- 送信時，本関数がコマンドメッセージにターミネータを付加するので，指定するコマンドメッセージには，ターミネータを含まないでください。
- 複数コマンドの同時送信，ターミネータを含むコマンドメッセージには対応していません。
- トーカ機能のデータ出力要求コマンドのように，応答を返信しないコマンドには対応していません。

戻り値

エラー番号を返します。

エラー：

Not descriptor　機器記述子がありません。

参照

`CDAQDA100::runCommand`

sendLineDA100

構文

```
int sendLineDA100(DAQDA100 daqda100, const char * strLine);
```

宣言

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);
```

引数

daqda100	機器記述子を指定します。
strLine	送信文字列を指定します。

説明

引数で指定された送信文字列を送信します。

- ターミネータを付加して送信します。
- 本関数は応答を受信しません。 別途，受信のための関数で返信されるデータを受信してください。
- 指定する文字列は，原則ascii文字列です。

戻り値

エラー番号を返します。

エラー：

Not descriptor　機器記述子がありません。

参照

`CDAQDA100::sendLine`

sendTriggerDA100

構文

```
int sendTriggerDA100(DAQDA100 daqda100);
```

宣言

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);
```

引数

daqda100　　機器記述子を指定します。

説明

トリガコマンド(ESC T)を送信し，応答を受信します。

戻り値

エラー番号を返します。

エラー：

Not descriptor　機器記述子がありません。

参照

```
CDAQDA100::sendTrigger
```

setChAlarmDA100

構文

```
int setChAlarmDA100(DAQDA100 daqda100, int chType, int chNo,
int levelNo, int iAlarmType, int value, int relayType, int
relayNo);
```

宣言

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);
```

引数

daqda100	機器記述子を指定します。
chType	チャネルタイプを指定します。
chNo	チャネル番号を指定します。
levelNo	アラームレベルを指定します。
iAlarmType	アラーム種類を指定します。
value	アラーム値を指定します。
relayType	リレータイプを指定します。
relayNo	リレー番号を指定します。

説明

指定されたチャネル(チャネルタイプ，チャネル番号で指定)に，指定されたアラーム(アラームレベル，アラーム種類)とアラーム値を設定します。

- チャネルタイプに，定数値の「全測定チャネルタイプ指定」を指定すると，全サブユニットを処理します。
- チャネル番号に，定数値の「全チャネル番号指定」を指定すると，チャネルタイプ内の全チャネルを処理します。
- アラームレベルに，定数値の「全アラームレベル指定」を指定すると，チャネル内の全アラームレベルを処理します。
- リレーの指定で，リレー番号が0以下の場合，リレーは設定されません(OFF)。
- 設定後，保持しているチャネル情報データを更新します。
- 本関数は「通信インターフェイス」のSAコマンドを実行します。

戻り値

エラー番号を返します。

エラー：

Not descriptor　　機器記述子がありません。

参照

`CDAQDA100::setChAlarm`

setChDELTADA100

構文

```
int setChDELTADA100(DAQDA100 daqda100, int chType, int chNo,
int refChNo, int spanMin, int spanMax);
```

宣言

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);
```

引数

daqda100	機器記述子を指定します。
chType	チャネルタイプを指定します。
chNo	チャネル番号を指定します。
refChNo	基準チャネルのチャネル番号を指定します。
spanMin	スパンのレフト値を指定します。
spanMax	スパンのライト値を指定します。

説明

指定されたチャネル(チャネルタイプ，チャネル番号で指定)に，指定された基準チャネルとの差演算を設定します。

- チャネルタイプに，定数値の「全測定チャネルタイプ指定」を指定すると，全サブユニットを処理します。
- チャネル番号に，定数値の「全チャネル番号指定」を指定すると，チャネルタイプ内の全チャネルを処理します。
- スパンの指定で，レフト値とライト値が等しい場合，省略されたものとみなします。
- 設定後，保持しているチャネル情報データを更新します。
- 本関数は「通信インターフェイス」のSRコマンドを実行します。

戻り値

エラー番号を返します。

エラー：

Not descriptor　機器記述子がありません。

参照

CDAQDA100::setChDELTA

setChRRJCDA100

構文

```
int setChRRJCDA100(DAQDA100 daqda100, int chType, int chNo,
int refChNo, int spanMin, int spanMax);
```

宣言

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);
```

引数

daqda100	機器記述子を指定します。
chType	チャネルタイプを指定します。
chNo	チャネル番号を指定します。
refChNo	基準チャネルのチャネル番号を指定します。
spanMin	スパンのレフト値を指定します。
spanMax	スパンのライト値を指定します。

説明

指定されたチャネル(チャネルタイプ，チャネル番号で指定)に，指定された基準チャネルとのリモートRJCを設定します。

- チャネルタイプに，定数値の「全測定チャネルタイプ指定」を指定すると，全サブユニットを処理します。
- チャネル番号に，定数値の「全チャネル番号指定」を指定すると，チャネルタイプ内の全チャネルを処理します。
- スパンの指定で，レフト値とライト値が等しい場合，省略されたものとみなします。
- 設定後，保持しているチャネル情報データを更新します。
- 本関数は「通信インターフェイス」のSRコマンドを実行します。

戻り値

エラー番号を返します。

エラー：

Not descriptor　機器記述子がありません。

参照

CDAQDA100::setChRRJC

setChUnitDA100

構文

```
int setChUnitDA100(DAQDA100 daqda100, int chType, int chNo,
const char * strUnit);
```

宣言

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);
```

引数

daqda100	機器記述子を指定します。
chType	チャネルタイプを指定します。
chNo	チャネル番号を指定します。
strUnit	単位名を指定します。

説明

指定されたチャネル(チャネルタイプ，チャネル番号で指定)に，指定された単位名を設定します。

- チャネルタイプに，定数値の「全測定チャネルタイプ指定」を指定すると，全サブユニットを処理します。
- チャネル番号に，定数値の「全チャネル番号指定」を指定すると，チャネルタイプ内の全チャネルを処理します。
- 設定後，保持しているチャネル情報データを更新します。
- 本関数は「通信インターフェイス」のSNコマンドを実行します。
- 指定する文字列は，原則ascii文字列です。

戻り値

エラー番号を返します。

エラー：

Not descriptor　機器記述子がありません。

参照

CDAQDA100::setChUnit

setDateTimeNowDA100

構文

```
int setDateTimeNowDA100(DAQDA100 daqda100);
```

宣言

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);
```

引数

daqda100　　機器記述子を指定します。

説明

PCの現在の日付時刻を設定します。

・ 処理の最後に状態更新を行います。

・ 本関数は「通信インターフェイス」のSDコマンドを実行します。

戻り値

エラー番号を返します。

エラー：

Not descriptor　機器記述子がありません。

参照

`CDAQDA100::setDateTime`

setRangeDA100

構文

```
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);
```

宣言

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);
```

引数

daqda100	機器記述子を指定します。
chType	チャネルタイプを指定します。
chNo	チャネル番号を指定します。
iRange	レンジ種類を指定します。
spanMin	スパンのレフト値を指定します。
spanMax	スパンのライト値を指定します。
scaleMin	スケールのレフト値を指定します。
scaleMax	スケールのライト値を指定します。
scalePoint	スケールの小数点位置を指定します。
bFilter	フィルタを有効無効値で指定します。
iItem	パワー測定項目を指定します。
iWire	パワー接続方法指定します。

説明

指定されたチャネル(チャネルタイプ，チャネル番号で指定)に，指定されたレンジ種類を設定します。

- チャネルタイプに，定数値の「全測定チャネルタイプ指定」を指定すると，全サブユニットを処理します。
- チャネル番号に，定数値の「全チャネル番号指定」を指定すると，チャネルタイプ内の全チャネルを処理します。
- スパン，スケールの指定で，レフト値とライト値が等しい場合，省略されたものとみなします。
- フィルタの指定は，パルスレンジでのみ有効です。
- パワー測定項目とパワー接続方法の指定は，パワーモニタレンジでのみ有効です。
- 設定後，保持しているチャネル情報データを更新します。
- 本関数は「通信インターフェイス」のSRコマンドを実行します。

戻り値

エラー番号を返します。

エラー：

Not descriptor　機器記述子がありません。

参照

`CDAQDA100::setRange`

switchCodeDA100

構文

```
int switchCodeDA100(DAQDA100 daqda100, int iCode);
```

宣言

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);
```

引数

daqda100	機器記述子を指定します。
iCode	取得コード種類を指定します。

説明

指定された取得コード種類に切り替えます。

・ 処理の最後に状態更新を行います。

戻り値

エラー番号を返します。

エラー：

Not descriptor　機器記述子がありません。

参照

```
CDAQDA100::switchCode
```

switchComputeDA100

構文

```
int switchComputeDA100(DAQDA100 daqda100, int iCompute);
```

宣言

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);
```

引数

daqdarwin　　機器記述子を指定します。
iCompute　　演算処理を指定します。

説明

演算のスタート/ストップを実行します。

・ 演算オプションがある場合に有効です。
・ 処理の最後に状態更新を行います。
・ 本関数は「通信インターフェイス」のEXコマンドを実行します。

戻り値

エラー番号を返します。
エラー：
Not descriptor　機器記述子がありません。

参照

```
CDAQDA100::switchCompute
```

switchModeDA100

構文

```
int switchModeDA100(DAQDA100 daqda100, int iMode);
```

宣言

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);
```

引数

daqda100	機器記述子を指定します。
iMode	操作モードを指定します。

説明

指定された操作モードに切り替えます。

- 「運転モード」に切り替えた場合，保持しているチャネル情報データを更新します。
- 処理の最後に状態更新を行います。
- 本関数は「通信インターフェイス」のDSコマンドを実行します。

戻り値

エラー番号を返します。

エラー：

Not descriptor　機器記述子がありません。

参照

`CDAQDA100::switchMode`

switchReportDA100

構文

```
int switchReportDA100(DAQDA100 daqda100, int iReportRun);
```

宣言

Visual Basic

```
Public Declare Function switchReportDA100 Lib "DAQDA100"
(ByVal daqda100 As Long, ByVal iReportRun As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function switchReportDA100 Lib "DAQDA100"
(ByVal daqda100 As Integer, ByVal iReportRun As Integer) As
Integer
```

C#

```
[DllImport("DAQDA100.dll", CharSet=CharSet.Auto,
EntryPoint="switchReportDA100")]
public static extern int switchReportDA100(int daqda100, int
iReportRun);
```

引数

daqdarwin	機器記述子を指定します。
iReportRun	レポート実行種類を指定します。

説明

レポートのスタート/ストップを実行します。

- レポートオプションがある場合に有効です。
- 処理の最後に状態更新を行います。
- 本関数は「通信インターフェイス」のDRコマンドを実行します。

戻り値

エラー番号を返します。

エラー：

Not descriptor	機器記述子がありません。

参照

```
CDAQDA100::switchReport
```

talkCalibrationChDataDA100

構文

```
int talkCalibrationChDataDA100(DAQDA100 daqda100, int chType,
int chNo);
```

宣言

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);
```

引数

daqda100	機器記述子を指定します。
chType	チャネルタイプを指定します。
chNo	チャネル番号を指定します。

説明

指定されたチャネル(チャネルタイプ，チャネル番号で指定)のA/D校正モードの設定データを取得する宣言を実行します。

- 操作モードを「A/D校正モード」に切り替えておく必要があります。
- チャネルタイプに，定数値の「全測定チャネルタイプ指定」を指定すると，全サブユニットを処理します。
- チャネル番号に，定数値の「全チャネル番号指定」を指定すると，チャネルタイプ内の全チャネルを処理します。
- 本関数は「通信インターフェイス」のTS,LFコマンドを実行します。
- 本関数の実行後，行単位ごとのデータ取得には，getSetDataByLineDA100関数を使用します。

戻り値

エラー番号を返します。

エラー：

Not descriptor　機器記述子がありません。

参照

CDAQDA100::talkCalibrationChData

talkCalibrationDataDA100

構文

```
int talkCalibrationDataDA100(DAQDA100 daqda100, int
startChType, int startChNo, int endChType, int endChNo);
```

宣言

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);
```

引数

daqda100	機器記述子を指定します。
startChType	開始チャネルタイプを指定します。
startChNo	開始チャネル番号を指定します。
endChType	終了チャネルタイプを指定します。
endChNo	終了チャネル番号を指定します。

説明

開始チャネル(開始チャネルタイプ，開始チャネル番号)から終了チャネル(終了チャネルタイプ，終了チャネル番号)までのA/D校正モードの設定データを取得する宣言を実行します。

- 操作モードを「A/D校正モード」に切り替えておく必要があります。
- 本関数は「通信インターフェイス」のTS,LFコマンドを実行します。
- 本関数の実行後，行単位ごとのデータ取得には，getSetDataByLineDA100関数を使用します。

戻り値

エラー番号を返します。

エラー：

Not descriptor　機器記述子がありません。

参照

CDAQDA100::talkCalibrationData

talkOperationChDataDA100

構文

```
int talkOperationChDataDA100(DAQDA100 daqda100, int chType,
int chNo);
```

宣言

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);
```

引数

daqda100	機器記述子を指定します。
chType	チャネルタイプを指定します。
chNo	チャネル番号を指定します。

説明

指定されたチャネル(チャネルタイプ，チャネル番号で指定)の運転モードの設定データを取得する宣言を実行します。

- チャネルタイプに，定数値の「全測定チャネルタイプ指定」を指定すると，全サブユニットを処理します。
- チャネル番号に，定数値の「全チャネル番号指定」を指定すると，チャネルタイプ内の全チャネルを処理します。
- 本関数は「通信インターフェイス」のTS，LFコマンドを実行します。
- 本関数の実行後，行単位ごとのデータ取得には，getSetDataByLineDA100関数を使用します。

戻り値

エラー番号を返します。

エラー：

Not descriptor　機器記述子がありません。

参照

CDAQDA100::talkOperationChData

talkOperationDataDA100

構文

```
int talkOperationDataDA100(DAQDA100 daqda100, int startChType,
int startChNo, int endChType, int endChNo);
```

宣言

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);
```

引数

daqda100	機器記述子を指定します。
startChType	開始チャネルタイプを指定します。
startChNo	開始チャネル番号を指定します。
endChType	終了チャネルタイプを指定します。
endChNo	終了チャネル番号を指定します。

説明

開始チャネル(開始チャネルタイプ，開始チャネル番号)から終了チャネル(終了チャネルタイプ，終了チャネル番号)までの運転モードの設定データを取得する宣言を実行します。

- 本関数は「通信インターフェイス」のTS,LFコマンドを実行します。
- 本関数の実行後，行単位ごとのデータ取得には，getSetDataByLineDA100関数を使用します。

戻り値

エラー番号を返します。

エラー：

Not descriptor　機器記述子がありません。

参照

CDAQDA100::talkOperationData

talkSetupChDataDA100

構文

```
int talkSetupChDataDA100(DAQDA100 daqda100, int chType, int
chNo);
```

宣言

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);
```

引数

daqda100	機器記述子を指定します。
chType	チャネルタイプを指定します。
chNo	チャネル番号を指定します。

説明

指定されたチャネル(チャネルタイプ，チャネル番号で指定)のセットアップモードの設定データを取得する宣言を実行します。

- チャネルタイプに，定数値の「全測定チャネルタイプ指定」を指定すると，全サブユニットを処理します。
- チャネル番号に，定数値の「全チャネル番号指定」を指定すると，チャネルタイプ内の全チャネルを処理します。
- 本関数は「通信インターフェイス」のTS,LFコマンドを実行します。
- 本関数の実行後，行単位ごとのデータ取得には，getSetDataByLineDA100関数を使用します。

戻り値

エラー番号を返します。

エラー：

Not descriptor　機器記述子がありません。

参照

CDAQDA100::talkSetupChData

talkSetupDataDA100

構文

```
int talkSetupDataDA100(DAQDA100 daqda100, int startChType, int
startChNo, int endChType, int endChNo);
```

宣言

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);
```

引数

daqda100	機器記述子を指定します。
startChType	開始チャネルタイプを指定します。
startChNo	開始チャネル番号を指定します。
endChType	終了チャネルタイプを指定します。
endChNo	終了チャネル番号を指定します。

説明

開始チャネル(開始チャネルタイプ，開始チャネル番号)から終了チャネル(終了チャネルタイプ，終了チャネル番号)までのセットアップモードの設定データを取得する宣言を実行します。

- 本関数は「通信インターフェイス」のTS,LFコマンドを実行します。
- 本関数の実行後，行単位ごとのデータ取得には，getSetDataByLineDA100関数を使用します。

戻り値

エラー番号を返します。

エラー：

Not descriptor　機器記述子がありません。

参照

CDAQDA100::talkSetupData

updateReportStatusDA100

構文

```
int updateReportStatusDA100(DAQDA100 daqda100);
```

宣言

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);
```

引数

daqda100　　機器記述子を指定します。

説明

レポートステータスを取得します。

・ レポートオプションがある場合に有効です。

・ 本関数は「通信インターフェイス」のTS,RFコマンドを実行します。

戻り値

エラー番号を返します。

エラー：

Not descriptor　機器記述子がありません。

参照

```
CDAQDA100::updateReportStatus
```

updateStatusDA100

構文

```
int updateStatusDA100(DAQDA100 daqda100);
```

宣言

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);
```

引数

daqda100　　機器記述子を指定します。

説明

ステータスの出力コマンド(ESC S)を送信し，ステータスバイトを取得します。

戻り値

エラー番号を返します。

エラー：

Not descriptor　機器記述子がありません。

参照

```
CDAQDA100::updateStatus
```

updateSystemConfigDA100

構文

```
int updateSystemConfigDA100(DAQDA100 daqda100);
```

宣言

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);
```

引数

daqda100　　機器記述子を指定します。

説明

システム構成データを取得します。

・ 本関数は「通信インターフェイス」のTS,CFコマンドを実行します。

戻り値

エラー番号を返します。

エラー：

Not descriptor　機器記述子がありません。

参照

`CDAQDA100::updateSystemConfig`

24.2 関数の詳細ーDARWIN(Visual C/Visual Basic/Visual Basic.NET/C#)ー取得関数

ここでは，Visual C，Visual Basic，Visual C，およびC#で使用するDARWIN用関数について説明しています。関数は，関数名のアルファベット順で並んでいます。

定数，型については第25章をご覧ください。
DARWINに関する用語については付録2をご覧ください。

alarmMaxLengthDA100

構文

```
int alarmMaxLengthDA100(void);
```

宣言

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();
```

説明

アラーム種類の文字列の最大長を取得します。

・ 戻り値に終端は含まれません。

戻り値

文字列の長さを返します。

参照

CDAQDARWINDataInfo::getMaxLenAlarmName

alarmTypeDA100

構文

```
int alarmTypeDA100(DAQDA100 daqda100, int chType, int chNo,
int levelNo);
```

宣言

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);
```

引数

daqda100	機器記述子を指定します。
chType	チャネルタイプを指定します。
chNo	チャネル番号を指定します。
levelNo	アラームレベルを指定します。

説明

保持している測定データから，指定されたチャネル（チャネルタイプ，チャネル番号），アラームレベルのアラーム種類を取得します。

・存在しない場合，「アラームなし」を返します。

戻り値

アラーム種類を返します。

参照

```
CDAQDA100::getClassDataBuffer
CDAQDARWINDataBuffer::getClassDARWINDataInfo
CDAQDARWINDataInfo::getAlarm
```

channelPointDA100

構文

```
int channelPointDA100(DAQDA100 daqda100, int chType, int
chNo);
```

宣言

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);
```

引数

daqda100	機器記述子を指定します。
chType	チャネルタイプを指定します。
chNo	チャネル番号を指定します。

説明

保持しているチャネル情報データから，指定されたチャネル（チャネルタイプ，チャネル番号）の小数点位置を取得します。

・存在しない場合，0を返します。

戻り値

小数点位置を返します。

参照

```
CDAQDA100::getClassDataBuffer
CDAQDARWINChInfo::getPoint
CDAQDARWINDataBuffer::getClassDARWINChInfo
```

channelStatusDA100

構文

```
int channelStatusDA100(DAQDA100 daqda100, int chType, int
chNo);
```

宣言

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);
```

引数

daqda100	機器記述子を指定します。
chType	チャネルタイプを指定します。
chNo	チャネル番号を指定します。

説明

保持しているチャネル情報データから，指定されたチャネル（チャネルタイプ，チャネル番号）のチャネルステータスを取得します。

・存在しない場合，「不明」を返します。

戻り値

チャネルステータスを返します。

参照

```
CDAQDA100::getClassDataBuffer
CDAQDARWINChInfo::getChStatus
CDAQDARWINDataBuffer::getClassDARWINChInfo
```

dataAlarmDA100

構文

```
int dataAlarmDA100(DAQDA100 daqda100, int chType, int chNo,
int levelNo);
```

宣言

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);
```

引数

daqda100	機器記述子を指定します。
chType	チャネルタイプを指定します。
chNo	チャネル番号を指定します。
levelNo	アラームレベルを指定します。

説明

保持している測定データから，指定されたチャネル(チャネルタイプ，チャネル番号)のアラームレベルに対応するアラームの有無を有効無効値で取得します。

・ 存在しない場合，「無効値」を返します。

戻り値

有効無効値を返します。

参照

```
CDAQDA100::getClassDataBuffer
CDAQDARWINDataBuffer::isAlarm
```

dataDayDA100

構文

```
int dataDayDA100(DAQDA100 daqda100, int chType, int chNo);
```

宣言

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);
```

引数

daqda100	機器記述子を指定します。
chType	チャネルタイプを指定します。
chNo	チャネル番号を指定します。

説明

保持している時刻情報データから，指定されたチャネル(チャネルタイプ，チャネル番号)の日を取得します。

・ 日は1から31の数値です。
・ 存在しない場合，0を返します。

戻り値

日の値を返します。

参照

```
CDAQDA100::getClassDataBuffer
CDAQDARWINDataBuffer::getClassDARWINDateTime
CDAQDARWINDateTime::getDay
```

dataDoubleValueDA100

構文

```
double dataDoubleValueDA100(DAQDA100 daqda100, int chType, int
chNo);
```

宣言

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);
```

引数

daqda100	機器記述子を指定します。
chType	チャネルタイプを指定します。
chNo	チャネル番号を指定します。

説明

保持している測定データから，指定されたチャネル(チャネルタイプ，チャネル番号)の測定値を取得します。

・ 存在しない場合，0.0を返します。

戻り値

測定値を倍精度浮動小数で返します。

参照

```
CDAQDA100::getClassDataBuffer
CDAQDARWINDataBuffer::getClassDARWINDataInfo
CDAQDARWINDataInfo::getDoubleValue
```

dataHourDA100

構文

```
int dataHourDA100(DAQDA100 daqda100, int chType, int chNo);
```

宣言

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);
```

引数

daqda100	機器記述子を指定します。
chType	チャネルタイプを指定します。
chNo	チャネル番号を指定します。

説明

保持している時刻情報データから，指定されたチャネル(チャネルタイプ，チャネル番号)の時を取得します。

- 時は0から23の数値です。
- 存在しない場合，0を返します。

戻り値

時の値を返します。

参照

```
CDAQDA100::getClassDataBuffer
CDAQDARWINDataBuffer::getClassDARWINDateTime
CDAQDARWINDateTime::getHour
```

dataMinuteDA100

構文

```
int dataMinuteDA100(DAQDA100 daqda100ÅCint chTypeÅCint chNo);
```

宣言

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);
```

引数

daqda100	機器記述子を指定します。
chType	チャネルタイプを指定します。
chNo	チャネル番号を指定します。

説明

保持している時刻情報データから，指定されたチャネル(チャネルタイプ，チャネル番号)の分を取得します。

- 分は0から59の数値です。
- 存在しない場合，0を返します。

戻り値

分の値を返します。

参照

```
CDAQDA100::getClassDataBuffer
CDAQDARWINDataBuffer::getClassDARWINDateTime
CDAQDARWINDateTime::getMinute
```

dataMonthDA100

構文

```
int dataMonthDA100(DAQDA100 daqda100, int chType, int chNo);
```

宣言

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);
```

引数

daqda100	機器記述子を指定します。
chType	チャネルタイプを指定します。
chNo	チャネル番号を指定します。

説明

保持している時刻情報データから，指定されたチャネル(チャネルタイプ，チャネル番号)の月を取得します。

・ 月は1から12の数値です。
・ 存在しない場合，0を返します。

戻り値

月の値を返します。

参照

```
CDAQDA100::getClassDataBuffer
CDAQDARWINDataBuffer::getClassDARWINDateTime
CDAQDARWINDateTime::getMonth
```

dataSecondDA100

構文

```
int dataSecondDA100(DAQDA100 daqda100, int chType, int chNo);
```

宣言

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);
```

引数

daqda100	機器記述子を指定します。
chType	チャネルタイプを指定します。
chNo	チャネル番号を指定します。

説明

保持している時刻情報データから，指定されたチャネル(チャネルタイプ，チャネル番号)の秒を取得します。

- 秒は0から59の数値です。
- 存在しない場合，0を返します。

戻り値

秒の値を返します。

参照

```
CDAQDA100::getClassDataBuffer
CDAQDARWINDataBuffer::getClassDARWINDateTime
CDAQDARWINDateTime::getSecond
```

dataStatusDA100

構文

```
int dataStatusDA100(DAQDA100 daqda100, int chType, int chNo);
```

宣言

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);
```

引数

daqda100	機器記述子を指定します。
chType	チャネルタイプを指定します。
chNo	チャネル番号を指定します。

説明

保持している測定データから，指定されたチャネル(チャネルタイプ，チャネル番号)のデータステータス値を取得します。

・存在しない場合，「不明」を返します。

戻り値

データステータス値を返します。

参照

```
CDAQDA100::getClassDataBuffer
CDAQDARWINDataBuffer::getClassDARWINDataInfo
CDAQDARWINDataInfo::getStatus
```

dataStringValueDA100

構文

```
int dataStringValueDA100(DAQDA100 daqda100, int chType, int
chNo, char * strValue, int lenValue);
```

宣言

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);
```

引数

daqda100	機器記述子を指定します。
chType	チャネルタイプを指定します。
chNo	チャネル番号を指定します。
strValue	文字列を格納する領域を指定します。
lenValue	文字列を格納する領域のバイト数を指定します。

説明

保持している測定データから，指定されたチャネル(チャネルタイプ，チャネル番号)の測定値を取得します。

- 文字列に変換して，指定された領域に格納します。
- 領域に格納する文字列には，終端(NULL)も含まれます。
- 実際の文字列の長さが戻り値になります。 戻り値に終端は含まれません。
- 存在しない場合，0を返します。
- 格納される文字列は，原則ascii文字列です。

戻り値

文字列の長さを返します。

参照

```
CDAQDA100::getClassDataBuffer
CDAQDARWINDataBuffer::getClassDARWINDataInfo
CDAQDARWINDataInfo::getStringValue
```

dataValueDA100

構文

```
int dataValueDA100(DAQDA100 daqda100, int chType, int chNo);
```

宣言

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);
```

引数

daqda100	機器記述子を指定します。
chType	チャネルタイプを指定します。
chNo	チャネル番号を指定します。

説明

保持している測定データから，指定されたチャネル(チャネルタイプ，チャネル番号)のデータ値を取得します。

・存在しない場合，0を返します。

戻り値

データ値を返します。

参照

```
CDAQDA100::getClassDataBuffer
CDAQDARWINDataBuffer::getClassDARWINDataInfo
CDAQDARWINDataInfo::getValue
```

dataYearDA100

構文

```
int dataYearDA100(DAQDA100 daqda100, int chType, int chNo);
```

宣言

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);
```

引数

daqda100	機器記述子を指定します。
chType	チャネルタイプを指定します。
chNo	チャネル番号を指定します。

説明

保持している時刻情報データから，指定されたチャネル(チャネルタイプ，チャネル番号)の年を取得します。

- 年は4桁の数値です。
- 存在しない場合，0を返します。

戻り値

年の値を返します。

参照

```
CDAQDA100::getClassDataBuffer
CDAQDARWINDataBuffer::getClassDARWINDateTime
CDAQDARWINDateTime::getFullYear
```

errorMaxLengthDA100

構文

```
int errorMaxLengthDA100(void);
```

宣言

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();
```

説明

エラーメッセージ文字列の最大長を取得します。

・ 戻り値に終端は含まれません。

戻り値

文字列の長さを返します。

参照

```
CDAQDA100::getMaxLenErrorMessage
```

getAlarmNameDA100　　[Visual Cのみ]

構文

```
const char * getAlarmNameDA100(int iAlarmType);
```

引数

iAlarmType　　アラーム種類を指定します。

説明

指定されたアラーム種類に対応する文字列を取得します。

・存在しない場合，「アラームなし」に対応する文字列へのポインタを返します。

戻り値

文字列へのポインタを返します。

参照

`CDAQDARWINDataInfo::getAlarmName`

getChannelUnitDA100 [Visual Cのみ]

構文

```
const char * getChannelUnitDA100(DAQDA100 daqda100, int
chType, int chNo);
```

引数

daqda100	機器記述子を指定します。
chType	チャネルタイプを指定します。
chNo	チャネル番号を指定します。

説明

保持しているチャネル情報データから，指定されたチャネル(チャネルタイプ，チャネル番号)の単位名を取得します。

・ 存在しない場合，NULLを返します。

戻り値

文字列へのポインタを返します。

参照

```
CDAQDA100::getClassDataBuffer
CDAQDARWINChInfo::getUnit
CDAQDARWINDataBuffer::getClassDARWINChInfo
```

getErrorMessageDA100　[Visual Cのみ]

構文

```
const char * getErrorMessageDA100(int errCode);
```

引数

errCode　エラー番号を指定します。

説明

指定されたエラー番号に対応するエラーメッセージ文字列を取得します。

・ 存在しない場合，文字列「Unknown」へのポインタを返します。

戻り値

文字列へのポインタを返します。

参照

`CDAQDA100::getErrorMessage`

getModuleNameDA100 [Visual Cのみ]

構文

```
const char * getModuleNameDA100(DAQDA100 daqda100, int unitNo,
int slotNo);
```

引数

daqda100	機器記述子を指定します。
unitNo	ユニット番号を指定します。
slotNo	スロット番号を指定します。

説明

保持しているシステム構成データから，指定されたユニット番号とスロット番号で示される位置のモジュール名を取得します。

・ 存在しない場合，NULLを返します。

戻り値

文字列へのポインタを返します。

参照

```
CDAQDA100::getClassSysInfo
CDAQDARWINSysInfo::getModuleName
```

moduleCodeDA100

構文

```
int moduleCodeDA100(DAQDA100 daqda100, int unitNo, int
slotNo);
```

宣言

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);
```

引数

daqda100	機器記述子を指定します。
unitNo	ユニット番号を指定します。
slotNo	スロット番号を指定します。

説明

保持しているシステム構成データから，指定されたユニット番号とスロット番号で示される位置の内部コードを取得します。

・ 存在しない場合，0を返します。

戻り値

内部コードを返します。

参照

```
CDAQDA100::getClassSysInfo
CDAQDARWINSysInfo::getModuleCode
```

revisionAPIDA100

構文

```
const int revisionAPIDA100(void);
```

宣言

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();
```

説明

本APIのリビジョン番号を取得します。

戻り値

リビジョン番号を返します。

参照

```
CDAQDA100::getRevisionAPI
```

statusByteDA100

構文

```
int statusByteDA100(DAQDA100 daqda100);
```

宣言

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);
```

引数

daqda100　　機器記述子を指定します。

説明

保持しているステータスバイトを取得します。

・ 存在しない場合，「全ステータスバイトが無効の場合の値」を返します。

戻り値

ステータスバイトを返します。

参照

CDAQDA100::getByte

statusCodeDA100

構文

```
int statusCodeDA100(DAQDA100 daqda100);
```

宣言

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);
```

引数

daqda100　　機器記述子を指定します。

説明

保持している取得コード種類を取得します。

・ 存在しない場合，「バイナリコード」を返します。

戻り値

取得コード種類を返します。

参照

CDAQDA100::getCode

statusReportDA100

構文

```
int statusReportDA100(DAQDA100 daqda100);
```

宣言

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);
```

引数

daqda100　　機器記述子を指定します。

説明

保持しているレポートステータスを取得します。

・存在しない場合，「全無効」を返します。

戻り値

レポートステータスを返します。

参照

CDAQDA100::getReport

toAlarmNameDA100

構文

```
int toAlarmNameDA100(int iAlarmTypeÅCchar * strAlarmÅCint
lenAlarm);
```

宣言

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);
```

引数

iAlarmType	アラーム種類を指定します。
strAlarm	文字列を格納する領域を指定します。
lenAlarm	文字列を格納する領域のバイト数を指定します。

説明

指定されたアラーム種類に対応する文字列を，指定された領域に格納します。

- 領域に格納する文字列には，終端(NULL)も含まれます。
- 実際の文字列の長さが戻り値になります。戻り値に終端は含まれません。
- 格納される文字列は，原則ascii文字列です。

戻り値

文字列の長さを返します。

参照

getAlarmNameDA100

toChannelUnitDA100

構文

```
int toChannelUnitDA100(DAQDA100 daqda100, int chType, int
chNo, char * strUnit, int lenUnit);
```

宣言

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);
```

引数

daqda100	機器記述子を指定します。
chType	チャネルタイプを指定します。
chNo	チャネル番号を指定します。
strUnit	文字列を格納する領域を指定します。
lenUnit	文字列を格納する領域のバイト数を指定します。

説明

保持しているチャネル情報データから，指定されたチャネル(チャネルタイプ，チャネル番号)の単位名を取得します。

- 指定された格納先に文字列を格納します。
- 領域に格納する文字列には，終端(NULL)も含まれます。
- 実際の文字列の長さが戻り値になります。戻り値に終端は含まれません。
- 存在しない場合，0を返します。
- 格納される文字列は，原則ascii文字列です。

戻り値

実際の文字列の長さを返します。

参照

getChannelUnitDA100

toDoubleValueDA100

構文

```
double toDoubleValueDA100(int dataValue, int point);
```

宣言

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);
```

引数

dataValue	データ値を指定します。
point	小数点位置を指定します。

説明

指定されたデータ値と小数点位置から測定値を生成します。

戻り値

測定値を倍精度浮動小数で返します。

参照

```
CDAQDARWINDataInfo::toDoubleValue
```

toErrorMessageDA100

構文

```
int toErrorMessageDA100(int errCode, char * errStr, int
errLen);
```

宣言

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);
```

引数

errorCode	エラー番号を指定します。
errStr	文字列を格納する領域を指定します。
errLen	文字列を格納する領域のバイト数を指定します。

説明

エラー番号に対応するエラーメッセージ文字列を，指定された領域に格納します。

- 領域に格納する文字列には，終端(NULL)も含まれます。
- 実際の文字列の長さが戻り値になります。戻り値に終端は含まれません。
- 格納される文字列は，原則ascii文字列です。

戻り値

文字列の長さを返します。

参照

getErrorMessageDA100

toModuleNameDA100

構文

```
int toModuleNameDA100(DAQDA100 daqda100, int unitNo, int
slotNo, char * strName, int lenName);
```

宣言

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);
```

引数

daqda100	機器記述子を指定します。
unitNo	ユニット番号を指定します。
slotNo	スロット番号を指定します。
strName	文字列を格納する領域を指定します。
lenName	文字列を格納する領域のバイト数を指定します。

説明

保持しているシステム構成データから，指定されたユニット番号とスロット番号で示される位置のモジュール名を取得します。

- 指定された格納先に文字列を格納します。
- 領域に格納する文字列には，終端(NULL)も含まれます。
- 実際の文字列の長さが戻り値になります。 戻り値に終端は含まれません。
- 存在しない場合，0を返します。
- 格納される文字列は，原則ascii文字列です。

戻り値

文字列の長さを返します。

参照

getModuleNameDA100

toStringValueDA100

構文

```
int toStringValueDA100(int dataValue, int point, char *
strValue, int lenValue);
```

宣言

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);
```

引数

dataValue	データ値を指定します。
point	小数点位置を指定します。
strValue	文字列を格納する領域を指定します。
lenValue	文字列を格納する領域のバイト数を指定します。

説明

指定されたデータ値と小数点位置から測定値を生成します。

- 生成された測定値を文字列に変換して，指定された領域に格納します。
- 領域に格納する文字列には，終端(NULL)も含まれます。
- 実際の文字列の長さが戻り値になります。戻り値に終端は含まれません。
- 格納される文字列は，原則ascii文字列です。

戻り値

文字列の長さを返します。

参照

`CDAQDARWINDataInfo::toStringValue`

unitIntervalDA100

構文

```
double unitIntervalDA100(DAQDA100 daqda100);
```

宣言

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);
```

引数

daqda100　　機器記述子を指定します。

説明

保持しているシステム構成データから，測定周期を取得します。

・ 存在しない場合，0.0を返します。

戻り値

測定周期を返します。

参照

```
CDAQDA100::getClassSysInfo
CDAQDARWINSysInfo::getInterval
```

unitValidDA100

構文

```
int unitValidDA100(DAQDA100 daqda100, int unitNo);
```

宣言

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);
```

引数

daqda100	機器記述子を指定します。
unitNo	ユニット番号を指定します。

説明

保持しているシステム構成データから，指定されたユニット番号のユニットの有無を有効無効値で取得します。

・ 存在しない場合，「無効値」を返します。

戻り値

有効無効値を返します。

参照

```
CDAQDA100::getClassSysInfo
CDAQDARWINSysInfo::isExist
```

versionAPIDA100

構文

```
const int versionAPIDA100(void);
```

宣言

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();
```

説明

本APIのバージョン番号を取得します。

戻り値

バージョン番号を返します。

参照

CDAQDA100::getVersionAPI

24.3 瞬時値データ読み込み用関数の詳細ーDARWIN (Visual C/Visual Basic/Visual Basic.NET/C#)ー状態遷移関数

Visual C，Visual Basic，Visual Basic.NET，およびC#で使用する瞬時値データ読み込みポートを使用した場合のDARWIN用の関数について説明します。

ほとんどの関数は戻り値として，エラー番号を返します。正常終了の場合は，エラー番号「0」を返します。

closeDA100Reader

構文

```
int closeDA100Reader(DAQDA100READER daqda100);
```

宣言

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);
```

引数

daqda100Reader　　機器記述子を指定します。

説明

指定された機器記述子による通信を切断をします。

・ 通信を切断すると，機器記述子の値は無意味になります。

・ 切断後は，機器記述子の値は使用しないでください。

戻り値

エラー番号を返します。

エラー：

Not descriptor　　機器記述子がありません。

参照

`CDAQDA100Reader::closes`

mathInfoChDA100Reader

構文

```
int mathInfoChDA100Reader(DAQDA100READER daqda100reader, int
chNo);
```

宣言

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);
```

引数

daqda100Reader	機器記述子を指定します。
chNo	チャネル番号を指定します。

説明

指定された演算チャネルのチャネル情報データを取得します。

- チャネル番号に，定数値の「全チャネル番号指定」を指定すると，全演算チャネルを処理します。
- 測定チャネルと演算チャネルは，別々に指定してください。
- 本関数は「通信インターフェイス」のELコマンドを実行します。

戻り値

エラー番号を返します。

エラー：

Not descriptor	機器記述子がありません。

参照

CDAQDA100Reader::mathInfoCh

mathInstChDA100Reader

構文

```
int mathInstChDA100Reader(DAQDA100READER daqda100reader, int
chNo);
```

宣言

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);
```

引数

daqda100Reader	機器記述子を指定します。
chNo	チャネル番号を指定します。

説明

指定された演算チャネルの測定データを取得します。

- チャネル番号に，定数値の「全チャネル番号指定」を指定すると，全演算チャネルを処理します。
- 測定データとアラームデータを取得します。
- 測定チャネルと演算チャネルは，別々に指定してください。
- 本関数は「通信インターフェイス」のEFコマンドを実行します。

戻り値

エラー番号を返します。

エラー：

Not descriptor	機器記述子がありません。

参照

```
CDAQDA100Reader::mathInstCh
```

measInfoChDA100Reader

構文

```
int measInfoChDA100Reader(DAQDA100READER daqda100reader, int
chType, int chNo);
```

宣言

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);
```

引数

daqda100Reader	機器記述子を指定します。
chType	チャネルタイプを指定します。
chNo	チャネル番号を指定します。

説明

指定された測定チャネル(チャネルタイプ， チャネル番号で指定)のチャネル情報データを取得します。

- チャネルタイプに，定数値の「全測定チャネルタイプ指定」を指定すると，全サブユニットを処理します。
- チャネル番号に， 定数値の「全チャネル番号指定」を指定すると，チャネルタイプ内の全チャネルを処理します。
- 測定チャネルと演算チャネルは，別々に指定してください。
- 本関数は「通信インターフェイス」のELコマンドを実行します。

戻り値

エラー番号を返します。

エラー：

Not descriptor	機器記述子がありません。

参照

CDAQDA100Reader::measInfoCh

measInstChDA100Reader

構文

```
int measInstChDA100Reader(DAQDA100READER daqda100reader, int
chType, int chNo);
```

宣言

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);
```

引数

daqda100Reader	機器記述子を指定します。
chType	チャネルタイプを指定します。
chNo	チャネル番号を指定します。

説明

指定された測定チャネル(チャネルタイプ，チャネル番号で指定)の測定データを取得します。

- チャネルタイプに，定数値の「全測定チャネルタイプ指定」を指定すると，全サブユニットを処理します。
- チャネル番号に，定数値の「全チャネル番号指定」を指定すると，チャネルタイプ内の全チャネルを処理します。
- 測定データとアラームデータを取得します。
- 測定チャネルと演算チャネルは，別々に指定してください。
- 本関数は「通信インターフェイス」のEFコマンドを実行します。

戻り値

エラー番号を返します。

エラー：

Not descriptor	機器記述子がありません。

参照

CDAQDA100Reader::measInstCh

openDA100Reader

構文

```
DAQDA100READER openDA100Reader(const char * strAddress, int *
errorCode);
```

宣言

Visual Basic

```
Public Declare Function openDA100Reader Lib "DAQDA100" (ByVal
strAddress As String, ByRef errorCode As Long) As Long
```

Visual Basic.NET

```
Public Declare Ansi Function openDA100Reader Lib "DAQDA100"
(ByVal strAddress As String, ByRef errorCode As Integer) As
Integer
```

C#

```
[DllImport("DAQDA100.dll", CharSet=CharSet.Auto,
EntryPoint="openDA100Reader")]
public static extern int openDA100Reader(byte[] strAddress,
out int errorCode);
```

引数

strAddress	IPアドレスを文字列で指定します。
errorCode	エラー番号の返却先を指定します。

説明

引数で指定されたアドレスの機器と通信接続をします。

- 機器記述子を作成し，戻り値として返却します。
- 返却先が指定されていれば，エラー番号を格納します。
- ポート番号は固定で，通信用定数の「瞬時値データ読み込み用ポート番号」になります。
- 保持しているデータを初期化します。チャネル情報データなど，機器の状態を取得して保持します。
- 指定する文字列は，原則ascii文字列です。
- 失敗した場合，Visual CではNULLを， Visual Basic，Visual Basic.NET，C#では0を返します。

戻り値

機器記述子を返します。

エラー：

Creating descriptor is failure	機器記述子の作成に失敗しました。

参照

`CDAQDA100Reader::open`

24.4 瞬時値データ読み込み用関数の詳細ーDARWIN(Visual C/Visual Basic/Visual Basic.NET/C#)ー取得関数

Visual C，Visual Basic，Visual Basic.NET，およびC#で使用する瞬時値データ読み込みポートを使用した場合のDARWIN用の関数について説明します。

alarmMaxLengthDA100Reader

構文

```
int alarmMaxLengthDA100Reader(void);
```

宣言

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();
```

説明

アラーム種類の文字列の最大長を取得します。

・ 戻り値に終端は含まれません。

戻り値

文字列の長さを返します。

参照

CDAQDARWINDataInfo::getMaxLenAlarmName

alarmTypeDA100Reader

構文

```
int alarmTypeDA100Reader(DAQDA100READER daqda100reader, int
chType, int chNo, int levelNo);
```

宣言

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);
```

引数

daqda100Reader	機器記述子を指定します。
chType	チャネルタイプを指定します。
chNo	チャネル番号を指定します。
levelNo	アラームレベルを指定します。

説明

保持している測定データから，指定されたチャネル(チャネルタイプ，チャネル番号)，アラームレベルのアラーム種類を取得します。

・ 存在しない場合，「アラームなし」を返します。

戻り値

アラーム種類を返します。

参照

alarmTypeDA100

channelPointDA100Reader

構文

```
int channelPointDA100Reader(DAQDA100READER daqda100reader, int
chType, int chNo);
```

宣言

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);
```

引数

daqda100Reader	機器記述子を指定します。
chType	チャネルタイプを指定します。
chNo	チャネル番号を指定します。

説明

保持しているチャネル情報データから，指定されたチャネル(チャネルタイプ，チャネル番号)の小数点位置を取得します。

・ 存在しない場合，0を返します。

戻り値

小数点位置を返します。

参照

channelPointDA100

channelStatusDA100Reader

構文

```
int channelStatusDA100Reader(DAQDA100READER daqda100reader,
int chType, int chNo);
```

宣言

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);
```

引数

daqda100Reader	機器記述子を指定します。
chType	チャネルタイプを指定します。
chNo	チャネル番号を指定します。

説明

保持しているチャネル情報データから，指定されたチャネル(チャネルタイプ，チャネル番号)のチャネルステータスを取得します。

・ 存在しない場合，「不明」を返します。

戻り値

チャネルステータスを返します。

参照

channelStatusDA100

dataAlarmDA100Reader

構文

```
int dataAlarmDA100Reader(DAQDA100READER daqda100reader, int
chType, int chNo, int levelNo);
```

宣言

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);
```

引数

daqda100Reader	機器記述子を指定します。
chType	チャネルタイプを指定します。
chNo	チャネル番号を指定します。
levelNo	アラームレベルを指定します。

説明

保持している測定データから，指定されたチャネル(チャネルタイプ，チャネル番号)のアラームレベルに対応するアラームの有無を有効無効値で取得します。

・ 存在しない場合，「無効値」を返します。

戻り値

有効無効値を返します。

参照

dataAlarmDA100

dataDayDA100Reader

構文

```
int dataDayDA100Reader(DAQDA100READER daqda100reader, int
chType, int chNo);
```

宣言

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);
```

引数

daqda100Reader	機器記述子を指定します。
chType	チャネルタイプを指定します。
chNo	チャネル番号を指定します。

説明

保持している時刻情報データから，指定されたチャネル(チャネルタイプ，チャネル番号)の日を取得します。

・ 日は1から31の数値です。

・ 存在しない場合，0を返します。

戻り値

日の値を返します。

参照

dataDayDA100

dataDoubleValueDA100Reader

構文

```
double dataDoubleValueDA100Reader(DAQDA100READER
daqda100reader, int chType, int chNo);
```

宣言

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);
```

引数

daqda100Reader	機器記述子を指定します。
chType	チャネルタイプを指定します。
chNo	チャネル番号を指定します。

説明

保持している測定データから，指定されたチャネル(チャネルタイプ，チャネル番号)の測定値を取得します。

・ 存在しない場合，0.0を返します。

戻り値

測定値を倍精度浮動小数で返します。

参照

dataDoubleValueDA100

dataHourDA100Reader

構文

```
int dataHourDA100Reader(DAQDA100READER daqda100reader, int
chType, int chNo);
```

宣言

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);
```

引数

daqda100Reader	機器記述子を指定します。
chType	チャネルタイプを指定します。
chNo	チャネル番号を指定します。

説明

保持している時刻情報データから，指定されたチャネル(チャネルタイプ，チャネル番号)の時を取得します。

・ 時は0から23の数値です。

・ 存在しない場合，0を返します。

戻り値

時の値を返します。

参照

dataHourDA100

dataMilliSecDA100Reader

構文

```
int dataMilliSecDA100Reader(DAQDA100READER daqda100redaer, int
chType, int chNo);
```

宣言

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);
```

引数

daqda100Reader	機器記述子を指定します。
chType	チャネルタイプを指定します。
chNo	チャネル番号を指定します。

説明

保持している時刻情報データから，指定されたチャネル(チャネルタイプ，チャネル番号)のミリ秒を取得します。

・ 存在しない場合，0を返します。

戻り値

ミリ秒の値を返します。

参照

```
CDAQDA100Reader::getClassDataBuffer
CDAQDARWINDataBuffer::getClassDARWINDateTime
CDAQDARWINDateTime::getMilliSecond
```

dataMinuteDA100Reader

構文

```
int dataMinuteDA100Reader(DAQDA100READER daqda100reader, int
chType, int chNo);
```

宣言

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);
```

引数

daqda100Reader	機器記述子を指定します。
chType	チャネルタイプを指定します。
chNo	チャネル番号を指定します。

説明

保持している時刻情報データから，指定されたチャネル(チャネルタイプ，チャネル番号)の分を取得します。

・ 分は0から59の数値です。

・ 存在しない場合，0を返します。

戻り値

分の値を返します。

参照

dataMinuteDA100

dataMonthDA100Reader

構文

```
int dataMonthDA100Reader(DAQDA100READER daqda100reader, int
chType, int chNo);
```

宣言

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);
```

引数

daqda100Reader	機器記述子を指定します。
chType	チャネルタイプを指定します。
chNo	チャネル番号を指定します。

説明

保持している時刻情報データから，指定されたチャネル(チャネルタイプ，チャネル番号)の月を取得します。

- 月は1から12の数値です。
- 存在しない場合，0を返します。

戻り値

月の値を返します。

参照

dataMonthDA100

dataSecondDA100Reader

構文

```
int dataSecondDA100Reader(DAQDA100READER daqda100redaer, int
chType, int chNo);
```

宣言

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);
```

引数

daqda100Reader	機器記述子を指定します。
chType	チャネルタイプを指定します。
chNo	チャネル番号を指定します。

説明

保持している時刻情報データから，指定されたチャネル(チャネルタイプ，チャネル番号)の秒を取得します。

・ 秒は0から59の数値です。

・ 存在しない場合，0を返します。

戻り値

秒の値を返します。

参照

dataSecondDA100

dataStatusDA100Reader

構文

```
int dataStatusDA100Reader(DAQDA100READER daqda100reader, int
chType, int chNo);
```

宣言

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);
```

引数

daqda100Reader	機器記述子を指定します。
chType	チャネルタイプを指定します。
chNo	チャネル番号を指定します。

説明

保持している測定データから，指定されたチャネル(チャネルタイプ，チャネル番号)のデータステータス値を取得します。

・ 存在しない場合，「不明」を返します。

戻り値

データステータス値を返します。

参照

dataStatusDA100

dataStringValueDA100Reader

構文

```
int dataStringValueDA100Reader(DAQDA100READER daqda100reader,
int chType, int chNo, char * strValue, int lenValue);
```

宣言

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);
```

引数

daqda100Reader	機器記述子を指定します。
chType	チャネルタイプを指定します。
chNo	チャネル番号を指定します。
strValue	文字列を格納する領域を指定します。
lenValue	文字列を格納する領域のバイト数を指定します。

説明

保持している測定データから，指定されたチャネル(チャネルタイプ，チャネル番号)の測定値を取得します。

- 文字列に変換して，指定された領域に格納します。
- 領域に格納する文字列には，終端(NULL)も含まれます。
- 実際の文字列の長さが戻り値になります。戻り値に終端は含まれません。
- 存在しない場合，0を返します。
- 格納される文字列は，原則ascii文字列です。

戻り値

文字列の長さを返します。

参照

dataStringValueDA100

dataValueDA100Reader

構文

```
int dataValueDA100Reader(DAQDA100READER daqda100reader, int
chType, int chNo);
```

宣言

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);
```

引数

daqda100Reader	機器記述子を指定します。
chType	チャネルタイプを指定します。
chNo	チャネル番号を指定します。

説明

保持している測定データから，指定されたチャネル(チャネルタイプ，チャネル番号)のデータ値を取得します。

・ 存在しない場合，0を返します。

戻り値

データ値を返します。

参照

dataValueDA100

dataYearDA100Reader

構文

```
int dataYearDA100Reader(DAQDA100READER daqda100reader, int
chType, int chNo);
```

宣言

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);
```

引数

daqda100Reader	機器記述子を指定します。
chType	チャネルタイプを指定します。
chNo	チャネル番号を指定します。

説明

保持している時刻情報データから，指定されたチャネル(チャネルタイプ，チャネル番号)の年を取得します。

・ 年は4桁の数値です。

・ 存在しない場合，0を返します。

戻り値

年の値を返します。

参照

dataYearDA100

errorMaxLengthDA100Reader

構文

```
int errorMaxLengthDA100Reader(void);
```

宣言

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();
```

説明

エラーメッセージ文字列の最大長を取得します。

・ 戻り値に終端は含まれません。

戻り値

文字列の長さを返します。

参照

```
CDAQDA100Reader::getMaxLenErrorMessage
```

getAlarmNameDA100Reader [Visual Cのみ]

構文

```
const char * getAlarmNameDA100Reader(int iAlarmType);
```

引数

iAlarmType　アラーム種類を指定します。

説明

指定されたアラーム種類に対応する文字列を取得します。

・存在しない場合，「アラームなし」に対応する文字列へのポインタを返します。

戻り値

文字列へのポインタを返します。

参照

`CDAQDARWINDataInfo::getAlarmName`

getChannelUnitDA100Reader [Visual Cのみ]

構文

```
const char * getChannelUnitDA100Reader(DAQDA100READER
daqda100reader, int chType, int chNo);
```

引数

daqda100Reader	機器記述子を指定します。
chType	チャネルタイプを指定します。
chNo	チャネル番号を指定します。

説明

保持しているチャネル情報データから，指定されたチャネル(チャネルタイプ，チャネル番号)の単位名を取得します。

・ 存在しない場合，NULLを返します。

戻り値

文字列へのポインタを返します。

参照

`getChannelUnitDA100`

getErrorMessageDA100Reader [Visual Cのみ]

構文

```
const char * getErrorMessageDA100Reader(int errCode);
```

引数

errCode エラー番号を指定します。

説明

指定されたエラー番号に対応するエラーメッセージ文字列を取得します。

・ 存在しない場合，文字列「Unknown」へのポインタを返します。

戻り値

文字列へのポインタを返します。

参照

`CDAQDA100Reader::getErrorMessage`

revisionAPIDA100Reader

構文

```
const int revisionAPIDA100Reader(void);
```

宣言

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();
```

説明

本APIのリビジョン番号を取得します。

戻り値

リビジョン番号を返します。

参照

```
CDAQDA100Reader::getRevisionAPI
```

toAlarmNameDA100Reader

構文

```
int toAlarmNameDA100Reader(int iAlarmType, char * strAlarm,
int lenAlarm);
```

宣言

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);
```

引数

iAlarmType	アラーム種類を指定します。
strAlarm	文字列を格納する領域を指定します。
lenAlarm	文字列を格納する領域のバイト数を指定します。

説明

指定されたアラーム種類に対応する文字列を，指定された領域に格納します。

- 領域に格納する文字列には，終端(NULL)も含まれます。
- 実際の文字列の長さが戻り値になります。戻り値に終端は含まれません。
- 格納される文字列は，原則ascii文字列です。

戻り値

文字列の長さを返します。

参照

getAlarmNameDA100Reader

toChannelUnitDA100Reader

構文

```
int toChannelUnitDA100Reader(DAQDA100READER daqda100reader,
int chType, int chNo, char * strUnit, int lenUnit);
```

宣言

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);
```

引数

daqda100Reader	機器記述子を指定します。
chType	チャネルタイプを指定します。
chNo	チャネル番号を指定します。
strUnit	文字列を格納する領域を指定します。
lenUnit	文字列を格納する領域のバイト数を指定します。

説明

保持しているチャネル情報データから，指定されたチャネル(チャネルタイプ，チャネル番号)の単位名を取得します。

- 指定された格納先に文字列を格納します。
- 領域に格納する文字列には，終端(NULL)も含まれます。
- 実際の文字列の長さが戻り値になります。戻り値に終端は含まれません。
- 存在しない場合，0を返します。
- 格納される文字列は，原則ascii文字列です。

戻り値

文字列の長さを返します。

参照

getChannelUnitDA100Reader

toDoubleValueDA100Reader

構文

```
double toDoubleValueDA100Reader(int dataValue, int point);
```

宣言

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);
```

引数

dataValue	データ値を指定します。
point	小数点位置を指定します。

説明

指定されたデータ値と小数点位置から測定値を生成します。

戻り値

測定値を倍精度浮動小数で返します。

参照

```
CDAQDARWINDataInfo::toDoubleValue
```

toErrorMessageDA100Reader

構文

```
int toErrorMessageDA100Reader(int errCode, char * errStr, int
errLen);
```

宣言

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);
```

引数

errorCode	エラー番号を指定します。
errStr	文字列を格納する領域を指定します。
errLen	文字列を格納する領域のバイト数を指定します。

説明

エラー番号に対応するエラーメッセージ文字列を，指定された領域に格納します。

- 領域に格納する文字列には，終端(NULL)も含まれます。
- 実際の文字列の長さが戻り値になります。戻り値に終端は含まれません。
- 格納される文字列は，原則ascii文字列です。

戻り値

文字列の長さを返します。

参照

getErrorMessageDA100Reader

toStringValueDA100Reader

構文

```
int toStringValueDA100Reader(int dataValue, int point, char *
strValue, int lenValue);
```

宣言

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);
```

引数

dataValue	データ値を指定します。
point	小数点位置を指定します。
strValue	文字列を格納する領域を指定します。
lenValue	文字列を格納する領域のバイト数を指定します。

説明

指定されたデータ値と小数点位置から測定値を生成します。

- 生成された測定値を文字列に変換して，指定された領域に格納します。
- 領域に格納する文字列には，終端(NULL)も含まれます。
- 実際の文字列の長さが戻り値になります。戻り値に終端は含まれません。
- 格納される文字列は，原則ascii文字列です。

戻り値

文字列の長さを返します。

参照

`CDAQDARWINDataInfo::toStringValue`

versionAPIDA100Reader

構文

```
const int versionAPIDA100Reader(void);
```

宣言

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();
```

説明

本APIのバージョン番号を取得します。

戻り値

バージョン番号を返します。

参照

```
CDAQDA100Reader::getVersionAPI
```

25.1 DARWINの定数の概要

本拡張APIでは，以下の種類の定数を用意しています。

Visual C/Visual C++では，11.1節の定数を継承します。また，拡張API用に定数値，取得レコード種類，レンジ種類およびスキップレンジの定数が追加されています。25.2節を参照してください。
Visual Basic，Visual Basic.NET，C#の定数は25.2節に記載します。

種類	説明	ページ
定数値	ユニット内チャネル数など	25-2, 25-5
取得コード種類	バイナリコード，ASCIIコード	25-2, 25-5
通信用定数	DARWINの通信ポート番号	11-2, 25-5
個数値	サブユニット数など	11-2, 25-5
最大値	チャネル名文字列最大長など	11-2, 25-6
有効無効値	有効(ON)設定，無効(OFF)設定	11-2, 25-6
フラグステータス	データ取得時に最終データを判別	11-3, 25-6
データステータス値	測定データの状態	11-3, 25-6
アラーム種類	上限アラームなど	11-3, 25-7
チャネル/リレータイプ	チャネルやリレーの種類	11-4, 25-7
操作モード	運転，セットアップ，A/D構成モード	11-4, 25-7
トーカ機能種類	出力データに対応するトーカ	11-4, 25-8
ステータスバイト値	各種状態	11-5, 25-8
セットアップ確定	破棄，確定	11-5, 25-8
ユニット番号	拡張モデル，スタンドアロンモデル	11-5, 25-8
演算処理	演算のスタート，ストップ，クリアなど	11-5, 25-9
レポート実行種類	レポートのスタート，ストップ	11-5, 25-9
レポート種類	時報，日報，月報，ステータス	11-6, 25-9
レポートステータス	レポート種類の全無効，最新情報，有効	11-6, 25-9
レンジ種類		
直流電圧レンジ	20mVレンジなど	11-6, 25-3, 25-10
熱電対レンジ	Type Rなど	11-6, 25-3, 25-10
測温抵抗体レンジ	Pt100: 1mAなど	11-7, 25-3, 25-11
接点入力(DI)レンジ	電圧入力または接点入力	11-7, 25-4, 25-11
ひずみ入力レンジ	2k，20k，200k	11-7, 25-4, 25-11
パルスレンジ	GATE，RATE	11-7, 25-4, 25-11
パワーモニタレンジ	25V 0.5A，25V 5A，250V 0.5A，250V 5A	11-8, 25-4, 25-12
パワー接続方法	単相2線式など	11-8, 25-12
直流電流レンジ	20mA	11-8, 25-4, 25-12
SKIPレンジ	スキップ	25-4, 25-12
パワー接続方法	単相2線式など	11-8, 25-12
パワー測定項目	実効電流1など	11-9, 25-13

25.2 DARWINの定数

定数のニーモニックと意味を説明しています。DARWINの用語については，付録2をご覧ください。

Visual C/Visual C++の定数

Visual C/Visual C++では，11.2節の定数を継承しています。また，以下の定数を追加しています。

定数値

ニーモニック	内容
DAQDA100_NUMCH_BYUNIT	ユニット内チャネル数
DAQDA100_CHTYPE_MEASALL	全測定チャネルタイプ指定 (演算チャネルは含みません)
DAQDA100_CHNO_ALL	全チャネル番号指定
DAQDA100_LEVELNO_ALL	全アラームレベル指定

取得コード種類

ニーモニック	内容
DAQDA100_CODE_BINARY	バイナリコード
DAQDA100_CODE_ASCII	ASCIIコード

取得コード種類は，測定データを取得する場合の出力フォーマットの種類です。

レンジ種類

本拡張APIは，レンジを一意に識別するための定義をしています。
論理演算で合成されています。

ニーモニック	内容
DAQDA100_RANGETYPE_VOLT	直流電圧レンジ
DAQDA100_RANGETYPE_DI	接点レンジ
DAQDA100_RANGETYPE_TC	熱電対レンジ
DAQDA100_RANGETYPE_RTD	測温抵抗体レンジ
DAQDA100_RANGETYPE_SKIP	SKIPレンジ
DAQDA100_RANGETYPE_MA	直流電流レンジ
DAQDA100_RANGETYPE_POWER	パワーモニタレンジ
DAQDA100_RANGETYPE_STRAIN	ひずみ入力レンジ
DAQDA100_RANGETYPE_PULSE	パルスレンジ

レンジを指定する場合，上記種類と合成された以下に示す一意のレンジ種類を指定してください。

直流電圧レンジ

ニーモニック	内容	設定範囲
DAQDA100_RANGE_VOLT_20MV	20mV	−20.000～20.000 mV
DAQDA100_RANGE_VOLT_60MV	60mV	−60.00～60.00 mV
DAQDA100_RANGE_VOLT_200MV	200mV	−200.00～200.00 mV
DAQDA100_RANGE_VOLT_2V	2V	−2.0000～2.0000 V
DAQDA100_RANGE_VOLT_6V	6V	−6.000～6.000 V
DAQDA100_RANGE_VOLT_20V	20V	−20.000～20.000 V
DAQDA100_RANGE_VOLT_50V	50V	−50.00～50.00 V

熱電対レンジ

ニーモニック	内容	設定範囲
DAQDA100_RANGE_TC_R	R	0.0～1760.0 ℃
DAQDA100_RANGE_TC_S	S	0.0～1760.0 ℃
DAQDA100_RANGE_TC_B	B	0.0～1820.0 ℃
DAQDA100_RANGE_TC_K	K	−200.0～1370.0 ℃
DAQDA100_RANGE_TC_E	E	−200.0～800.0 ℃
DAQDA100_RANGE_TC_J	J	−200.0～1100.0 ℃
DAQDA100_RANGE_TC_T	T	−200.0～400.0 ℃
DAQDA100_RANGE_TC_N	N	0.0～1300.0 ℃
DAQDA100_RANGE_TC_W	W	0.0～2315.0 ℃
DAQDA100_RANGE_TC_L	L	−200.0～900.0 ℃
DAQDA100_RANGE_TC_U	U	−200.0～400.0 ℃
DAQDA100_RANGE_TC_KP	KpAu7Fe	0.0～300.0 K

測温抵抗体レンジ

ニーモニック	内容	設定範囲
DAQDA100_RANGE_RTD_1MAPT	Pt100:1mA	−200.0～600.0 ℃
DAQDA100_RANGE_RTD_2MAPT	Pt100:2mA	−200.0～250.0 ℃
DAQDA100_RANGE_RTD_1MAJPT	JPt100:1mA	−200.0～550.0 ℃
DAQDA100_RANGE_RTD_2MAJPT	JPt100:2mA	−200.0～250.0 ℃
DAQDA100_RANGE_RTD_2MAPT50	Pt50:2mA	−200.0～550.0 ℃
DAQDA100_RANGE_RTD_1MAPTH	Pt100:1mA−H	−140.00～150.00 ℃
DAQDA100_RANGE_RTD_2MAPTH	Pt100:2mA−H	−70.00～70.00 ℃
DAQDA100_RANGE_RTD_1MAJPTH	JPt100:1mA−H	−140.00～150.00 ℃
DAQDA100_RANGE_RTD_2MAJPTH	JPt100:2mA−H	−70.00～70.00 ℃
DAQDA100_RANGE_RTD_1MANIS	Ni100:1mA−S	−200.0～250.0 ℃
DAQDA100_RANGE_RTD_1MANID	Ni100:1mA−D	−60.0～180.0 ℃
DAQDA100_RANGE_RTD_1MANI120	Ni120:1mA	−70.0～200.0 ℃
DAQDA100_RANGE_RTD_CU10GE	Cu10:GE	−200.0～300.0 ℃
DAQDA100_RANGE_RTD_CU10LN	Cu10:L&N	−200.0～300.0 ℃
DAQDA100_RANGE_RTD_CU10WEED	Cu10:WEED	−200.0～300.0 ℃
DAQDA100_RANGE_RTD_CU10BAILEY	Cu10:BAILEY	−200.0～300.0 ℃
DAQDA100_RANGE_RTD_J263B	J263*B	−0.0～300.0 K

接点入力(DI)レンジ

ニーモニック	内容	設定範囲
DAQDA100_RANGE_DI_LEVEL	電圧入力	0：2.4V未満，1：2.4V以上
DAQDA100_RANGE_DI_CONTACT	接点入力	0：open，1：close

ひずみ入力レンジ

ニーモニック	内容	設定範囲
DAQDA100_RANGE_STRAIN_2K	2k	−2000～2000μひずみ(1ゲージ法) −1000～1000μひずみ(2ゲージ法) −500～500μひずみ(4ゲージ法)
DAQDA100_RANGE_STRAIN_20K	20k	−20000～20000μひずみ(1ゲージ法) −10000～10000μひずみ(2ゲージ法) −5000～5000μひずみ(4ゲージ法)
DAQDA100_RANGE_STRAIN_200K	200k	−200000～200000μひずみ(1ゲージ法) −100000～100000μひずみ(2ゲージ法) −50000～50000μひずみ(4ゲージ法)

パルスレンジ

ニーモニック	内容	設定範囲
DAQDA100_RANGE_PULSE_RATE	RATE	0 ～ 30000
DAQDA100_RANGE_PULSE_GATE	GATE	0 ～ 30000

パワーモニタレンジ

ニーモニック	内容	設定範囲
DAQDA100_RANGE_POWER_25V05A	25V 0.5A	電圧25V，電流0.5A
DAQDA100_RANGE_POWER_25V5A	25V 5A	電圧25V，電流5A
DAQDA100_RANGE_POWER_250V05A	250V 0.5A	電圧250V，電流0.5A
DAQDA100_RANGE_POWER_250V5A	250V 5A	電圧250V，電流5A

直流電流レンジ

ニーモニック	内容	設定範囲
DAQDA100_RANGE_MA_20MA	20mA	−20.000～20.000mA

SKIPレンジ

ニーモニック	内容
DAQDA100_RANGE_SKIP SKIP	スキップ(未使用)レンジ

Visual Basic，Visual Basic.NET，C#の定数

定数のニーモニックと意味を説明しています。DARWINの機能の詳細については，それぞれのユーザーズマニュアルを参照してください。

C#では，DAQDA100クラスの定数データになります。各定数の前に「DAQDA100.」をつけてご使用ください(例　DAQDA100.DAQDA100_NUMCHANNEL)。

定数値

ニーモニック	内容
DAQDA100_NUMCH_BYUNIT	ユニット内チャネル数
DAQDA100_CHTYPE_MEASALL	全測定チャネルタイプ指定(演算チャネルは含みません)
DAQDA100_CHNO_ALL	全チャネル番号指定
DAQDA100_LEVELNO_ALL	全アラームレベル指定

取得コード種類

ニーモニック	内容
DAQDA100_CODE_BINARY	バイナリコード
DAQDA100_CODE_ASCII	ASCIIコード

取得コード種類は，測定データを取得する場合の出力フォーマットの種類です。

通信用定数

ニーモニック	内容
DAQDA100_COMMPORT	DARWINの通信ポート番号です。

個数値

ニーモニック	内容
DAQDA100_NUMCHANNEL	チャネル個数です。
DAQDA100_NUMALARM	アラーム個数です。
DAQDA100_NUMUNIT	サブユニット個数です。
DAQDA100_NUMSLOT	サブユニットごとのスロット個数です。
DAQDA100_NUMTERM	スロット(モジュール)ごとの端子個数です。

個数値には，対象となるモジュールやユニットのそれぞれの個数値が設定されます。

最大値

ニーモニック	内容
DAQDA100_MAXCHNAMELEN	チャネル名文字列最大長です。
DAQDA100_MAXCHRANGLEN	チャネル範囲名文字列最大長です。
DAQDA100_MAXUNITLEN	単位名文字列最大長です。
DAQDA100_MAXMODULELEN	モジュール名文字列最大長です。
DAQDA100_MAXRELAYLEN	リレー名文字列最大長です。チャネル名文字列最大長と同じです。リレーとは，アラーム出力モジュールまたはDI/DOモジュールの出力リレーのことです。
DAQDA100_MAXDECIMALPOINT	小数点位置の最大値です。

文字列の最大長は，終端(NULL)を含みません。

有効無効値

ニーモニック	内容
DAQDA100_VALID_OFF	無効(OFF)値
DAQDA100_VALID_ON	有効(ON)値

フラグステータス

ニーモニック	内容
DAQDA100_FLAG_OFF	全OFF。
DAQDA100_FLAG_ENDDATA	ASCIIコードや行単位で取得するデータ行が最終データです。

論理OR演算で合成できます。

データステータス値

ニーモニック	内容
DAQDA100_UNKNWON	データステータスがセットされていない状態です。
DAQDA100_DATA_NORMAL	正常です。
DAQDA100_DATA_DIFFINPUT	チャネル間差演算状態です。
DAQDA100_DATA_PLUSOVER	プラスオーバ状態です。
DAQDA100_DATA_MINUSOVER	マイナスオーバ状態です。
DAQDA100_DATA_SKIP	スキップ(未使用)状態です。
DAQDA100_DATA_ILLEGAL	不明な不正データ状態です。
DAQDA100_DATA_ABNORMAL	異常データ状態です。
DAQDA100_DATA_NODATA	データなし状態です。
DAQDA100_DATA_READER	瞬時値データ読み込み通信時の状態です。

瞬時値データ読み込み通信時の状態は，瞬時値データ読み込み通信ポート使用時，チャネル情報データ取得によるチャネルステータスです。

アラーム種類

□はスペースを示します。

ニーモニック	内容	文字列
DAQDA100_ALARM_NONE	アラームなし(アラームOFF)	□□
DAQDA100_ALARM_UPPER	上限アラーム	H□
DAQDA100_ALARM_LOWER	下限アラーム	L□
DAQDA100_ALARM_UPDIFF	差上限アラーム	dH
DAQDA100_ALARM_LOWDIFF	差下限アラーム	dL
DAQDA100_ALARM_INCRATE	変化率上昇限アラーム	RH
DAQDA100_ALARM_DECRATE	変化率下降限アラーム	RL

チャネル/リレータイプ

チャネル，リレー，通信入力，演算定数のタイプ値です。チャネルやリレーを指定するときに使用できます。

ニーモニック	内容	チャネル	リレー
DAQDA100_CHTYPE_MAINUNIT	拡張モデルのメインユニットを表す値です。	-	あり
DAQDA100_CHTYPE_STANDALONE	スタンドアロンモデルのユニットを表す値です。サブユニット番号の0と同じです。	あり	あり
DAQDA100_CHTYPE_MATHTYPE	演算チャネルを表す値です。	あり	-
DAQDA100_CHTYPE_SWITCH	内部スイッチを表す値です。	-	あり
DAQDA100_CHTYPE_COMMDATA	通信入力を表す値です。	-	-
DAQDA100_CHTYPE_CONSTANT	演算定数を表す値です。	-	-
DAQDA100_CHTYPE_REPORT	レポートを表す値です。	-	-

あり　：そのタイプのチャネル/リレーが存在します。
-　　：そのタイプのチャネル/リレーはありません。

Note

拡張モデルに接続されるサブユニットを識別するサブユニット番号もタイプ値です。サブユニット番号は0から5の整数値です。付録2を参照してください。

操作モード

ニーモニック	内容
DAQDA100_MODE_OPE	運転モード
DAQDA100_MODE_SETUP	セットアップモード
DAQDA100_MODE_CALIB	A/D校正モード

トーカ機能種類

ニーモニック	内容
DAQDA100_TALK_MEASUREDDATA	測定データ，演算データの出力
DAQDA100_TALK_OPEDATA	運転モードの設定データの出力
DAQDA100_TALK_CHINFODATA	チャネル情報データの出力
DAQDA100_TALK_SYSINFODATA	システム構成データの出力
DAQDA100_TALK_CALIBDATA	校正データ(A/D校正モードの設定データ)の出力
DAQDA100_TALK_SETUPDATA	セットアップモードの設定データの出力
DAQDA100_TALK_REPORTDATA	レポートステータスの出力

ステータスバイト値

論理OR演算で合成できます。

ニーモニック	内容
DAQDA100_STATUS_OFF	全ステータスバイトが無効の場合の値
DAQDA100_STATUS_ADCONV	A/D変換終了
DAQDA100_STATUS_SYNTAX	コマンド文法エラー
DAQDA100_STATUS_TIMER	内部タイマ起動/レポート作成
DAQDA100_STATUS_MEDIA	メディアへのアクセス(DC100)
DAQDA100_STATUS_RELEASE	演算中の測定抜け
DAQDA100_STATUS_ALL	全ステータスバイトを有効にするマスク値
DAQDA100_STATUS_SRQ	SRQ

ステータスバイト値の意味の詳細については，DARWIN機器の通信インターフェースユーザーズマニュアルをご覧ください。

セットアップ確定

ニーモニック	内容
DAQDA100_SETUP_ABORT	破棄
DAQDA100_SETUP_STORE	確定

ユニット番号

ニーモニック	内容
DAQDA100_UNITNO_MAINUNIT	拡張モデルのメインユニット
DAQDA100_UNITNO_STANDALONE	スタンドアロンモデルのユニット

サブユニット番号は，数値です。 チャネル/リレータイプを参照。

演算処理

ニーモニック	内容
DAQDA100_COMPUTE_START	演算のスタート
DAQDA100_COMPUTE_STOP	演算のストップ
DAQDA100_COMPUTE_RESTART	演算データクリア後，再度スタート
DAQDA100_COMPUTE_CLEAR	演算データクリア
DAQDA100_COMPUTE_RELEASE	測定抜けのステータス表示解除

レポート実行種類

ニーモニック	内容
DAQDA100_REPORT_RUN_START	レポートのスタート
DAQDA100_REPORT_RUN_STOP	レポートのストップ

レポート種類

ニーモニック	内容
DAQDA100_REPORT_HOURLY	時報
DAQDA100_REPORT_DAILY	日報
DAQDA100_REPORT_MONTHLY	月報
DAQDA100_REPORT_STATUS	ステータス

レポートステータス

論理OR演算で合成できます。

ニーモニック	内容
DAQDA100_REPSTATUS_NONE	全無効
DAQDA100_REPSTATUS_HOURLY_NEW	最新時報
DAQDA100_REPSTATUS_HOURLY_VALID	時報の有効
DAQDA100_REPSTATUS_DAILY_NEW	最新日報
DAQDA100_REPSTATUS_DAILY_VALID	日報の有効
DAQDA100_REPSTATUS_MONTHLY_NEW	最新月報
DAQDA100_REPSTATUS_MONTHLY_VALID	月報の有効

レンジ種類

本拡張APIは，レンジを一意に識別するための定義をしています。
論理演算で合成されています。

ニーモニック	内容
DAQDA100_RANGETYPE_VOLT	直流電圧レンジ
DAQDA100_RANGETYPE_DI	接点レンジ
DAQDA100_RANGETYPE_TC	熱電対レンジ
DAQDA100_RANGETYPE_RTD	測温抵抗体レンジ
DAQDA100_RANGETYPE_SKIP	SKIPレンジ
DAQDA100_RANGETYPE_MA	直流電流レンジ
DAQDA100_RANGETYPE_POWER	パワーモニタレンジ
DAQDA100_RANGETYPE_STRAIN	ひずみ入力レンジ
DAQDA100_RANGETYPE_PULSE	パルスレンジ

レンジを指定する場合，上記種類と合成された以下に示す一意のレンジ種類を指定してください。

直流電圧レンジ

ニーモニック	内容	設定範囲
DAQDA100_RANGE_VOLT_20MV	20mV	−20.000～20.000 mV
DAQDA100_RANGE_VOLT_60MV	60mV	−60.00～60.00 mV
DAQDA100_RANGE_VOLT_200MV	200mV	−200.00～200.00 mV
DAQDA100_RANGE_VOLT_2V	2V	−2.0000～2.0000 V
DAQDA100_RANGE_VOLT_6V	6V	−6.000～6.000 V
DAQDA100_RANGE_VOLT_20V	20V	−20.000～20.000 V
DAQDA100_RANGE_VOLT_50V	50V	−50.00～50.00 V

熱電対レンジ

ニーモニック	内容	設定範囲
DAQDA100_RANGE_TC_R	R	0.0～1760.0 ℃
DAQDA100_RANGE_TC_S	S	0.0～1760.0 ℃
DAQDA100_RANGE_TC_B	B	0.0～1820.0 ℃
DAQDA100_RANGE_TC_K	K	−200.0～1370.0 ℃
DAQDA100_RANGE_TC_E	E	−200.0～800.0 ℃
DAQDA100_RANGE_TC_J	J	−200.0～1100.0 ℃
DAQDA100_RANGE_TC_T	T	−200.0～400.0 ℃
DAQDA100_RANGE_TC_N	N	0.0～1300.0 ℃
DAQDA100_RANGE_TC_W	W	0.0～2315.0 ℃
DAQDA100_RANGE_TC_L	L	−200.0～900.0 ℃
DAQDA100_RANGE_TC_U	U	−200.0～400.0 ℃
DAQDA100_RANGE_TC_KP	KpAu7Fe	0.0～300.0 K

測温抵抗体レンジ

ニーモニック	内容	設定範囲
DAQDA100_RANGE_RTD_1MAPT	Pt100:1mA	−200.0～600.0 ℃
DAQDA100_RANGE_RTD_2MAPT	Pt100:2mA	−200.0～250.0 ℃
DAQDA100_RANGE_RTD_1MAJPT	JPt100:1mA	−200.0～550.0 ℃
DAQDA100_RANGE_RTD_2MAJPT	JPt100:2mA	−200.0～250.0 ℃
DAQDA100_RANGE_RTD_2MAPT50	Pt50:2mA	−200.0～550.0 ℃
DAQDA100_RANGE_RTD_1MAPTH	Pt100:1mA−H	−140.00～150.00 ℃
DAQDA100_RANGE_RTD_2MAPTH	Pt100:2mA−H	−70.00～70.00 ℃
DAQDA100_RANGE_RTD_1MAJPTH	JPt100:1mA−H	−140.00～150.00 ℃
DAQDA100_RANGE_RTD_2MAJPTH	JPt100:2mA−H	−70.00～70.00 ℃
DAQDA100_RANGE_RTD_1MANIS	Ni100:1mA−S	−200.0～250.0 ℃
DAQDA100_RANGE_RTD_1MANID	Ni100:1mA−D	−60.0～180.0 ℃
DAQDA100_RANGE_RTD_1MANI120	Ni120:1mA	−70.0～200.0 ℃
DAQDA100_RANGE_RTD_CU10GE	Cu10:GE	−200.0～300.0 ℃
DAQDA100_RANGE_RTD_CU10LN	Cu10:L&N	−200.0～300.0 ℃
DAQDA100_RANGE_RTD_CU10WEED	Cu10:WEED	−200.0～300.0 ℃
DAQDA100_RANGE_RTD_CU10BAILEY	Cu10:BAILEY	−200.0～300.0 ℃
DAQDA100_RANGE_RTD_J263B	J263*B	−0.0～300.0 K

接点入力(DI)レンジ

ニーモニック	内容	設定範囲
DAQDA100_RANGE_DI_LEVEL	電圧入力	0：2.4V未満，1：2.4V以上
DAQDA100_RANGE_DI_CONTACT	接点入力	0：open，1：close

ひずみ入力レンジ

ニーモニック	内容	設定範囲
DAQDA100_RANGE_STRAIN_2K	2k	−2000～2000μひずみ(1ゲージ法) −1000～1000μひずみ(2ゲージ法) −500～500μひずみ(4ゲージ法)
DAQDA100_RANGE_STRAIN_20K	20k	−20000～20000μひずみ(1ゲージ法) −10000～10000μひずみ(2ゲージ法) −5000～5000μひずみ(4ゲージ法)
DAQDA100_RANGE_STRAIN_200K	200k	−200000～200000μひずみ(1ゲージ法) −100000～100000μひずみ(2ゲージ法) −50000～50000μひずみ(4ゲージ法)

パルスレンジ

ニーモニック	内容	設定範囲
DAQDA100_RANGE_PULSE_RATE	RATE	0 ～ 30000
DAQDA100_RANGE_PULSE_GATE	GATE	0 ～ 30000

パワーモニタレンジ

ニーモニック	内容	設定範囲
DAQDA100_RANGE_POWER_25V05A	25V 0.5A	電圧25V，電流0.5A
DAQDA100_RANGE_POWER_25V5A	25V 5A	電圧25V，電流5A
DAQDA100_RANGE_POWER_250V05A	250V 0.5A	電圧250V，電流0.5A
DAQDA100_RANGE_POWER_250V5A	250V 5A	電圧250V，電流5A

直流電流レンジ

ニーモニック	内容	設定範囲
DAQDA100_RANGE_MA_20MA	20mA	−20.000～20.000mA

SKIPレンジ

ニーモニック	内容
DAQDA100_RANGE_SKIP SKIP	(未使用)

パワー接続方法

ニーモニック	内容
DAQDA100_WIRE_1PH2W	単相2線式
DAQDA100_WIRE_1PH3W	単相3線式(3線式用だけ)
DAQDA100_WIRE_3PH3W2I	3相3線式(2電圧2電流 3線式用だけ)
DAQDA100_WIRE_3PH3W3I	3相3線式(3電圧3電流 3線式用だけ)
DAQDA100_WIRE_3PH4W	3相4線式(3線式用だけ)

パワー測定項目

ニーモニック	内容
DAQDA100_POWERITEM_I0	(I1+I2+I3)/3
DAQDA100_POWERITEM_I1	実効電流1
DAQDA100_POWERITEM_I2	実効電流2
DAQDA100_POWERITEM_I3	実効電流3
DAQDA100_POWERITEM_I13	(I1+I3)/2
DAQDA100_POWERITEM_P0	P1+P2+P3
DAQDA100_POWERITEM_P1	有効電力1
DAQDA100_POWERITEM_P2	有効電力2
DAQDA100_POWERITEM_P3	有効電力3
DAQDA100_POWERITEM_P13	P1+P3
DAQDA100_POWERITEM_PF0	$P0/(P0^2+VAR0^2)^{1/2}=P0/VA0$
DAQDA100_POWERITEM_PF1	力率1
DAQDA100_POWERITEM_PF2	力率2
DAQDA100_POWERITEM_PF3	力率3
DAQDA100_POWERITEM_PF13	$P13/(P13^2+VAR132)^{1/2}=P13/VA13$
DAQDA100_POWERITEM_PH0	$\tan^{-1}(VAR0/P0)$
DAQDA100_POWERITEM_PH1	位相1
DAQDA100_POWERITEM_PH2	位相2
DAQDA100_POWERITEM_PH3	位相3
DAQDA100_POWERITEM_PH13	$\tan^{-1}(VAR13/P13)$
DAQDA100_POWERITEM_V0	(V1+V2+V3)/3
DAQDA100_POWERITEM_V1	実効電力1
DAQDA100_POWERITEM_V2	実効電力2
DAQDA100_POWERITEM_V3	実効電力3
DAQDA100_POWERITEM_V13	(V1+V3)/2
DAQDA100_POWERITEM_VA0	VA1+VA2+VA3
DAQDA100_POWERITEM_VA1	皮相電力1
DAQDA100_POWERITEM_VA2	皮相電力2
DAQDA100_POWERITEM_VA3	皮相電力3
DAQDA100_POWERITEM_VA13	VA1+VA3
DAQDA100_POWERITEM_VAR0	VAR1+VAR2+VAR3
DAQDA100_POWERITEM_VAR1	無効電力1
DAQDA100_POWERITEM_VAR2	無効電力2
DAQDA100_POWERITEM_VAR3	無効電力3
DAQDA100_POWERITEM_VAR13	VAR1+VAR3
DAQDA100_POWERITEM_FREQ	周波数

25.3 DARWINの型

DAQDA100

本機能用の機器記述子を格納するための型です。
Visual BasicではLong型，Visual Cではintで扱います。Visual Basic.NETではInteger型で扱います。 C#ではint型で扱います。

コールバック型

型	説明
コールバック型	関数名に接頭辞「DLL」を付加し，大文字で記述します。 例. openDA100関数のコールバック型：DLLOPENDA100

コールバック型は，Visual Cを使用のときに，実行可能モジュール(.dll)とリンクさせるために使用します。

25.4 瞬時値データ読み込み用DARWINの定数の概要

以下の種類の定数を用意しています。

種類	説明	ページ
定数値	ユニット内チャネル数など	25-2，25-17
個数値	サブユニット数など	11-2，25-17
最大値	チャネル名文字列最大長など	11-2，25-17
有効無効値	有効(ON)設定，無効(OFF)設定	11-2，25-17
通信用定数	瞬時値データの読み込み用ポート番号	25-16，25-18
データステータス値	測定データの状態	11-3，25-18
アラーム種類	上限アラームなど	11-3，25-18
チャネルタイプ/リレータイプ	チャネルやリレーの種類	11-4，25-19
ユニット番号	拡張モデル，スタンドアロンモデル	11-5，25-19

25.5 瞬時値データ読み込み用DARWINの定数

Visual C/Visual C++の定数

Visual C，Visual C++では，11.2節と25.2節の定数を継承しています。
以下の定数を追加しています。

通信用定数

ニーモニック	内容
DAQDA100READER_DATAPORT	瞬時値データ読み込み用ポート番号

Visual Basic，Visual Basic.NET，C#の定数

定数のニーモニックと意味を説明しています。DARWINの機能の詳細については，それぞれのユーザーズマニュアルを参照してください。

C#では，DAQDA100Readerクラスの定数データになります。各定数の前に「DAQDA100Reader.」をつけて使用してください。

(例　DAQDA100Reader.DAQDA100READER_NUMCHANNEL)

定数値

ニーモニック	内容
DAQDA100_CHTYPE_MEASALL	全測定チャネルタイプ指定 （演算チャネルは含みません）
DAQDA100_CHNO_ALL	全チャネル番号指定

個数値

ニーモニック	内容
DAQDA100READER_NUMCHANNEL	チャネル個数です。
DAQDA100READER_NUMALARM	アラーム個数です。
DAQDA100READER_NUMUNIT	サブユニット個数です。
DAQDA100READER_NUMSLOT	サブユニット毎のスロット個数です。
DAQDA100READER_NUMTERM	スロット(モジュール)毎の端子個数です。

最大値

ニーモニック	内容
DAQDA100READER_MAXUNITLEN	単位名文字列最大長です。

文字列の最大長は，終端(NULL)を含みません。

有効無効値

ニーモニック	内容
DAQDA100READER_VALID_OFF	無効(OFF)値
DAQDA100READER_VALID_ON	有効(ON)値

通信用定数

ニーモニック	内容
DAQDA100READER_DATAPORT	瞬時値データ読み込み用ポート番号

データステータス値

ニーモニック	内容
DAQDA100READER_UNKNWON	不明。 データステータスがセットされていない状態です。
DAQDA100READER_DATA_NORMAL	ノーマル状態です。
DAQDA100READER_DATA_DIFFINPUT	差入力状態です。
DAQDA100READER_DATA_PLUSOVER	プラスオーバ状態です。
DAQDA100READER_DATA_MINUSOVER	マイナスオーバ状態です。
DAQDA100READER_DATA_SKIP	スキップ状態です。
DAQDA100READER_DATA_ILLEGAL	不明な不正データ状態です。
DAQDA100READER_DATA_ABNORMAL	異常データ状態です。
DAQDA100READER_DATA_NODATA	データなし状態です。
DAQDA100READER_DATA_READER	瞬時値データ読み込み 通信時の状態です。

瞬時値データ読み込み通信時の状態は，瞬時値データ読み込み通信ポート使用時，チャネル情報データ取得によるチャネルステータスです。

アラーム種類

□はスペースを示します。

ニーモニック	内容	文字列
DAQDA100READER_ALARM_NONE	アラームなし (アラームOFF)	□□
DAQDA100READER_ALARM_UPPER	上限アラーム	H□
DAQDA100READER_ALARM_LOWER	下限アラーム	L□
DAQDA100READER_ALARM_UPDIFF	差上限アラーム	dH
DAQDA100READER_ALARM_LOWDIFF	差下限アラーム	dL
DAQDA100READER_ALARM_INCRATE	変化率上昇限アラーム	RH
DAQDA100READER_ALARM_DECRATE	変化率下降限アラーム	RL

チャネル/リレータイプ

ニーモニック	内容
DAQDA100READER_CHTYPE_MAINUNIT	拡張モデルのメインユニットを表す値です。
DAQDA100READER_CHTYPE_STANDALONE	スタンドアロンモデルのユニットを表す値です。 サブユニット番号の0と同じです。
DAQDA100READER_CHTYPE_MATHTYPE	演算を表す値です。

ユニット番号

ニーモニック	内容
DAQDA100READER_UNITNO_MAINUNIT	拡張モデルのメインユニット
DAQDA100READER_UNITNO_STANDALONE	スタンドアロンモデルのユニット

サブユニット番号は，数値です。 チャネル/リレータイプを参照。

25.6 瞬時値データ読み込み用DARWINの型

DAQDA100READER

本機能用の機器記述子を格納するための型です。
Visual BasicではLong型，Visual Cではintで扱います。Visual Basic.NETではInteger型で扱います。 C#ではint型で扱います。

コールバック型

型	説明
コールバック型	関数名に接頭辞「DLL」を付加し，大文字で記述します。 例. ｏｐｅｎＤＡ１００Ｒｅａｄｅｒ関数のコールバック型： DLLOPENDA100READER

コールバック型は，Visual Cを使用のときに，実行可能モジュール(.dll)とリンクさせるために使用します。

26.1 APIによるエラーメッセージ

関数が実行処理に失敗した場合，エラー番号を返します。以下は，エラー番号とメッセージ文字列，および対処方法の一覧です。MX100とDARWINで共通です。

エラー番号	メッセージ文字列 説明 対処，備考
0	Success 正常終了しました。 -
1	Communication error 通信エラーが発生しました。 通信環境(アドレス，ケーブル，機器の電源等)を確認してください。
2	Timeout タイムアウトが発生しました。 通信環境(負荷状態など)を確認してください。
3	Receive continue 受信データが継続しています。 受信するデータが長すぎます。残りのデータを受信するか，通信手順を確認してください。
4	Creating connection is failure 通信記述子の作成に失敗しました。 メモリやリソース不足が考えられます。PCの環境を確認してください。
5	Creating descriptor is failure 機器記述子の作成に失敗しました。 メモリやリソース不足が考えられます。PCの環境を確認してください。
6	Connection exists already 既に通信が確立しています。 通信接続されているところに，通信を接続しないでください。
7	Not connected 通信接続されていません。 通信接続を実行せずにコマンドを実行していると考えられます。通信接続してから実行してください。
8	Not descriptor 機器記述子がありません。 機器記述子の指定が間違っていると考えられます。機器記述子を指定してください。
9	Commands are not processed successfully コマンド実行処理に失敗しました。 測定機器本体において，エラーが発生しました。送信したコマンド，または，本体の操作モードを確認してください。

エラー番号	メッセージ文字列 説明 対処，備考
10	Not acknowledge 対応していない応答を受信しました。 測定機器本体からの応答が，予想される応答と異なると考えられます。送信したコマンドや手順を確認してください。
11	Not support サポートしていない機能が指定されました。 範囲外の値を指定したことが考えられます。関数に与えた引数の値を確認してください。
12	Not data データがありません。 関数の入力指定が不正です。引数で指定されたデータが間違っていることが考えられます。または，文字列の場合，間違っていることが考えられます。関数に与えた引数の値を確認してください。
13	Exception 例外を検出しました。 システム例外が発生したり，領域確保に失敗したことなどが考えられます。PCの環境を確認してください。

26.2 MX100固有エラーメッセージ

MX100が発するエラーの値，説明，および対処方法の一覧です。
関数(CDAQMX::getLastError，getLastErrorMX)で値を取得できます。
拡張APIでは，関数(DAQMX100::getLastError，lastErrorMX100)で値を取得できます。

値	説明 対処方法，備考
0	エラーはありません。 -
1	Versionが異なるため対応できません。 MX100とMXAPIのバージョンを確認してください。
2	パケットサイズが大きすぎます。 パケットサイズを確認してください。
3	不明なリクエストです。 リクエストを確認してください。
4	整合が取れていないリクエストです。 パケットの構成を確認してください。
6	セッション番号が間違っています。 セッション番号を確認してください。
7	FIFO番号が間違っています。 FIFO番号を確認してください。
8	チャネル番号が間違っています。 チャネル番号を確認してください。
9	指定範囲にデータが存在しません。 データ番号を確認してください。
10	設定に失敗しました。 モジュールの種類，状態を確認してください。
11	ステータスが原因で失敗しました。 動作モードを確認してください。アイドリングモードで実施してください。
12	DOに対する不適切なリクエストです。 DO出力設定を確認してください。
13	CFカードがありません。 CFカードを装着してください。
14	CFカードをフォーマットできません(CFカードは装着されています)。 CFカードが故障している可能性があります。CFカードを交換してください。
23	初期バランスが正しく行えませんでした。モジュールの装着状態を確認して，もう一度やり直してください。それでもエラーが発生する場合は，お買い求め先までご連絡ください。
255	通信エラー。 通信エラーです。お買い求め先までご連絡ください。
256	その他のエラー。 上記以外のエラーです。お買い求め先までご連絡ください。

付録1 MX100に関する用語

本ソフトウエアの用語とMX100の用語について説明しています。大項目はアルファベット順，五十音順に並んでいます。
詳細は，MX100本体の取扱説明書をご覧ください。

7セグメントLED

7セグメントLEDの表示を表すデータです。
MXSegment構造体を参照してください。
MX100には，2個の7セグメントLEDが付いています。必ず，2個の7セグメントLEDを組にして扱います。

セグメント番号

7セグメントLEDの位置を識別するための値です。
0から1の整数値です。
個数値が定義されています。

表示形式

全7セグメントLEDの表示状態を表示形式値で表します。

表示時間

表示パターンの表示時間です。
単位は，ミリ秒です。
最大値を超えることはできません。

表示パターン

各セグメントの表示パターンの値です。
0からFの16進数整数値です。
範囲外の値の場合，無表示パターンになります。

AO/PWMデータ

AO/PWMチャネルの出力を表すデータです。
MXAOPWM構造体を参照してください。
MXAOPWMData構造体は，全チャネルを集約したものです。
有効無効値と出力データ値で構成されます。
データ取得した場合，コマンドAO/PWMのチャネルを有効無効値で表します。
データ送信する場合，「有効値」指定されたチャネルだけが送信されます。

出力データ値

機器本体が認識する，出力値を表すデータ値です。
レンジ種類により異なります。
実際の出力値と異なる値であるため，ユーティリティの出力値と出力データ値の変換関数を使用して求めます。
データ値やスパンと異なることに注意してください。

AO/PWMデータ番号(AOデータ番号，PWMデータ番号)

AO，または，PWMチャネルの位置を識別するための値です。
1から60の整数値です。
個数値が定義されています。

DOデータ

DOチャネルの出力を表すデータです。
MXDO構造体を参照してください。
MXDOData構造体は，全チャネルを集約したものです。
有効無効値とON/OFFで構成されます。
データ取得した場合，コマンドDOのチャネルを有効無効値で表します。
データ送信する場合，「有効値」指定されたチャネルだけが送信されます。

DOデータ番号

DOチャネルの位置を識別するための値です。
1から60の整数値です。
個数値が定義されています。

FIFO

測定データをFIFOバッファに書き込む動作です。
MX100では，FIFOを開始しなければ，測定データを収集することができません。
ユーザがFIFOの開始/停止を指定するとこが可能です。また，FIFOの自動制御を指定することができます。
MX100の場合，FIFOは測定周期種類別に分かれています。
FIFO番号で識別します。
測定周期とFIFOの構成は以下のとおりです。

- 同じ測定周期の入力モジュールの測定データは，同一のFIFOにチャネル番号順に収録されます。
- 測定周期の速いものからインターバル番号(FIFO番号)が割り付けられます。
- FIFO(Data Buffer)への書き込みは測定ごとに更新されるデータ番号により管理されます。
- データの読み出しはFIFO番号とデータ番号を指定して行います。
- 読み出し可能なデータ範囲は，データ番号により提供されます。

FIFO番号

FIFOにつけられた番号です。
測定周期種類が速い順に番号が割り振られます。
0から2の整数値です。
個数値が定義されています。

自動制御

FIFO中に設定コマンドを実行すると，FIFOを停止します。
自動制御が「有効値」に設定されている場合，各設定のコマンド実行後，自動的にFIFOを開始をします。

データ番号

格納されているデータに測定順につけられたシーケンシャルな番号です。
FIFO番号ごとに異なります。

RJC電圧値

外部のRJC機能を使用するときの補償電圧値です。
単位はμVの整数値です。

アラーム

アラーム機能です。
MXAlarm構造体を参照してください。

アラームレベル

チャネル内のアラーム機能の識別番号です。
1から2の整数値です。
個数値が定義されています。

アラーム値

アラーム発生のしきい値(On値)です。
小数点を除いた整数値です。

ヒステリシス

アラーム値と，アラーム停止のしきい値(Off値)に差を持たせるときの差分です。
小数点を除いた正の整数値です。

応答

要求コマンドに対して，対応する返信が存在します。応答は，次のいずれかになります。

・ 処理が正常に行われた
・ 処理が正常に行われなかった

処理が正常に行われず，コマンド実行処理に失敗した関数は，エラー番号を返します。詳細なエラー番号は，「MX固有エラー」で表されます。別途取得することができます。

機器記述子

測定機器本体を識別するための値です。
各関数を実行する場合，この機器記述子が必要になります。
APIの場合，実体は，CDAQMXオブジェクトへの参照になります。DAQMX型で格納，指定をします。
拡張APIの場合，実体は，CDAQMX100オブジェクトへの参照になります。
DAQMX100型で格納，指定をします。

基準チャネル番号

AI/DIチャネルで差演算，またはAIチャネルでリモートRJCの場合，基準として参照するチャネル番号です。
AO/PWMチャネルで伝送出力の場合，伝送元のチャネル番号です。

参照アラーム

DOチャネルが，どのアラームに反応すべきかを示します。チャネル番号とアラームレベルでアラームを指定します。

時刻情報データ

計測時刻を示す日付時刻のデータです。
原則，1970年01月01日からの秒数です。
MXDateTime構造体を参照してください。
ミリ秒の値も含みます。

システム構成データ

システムは複数のモジュールから構成されます。
MXSystemInfo構造体を参照してください。
以下の項目から構成されています。
・ ユニット情報
・ 各モジュール情報

出力チャネルデータ

出力チャネル(AO/PWMチャネル)で，出力されるデータの制御方法を指定するデータです。
指定した動作の詳細は，「MX100データアクイジションユニットユーザーズマニュアル」(IMMX100-01)と「MX100スタンダードソフトウエアユーザーズマニュアル」(IMMX180-01)を参照してください。
MXOutput構造体を参照してください。
MXOutputData構造体は，全チャネルを集約したものです。
設定データの一部です。
以下の項目から構成されています。
・ 出力種類
・ 選択値(アイドル時，エラー時)
・ ユーザ指定の出力値
・ パルス周期倍率(PWMチャネルのみ)
ここで，ユーザ指定の出力値は，データ値やスパンと同様に整数値で指定します。

出力チャネルデータ番号

出力チャネルの位置を識別するための値です。
1から60の整数値です。
個数値が定義されています。

初期バランスデータ

ひずみチャネルの初期バランス値を表すデータです。
MXBalance構造体を参照してください。
MXBalanceData構造体は，全チャネルを集約したものです。
有効無効値と初期バランス値で構成されます。
データ取得した場合，ひずみチャネルを有効無効値で表します。
データ送信する場合，「有効値」指定されたチャネルだけが変更，送信されます。

初期バランスデータ番号

ひずみチャネルの位置を識別するための値です。
1から60の整数値です。
個数値が定義されています。

ステータスデータ

システム(ユニット)の状態です。
MXStatus構造体を参照してください。
以下のステータス情報を含みます。

- ユニットステータス値
- CFステータス情報
- FIFOステータス情報

設定項目番号

設定データの構造体の各項目に一意になるように付けられた番号です。
設定データを送信するとき，整合性のチェックでエラーを検出した場所を表す番号です。
定義ファイルが用意されています。「6.3　MX100の設定項目番号」を参照してください。

設定データ

MX100の設定情報です。
MXConfigData構造体を参照してください。
全設定データは，基本設定，システム構成データ，ステータスデータの取得を実行して，情報をマージします。
各データ取得で取得される項目については，「型」の項の各データ構造体を参照してください。
以下のデータから構成されます。

- システム構成データ
- ネットワーク情報データ
- ステータスデータ
- チャネル設定データ
- 初期バランスデータ
- 出力チャネルデータ

基本設定

設定情報のうち，システムの基本的な設定や静的情報です。

測定値

データ値と小数点位置で表されます。
データ値と小数点位置から，工業量を生成します。

データ値

測定値の仮数部を整数値で表した値です。

小数点位置

測定値の指数部を表した値です。
0から4の整数値です。
ひずみの「200000μSTR」レンジの場合のみ，－1です。

測定データ

測定点のチャネルごとのデータ値の情報です。
MXDataInfo構造体を参照してください。
本APIでは，瞬時値とFIFO値を取得できます。
以下の情報を含みます。

- データ値
- データステータス値
- アラームの有無(有効無効値)

FIFO値

FIFOに格納されているデータ値です。
測定データを取得するときに，開始と終了のデータ番号で取得範囲を指定して取得します。
データ番号は，ステータスデータから取得することができます。

瞬時値

FIFOに格納されている最新のデータ値です。
測定データを取得するときに，取得範囲を示すデータ番号を省略することで取得できます。
「瞬時値指定用データ番号」の定数値も定義されています。
最速でも100msごとの取得になります。

チャネル

チャネルは，チャネルタイプとチャネル番号で表されます。
チャネル番号は，ユニット上のスロット位置と，そのスロットに搭載したモジュールの端子位置で決まります。

チャネルタイプ

チャネルの位置と種類を識別するための値です。
MX100の場合，必要ありません。
DARWINとの整合用に，「0(測定チャネル)」としています。

チャネル番号

整数値で指定します。
1から60の整数値です。
個数値が定義されています。

例　：スロット番号3のモジュールの端子番号2のチャネルの場合，チャネル番号は「32」です。
　　このとき，モジュール番号は「3」になります。

チャネル範囲

同じチャネルタイプ内の連続するチャネルを表します。
チャネルタイプ，開始チャネル番号，および終了チャネル番号で指定します。
終了チャネル番号が開始チャネル番号以下の場合，開始チャネル番号による単独のチャネルとみなします。
チャネルタイプは省略されていることもあります。

チャネル名

ユニット番号とユニット内のチャネル番号から生成される名称です。
整数値で表されます。
文字列の場合，5桁の十進整数(例「00001」)で表記されます。
本APIでは使用していません。

チャネル識別情報

チャネル設定データとチャネル情報データに共通の情報で，チャネルを識別するための値です。
MXChID構造体を参照してください。
以下の情報を含みます。

- チャネル番号
- 小数点位置
- チャネルステータス(有効無効値)
- チャネル種類
- レンジ種類
- スケール種類
- 単位名
- タグ
- コメント
- アラーム

コメント

最大30バイトの任意の文字列です。
終端はNULLです。

タグ

最大15バイトの任意の文字列です。
終端はNULLです。

単位名

最大6バイトの任意の文字列です。
終端はNULLです。
レンジ種類に関係なく設定できます。

チャネル情報データ

入力チャネルの測定データを取得するときに，FIFO内の位置を識別するためなどの情報です。
MXChInfo構造体を参照してください。
チャネル識別情報と以下の情報を含みます。

- FIFO番号
- FIFO内チャネル順序番号
- 基準範囲(未使用)
- 表示範囲(スパン，スケール指定による有効範囲)
- 実範囲(レンジ種類の測定可能範囲)

チャネル設定データ

チャネルごとの設定情報です。
MXChConfig構造体を参照してください。
MXChConfigData構造体は，全チャネルを集約したものです。
チャネル識別情報と各チャネル種類別設定の情報を含みます。

データ識別子

拡張APIのデータ操作機能でユーザが作成したデータを識別する値です。
0からの整数です。上限はシステムに依存します。
各データ種類ごとに割り当てられます。以下の種類があります。

- DOデータ識別子
- AP/PWMデータ識別子
- 初期バランスデータ識別子
- 伝送出力データ識別子

伝送出力データ

AO/PWMチャネルの伝送状態を表すデータです。
MXTransmit構造体を参照してください。
コマンドAO/PWMのチャネルについては，AO/PWMデータを参照してください。
データ取得した場合，現在の伝送出力の状態を表します。
データ送信する場合，伝送出力の開始，停止を制御する指定になります。

ネットワーク情報データ

ネットワークの設定情報です。
MXNetInfo構造体を参照してください。

パケット

MX100の場合，通信はパケットによるバイナリの送受信で行われます。
要求コマンドの指定により，クラス内部でパケットを生成して送信しています。応答のパケットを受信し解析して，構造体に必要なデータを格納し返却しています。

バックアップ

通信切断時，CFカードに測定データを記録する機能です。

パルス周期倍率

PWMチャネルで，パルス幅を示す値です。
レンジ種類の分解能の倍数で指定します。
1から30000の整数値です。
最大値が定義されています。

フラグ

データを取得した場合の状態を表します。
データ取得で，データの終端を示すのに使用されます。
複数データを取得する場合，各データごとの状態を表します。
フラグステータスを論理OR演算で合成した値です。

モジュール情報

メインモジュールを除く各モジュールごとの情報です。
ユニット上のスロット位置で識別されます。
以下の項目から構成されています。

- プロダクト情報
- モジュール種類(起動時，実際)
- チャネル数
- 周期種類
- AD積分時間種類
- 端子種類
- 有効無効値
- モジュールバージョン
- FIFO番号

モジュール番号

モジュールの位置(スロット位置)を識別するための番号です。
0から5の整数値です。
個数値が定義されています。

ユーザカウント

ユーザ定義による順序情報です。
設定後の通信から本体に送信されます。
本体は受信後のデータからデータ番号に対応させて格納します。
測定データの取得において，データ番号に対応する時刻情報データと同時にこの値を取得することができます。

ユニット情報

ユニットはメインモジュールを中心としたひとつのシステム単位です。
以下の項目から構成されています。

- プロダクト情報
- ユニット種類
- スタイル
- 温度単位種類
- ユニット番号
- タイムアウト値
- CF書き込み種類
- 電源周波数
- パート番号

ユニット番号

ユニットの識別番号です。ユーザが指定できます。
0から98の整数値です。
メインモジュールの7セグメントLEDに表示されます。

タイムアウト値

通信切断時，測定データをCFに保存開始するまでの時間です。
60以上の整数値です。単位は秒です。
付録3を参照してください。

付録2 DARWINに関する用語

本APIとDARWINに関する用語を説明しています。大項目はアルファベット順，五十音順に並んでいます。
詳細は，DARWIN本体の取扱説明書や「通信インタフェースマニュアル」をご覧ください。

アラーム

アラーム機能です。

アラームレベル

チャネル内のアラーム機能の識別番号です。
1から4の整数値です。
個数値が定義されています。

アラーム種類

DARWINの場合，アラーム種類一覧で示されるアラームの種類です。

アラーム値

アラームONにする値です。
小数点を除いた整数値です。

応答

コマンド送信後，測定機器本体から応答を受信します。トーカとしてデータ出力要求をした場合以外は，次のいずれかを受信します。

・ 処理が正常に行われた
・ 処理が正常に行われなかった

処理が正常に行われず，コマンド実行処理に失敗した関数は，エラー番号を返します。

機器記述子

測定機器本体を識別するための値です。
各関数を実行する場合，この機器記述子が必要になります。
APIの場合，実体は，CDAQDARWINオブジェクトへの参照になります。
DAQDARWIN型で格納，指定します。
拡張APIの場合，実体は，CDAQDA100オブジェクトへの参照になります。
CDAQDA100型で格納，指定します。

時刻情報データ

測定時刻を示す日付時刻のデータです。原則，1970年01月01日からの秒数です。
DarwinDateTime構造体を参照してください。
ミリ秒の値は未使用です。
瞬時値データ読み込み通信ポートを使用した場合のみ，ミリ秒を使用します。

システム構成データ

ユニット，モジュールの情報です。
DarwinSystemInfo構造体を参照してください。

ユニット番号

システム構成のユニットを識別する番号です。
チャネル/リレータイプを参照してください。
メインユニット，サブユニット番号のいずれかです。
スタンドアロンモデルの場合，有効なユニット番号は0だけになります。

サブユニット番号

拡張モデルの場合に接続されているサブユニットを識別する番号です。
0から5の整数値です。
スタンドアロンモデルの場合，ユニット番号と同一になり，有効なサブユニット番号は0だけになります。

スロット番号

ユニット毎のモジュール接続位置を表します。
0から5の整数値です。

端子番号

スロットに接続されたモジュール毎のチャネル/リレーの位置を識別する番号です。
1から10の整数値です。
モジュールの種類により異なります。

スケール

レフト値，ライト値，小数点位置から構成されます。値は，-30000から30000の範囲です。
本API内の指定では，レフト値とライト値が等しい場合，省略されたものとみなします。

ステータスバイト

機器本体の状態を示す値です。ステータスバイト値を論理OR演算で合成した値です。

スパン

レフト値，ライト値から構成されます。値は，測定レンジの種類によって異なります。

本API内の指定では，レフト値とライト値が等しい場合，省略されたものとみなします。

測定値

データ値と小数点位置で表されます。

データ値と小数点位置から工業量を生成します。

データ値

測定値の仮数部を整数値で表した値です。

小数点位置

測定値の指数部を表した値です。

0から4の整数値です。

測定データ

測定点のチャネルごとのデータ値の情報です。

DarwinDataInfo構造体を参照してください。

トーカの測定データの出力で取得できます。

アラーム有無は，アラームタイプで表されます。

データステータス

チャネルステータスと共通な値を使用しています。データステータス値を参照してください。

測定周期

測定インターバルです。

単位は秒です。

ターミネータ

コマンド終端を表す文字列です。

チャネル

チャネルは，チャネルタイプとチャネル番号で表されます。
DARWINの場合，チャネルは，測定チャネルと演算チャネルがあります。

測定チャネル

チャネルタイプが，拡張モデルのサブユニット番号，または，スタンドアロンモデルのユニットの場合のチャネルです。
モジュール接続位置で定まる入力位置識別番号を表します。
チャネル番号は，スロット番号と端子番号から生成されます。
システム構成によっては非連続になります。

演算チャネル

チャネルタイプが，「演算」を表す値の場合のチャネルです。
演算オプション付きモデルで利用可能です。
スタンドアロンモデルの場合，チャネル番号が1から30の整数値になります。
拡張モデルの場合，チャネル番号が1から60の整数値になります。

チャネルタイプ

チャネルの位置と種類を識別するため値です。
DARWINの場合，チャネルの位置と種類を識別するための値です。
チャネル/リレータイプを参照してください。
拡張モデルのサブユニット番号，スタンドアロンモデルのユニット，または，演算のいずれかです。

チャネル番号

整数値で指定します。
1から60の整数値です。
例　：スロット番号3の10Ch中速ユニバーサル入力モジュールの端子番号2のチャネルの場合，チャネル番号は「32」です。
演算チャネルの場合，拡張モデルで1から60の整数値，スタンドアロンモデルで1から30の整数値です。

チャネル範囲

同じチャネルタイプ内の連続するチャネルを表します。
チャネルタイプ，開始チャネル番号，終了チャネル番号で指定します。
終了チャネル番号が開始チャネル番号以外の場合，開始チャネル番号による単独のチャネルとみなします。

チャネル情報データ

チャネルごとのチャネルタイプ，チャネル番号，小数点位置などの静的な情報です。
DarwinChInfo構造体を参照してください。
トーカのチャネル情報データの出力で取得できます。

単位名

最大6バイトの任意の文字列です。

チャネルステータス

データステータスと共通の値を使用しています。
データステータス値を参照してくさい。

トーカ

データ出力を行う機能です。トーカ機能種類に示す種類をサポートします。
複数チャネル/行のデータが出力されるため，開始コマンド(宣言)を実行後，チャネル/行単位で取得コマンドを実行します。

フラグ

データを取得した場合の状態を表します。
データ取得で，データの終端を示すのに使用されます。
フラグステータスを論理OR演算で合成した値です。

リレー

リレーは，リレータイプとリレー番号で表されます。

リレータイプ

リレーの位置と種類を識別するための値です。チャネル/リレータイプを参照してください。
拡張モデルのメインユニットとサブユニット番号，または，内部スイッチのいずれかです。

リレー番号

リレータイプ間で非連続です。
1から60の整数値です。
リレータイプがユニット番号の場合，モジュール接続位置で定まるリレー位置識別番号を表します。スロット番号と 端子番号から生成されます。システム構成によっては非連続になります。

付録3 MX100のタイムアウト値の算出

タイムアウト値は，通信切断時，測定周期でサンプリングしたデータをCFに保存開始するまでの時間です。60以上の整数値です。単位は秒です。初期値は60sです。
通信切断時に，サンプリングデータを欠落なくCFカードに保存するためには，FIFOバッファの未保存のデータが上書きされる前に，CFカードに保存することが必要です。値が適切でないと，CFカードへの保存開始時にエラーとなり，サンプリングデータが保存されません。
以下のタイムアウト値の算出方法を参考にしてください。

タイムアウト値(s) < (FIFOバッファ容量/1sあたりのデータ量)−20s

FIFOバッファ容量＝2Mバイト(2097152バイト)
1sあたりのFIFOデータ量＝(時刻バイト数＋測定値バイト数×チャネル数)×1s間の測定回数
時刻バイト数＝16バイト
測定値バイト数＝4バイト
20s：CFカードへの保存に要する時間(目安)

算出例1
FIFO：60チャネル/測定周期10msの場合

1sあたりのデータ量＝(16＋4×60ch)×1000/10＝25600バイト
設定値=(2Mバイト/25600バイト)−20s＝61.9 > 60s(設定値目安)

算出例2
下記の3つのFIFOが動作している場合
FIFO：40チャネル/測定周期10ms，4チャネル/測定周期50ms，10チャネル/測定周期100ms

1sあたりのデータ量＝(16＋4×40ch)×1000/10＝17600バイト
1sあたりのデータ量＝(16＋4×4ch)×1000/50＝640バイト
1sあたりのデータ量＝(16＋4×10ch)×1000/100＝560バイト
設定値=(2Mバイト/(17600＋640＋560バイト))−20s＝91.5 > 90s(設定値目安)

付録4 API改訂履歴(R2.01)

本API R2.01で追加・削除された関数について記載します。

Visual C/Visual Basic用関数

新しく追加されたMX100用関数

changeAOPWMDataMX
changeBalanceMX
changeTransmitMX
getAlarmNameMX（Visual Cのみ）
getAOPWMDataMX
getBalanceMX
getItemErrorMX
getMaxLenAlarmNameMX
getOutputMX
isDataNoMX（Visual Cのみ）
isDataNoVBMX
resetBalanceMX
runBalanceMX
setAOMX
setAOPWMDataMX
setAOTypeMX
setBalanceMX
setChoiceMX
setOutputMX
setOutputTypeMX
setPulseTimeMX
setPWMMX
setPWMTypeMX
setRESMX
setSTRAINMX
setTransmitMX
toAlarmNameMX
toAOPWMValueMX
toRealValueMX
toStyleVersionMX

新しく追加されたDARWIN用関数

computeDARWIN
establishDARWIN
getAlarmNameDARWIN (Visual Cのみ)
getMaxLenAlarmNameDARWIN
getReportStatusDARWIN
receiveByteDARWIN
reportingDARWIN
setMADARWIN
setPOWERDARWIN
setPULSEDARWIN
setSTRAINDARWIN

Visual C++用関数

新しく追加されたMX100用クラス

CDAQMXAOPWMDataクラス
CDAQMXBalanceDataクラス
CDAQMXBalanceResultクラス
CDAQMXOutputDataクラス
CDAQMXTransmitクラス

新しく追加されたMX100用メンバ

CDAQMX::clearLastDataNoCh
CDAQMX::clearLastDataNoFIFO
CDAQMX::getAOPWMData
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::setAOPWMData
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::setPWMType
CDAQMXConfig::setRES
CDAQMXConfig::setSTRAIN
CDAQMXDataInfo::initMXDataInfo
CDAQMXDataInfo::isObject
CDAQMXDateTime::initMXDateTime
CDAQMXDateTime::isObject

CDAQMXDOData::initMXDOData
CDAQMXDOData::isObject
CDAQMXDOData::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

削除されたMX100用メンバ

バージョン非互換のため，削除されました。
CDAQMXChConfig::setFromBlockChConfig
CDAQMXChConfigData::setFromAckGetConfigPacket
CDAQMXChInfo::setFromBlockChInfo
CDAQMXConfig::setFromAckGetConfigPacket
CDAQMXDataInfo::setFromBlockData
CDAQMXNetInfo::setFromAckGetConfigPacket
CDAQMXStatus::setFromAckGetConfigPacket
CDAQMXStatus::setFromAckGetStatusPacket
CDAQMXSysInfo::setFromAckGetConfigPacket
CDAQMXSysInfo::setFromAckGetUnitInfoPacket

新しく追加されたDARWIN用メンバ

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

付録5 API改訂履歴(R3.01)

本API R3.01で追加・変更された内容について記載します。

API

Visual C++

新しく追加されたMX100用メンバ

CDAQMXChConfig::isChatFilter
CDAQMXChConfig::setChatFilter
CDAQMXChConfig::setCOM
CDAQMXChConfig::setPULSE
CDAQMXConfig::setCOM
CDAQMXConfig::setPULSE

変更されたMX100用メンバ

CDAQMXChConfig::setDELTA
CDAQMXConfig::setDELTA

Visual C / Visual Basic

新しく追加されたMX100用関数

setChatFilterMX
setCOMMX
setPULSEMX

定数

新しく追加されたMX100用定数

DAQMX_CHKIND_PI
DAQMX_CHKIND_PIDIFF
DAQMX_CHKIND_CI
DAQMX_CHKIND_CIDIFF
DAQMX_MODULE_HIDDEN
DAQMX_MODULE_MX114PLSM10
DAQMX_MODULE_MX110VTDL30
DAQMX_MODULE_MX118CANM10
DAQMX_MODULE_MX118CANM20
DAQMX_MODULE_MX118CANM30
DAQMX_MODULE_MX118CANSUB
DAQMX_MODULE_MX118CANMERR
DAQMX_MODULE_MX118CANSERR
DAQMX_CHNUM_30
DAQMX_TERMINAL_DSUB
DAQMX_RANGE_TC_XK
DAQMX_RANGE_RTD_1MAPTG
DAQMX_RANGE_RTD_1MACU100G
DAQMX_RANGE_RTD_1MACU50G
DAQMX_RANGE_RTD_1MACU10G
DAQMX_RANGE_RTD_2MACU100G
DAQMX_RANGE_RTD_2MACU50G
DAQMX_RANGE_RTD_2MACU10G
DAQMX_RANGE_DI_CONTACT_AI30
DAQMX_RANGE_COM_CAN
DAQMX_RANGE_PI_LEVEL
DAQMX_RANGE_PI_CONTACT

変更されたMX100用定数

DAQMX_NUMALARM

設定項目番号

新しく追加されたMX100用設定項目番号

DAQMX_ITEM_CHCHATFILTER
DAQMX_ITEM_ALARMTYPE2
DAQMX_ITEM_ALARMON2
DAQMX_ITEM_ALARMOFF2
DAQMX_ITEM_CHREFALARM2

新しく追加されたMX100用定数

DAQMX_MAX_INDEX_FIFO
DAQMX_MAX_INDEX_MODULE
DAQMX_MAX_INDEX_CHANNEL
DAQMX_ITEM_ALL_END_R3

変更されたMX100用定数

DAQMX_ITEM_ALL_END

型

変更されたMX100用型

DAQMX
MXDataInfo
MXChConfigAIDI
MXChConfigAI
MXChConfigDO
MXChID
MXChConfig
MXChConfigData
MXChInfo

拡張API

Visual C++

新しく追加されたMX100用メンバ

CDAQMX100::setChatFilter

変更されたMX100用メンバ

CDAQMX100::measDataCh
CDAQMX100::measDataFIFO
CDAQMX100::measInstCh
CDAQMX100::measInstFIFO

Visual C / Visual Basic

新しく追加されたMX100用関数

setChatFilterMX100
channelChatFilterMX100

定数

新しく追加されたMX100用定数

DAQMX100_CHKIND_PI
DAQMX100_CHKIND_PIDIFF
DAQMX100_CHKIND_CI
DAQMX100_CHKIND_CIDIFF
DAQMX100_MODULE_HIDDEN
DAQMX100_MODULE_MX114PLSM10
DAQMX100_MODULE_MX110VTDL30
DAQMX100_MODULE_MX118CANM10
DAQMX100_MODULE_MX118CANM20
DAQMX100_MODULE_MX118CANM30
DAQMX100_MODULE_MX118CANSUB
DAQMX100_MODULE_MX118CANMERR
DAQMX100_MODULE_MX118CANSERR
DAQMX100_CHNUM_30
DAQMX100_TERMINAL_DSUB
DAQMX100_RANGE_TC_XK

DAQMX100_RANGE_RTD_1MAPTG
DAQMX100_RANGE_RTD_1MACU100G
DAQMX100_RANGE_RTD_1MACU50G
DAQMX100_RANGE_RTD_1MACU10G
DAQMX100_RANGE_RTD_2MACU100G
DAQMX100_RANGE_RTD_2MACU50G
DAQMX100_RANGE_RTD_2MACU10G
DAQMX100_RANGE_DI_CONTACT_AI30
DAQMX100_RANGE_COM_CAN
DAQMX100_RANGE_PI_LEVEL
DAQMX100_RANGE_PI_CONTACT

変更されたMX100用定数

DAQMX100_NUMALARM

型

変更されたMX100用型

DAQMX100

索引

記号

.bas 1-5
.cs 1-5
.dll 1-5
.h 1-5
.lib 1-5
.txt 1-5
.vb 1-5
[..... 12-66, 13-11, 15-10
7 セグメント LED ..... 付 -1

A

ackAlarmDA100 24-2
ackAlarmMX100 17-2
addressPartMX100 17-92
A/D 積分時間種類 6-8, 18-9
alarmDoubleHisterisysMX100 17-93
alarmDoubleValueOFFMX100 17-94
alarmDoubleValueONMX100 17-95
alarmHisterisysMX100 17-96
alarmMaxLengthDA100 24-40
alarmMaxLengthDA100Reader 24-82
alarmMaxLengthMX100 17-97
alarmTypeDA100 24-41
alarmTypeDA100Reader 24-83
alarmTypeMX100 17-98
alarmValueOFFMX100 17-99
alarmValueONMX100 17-100
AO/PWM データ ..... 付 -2
AO/PWM データ番号 ..... 付 -2
AO レンジ 6-16, 18-17
API 改訂履歴 ..... 付 -20, 付 -26
API 用 MX100 のクラス 2-1
autoFIFOMX 5-2

C

CD-ROM の取り扱い iv
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
CDAQChInfo クラス 2-15
CDAQDA100::ackAlarm 19-19
CDAQDA100::CDAQDA100 19-20
CDAQDA100::chNumMax 19-20
CDAQDA100::chNumMaxReport 19-21
CDAQDA100::getByte 19-21
CDAQDA100::getChannel 19-21
CDAQDA100::getClassDataBuffer 19-22
CDAQDA100::getClassSysInfo 19-22
CDAQDA100::getCode 19-22
CDAQDA100::getData 19-23
CDAQDA100::getInfoCh 19-23
CDAQDA100::getInstChASCII 19-24
CDAQDA100::getInstChBINARY 19-24
CDAQDA100::getReport 19-25
CDAQDA100::getRevisionDA100DLL 19-25
CDAQDA100::getVersionDA100DLL 19-25
CDAQDA100::initSetValue 19-25
CDAQDA100::isObject 19-26
CDAQDA100::mathInfoCh 19-26
CDAQDA100::mathInstCh 19-27
CDAQDA100::measClear 19-27
CDAQDA100::measInfoCh 19-28
CDAQDA100::measInstCh 19-29
CDAQDA100::open 19-29
CDAQDA100::reconstruct 19-30
CDAQDA100::setChAlarm 19-30
CDAQDA100::setChDELTA 19-31
CDAQDA100::setChRRJC 19-32
CDAQDA100::setChUnit 19-33
CDAQDA100::setDateTime 19-33
CDAQDA100::setRange 19-34
CDAQDA100::switchCode 19-35
CDAQDA100::switchCompute 19-35
CDAQDA100::switchMode 19-35
CDAQDA100::switchReport 19-36
CDAQDA100::talkCalibrationChData 19-36
CDAQDA100::talkOperationChData 19-37
CDAQDA100::talkSetupChData 19-37
CDAQDA100::updateAll 19-38
CDAQDA100::updateChInfo 19-38
CDAQDA100::updateRenew 19-38
CDAQDA100::updateReportStatus 19-39
CDAQDA100::updateStatus 19-39
CDAQDA100::updateSystemConfig 19-39
CDAQDA100Reader::CDAQDA100Reader 19-42
CDAQDA100Reader::getInfoCh 19-43
CDAQDA100Reader::getInstCh 19-43
CDAQDA100Reader::isObject 19-44
CDAQDA100Reader::measInfoCh 19-44
CDAQDA100Reader::measInstCh 19-45
CDAQDA100Reader::open 19-45
CDAQDA100Reader クラス 19-40
CDAQDA100 クラス 19-16
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::getReportStatus 7-20
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

CDAQDARWIN::setDELTA 7-27
CDAQDARWIN::setDI 7-28
CDAQDARWIN::setMA 7-29
CDAQDARWIN::setPOWER 7-30
CDAQDARWIN::setPULSE 7-31
CDAQDARWIN::setRRJC 7-32
CDAQDARWIN::setRTD 7-33
CDAQDARWIN::setScallingUnit 7-34
CDAQDARWIN::setSKIP 7-34
CDAQDARWIN::setSTRAIN 7-35
CDAQDARWIN::setTC 7-36
CDAQDARWIN::setVOLT 7-37
CDAQDARWIN::startTalker 7-37
CDAQDARWIN::talkCalibrationData 7-38
CDAQDARWIN::talkChInfo 7-38
CDAQDARWIN::talkDataByASCII 7-39
CDAQDARWIN::talkDataByBinary 7-40
CDAQDARWIN::talkOperationData 7-41
CDAQDARWIN::talkSetupData 7-41
CDAQDARWIN::transMode 7-42
CDAQDARWINChInfo::CDAQDARWINChInfo 7-45
CDAQDARWINChInfo::getChName 7-45
CDAQDARWINChInfo::getChStatus 7-46
CDAQDARWINChInfo::getDarwinChInfo 7-46
CDAQDARWINChInfo::getStatusName 7-46
CDAQDARWINChInfo::getUnit 7-46
CDAQDARWINChInfo::initDarwinChInfo 7-47
CDAQDARWINChInfo::initialize 7-47
CDAQDARWINChInfo::isObject 7-47
CDAQDARWINChInfo::operator= 7-48
CDAQDARWINChInfo::setCHStatus 7-48
CDAQDARWINChInfo::setDarwinChInfo 7-48
CDAQDARWINChInfo::setLine 7-49
CDAQDARWINChInfo::setUnit 7-49
CDAQDARWINChInfo::toChName 7-50
CDAQDARWINChInfo::toChRange 7-50
CDAQDARWINChInfo::toChType 7-51
CDAQDARWINChInfo::toFlag 7-51
CDAQDARWINChInfo::toStatus 7-52
CDAQDARWINChInfo クラス 7-43, 24-15
CDAQDARWINDataBuffer::CDAQDARWINDataBuffer . 19-47
CDAQDARWINDataBuffer::getClassDARWINChInfo 19-47
CDAQDARWINDataBuffer::getClassDARWINDataInfo . 19-47
CDAQDARWINDataBuffer::getClassDARWINDateTime 19-47
CDAQDARWINDataBuffer::initialize 19-48
CDAQDARWINDataBuffer::isAlarm 19-48
CDAQDARWINDataBuffer::setChInfo 19-48
CDAQDARWINDataBuffer::setDataInfo 19-49
CDAQDARWINDataBuffer::setDateTime 19-49
CDAQDARWINDataBuffer クラス 19-46
CDAQDARWINDataInfo::CDAQDARWINDataInfo 7-55
CDAQDARWINDataInfo::getAlarm 7-55
CDAQDARWINDataInfo::getAlarmName 7-56
CDAQDARWINDataInfo::getClassDARWINChInfo 7-56
CDAQDARWINDataInfo::getDarwinDataInfo 7-56
CDAQDARWINDataInfo::getMaxLenAlarmName 7-57
CDAQDARWINDataInfo::getStatus 7-57
CDAQDARWINDataInfo::initDarwinDataInfo 7-57
CDAQDARWINDataInfo::initialize 7-57
CDAQDARWINDataInfo::isObject 7-58
CDAQDARWINDataInfo::operator= 7-58
CDAQDARWINDataInfo::setAlarm 7-58
CDAQDARWINDataInfo::setByte 7-59
CDAQDARWINDataInfo::setClassDARWINChInfo 7-59
CDAQDARWINDataInfo::setDarwinDataInfo 7-59
CDAQDARWINDataInfo::setLine 7-60
CDAQDARWINDataInfo::setStatus 7-60
CDAQDARWINDataInfo::toAlarmType 7-60
CDAQDARWINDataInfo クラス 7-53
CDAQDARWINDateTime::CDAQDARWINDateTime 7-63
CDAQDARWINDateTime::getDarwinDateTime 7-63
CDAQDARWINDateTime::getDay 7-63
CDAQDARWINDateTime::getFullYear 7-64
CDAQDARWINDateTime::getHour 7-64
CDAQDARWINDateTime::getMinute 7-64
CDAQDARWINDateTime::getMonth 7-64
CDAQDARWINDateTime::getSecond 7-65
CDAQDARWINDateTime::getYear 7-65
CDAQDARWINDateTime::initDarwinDateTime 7-65
CDAQDARWINDateTime::initialize 7-65
CDAQDARWINDateTime::isObject 7-66
CDAQDARWINDateTime::operator= 7-66
CDAQDARWINDateTime::setByte 7-67
CDAQDARWINDateTime::setDarwinDateTime 7-67
CDAQDARWINDateTime::setLine 7-68
CDAQDARWINDateTime::setNow 7-68
CDAQDARWINDateTime::toDateTime 7-68
CDAQDARWINDateTime::toString 7-69
CDAQDARWINDateTime クラス 7-61
CDAQDARWINSysInfo::CDAQDARWINSysInfo 7-71
CDAQDARWINSysInfo::getDarwinModuleInfo 7-71
CDAQDARWINSysInfo::getDarwinSystemInfo 7-72
CDAQDARWINSysInfo::getDarwinUnitInfo 7-72
CDAQDARWINSysInfo::getInterval 7-72
CDAQDARWINSysInfo::getModuleCode 7-73
CDAQDARWINSysInfo::getModuleName 7-73
CDAQDARWINSysInfo::initDarwinSystemInfo 7-73
CDAQDARWINSysInfo::initialize 7-74
CDAQDARWINSysInfo::isExist 7-74
CDAQDARWINSysInfo::isObject 7-74
CDAQDARWINSysInfo::operator= 7-75
CDAQDARWINSysInfo::setDarwinSystemInfo 7-75
CDAQDARWINSysInfo::setLine 7-76
CDAQDARWINSysInfo::toRelayName 7-76
CDAQDARWINSysInfo クラス 7-70
CDAQDARWIN クラス 7-13
CDAQDataInfo::CDAQDataInfo 2-20
CDAQDataInfo::getClassChInfo 2-20
CDAQDataInfo::getDoubleValue 2-20
CDAQDataInfo::getStringValue 2-21
CDAQDataInfo::getValue 2-21
CDAQDataInfo::initialize 2-21
CDAQDataInfo::isObject 2-22
CDAQDataInfo::operator= 2-22
CDAQDataInfo::setClassChInfo 2-22
CDAQDataInfo::setValue 2-23
CDAQDataInfo::toDoubleValue 2-23
CDAQDataInfo::toStringValue 2-23
CDAQDataInfo クラス 2-19
CDAQDateTime::CDAQDateTime 2-25
CDAQDateTime::getMilliSecond 2-25
CDAQDateTime::getTime 2-25
CDAQDateTime::initialize 2-25
CDAQDateTime::isObject 2-26
CDAQDateTime::operator= 2-26
CDAQDateTime::setMilliSecond 2-26
CDAQDateTime::setNow 2-26
CDAQDateTime::setTime 2-27
CDAQDateTime::toLocalDateTime 2-27
CDAQDateTime クラス 2-24
CDAQHandler::CDAQHandler 2-29
CDAQHandler::close 2-30
CDAQHandler::getChannel 2-30
CDAQHandler::getData 2-31
CDAQHandler::getErrorMessage 2-31
CDAQHandler::getMaxLenErrorMessage 2-31

CDAQHandler::getRevisionAPI 2-32
CDAQHandler::getRevisionDLL 2-32
CDAQHandler::getVersionAPI 2-32
CDAQHandler::getVersionDLL 2-32
CDAQHandler::isObject 2-33
CDAQHandler::open 2-33
CDAQHandler::receive 2-34
CDAQHandler::receiveLine 2-34
CDAQHandler::receiveRemain 2-35
CDAQHandler::send 2-35
CDAQHandler::sendLine 2-36
CDAQHandler::setTimeOut 2-36
CDAQHandler クラス 2-28
CDAQMX::autoFIFO 2-41
CDAQMX::CDAQMX 2-41
CDAQMX::clearAttr 2-42
CDAQMX::clearData 2-42
CDAQMX::clearLastDataNoCh 2-42
CDAQMX::clearLastDataNoFIFO 2-42
CDAQMX::formatCF 2-43
CDAQMX::getAOPWMData 2-43
CDAQMX::getBalance 2-44
CDAQMX::getChannel 2-44
CDAQMX::getChConfig 2-45
CDAQMX::getChData 2-46
CDAQMX::getChDataNo 2-46
CDAQMX::getChInfo 2-47
CDAQMX::getConfig 2-47
CDAQMX::getData 2-48
CDAQMX::getDataNo 2-48
CDAQMX::getDOData 2-49
CDAQMX::getFIFODataNo 2-49
CDAQMX::getItemError 2-49
CDAQMX::getLastError 2-50
CDAQMX::getMXConfig 2-50
CDAQMX::getNo 2-50
CDAQMX::getOutput 2-51
CDAQMX::getPacketVersion 2-51
CDAQMX::getRevisionDLL 2-51
CDAQMX::getStatusData 2-52
CDAQMX::getSystemConfig 2-52
CDAQMX::getTimeData 2-53
CDAQMX::getUserTime 2-53
CDAQMX::getVersionDLL 2-53
CDAQMX::incCurDataNo 2-54
CDAQMX::incCurFIFOIdx 2-54
CDAQMX::initSystem 2-54
CDAQMX::isObject 2-55
CDAQMX::nop 2-55
CDAQMX::open 2-56
CDAQMX::receiveBlock 2-56
CDAQMX::receiveBuffer 2-57
CDAQMX::receivePacket 2-57
CDAQMX::registry 2-58
CDAQMX::resetBalance 2-58
CDAQMX::runBalance 2-59
CDAQMX::runCommand 2-59
CDAQMX::runPacket 2-60
CDAQMX::searchChNo 2-60
CDAQMX::sendPacket 2-61
CDAQMX::setAOPWMData 2-61
CDAQMX::setBackup 2-62
CDAQMX::setBalance 2-62, 2-63
CDAQMX::setConfig 2-63
CDAQMX::setDateTime 2-64
CDAQMX::setDOData 2-64
CDAQMX::setMXConfig 2-65
CDAQMX::setOutput 2-65
CDAQMX::setSegment 2-66
CDAQMX::setTransmit 2-66
CDAQMX::setUserTime 2-67
CDAQMX::startFIFO 2-67
CDAQMX::stopFIFO 2-67
CDAQMX::talkChData 2-68
CDAQMX::talkChInfo 2-68
CDAQMX::talkConfig 2-69
CDAQMX::talkFIFOData 2-69
CDAQMX100::ackAlarm 12-25
CDAQMX100::CDAQMX100 12-25
CDAQMX100::CDAQMXItemConfig 12-29
CDAQMX100::changeAOPWMValue 12-26
CDAQMX100::clearBalance 12-26
CDAQMX100::commandAOPWM 12-27
CDAQMX100::commandDO 12-27
CDAQMX100::commandTransmit 12-28
CDAQMX100::currentDoubleAOPWMValue 12-28
CDAQMX100::displaySegment 12-29
CDAQMX100::formatCF 12-29
CDAQMX100::getClassMXAOPWMList 12-30
CDAQMX100::getClassMXBalanceList 12-30
CDAQMX100::getClassMXDataBuffer 12-30
CDAQMX100::getClassMXDOList 12-30
CDAQMX100::getClassMXItemConfig 12-31
CDAQMX100::getClassMXTransmitList 12-31
CDAQMX100::getDataCh 12-31
CDAQMX100::getDataFIFO 12-32
CDAQMX100::getDataNum 12-32
CDAQMX100::getInstCh 12-33
CDAQMX100::getInstFIFO 12-33
CDAQMX100::getItemAll 12-34
CDAQMX100::getRevisionMX100DLL 12-34
CDAQMX100::getVersionMX100DLL 12-34
CDAQMX100::initBalance 12-34
CDAQMX100::initDataCh 12-35
CDAQMX100::initDataFIFO 12-35
CDAQMX100::initSetValue 12-35
CDAQMX100::isObject 12-36
CDAQMX100::measClear 12-36
CDAQMX100::measDataCh 12-37
CDAQMX100::measDataFIFO 12-37
CDAQMX100::measInstCh 12-38
CDAQMX100::measInstFIFO 12-38
CDAQMX100::measStart 12-39
CDAQMX100::measStop 12-39
CDAQMX100::nextFIFO 12-39
CDAQMX100::open 12-40
CDAQMX100::reconstruct 12-40
CDAQMX100::reloadBalance 12-41
CDAQMX100::sendConfig 12-41
CDAQMX100::setAlarm 12-42
CDAQMX100::setBurnout 12-42
CDAQMX100::setCFWriteMode 12-43
CDAQMX100::setChatFilter 12-43
CDAQMX100::setChComment 12-44
CDAQMX100::setChDELTA 12-44
CDAQMX100::setChKind 12-45
CDAQMX100::setChoice 12-46
CDAQMX100::setChRRJC 12-46
CDAQMX100::setChTag 12-47
CDAQMX100::setChUnit 12-47
CDAQMX100::setDateTime 12-48
CDAQMX100::setDeenergize 12-48
CDAQMX100::setFilter 12-49
CDAQMX100::setHisterisys 12-49
CDAQMX100::setHold 12-50
CDAQMX100::setIntegral 12-50

CDAQMX100::setInterval 12-51
CDAQMX100::setItemAll 12-51
CDAQMX100::setOutputType 12-52
CDAQMX100::setPulseTime 12-52
CDAQMX100::setRange 12-53
CDAQMX100::setRefAlarm 12-54
CDAQMX100::setRJCType 12-54
CDAQMX100::setScale 12-55
CDAQMX100::setSpan 12-55
CDAQMX100::setUnitNo 12-56
CDAQMX100::setUnitTemp 12-56
CDAQMX100::switchBackup 12-57
CDAQMX100::switchDO 12-57
CDAQMX100::switchTransmit 12-58
CDAQMX100::toChNo 12-58
CDAQMX100::updateAll 12-59
CDAQMX100::updateAOPWMData 12-59
CDAQMX100::updateBalance 12-60
CDAQMX100::updateConfig 12-60
CDAQMX100::updateDOData 12-60
CDAQMX100::updateInfoCh 12-61
CDAQMX100::updateOutput 12-61
CDAQMX100::updateRenew 12-61
CDAQMX100::updateStatus 12-62
CDAQMX100::updateSystem 12-62
CDAQMX100::userClear 12-62
CDAQMX100::userDoubleAOPWMValue 12-63
CDAQMX100 クラス 12-20
CDAQMXAOPWMData::CDAQMXAOPWMData 2-71
CDAQMXAOPWMData::getAOPWMValid 2-71
CDAQMXAOPWMData::getAOPWMValue 2-72
CDAQMXAOPWMData::getMXAOPWM 2-72
CDAQMXAOPWMData::getMXAOPWMData 2-72
CDAQMXAOPWMData::initialize 2-72
CDAQMXAOPWMData::initMXAOPWMData 2-73
CDAQMXAOPWMData::isObject 2-73
CDAQMXAOPWMData::operator= 2-73
CDAQMXAOPWMData::setAOPWM 2-74
CDAQMXAOPWMData::setMXAOPWMData 2-74
CDAQMXAOPWMData::toAOPWMValue 2-74
CDAQMXAOPWMData::toRealValue 2-75
CDAQMXAOPWMData クラス 2-70
CDAQMXAOPWMList::add 12-65
CDAQMXAOPWMList::CDAQMXAOPWMList 12-66
CDAQMXAOPWMList::copy 12-66
CDAQMXAOPWMList::copyData 12-67
CDAQMXAOPWMList::create 12-67
CDAQMXAOPWMList::getClassMXAOPWMData 12-67
CDAQMXAOPWMList::getCurrent 12-68
CDAQMXAOPWMList::initCurrent 12-68
CDAQMXAOPWMList クラス 12-64
CDAQMXBalanceData::CDAQMXBalanceData 2-77
CDAQMXBalanceData::getBalanceValid 2-77
CDAQMXBalanceData::getBalanceValue 2-78
CDAQMXBalanceData::getMXBalance 2-78
CDAQMXBalanceData::getMXBalanceData 2-78
CDAQMXBalanceData::initialize 2-79
CDAQMXBalanceData::isObject 2-79
CDAQMXBalanceData::operator= 2-79
CDAQMXBalanceData::setBalance 2-80
CDAQMXBalanceData::setMXBalanceData 2-80
CDAQMXBalanceData クラス 2-76
CDAQMXBalanceList::add 12-70
CDAQMXBalanceList::CDAQMXBalanceList 12-70
CDAQMXBalanceList::change 12-71
CDAQMXBalanceList::copy 12-71
CDAQMXBalanceList::copyData 12-72
CDAQMXBalanceList::create 12-72
CDAQMXBalanceList::getClassMXBalanceData 12-72
CDAQMXBalanceList::getCurrent 12-73
CDAQMXBalanceList::initCurrent 12-73
CDAQMXBalanceList クラス 12-69
CDAQMXBalanceResult::CDAQMXBalanceResult 2-82
CDAQMXBalanceResult::getMXBalanceResult 2-83
CDAQMXBalanceResult::getResult 2-83
CDAQMXBalanceResult::initialize 2-83
CDAQMXBalanceResult::initMXBalanceData 2-80
CDAQMXBalanceResult::initMXBalanceResult 2-83
CDAQMXBalanceResult::isObject 2-84
CDAQMXBalanceResult::operator= 2-84
CDAQMXBalanceResult::setMXBalanceResult 2-85
CDAQMXBalanceResult::setResult 2-85
CDAQMXBalanceResult クラス 2-81
CDAQMXChConfig::CDAQMXChConfig 2-88
CDAQMXChConfig::changeRange 2-89
CDAQMXChConfig::getBurnout 2-89
CDAQMXChConfig::getFilter 2-89
CDAQMXChConfig::getItemError 2-89
CDAQMXChConfig::getMXChConfig 2-90
CDAQMXChConfig::getRangeMax 2-90
CDAQMXChConfig::getRangeMin 2-91
CDAQMXChConfig::getRangePoint 2-91
CDAQMXChConfig::getRefChNo 2-91
CDAQMXChConfig::getRJCType 2-92
CDAQMXChConfig::getRJCVolt 2-92
CDAQMXChConfig::getScaleMax 2-92
CDAQMXChConfig::getScaleMin 2-92
CDAQMXChConfig::getSpanMax 2-93
CDAQMXChConfig::getSpanMin 2-93
CDAQMXChConfig::initialize 2-93
CDAQMXChConfig::initMXChConfig 2-93
CDAQMXChConfig::isChatFilter 2-94
CDAQMXChConfig::isCorrect 2-94
CDAQMXChConfig::isDeenergize 2-94
CDAQMXChConfig::isHold 2-95
CDAQMXChConfig::isObject 2-95
CDAQMXChConfig::isRefAlarm 2-95
CDAQMXChConfig::operator= 2-96
CDAQMXChConfig::setAlarm 2-96
CDAQMXChConfig::setAO 2-97
CDAQMXChConfig::setBurnout 2-97
CDAQMXChConfig::setChatFilter 2-97
CDAQMXChConfig::setCOM 2-98
CDAQMXChConfig::setDeenergize 2-98
CDAQMXChConfig::setDELTA 2-99
CDAQMXChConfig::setDI 2-99
CDAQMXChConfig::setFilter 2-100
CDAQMXChConfig::setHold 2-100
CDAQMXChConfig::setMXChConfig 2-100
CDAQMXChConfig::setPULSE 2-101
CDAQMXChConfig::setPWM 2-101
CDAQMXChConfig::setRefAlarm 2-102
CDAQMXChConfig::setRefChNo 2-102
CDAQMXChConfig::setRES 2-103
CDAQMXChConfig::setRJCType 2-103
CDAQMXChConfig::setRTD 2-104
CDAQMXChConfig::setScalling 2-104
CDAQMXChConfig::setSKIP 2-105
CDAQMXChConfig::setSpan 2-105
CDAQMXChConfig::setSTRAIN 2-105
CDAQMXChConfig::setTC 2-106
CDAQMXChConfig::setVOLT 2-106
CDAQMXChConfigData::CDAQMXChConfigData 2-108
CDAQMXChConfigData::changeRange 2-108
CDAQMXChConfigData::getClassMXChConfig 2-109
CDAQMXChConfigData::getItemError 2-109

CDAQMXChConfigData::getMXChConfigData 2-109
CDAQMXChConfigData::initialize 2-109
CDAQMXChConfigData::initMXChConfigData 2-110
CDAQMXChConfigData::isCorrect 2-110
CDAQMXChConfigData::isObject 2-111
CDAQMXChConfigData::operator= 2-111
CDAQMXChConfigData::setMXChConfig 2-111
CDAQMXChConfigData::setMXChConfigData 2-112
CDAQMXChConfigData::setRRJC 2-112
CDAQMXChConfigData クラス 2-107
CDAQMXChConfig クラス 2-86
CDAQMXChID::CDAQMXChID 2-115
CDAQMXChID::getAlarmType 2-115
CDAQMXChID::getAlarmValueOFF 2-115
CDAQMXChID::getAlarmValueON 2-116
CDAQMXChID::getChName 2-116
CDAQMXChID::getChType 2-116
CDAQMXChID::getComment 2-117
CDAQMXChID::getKind 2-117
CDAQMXChID::getMXAlarm 2-117
CDAQMXChID::getMXChID 2-117
CDAQMXChID::getRange 2-118
CDAQMXChID::getScale 2-118
CDAQMXChID::getTag 2-118
CDAQMXChID::getUnit 2-118
CDAQMXChID::initialize 2-118
CDAQMXChID::initMXChID 2-119
CDAQMXChID::isObject 2-119
CDAQMXChID::isValid 2-119
CDAQMXChID::operator= 2-120
CDAQMXChID::setAlarmValue 2-120
CDAQMXChID::setChType 2-120
CDAQMXChID::setComment 2-121
CDAQMXChID::setMXChID 2-121
CDAQMXChID::setTag 2-121
CDAQMXChID::setType 2-122
CDAQMXChID::setUnit 2-122
CDAQMXChID::setValid 2-122
CDAQMXChID::toChName 2-122
CDAQMXChID::toChNo 2-123
CDAQMXChID::toUnitNo 2-123
CDAQMXChID クラス 2-113
CDAQMXChInfo::CDAQMXChInfo 2-126
CDAQMXChInfo::getDisplayMax 2-126
CDAQMXChInfo::getDisplayMin 2-126
CDAQMXChInfo::getFIFOIndex 2-126
CDAQMXChInfo::getFIFONo 2-127
CDAQMXChInfo::getMXChInfo 2-127
CDAQMXChInfo::getOriginalMax 2-127
CDAQMXChInfo::getOriginalMin 2-127
CDAQMXChInfo::getRealMax 2-128
CDAQMXChInfo::getRealMin 2-128
CDAQMXChInfo::initialize 2-128
CDAQMXChInfo::isObject 2-129
CDAQMXChInfo::operator= 2-129
CDAQMXChInfo::setFIFOIndex 2-129
CDAQMXChInfo::setFIFONo 2-130
CDAQMXChInfo::setMXChInfo 2-130
CDAQMXChInfo クラス 2-124
CDAQMXConfig::CDAQMXConfig 2-133
CDAQMXConfig::getChName 2-133
CDAQMXConfig::getClassMXBalanceData 2-133
CDAQMXConfig::getClassMXChConfig 2-134
CDAQMXConfig::getClassMXChConfigData 2-134
CDAQMXConfig::getClassMXNetInfo 2-134
CDAQMXConfig::getClassMXOutputData 2-134
CDAQMXConfig::getClassMXStatus 2-135
CDAQMXConfig::getClassMXSysInfo 2-135
CDAQMXConfig::getItemError 2-135
CDAQMXConfig::getMXConfigData 2-135
CDAQMXConfig::getRangePoint 2-136
CDAQMXConfig::getSpanPoint 2-136
CDAQMXConfig::initialize 2-136
CDAQMXConfig::initMXConfigData 2-137
CDAQMXConfig::isCorrect 2-137
CDAQMXConfig::isObject 2-138, 12-92
CDAQMXConfig::operator= 2-138
CDAQMXConfig::reconstruct 2-139
CDAQMXConfig::setAO 2-140
CDAQMXConfig::setAOType 2-141
CDAQMXConfig::setChKind 2-142
CDAQMXConfig::setCOM 2-142
CDAQMXConfig::setDELTA 2-143
CDAQMXConfig::setDI 2-144
CDAQMXConfig::setDOType 2-144
CDAQMXConfig::setInterval 2-145
CDAQMXConfig::setMXConfigData 2-145
CDAQMXConfig::setPULSE 2-146
CDAQMXConfig::setPWM 2-146
CDAQMXConfig::setPWMType 2-147
CDAQMXConfig::setRES 2-148
CDAQMXConfig::setRRJC 2-148
CDAQMXConfig::setRTD 2-149
CDAQMXConfig::setScalling 2-149
CDAQMXConfig::setSKIP 2-150
CDAQMXConfig::setSTRAIN 2-150
CDAQMXConfig::setTC 2-151
CDAQMXConfig::setTempUnit 2-151
CDAQMXConfig::setVOLT 2-152
CDAQMXConfig クラス 2-131
CDAQMXDataBuffer::CDAQMXDataBuffer 12-75
CDAQMXDataBuffer::create 12-75
CDAQMXDataBuffer::currentDataInfo 12-76
CDAQMXDataBuffer::currentDateTime 12-76
CDAQMXDataBuffer::getClassMXChInfo 12-76
CDAQMXDataBuffer::getDataNum 12-77
CDAQMXDataBuffer::getDateTime 12-77
CDAQMXDataBuffer::initialize 12-77
CDAQMXDataBuffer::isCurrent 12-77
CDAQMXDataBuffer::next 12-78
CDAQMXDataBuffer::setChInfo 12-78
CDAQMXDataBuffer::setDataInfo 12-76, 12-78
CDAQMXDataBuffer::setDateTime 12-79
CDAQMXDataBuffer クラス 12-74
CDAQMXDataInfo::CDAQMXDataInfo 2-154
CDAQMXDataInfo::getAlarmName 2-155
CDAQMXDataInfo::getClassMXChInfo 2-155
CDAQMXDataInfo::getMXDataInfo 2-155
CDAQMXDataInfo::getStatus 2-156
CDAQMXDataInfo::isAlarm 2-156
CDAQMXDataInfo::isObject 2-157
CDAQMXDataInfo::operator= 2-157
CDAQMXDataInfo::setAlarm 2-158
CDAQMXDataInfo::setClassMXChInfo 2-158
CDAQMXDataInfo::setMXDataInfo 2-158
CDAQMXDataInfo::setStatus 2-158
CDAQMXDataInfo クラス 2-153
CDAQMXDateTime::CDAQMXDateTime 2-160
CDAQMXDateTime::getMXDateTime 2-160
CDAQMXDateTime::initMXDateTime 2-161
CDAQMXDateTime::isObject 2-161
CDAQMXDateTime::operator= 2-161
CDAQMXDateTime::setMXDateTime 2-162
CDAQMXDateTime クラス 2-159
CDAQMXDOData::CDAQMXDOData 2-164
CDAQMXDOData::getDOONOFF 2-164

CDAQMXDOData::getDOValid 2-165
CDAQMXDOData::getMXDO 2-165
CDAQMXDOData::getMXDOData 2-165
CDAQMXDOData::initialize 2-166
CDAQMXDOData::initMXDOData 2-166
CDAQMXDOData::isObject 2-166
CDAQMXDOData::operator= 2-167
CDAQMXDOData::setDO 2-167
CDAQMXDOData::setDOONOFF 2-167
CDAQMXDOData::setMXDOData 2-168
CDAQMXDOData クラス 2-163
CDAQMXDOList::add 12-81
CDAQMXDOList::CDAQMXDOList 12-81
CDAQMXDOList::change 12-82
CDAQMXDOList::copyData 12-83
CDAQMXDOList::create 12-83
CDAQMXDOList::getClassMXDOData 12-83
CDAQMXDOList::getCurrent 12-84
CDAQMXDOList::initCurrent 12-84
CDAQMXDOList クラス 12-80
CDAQMXItemConfig::CDAQMXItemConfig 12-86
CDAQMXItemConfig::getDoubleAlarmOFF 12-87
CDAQMXItemConfig::getDoubleAlarmON 12-87
CDAQMXItemConfig::getDoubleHisterisys 12-88
CDAQMXItemConfig::getDoublePresetValue 12-88
CDAQMXItemConfig::getDoubleScaleMax 12-89
CDAQMXItemConfig::getDoubleScaleMin 12-89
CDAQMXItemConfig::getDoubleSpanMax 12-90
CDAQMXItemConfig::getDoubleSpanMin 12-90
CDAQMXItemConfig::getHisterisys 12-91
CDAQMXItemConfig::getMaxLenItemName 12-91
CDAQMXItemConfig::readItem 12-92
CDAQMXItemConfig::toItemName 12-93
CDAQMXItemConfig::toItemNo 12-93
CDAQMXItemConfig::writeItem 12-94
CDAQMXItemConfig クラス 12-85
CDAQMXList::addData 12-96
CDAQMXList::CDAQMXList 12-96
CDAQMXList::copy 12-82, 12-96
CDAQMXList::create 12-97
CDAQMXList::del 12-97
CDAQMXList::delData 12-97
CDAQMXList::getData 12-98
CDAQMXList::getMaxNo 12-98
CDAQMXList::getNum 12-98
CDAQMXList::initialize 12-98
CDAQMXList::isData 12-99
CDAQMXList クラス 12-95
CDAQMXNetInfo::CDAQMXNetInfo 2-170
CDAQMXNetInfo::getAddress 2-170
CDAQMXNetInfo::getGateway 2-170
CDAQMXNetInfo::getHost 2-170
CDAQMXNetInfo::getMXNetInfo 2-171
CDAQMXNetInfo::getPart 2-171
CDAQMXNetInfo::getPort 2-171
CDAQMXNetInfo::getSubMask 2-172
CDAQMXNetInfo::initialize 2-172
CDAQMXNetInfo::initMXNetInfo 2-172
CDAQMXNetInfo::isObject 2-173
CDAQMXNetInfo::operator= 2-173
CDAQMXNetInfo::setMXNetInfo 2-173
CDAQMXNetInfo クラス 2-169
CDAQMXOutputData::CDAQMXOutputData 2-175
CDAQMXOutputData::getErrorChoice 2-175
CDAQMXOutputData::getIdleChoice 2-176
CDAQMXOutputData::getMXOutput 2-176
CDAQMXOutputData::getMXOutputData 2-176
CDAQMXOutputData::getOutputType 2-177
CDAQMXOutputData::getPresetValue 2-177
CDAQMXOutputData::getPulseTime 2-177
CDAQMXOutputData::initialize 2-178
CDAQMXOutputData::initMXOutputData 2-178
CDAQMXOutputData::isObject 2-178
CDAQMXOutputData::operator= 2-179
CDAQMXOutputData::setChoice 2-179
CDAQMXOutputData::setMXOutputData 2-179
CDAQMXOutputData::setOutputType 2-180
CDAQMXOutputData::setPulseTime 2-180
CDAQMXSegment::CDAQMXSegment 2-182
CDAQMXSegment::getMXSegment 2-182
CDAQMXSegment::getPattern 2-182
CDAQMXSegment::initialize 2-183
CDAQMXSegment::initMXSegment 2-183
CDAQMXSegment::isObject 2-183
CDAQMXSegment::operator= 2-184
CDAQMXSegment::setMXSegment 2-184
CDAQMXSegment::setPattern 2-184
CDAQMXSegment クラス 2-174, 2-181
CDAQMXStatus::CDAQMXStatus 2-186
CDAQMXStatus::getCFRemain 2-186
CDAQMXStatus::getCFSize 2-187
CDAQMXStatus::getCFStatus 2-187
CDAQMXStatus::getConfigCnt 2-187
CDAQMXStatus::getDateTime 2-187
CDAQMXStatus::getFIFONum 2-188
CDAQMXStatus::getFIFOStatus 2-188
CDAQMXStatus::getInterval 2-188
CDAQMXStatus::getMilliSecond 2-189
CDAQMXStatus::getMXFIFOInfo 2-189
CDAQMXStatus::getMXStatus 2-189
CDAQMXStatus::getNewDataNo 2-190
CDAQMXStatus::getOldDataNo 2-190
CDAQMXStatus::getTime 2-190
CDAQMXStatus::getTimeCnt 2-191
CDAQMXStatus::getUnitStatus 2-191
CDAQMXStatus::initialize 2-191
CDAQMXStatus::initMXStatus 2-191
CDAQMXStatus::isBackup 2-192
CDAQMXStatus::isDataNo 2-192
CDAQMXStatus::isObject 2-192
CDAQMXStatus::operator= 2-193
CDAQMXStatus::setMXStatus 2-193
CDAQMXStatus クラス 2-185
CDAQMXSysInfo::CDAQMXSysInfo 2-195
CDAQMXSysInfo::getCFTimeout 2-196
CDAQMXSysInfo::getCFWriteMode 2-196
CDAQMXSysInfo::getChNum 2-196
CDAQMXSysInfo::getFIFONo 2-197
CDAQMXSysInfo::getFrequency 2-197
CDAQMXSysInfo::getIntegral 2-197
CDAQMXSysInfo::getInterval 2-198
CDAQMXSysInfo::getItemError 2-198
CDAQMXSysInfo::getMAC 2-198
CDAQMXSysInfo::getModuleSerial 2-199
CDAQMXSysInfo::getModuleType 2-199
CDAQMXSysInfo::getModuleVersion 2-200
CDAQMXSysInfo::getMXModuleData 2-200
CDAQMXSysInfo::getMXSystemInfo 2-200
CDAQMXSysInfo::getOption 2-201
CDAQMXSysInfo::getPartNo 2-201
CDAQMXSysInfo::getRealType 2-201
CDAQMXSysInfo::getStandbyType 2-202
CDAQMXSysInfo::getStyle 2-202
CDAQMXSysInfo::getTempUnit 2-202
CDAQMXSysInfo::getTerminalType 2-203
CDAQMXSysInfo::getUnitNo 2-203

CDAQMXSysInfo::getUnitSerial 2-203
CDAQMXSysInfo::getUnitType 2-203
CDAQMXSysInfo::initialize 2-204
CDAQMXSysInfo::initMXSystemInfo 2-204
CDAQMXSysInfo::isCorrect 2-204
CDAQMXSysInfo::isModuleValid 2-205
CDAQMXSysInfo::isObject 2-205
CDAQMXSysInfo::operator= 2-205
CDAQMXSysInfo::setCFTimeout 2-206
CDAQMXSysInfo::setCFWriteMode 2-206
CDAQMXSysInfo::setModule 2-206
CDAQMXSysInfo::setMXSystemInfo 2-207
CDAQMXSysInfo::setRealModule 2-207
CDAQMXSysInfo::setTempUnit 2-207
CDAQMXSysInfo::setUnitNo 2-208
CDAQMXSysInfo クラス 2-194
CDAQMXTransmit::CDAQMXTransmit 2-210
CDAQMXTransmit::getMXTransmit 2-210
CDAQMXTransmit::getTransmit 2-210
CDAQMXTransmit::initialize 2-211
CDAQMXTransmit::initMXTransmit 2-211
CDAQMXTransmit::isObject 2-211
CDAQMXTransmit::operator= 2-212
CDAQMXTransmit::setMXTransmit 2-212
CDAQMXTransmit::setTransmit 2-212
CDAQMXTransmitList::add 12-101
CDAQMXTransmitList::CDAQMXTransmitList 12-101
CDAQMXTransmitList::change 12-102
CDAQMXTransmitList::copy 12-102
CDAQMXTransmitList::copyData 12-103
CDAQMXTransmitList::create 12-103
CDAQMXTransmitList::getClassMXTransmit 12-103
CDAQMXTransmitList::getCurrent 12-104
CDAQMXTransmitList::initCurrent 12-104
CDAQMXTransmitList クラス 12-100
CDAQMXTransmit クラス 2-209
CDAQMX クラス 2-37
CF 書き込み種類 6-8, 18-9
CF ステータス種類 6-8, 18-9
changeAOPWMDataMX 5-3
changeAOPWMMX100 17-3
changeAOPWMValueMX100 17-4
changeBalanceMX 5-4
changeBalanceMX100 17-5
changeDODataMX 5-5
changeDOMX100 17-6
changeTransmitMX 5-6
changeTransmitMX100 17-7
channelBalanceValidMX100 17-101
channelBalanceValueMX100 17-102
channelBurnoutMX100 17-103
channelChatFilterMX100 17-104
channelDeenergizeMX100 17-105
channelDisplayMaxMX100 17-106
channelDisplayMinMX100 17-107
channelDoublePresetValueMX100 17-108
channelDoubleScaleMaxMX100 17-109
channelDoubleScaleMinMX100 17-110
channelDoubleSpanMaxMX100 17-111
channelDoubleSpanMinMX100 17-112
channelErrorChoiceMX100 17-113
channelFIFOIndexMX100 17-114
channelFIFONoMX100 17-115
channelFilterMX100 17-116
channelHoldMX100 17-117
channelIdleChoiceMX100 17-118
channelKindMX100 17-119
channelNumberMX100 17-120
channelOutputTypeMX100 17-121
channelPointDA100 24-42
channelPointDA100Reader 24-84
channelPointMX100 17-122
channelPresetValueMX100 17-123
channelPulseTimeMX100 17-124
channelRangeMX100 17-125
channelRealMaxMX100 17-126
channelRealMinMX100 17-127
channelRefAlarmMX100 17-128
channelRefChNoMX100 17-129
channelRJCTypeMX100 17-130
channelRJCVoltMX100 17-131
channelScaleMaxMX100 17-132
channelScaleMinMX100 17-133
channelScaleTypeMX100 17-134
channelSpanMaxMX100 17-135
channelSpanMinMX100 17-136
channelStatusDA100 24-43
channelStatusDA100Reader 24-85
channelValidMX100 17-137
clearBalanceMX100 17-8
closeDA100 24-3
closeDA100Reader 24-75
closeDARWIN 10-2
closeMX 5-7
closeMX100 17-9
commandAOPWMMX100 17-10
commandBalanceMX100 17-11
commandDOMX100 17-12
commandTransmitMX100 17-13
compareDataNoMX 5-8
computeDARWIN 10-3
copyAOPWMMX100 17-14
copyBalanceMX100 17-15
copyDOMX100 17-16
copyTransmitMX100 17-17
createAOPWMMX100 17-18
createBalanceMX100 17-19
createDOMX100 17-20
createTransmitMX100 17-21
currentAOPWMValidMX100 17-138
currentAOPWMValueMX100 17-139
currentBalanceResultMX100 17-140
currentBalanceValidMX100 17-141
currentBalanceValueMX100 17-142
currentDoubleAOPWMValueMX100 17-143
currentDOValidMX100 17-144
currentDOValueMX100 17-145
currentTransmitMX100 17-146

D

DAQDA100 25-14
DAQDA100READER 25-20
DAQDARWIN 11-12
DAQINT64 6-27
DAQMX 6-27
DAQMX100 18-20
DARWIN 1-7
DarwinChInfo 構造体 11-12
DarwinDataInfok 構造体 11-13
DarwinDateTime 構造体 11-12
DarwinModuleInfoko 構造体 11-13
DarwinSystemInfo 構造体 11-14
DarwinUnitInfo 構造体 11-14
DARWIN 専用のクラス 7-1
DARWIN 用 API の機能 1-2

dataAlarmDA100 24-44
dataAlarmDA100Reader 24-86
dataAlarmMX100 17-147
dataDayDA100 24-45
dataDayDA100Reader 24-87
dataDayMX100 17-148
dataDoubleValueDA100 24-46
dataDoubleValueDA100Reader 24-88
dataDoubleValueMX100 17-149
dataHourDA100 24-47
dataHourDA100Reader 24-89
dataHourMX100 17-150
dataMilliSecDA100Reader 24-90
dataMilliSecMX100 17-151
dataMinuteDA100 24-48
dataMinuteDA100Reader 24-91
dataMinuteMX100 17-152
dataMonthDA100 24-49
dataMonthDA100Reader 24-92
dataMonthMX100 17-153
dataNumChMX100 17-154
dataNumFIFOMX100 17-155
dataSecondDA100 24-50
dataSecondDA100Reader 24-93
dataSecondMX100 17-156
dataStatusDA100 24-51
dataStatusDA100Reader 24-94
dataStatusMX100 17-157
dataStringValueDA100 24-52
dataStringValueDA100Reader 24-95
dataStringValueMX100 17-158
dataTimeMX100 17-159
dataValidMX100 17-160
dataValueDA100 24-53
dataValueDA100Reader 24-96
dataValueMX100 17-161
dataYearDA100 24-54
dataYearDA100Reader 24-97
dataYearMX100 17-162
decrementDataNoMX 5-9
deleteAOPWMMX100 17-22
deleteBalanceMX100 17-23
deleteDOMX100 17-24
deleteTransmitMX100 17-25
displaySegmentMX100 17-26
DO データ ..... 付 -2
DO データ番号 ..... 付 -2

E

errorMaxLengthDA100 24-55
errorMaxLengthDA100Reader 24-98
errorMaxLengthMX100 17-163
establishDA100 24-4
establishDARWIN 10-4

F

FIFO ..... 付 -3
FIFO ステータス値 6-9, 18-10
FIFO 値 ..... 付 -8
FIFO の開始 / 停止 14-1, 15-1, 16-1
FIFO 番号 ..... 付 -3
formatCFMX 5-10
formatCFMX100 17-27

G

getAlarmNameDA100 24-56
getAlarmNameDA100Reader 24-99
getAlarmNameDARWIN 10-5
getAlarmNameMX 5-11
getAlarmNameMX100 17-164
getAOPWMDataMX 5-12
getBalanceMX 5-13
getChannel 7-19
getChannelCommentMX100 17-165
getChannelTagMX100 17-166
getChannelUnitDA100 24-57
getChannelUnitDA100Reader 24-100
getChannelUnitMX100 17-167
getChConfigMX 5-14
getChDataByASCIIDARWIN 10-6
getChDataByBinaryDARWIN 10-7
getChDataMX 5-15
getChDataNoMX 5-16
getChInfoDARWIN 10-8
getChInfoMX 5-17
getConfigDataMX 5-18
getData 7-19
getDODataMX 5-19
getErrorMessageDA100 24-58
getErrorMessageDA100Reader 24-101
getErrorMessageDARWIN 10-9
getErrorMessageMX 5-20
getErrorMessageMX100 17-168
getFIFODataNoMX 5-21
getItemAllMX100 17-28
getItemErrorMX 5-22
getLastErrorMX 5-23
getMaxLenAlarmNameMX 5-24
getMaxLenErrorMessageDARWIN 10-11
getMaxLenErrorMessageMX 5-25
getModuleNameDA100 24-59
getModuleSerialMX100 17-169
getNetHostMX100 17-170
getOutputMX 5-26
getReportStatusDARWIN 10-12
getRevisionAPIDARWIN 10-13
getRevisionAPIMX 5-27
getSetDataByLineDA100 24-5
getSetDataByLineDARWIN 10-14
getStatusByteDARWIN 10-15
getStatusDataMX 5-28
getSystemConfigDARWIN 10-16
getSystemConfigMX 5-29
getTimeDataMX 5-30
getUnitPartNoMX100 17-171
getUnitSerialMX100 17-172
getVersionAPIDARWIN 10-17
getVersionAPIMX 5-31

I

incrementDataNoMX 5-32
initBalanceMX100 17-29
initDataChMX100 17-30
initDataFIFOMX100 17-31
initItemMX100 17-32
initSetValueDA100 24-6
initSetValueMX100 17-33
initSystemDARWIN 10-18
initSystemMX 5-33
isDataNoMX 5-34

isDataNoVBMX 5-35
itemErrorMX100 17-173
itemMaxLengthMX100 17-174

L

lastErrorMX100 17-175

M

mathInfoChDA100 24-7
mathInfoChDA100Reader 24-76
mathInstChDA100 24-8
mathInstChDA100Reader 24-77
measDataChMX100 17-34
measDataFIFOMX100 17-35
measInfoChDA100 24-9
measInfoChDA100Reader 24-78
measInstChDA100 24-10
measInstChDA100Reader 24-79
measInstChMX100 17-36
measInstFIFOMX100 17-37
measStartMX100 17-38
measStopMX100 17-39
moduleChNumMX100 17-176
moduleCodeDA100 24-60
moduleFIFONoMX100 17-177
moduleIntegralMX100 17-178
moduleIntervalMX100 17-179
moduleRealTypeMX100 17-180
moduleStandbyTypeMX100 17-181
moduleTerminalMX100 17-182
moduleTypeMX100 17-183
moduleValidMX100 17-184
moduleVersionMX100 17-185
MX100/DARWIN 共通クラス 2-1, 2-15
MX100 用クラス詳細 12-20
MX100 専用のクラス 2-1
MX100 用 API の機能 1-1
MX100 用クラス詳細 2-37, 12-20
MXAlarm 構造体 6-28
MXAOPWM 6-38
MXAOPWMData 6-38
MXBalance 6-36
MXBalanceData 6-36
MXBalanceResult 6-36
MXCFInfo 構造体 6-34
MXChConfigAIDI 構造体 6-28
MXChConfigAI 構造体 6-29
MXChConfigData 構造体 6-31
MXChConfigDO 構造体 6-29
MXChConfig 構造体 6-30
MXChID 構造体 6-30
MXChInfo 構造体 6-31
MXConfigData 構造体 6-37
MXDataInfo 構造体 6-28
MXDataNo 6-27
MXDateTime 構造体 6-27
MXDOData 構造体 6-38
MXDO 構造体 6-38
MXFIFOInfo 構造体 6-34
MXINT64 構造体 6-27
MXModuleData 構造体 6-33
MXNetInfo 構造体 6-36
MXOutput 6-37
MXOutputData 6-37
MXProductInfo 構造体 6-32
MXSegment 構造体 6-38
MXStatus 構造体 6-35
MXSystemInfo 構造体 6-34
MXTransmit 6-38
MXUnitData 構造体 6-32
MXUserTime 6-27

N

netAddressMX100 17-186
netGatewayMX100 17-187
netPortMX100 17-188
netSubmaskMX100 17-189

O

openDA100 24-11
openDA100Reader 24-80
openDARWIN 10-19
openMX 5-36
openMX100 17-40

P

PC(パーソナルコンピュータ) 1-6
PWM レンジ 6-16, 18-17, 18-18

R

rangePointMX100 17-190
readItemMX100 17-41
receiveByteDA100 24-12
receiveByteDARWIN 10-20
receiveLineDA100 24-13
receiveLineDARWIN 10-21
reconstructDA100 24-14
reconstructMX100 17-42
reportingDARWIN 10-22
resetBalanceMX 5-37
revisionAPIDA100 24-61
revisionAPIDA100Reader 24-102
revisionAPIMX100 17-191
RJC 種類 6-7, 18-8
RJC 電圧値 ..... 付 -3
runBalanceMX 5-38
runCommandDA100 24-15
runCommandDARWIN 10-23

S

sendConfigMX100 17-43
sendLineDA100 24-16
sendLineDARWIN 10-24
sendTriggerDA100 24-17
sendTriggerDARWIN 10-25
setAlarmDARWIN 10-26
setAlarmMX 5-39
setAlarmMX100 17-44
setAlarmValueMX100 17-45
setAOMX 5-40
setAOPWMDataMX 5-41
setAOTypeMX 5-42
setBackupMX 5-43
setBalanceMX 5-44
setBurnoutMX 5-45
setBurnoutMX100 17-46
setCFWriteModeMX100 17-47
setChAlarmDA100 24-18

setChatFilterMX 5-46
setChatFilterMX100 17-48
setChCommentMX100 17-49
setChConfigMX 5-47
setChDELTADA100 24-20
setChDELTAMX100 17-50
setChKindMX100 17-51
setChoiceMX 5-48
setChoiceMX100 17-52
setChRRJCDA100 24-21
setChRRJCMX100 17-53
setChTagMX100 17-54
setChUnitDA100 24-22
setChUnitMX100 17-55
setCommentMX 5-49
setCOMMX 5-50
setConfigDataMX 5-51
setDateTimeDARWIN 10-27
setDateTimeMX 5-52
setDateTimeNowDA100 24-23
setDateTimeNowDARWIN 10-28
setDateTimeNowMX 5-53
setDateTimeNowMX100 17-56
setDeenergizeMX100 17-57
setDELTADARWIN 10-29
setDELTAMX 5-54
setDIDARWIN 10-30
setDIMX 5-55
setDODataMX 5-56
setDOTypeMX 5-57
setDoubleAlarmMX100 17-58
setDoubleAlarmValueMX100 17-59
setDoubleChoiceMX100 17-60
setDoubleHisterisysMX100 17-61
setDoubleScaleMX100 17-62
setDoubleSpanMX100 17-63
setFilterMX 5-58
setFilterMX100 17-64
setHisterisysMX100 17-65
setHoldMX100 17-66
setIntegralMX100 17-67
setIntervalMX 5-59
setIntervalMX100 17-68
setItemAllMX100 17-69
setMADARWIN 10-31
setOutputMX 5-60
setOutputTypeMX 5-61
setOutputTypeMX100 17-70
setPOWERDARWIN 10-32
setPULSEMX 5-62
setPulseTimeMX 5-63
setPulseTimeMX100 17-71
setPWMMX 5-64
setPWMTypeMX 5-65
setRangeDA100 24-24
setRangeMX100 17-72
setRefAlarmMX 5-66
setRefAlarmMX100 17-73
setRESMX 5-67
setRJCTypeMX 5-68
setRJCTypeMX100 17-74
setRRJCDARWIN 10-34
setRRJCMX 5-69
setRTDDARWIN 10-35
setRTDMX 5-70
setScaleMX100 17-75
setScallingUnitDARWIN 10-36
setScallingUnitMX 5-71
setSegmentMX 5-72
setSKIPDARWIN 10-37
setSKIPMX 5-73
setSpanMX100 17-76
setSTRAINDARWIN 10-38
setSTRAINMX 5-74
setSystemConfigMX 5-75
setSystemTimeoutMX 5-76
setTagMX 5-77
setTCDARWIN 10-39
setTCMX 5-78
setTempUnitMX 5-79
setTimeOutDARWIN 10-40
setTimeOutMX 5-80
setTransmitMX 5-81
setUnitNoMX 5-82
setUnitNoMX100 17-77
setUnitTempMX100 17-78
setUserTimeMX 5-83
setUserTimeVBMX 5-84
setVOLTDARWIN 10-41
setVOLTMX 5-85
SKIP レンジ 25-4, 25-12
startFIFOMX 5-86
statusBackupMX100 17-192
statusByteDA100 24-62
statusCFMX100 17-193
statusCFRemainMX100 17-194
statusCFSizeMX100 17-195
statusCodeDA100 24-63
statusDayMX100 17-196
statusFIFOIntervalMX100 17-197
statusFIFOMX100 17-198
statusFIFONumMX100 17-199
statusHourMX100 17-200
statusMilliSecMX100 17-201
statusMinuteMX100 17-202
statusMonthMX100 17-203
statusReportDA100 24-64
statusSecondMX100 17-204
statusTimeMX100 17-205
statusUnitMX100 17-206
statusYearMX100 17-207
stopFIFOMX 5-87
switchBackupMX100 17-79
switchCodeDA100 24-26
switchComputeDA100 24-27
switchDOMX100 17-80
switchModeDA100 24-28
switchReportDA100 24-29
switchTransmitMX100 17-81

T

talkCalibrationChDataDA100 24-30
talkCalibrationDataDA100 24-31
talkCalibrationDataDARWIN 10-42
talkChDataInstMX 5-88
talkChDataMX 5-89
talkChDataVBMX 5-90
talkChInfoDARWIN 10-43
talkChInfoMX 5-91
talkConfigMX 5-92
talkDataByASCIIDARWIN 10-44
talkDataByBinaryDARWIN 10-45
talkFIFODataInstMX 5-93
talkFIFODataMX 5-94
talkFIFODataVBMX 5-95

talkOperationChDataDA100 24-32
talkOperationDataDA100 24-33
talkOperationDataDARWIN 10-46
talkSetupChDataDA100 24-34
talkSetupDataDA100 24-35
talkSetupDataDARWIN 10-47
toAlarmNameDA100 24-65
toAlarmNameDA100Reader 24-103
toAlarmNameDARWIN 10-48
toAlarmNameMX 5-96
toAlarmNameMX100 17-208
toAOPWMValueMX100 17-209
toChannelCommentMX100 17-210
toChannelTagMX100 17-211
toChannelUnitDA100 24-66
toChannelUnitDA100Reader 24-104
toChannelUnitMX100 17-212
toDateTimeMX 5-98
toDoubleValueDA100 24-67
toDoubleValueDA100Reader 24-105
toDoubleValueDARWIN 10-49
toDoubleValueMX 5-99
toDoubleValueMX100 17-213
toErrorMessageDA100 24-68
toErrorMessageDA100Reader 24-106
toErrorMessageDARWIN 10-50
toErrorMessageMX 5-100
toErrorMessageMX100 17-214
toItemNameMX100 17-215
toItemNoMX100 17-216
toModuleNameDA100 24-69
toModuleSerialMX100 17-217
toNetHostMX100 17-218
toRealValueMX100 17-219
toStringValueDA100 24-70
toStringValueDA100Reader 24-107
toStringValueDARWIN 10-51
toStringValueMX 5-102
toStringValueMX100 17-220
toStyleVersionMX 5-103
toStyleVersionMX100 17-221
toUnitPartNoMX100 17-222
toUnitSerialMX100 17-223
transModeDARWIN 10-52

U

unitCFWriteModeMX100 17-224
unitFrequencyMX100 17-225
unitIntervalDA100 24-71
unitMACMX100 17-226
unitNoMX100 17-227
unitOptionMX100 17-228
unitStyleMX100 17-229
unitTempMX100 17-230
unitTypeMX100 17-231
unitValidDA100 24-72
updateAOPWMDataMX100 17-82
updateBalanceMX100 ..... 17-83, 17-84
updateConfigMX100 ..... 17-84, 17-85
updateDODataMX100 ..... 17-85, 17-86
updateInfoChMX100 17-86
updateOutputMX100 ..... 17-87, 17-88
updateReportStatusDA100 24-36
updateStatusDA100 24-37
updateStatusMX100 ..... 17-88, 17-89
updateSystemConfigDA100 24-38
updateSystemMX100 ..... 17-89, 17-90
userAOPWMValidMX100 17-232
userAOPWMValueMX100 17-233
userBalanceValidMX100 17-234
userBalanceValueMX100 17-235
userDoubleAOPWMValueMX100 17-236
userDOValidMX100 17-237
userDOValueMX100 17-238
userTransmitMX100 17-239

V

versionAPIDA100 24-73
versionAPIDA100Reader 24-108
versionAPIMX100 17-240
Visual C/Visual C++ の定数 25-2
VisualStudio2005 1-6

W

writeItemMX100 17-90

ア

新しく追加された DARWIN 用関数 ..... 付 -21
新しく追加された DARWIN 用メンバ ..... 付 -25
新しく追加された MX100 用関数 ..... 付 -20
新しく追加された MX100 用クラス ..... 付 -22
新しく追加された MX100 用メンバ ..... 付 -22
アラーム ..... 付 -4, 付 -14
アラーム種類 ..... 6-5, 11-3, 18-6, 25-7, 25-18, 付 -14
アラーム値 ..... 付 -4, 付 -14
アラームレベル ..... 付 -4, 付 -14

イ

インクルードファイルのパス 2-9, 3-6, 7-5, 8-4, 13-12
インクルードファイルのパスを追加 19-7, 20-6, 20-11
インストール 1-8

エ

エラー(MX100 固有) 26-3
エラー処理 2-14, 3-12, 4-11, 7-12, 8-14, 9-10
エラー番号 26-1
演算処理 11-5, 25-9
演算チャネル ..... 付 -17

オ

応答 ..... 付 -4, 付 -14
オプション 6-10, 18-11
オペレーティングシステム 1-6
温度単位種類 6-8, 18-9

カ

改版履歴 vi
拡張 API 用 MX100 のクラス 12-1
型 6-26
型(DARWIN) 11-10
型(MX100) 6-24, 18-1
形名 iii
カレントデータ 12-12, 13-10, 14-10, 15-10, 16-10
関数の記述方法 viii

キ

機器記述子 ..... 付 -4, 付 -14
機能コマンドの実装 8-3
基準チャネル番号 ..... 付 -4
機能コマンドの実装 7-4
基本設定 ..... 6-26, 付 -7

コ

コールバック 18-20
コールバック型 25-14, 25-20
個数値 6-3, 11-2, 18-4, 25-5, 25-17
コメント ..... 付 -10

サ

再コンパイル 1-3
最大値 6-3, 11-2, 18-4, 25-6, 25-17
再リンク 1-3
削除された MX100 用メンバ ..... 付 -24
サブユニット番号 ..... 付 -15
参照 ix
参照アラーム ..... 付 -5
参照レンジ 6-10
サンプルプログラム vii

シ

時刻情報データ ..... 付 -5, 付 -15
システム構成データ 12-11, 13-9, 14-9, 15-9, 16-9, 19-5, 20-4, 21-4, 22-4, 23-4, 付 -5, 付 -15
システム制御種類 6-5, 11-3
自動制御 ..... 付 -3
周期種類 6-7, 18-8
出力種類 6-9, 18-10
出力チャネルデータ ..... 付 -5
出力チャネルデータの取得 2-8
出力チャネルデータ番号 ..... 付 -5
出力データ値 ..... 付 -2
取得関数 13-7, 14-7, 15-7, 16-7, 17-91, 19-5, 19-11, 20-4, 20-10, 21-4, 21-9, 22-4, 22-9, 23-4, 23-10
取得機能 12-8, 13-6, 14-6, 15-6, 16-6
取得コード種類 25-2, 25-5
瞬時値 ..... 付 -8
小数点位置 ..... 付 -7, 付 -16
状態取得関数 13-1, 14-1, 15-1, 16-1, 21-8, 22-8
状態遷移関数 22-1, 17-1, 19-2, 19-10, 20-1, 20-9, 21-1, 23-1, 23-9
状態データ 19-6, 20-4, 21-4, 22-4, 23-4
初期バランス結果 6-9, 18-10
初期バランスデータ ..... 付 -6
初期バランスデータの取得 2-8
初期バランスデータ番号 ..... 付 -6

ス

スキップ 18-11
スケール ..... 付 -15
スケール種類 6-6, 18-7
ステータスデータ ..... 12-12, 13-10, 14-10, 16-10, 付 -6
ステータスバイト ..... 付 -15
ステータスバイト値 11-5, 25-8
スパン ..... 付 -16
スロット番号 ..... 付 -15

セ

制御機能 .. 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
セグメント番号 ..... 付 -1
設定(運転モード)機能 19-3, 20-2, 21-2, 22-2, 23-2
設定機能 2-5, 4-2, 7-3, 8-2, 9-2, 12-4, 13-2, 14-2, 15-2, 16-2
設定項目 13-6, 14-6, 15-6, 16-6
設定項目番号 ..... 6-17, 付 -6
設定データ ..... 付 -7
設定変更機能 12-5, 13-3, 14-3, 15-3, 16-3
接点入力(DI)レンジ 11-7, 25-4, 25-11
セットアップ確定 11-5, 25-8
宣言(Visual Basic) 4-6, 9-4, 14-12, 15-12, 16-12, 21-6, 21-10
選択値 6-9, 18-10

ソ

ソースファイルでの宣言 2-9, 3-6, 7-5, 8-4, 12-15, 13-12, 19-7, 19-13, 20-6, 20-11
操作モード 11-4, 25-7
測温抵抗体(1mA)レンジ 6-12, 18-13
測温抵抗体(2mA)レンジ 6-14, 18-15
測温抵抗体(その他)のレンジ 6-15, 18-16
測温抵抗体レンジ 11-7, 25-3, 25-11
測定周期 ..... 付 -16
測定値 ..... 付 -7, 付 -16
測定チャネル ..... 付 -17
測定データ .. 12-9, 13-7, 14-7, 16-7, 19-5, 19-11, 20-4, 20-10, 21-9, 22-9, 23-4, 23-10, 付 -8, 付 -16
測定データの取得 2-8
ソフトウエア使用許諾契約書 ii
ソフトウエアの構成 1-4

タ

ターミネータ ..... 付 -16
対応機種 1-1
対応言語 1-1
タイムアウト値 ..... 付 -13, 付 -19
タイムアウト値の算出 ..... 付 -19
タグ ..... 付 -10
単位名 ..... 付 -10, 付 -18
端子種類 6-8, 18-9
端子番号 ..... 付 -15

チ

チャネル ..... 付 -9, 付 -17
チャネル / リレータイプ 25-7, 25-19
チャネル／リレータイプ 11-4
チャネル識別情報 ..... 付 -10
チャネル種類 6-5, 18-6
チャネル情報 19-5, 20-4, 21-4
チャネル情報データ 12-9, 13-7, 14-7, 15-7, 16-7, 19-11, 20-10, 21-9, 22-9, 23-4, 23-10, 付 -18
チャネル情報 22-4
チャネル数 6-6, 18-7
チャネルステータス ..... 付 -18
チャネル設定データ .... 12-10, 13-8, 14-8, 15-8, 16-8, 付 -11
チャネルタイプ ..... 付 -9, 付 -17
チャネル範囲 ..... 付 -9, 付 -17
チャネル番号 ..... 付 -9, 付 -17
チャネル名 ..... 付 -9
直流電圧レンジ 6-10, 11-6, 18-12, 25-3, 25-10
直流電流レンジ 11-8, 25-4, 25-12

ツ

通信機能 .. 2-4, 3-1, 4-1, 7-2, 8-1, 9-1, 12-3, 13-1, 14-1, 15-1, 16-1, 19-2, 19-10, 20-1, 20-9, 21-1, 21-8, 22-1, 23-1, 23-9
通信用定数 6-3, 11-2, 25-5, 25-16, 25-18
通信レンジ 6-16
通信機能 22-8

テ

データ識別子 ..... 付 -11
データ取得機能 2-7, 3-3, 4-3, 7-3, 8-3, 9-3, 19-4, 19-10, 20-3, 20-9, 21-3, 21-8, 22-3, 22-8, 23-9
データ取得機能一覧 23-3
データステータス ..... 付 -16
データステータス値 6-4, 11-3, 18-5, 25-6, 25-18
データ操作機能 12-7, 13-5, 14-5, 15-5, 16-5
データ値 ..... 付 -7, 付 -16
データ番号 ..... 付 -3
抵抗レンジ 6-15
定数(DARWIN) 11-1
定数(MX100) 6-1, 18-1
定数値 6-4, 18-3, 18-5, 25-2, 25-5, 25-17
ディジタル入力(DI)詳細レンジ 6-16
ディジタル入力(DI)レンジ 6-15
ディジタル入力(DI)詳細レンジ 18-17
伝送出力データ ..... 付 -11
伝送状態 6-9, 18-10

ト

トーカ ..... 付 -18
トーカ機能種類 11-4, 25-8

ネ

ネットワーク情報データ 12-11, 13-9, 14-9, 15-9, 16-9, 付 -11
熱電対レンジ 6-11, 11-6, 18-12, 25-3, 25-10

ハ

バーンアウト種類 6-7, 18-8
パケット ..... 付 -11
バックアップ ..... 付 -11
パッケージの構成 iii
パルス周期倍率 ..... 付 -11
パルスレンジ 6-16, 11-7, 25-4, 25-11
パワー測定項目 25-13
パワー接続方法 11-8, 25-12
パワー測定項目 11-9
パワーモニタレンジ 11-8, 25-4, 25-12

ヒ

ヒステリシス ..... 付 -4
ひずみ入力レンジ 25-4, 25-11
ひずみレンジ 18-17, 6-16
表示形式 ..... 付 -1
表示形式値 6-9, 18-10
表示時間 ..... 付 -1
表示パターン ..... 付 -1

フ

ファイル構成 1-5
フィルタ係数 6-7
フィルタ時定数 18-8
フラグ ..... 付 -12, 付 -18
フラグステータス 6-4, 11-3, 25-6
プログラムの記述 ix
プログラム例
機能コマンドの実装 7-10, 8-10, 9-8
設定データの取得 / 設定 2-13, 3-11, 4-10, 7-8, 8-8, 9-7
設定データの読み出しと書き込み 12-18, 13-15, 14-15, 15-15, 16-15
測定データの取得 2-10, 3-7, 4-7, 7-6, 8-5, 9-5, 12-16, 13-13, 14-13, 15-13, 16-13, 19-8, 19-14, 20-7, 20-12, 21-7, 21-11, 22-7, 22-11, 23-7, 23-12
トーカ機能の実装 7-11, 8-12, 9-9

マ

マニュアルの構成 v

モ

戻り値 viii
モジュール種類 6-6, 18-7
モジュール情報 ..... 付 -12
モジュール設定 16-4
モジュール番号 ..... 付 -12
文字列 11-2

ヤ

チャネル情報データ ..... 付 -10

ユ

ユーザー開発環境 1-6
ユーザカウント ..... 付 -12
ユーザデータ 12-13, 14-11, 15-11, 16-11
ユーティリティ 2-8, 3-5, 4-5, 7-4, 8-3, 9-3, 12-14, 13-11, 14-11, 15-11, 16-11, 19-6, 19-12, 20-5, 20-10, 21-5, 21-9, 22-5, 22-9, 23-5, 23-10
有効無効値 6-4, 11-2, 18-5, 25-6, 25-17
ユニット種類 6-7, 18-8
ユニット情報 ..... 付 -13
ユニットステータス値 6-8, 18-9
ユニット番号 ..... 11-5, 25-8, 25-19, 付 -13, 付 -15

ヨ

用語の参照 ix

ラ

ライブラリの指定 7-5, 19-7, 19-13, 2-9, 12-15

リ

リレー ..... 付 -18
リレータイプ ..... 付 -18
リレー番号 ..... 付 -18

レ

レポート実行種類 11-5, 25-9
レポート種類 11-6, 25-9
レポートステータス 11-6, 25-9
レンジ種類 6-10, 18-3, 18-11, 25-2, 25-10

ロ

ロードライブラリの記述 3-6, 8-4, 13-12, 20-6, 20-11