CENTUM VP

APCS ユーザカスタムブロック ライブラリ

IM 33J15U22-01JA

IM 33J15U22-01JA
2 版

はじめに

ユーザカスタムアルゴリズムは、APCS（Advanced Process Control Station）のユーザカスタムブロックで使用されるアルゴリズムです。
ユーザカスタムアルゴリズムは、Microsoft 社の Visual C++ でプログラミングします。

本書では、ユーザカスタムアルゴリズムを作成するためのライブラリを説明します。
なお、本書に関連するドキュメントとして次のものがあります。

- APCS（IM 33J15U10-01JA）
- APCS ユーザカスタムブロック（IM 33J15U20-01JA）
- APCS ユーザカスタムブロックプログラミングガイド（IM 33J15U21-01JA）
- APCS ユーザカスタムブロック開発環境（IM 33J15U23-01JA）

2nd Edition : Aug. 2019 (YK)

安全に使用するための注意事項

■ 本製品の保護・安全および改造に関する注意

- 本製品によって制御されるシステムおよび本製品自体を保護し、安全に操作するために、本書に記載されている安全に使用するための注意事項に従ってください。指示事項に反する扱いをされた場合、横河電機株式会社（以下、当社といいます）は安全性の保証をいたしかねます。
- ユーザーズマニュアルで指定していない方法で製品を使用した場合は、本製品で提供される保護機能が損なわれる可能性があります。
- 本製品によって制御されるシステムおよび本製品そのものに保護または安全回路が必要な場合は、本製品外部に別途ご用意ください。
- 本製品と組み合わせて使用する機器の仕様と設定については、必ず、機器の取扱説明書などで確認してください。
- 本製品の部品または消耗品を交換する場合は、当社が指定する部品のみを使用してください。
- 本製品および本製品の電源コードセットなどの付属品を、当社が指定する機器や用途以外に使用しないでください。
- 本製品を改造することは、固くお断りいたします。
- 本製品およびユーザーズマニュアルでは、安全に関する次の記号を使用しています。

「注意」を示します。本製品においては、感電など、人体への危険や機器損傷の恐れがあることを示すとともに、ユーザーズマニュアルを参照する必要があることを示します。また、ユーザーズマニュアルにおいては、人体への危険や機器損傷を避けるための注意事項が記載されている箇所に、本記号を「注意」「警告」の用語と一緒に使用しています。

「注意、高温表面」を示します。このマークの付いた機器は熱くなりますのでご注意ください。接触するとやけどなどの危険があります。

「保護導体端子」を示しています。感電防止のため、本製品を使用する前に、保護導体端子を必ず接地してください。

「機能接地端子」を示しています。「FG」と表示された端子も同じ機能を備えています。保護接地以外を目的とした接地端子です。本製品を使用する前に、機能接地端子を必ず接地してください。

「AC 電源」を示します。

「DC 電源」を示します。

「オン」を示します。電源スイッチなどの状態を示します。

「オフ」を示します。電源スイッチなどの状態を示します。

■ ユーザーズマニュアルに対する注意

- ユーザーズマニュアルは、最終ユーザまでお届けいただき、最終ユーザがお手元に保管して随時参照できるようにしてください。
- ユーザーズマニュアルをよく読んで、内容を理解したのちに本製品を操作してください。
- ユーザーズマニュアルは、本製品に含まれる機能詳細を説明するものであり、お客様の特定目的に適合することを保証するものではありません。
- ユーザーズマニュアルの内容については、将来予告なしに変更することがあります。
- ユーザーズマニュアルの内容について万全を期していますが、もしご不審な点や誤り、記載もれなどお気付きのことがありましたら、当社またはお買い求め先代理店までご連絡ください。乱丁、落丁はお取り替えいたします。

■ 本製品の免責について

- 当社は、保証条項に定める場合を除き、本製品に関していかなる保証も行いません。
- 本製品のご使用または使用不能から生じる間接損害については、当社は一切責任を負いかねますのでご了承ください。

■ ソフトウェア製品について

- 当社は、保証条項に定める場合を除き、本ソフトウェアに関していかなる保証も行いません。
- 本製品の各ソフトウェアに対するライセンスは、ご使用になるコンピュータの台数に応じて適正にご購入ください。
- バックアップ以外の目的で本ソフトウェアを複製することは、当社の知的所有権を侵害する行為であり、固くお断りいたします。
- 本ソフトウェアが収められているソフトウェアメディアは、大切に保管してください。
- 本ソフトウェアをリバースコンパイル、リバースアセンブリ、リバースエンジニアリング、その他の方法により人間が読み取り可能な形にすることは、固くお断りします。
- 当社から事前の書面による承認を得ることなく、本ソフトウェアの全部または一部を譲渡、交換、転貸などによって第三者に使用させることは、固くお断りいたします。

ユーザーズマニュアル中の凡例

■ ユーザーズマニュアル中のシンボルマーク

ユーザーズマニュアルの本文中では、次の各種記号が使用されています。

警告

死亡または重傷を招く可能性がある危険な状況を避けるための注意事項を記載しています。

注意

軽傷または物的損害を招く可能性がある危険な状況を避けるための注意事項を記載しています。

重要

操作や機能を知る上で、注意すべき事柄を記載しています。

補足

説明を補足するための事柄を記載しています。

参照

参照先を示します。
オンラインマニュアルでは、緑色の参照先をクリックすると、該当箇所が表示されます。黒色の参照先は、該当箇所が表示されません。

■ ユーザーズマニュアル中の表記

ユーザーズマニュアル中の表記は、次の内容を示します。

● ユーザーズマニュアル全体を通して共通に使用されている表記

入力文字列

次の書体の文字列は、ユーザが実際の操作において入力する内容を示します。
例：

```
FIC100.SV=50.0
```

▼記号

本製品のエンジニアリングを行うウィンドウの定義項目に関する説明箇所であることを示します。
本製品のエンジニアリングを行うウィンドウのヘルプメニューから「ビルダ定義項目一覧」を選択したときに開くウィンドウを経由して、選択した項目の説明を表示できます。なお、複数の定義項目が併記されている場合には、複数の定義項目に関する説明箇所であることを示します。
例：

▼ タグ名、ステーション名

Δ 記号

ユーザが入力する文字列で、空白文字（スペース）を示します。
例：

.AL Δ PIC010 Δ -SC

{} で囲った文字

ユーザが入力する文字列で、省略可能な文字列を示します。
例：

.PR Δ TAG {Δ . シート名}

● キーまたはボタン操作を示すために使用されている表記

[] で囲った文字

キーまたはボタンの操作説明において［］で囲まれている文字は、キーボードのキー、オペレーションキーボードのキー、ウィンドウに表示されるボタン名、またはウィンドウに表示されるリストボックスの選択項目のいずれかを示します。
例：

機能を切り替えるには［ESC］キーを押します。

● コマンド文やプログラム文などの書式説明の中で使用されている表記

コマンド文やプログラム文などの書式説明の中で使用されている表記は、次の内容を示します。

＜＞で囲った文字

ユーザが一定の規則に沿って任意に指定できる文字列を示します。

例：

```
#define ＜識別子＞ ＜文字列＞
```

…記号

直前のコマンドや引数が繰り返し可能であることを示します。

例：

```
lmax (arg1, arg2, …)
```

[ ] で囲った文字

省略可能な文字列を示します。

例：

```
sysalarm ＜フォーマット文字列＞ [, ＜出力値＞…]
```

| | で囲った文字

ユーザが複数候補から任意に選択できる文字列を示します。

例：

```
opeguide | ＜フォーマット文字列＞[,＜出力値＞…] |
         | OG,＜素子番号＞                       |
```

■ 図の表記

ユーザーズマニュアルに記載されている図は、説明の都合上、部分的に強調、簡略化、または省略されていることがあります。

ウィンドウの図では、機能理解や操作監視に支障を与えない範囲で、実際の表示と部品の表示位置や、大文字小文字など文字の種類が異なっている場合があります。

■ 入力文字

Windows では半角カタカナを使用できますが、本製品のソフトウェアへ入力する文字列には、半角カタカナを使用しないでください。

著作権および商標

■ 著作権

■ 商標

- CENTUM、ProSafe、Vnet/IP、PRM、Exaopc、Exaplog、Exapilot、Exaquantum、Exasmoc、Exarqe、Multivariable Optimizing Control/Robust Quality Estimation、StoryVIEW および FieldMate Validator は、横河電機株式会社の登録商標または商標です。
- 本製品で使用されている会社名、団体名、商品名およびロゴ等は、横河電機株式会社、各社または各団体の登録商標または商標です。

Blank Page

APCS ユーザカスタムブロック ライブラリ

IM 33J15U22-01JA 2 版

目　次

Blank Page

1. ユーザカスタムアルゴリズム作成用ライブラリ

本書は、ユーザカスタムアルゴリズム作成用ライブラリの関数インタフェースについて説明します。

■ 本書で使用する記号

本書で使用する記号を以下に示します。

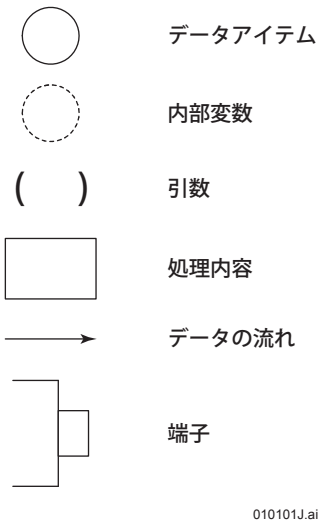


図 記号

■ 本書を読む際の注意事項

本書で記述されている「ビルダ定義項目」は、機能ブロック詳細ビルダの定義項目を意味します。

Blank Page

2. ブロックモードおよびステータス

ユーザカスタムブロックのブロックモード、アラームステータスを検査・設定する関数が用意されています。また、データステータスを検査・設定する関数が用意されています。

参照 CENTUM VP のブロックモード、ブロックステータス、データステータスの詳細については、以下を参照してください。
機能ブロック共通機能リファレンス（IM 33J15A20-01JA）「6. ブロックモードおよびステータス」

2.1 ブロックモード

本節では、ブロックモードについて説明します。

■ ブロックモード

連続制御形ユーザカスタムブロック（CSTM-C）の基本ブロックモードを以下に示します。これは、標準ブロックの PID 調節ブロックと同じです。

表　連続制御形ユーザカスタムブロックの基本ブロックモード

シンボル	名称	説明
O/S	サービスオフ（Out of Service）	機能ブロックの機能がすべて停止している状態
IMAN	初期化手動 （Initialization MANual）	演算処理と出力処理が停止している状態
TRK	トラッキング（TRacKing）	演算処理が停止していて、指定値を強制出力している状態
MAN	手動（MANual）	演算処理が停止していて、手動で設定された操作出力値を出力している状態
AUT	自動（AUTomatic）	演算処理を実行していて、演算結果を出力している状態
CAS	カスケード（CAScade）	カスケード上流ブロックから設定された設定値（CSV）を用いて演算処理を実行し、演算結果を出力している状態
PRD	プライマリダイレクト （PRimary Direct）	演算処理が停止していて、カスケード上流ブロックからの設定された設定値（CSV）を直接出力している状態
RCAS	リモートカスケード （Remote CAScade）	上位システムのコンピュータからリモート設定された設定値（RSV）を用いて制御演算処理を実行し、演算結果を出力している状態
ROUT	リモート出力（Remote OUTput）	演算処理が停止していて、上位システムのコンピュータからリモート設定された操作出力値（RMV）を直接出力している状態

汎用演算形ユーザカスタムブロック（CSTM-A）のブロックモードを以下に示します。

表　連続制御形ユーザカスタムブロックのブロックモード

シンボル	名称	説明
O/S	サービスオフ（Out of Service）	機能ブロックの機能がすべて停止している状態
AUT	自動（AUTomatic）	演算処理を実行していている状態

連続制御形ユーザカスタムブロックのブロックモードについて説明します。基本ブロックモードの中の初期化手動（IMAN）モードとトラッキング（TRK）モードは、単独では存在できず、他の基本ブロックモードと組み合わされて初めて意味のある動作状態を表します。基本ブロックモードの間には、自動（AUT）モード、手動（MAN）モード、カスケード（CAS）モード、およびプライマリダイレクト（PRD）モードの各ブロックモード間のように同時に成立できない互いに排他的な関係と、自動（AUT）モードとトラッキング（TRK）モードの間のように同時に成立ができる互いに補完的な関係があります。基本ブロックモード間の関係を以下の図に示します。

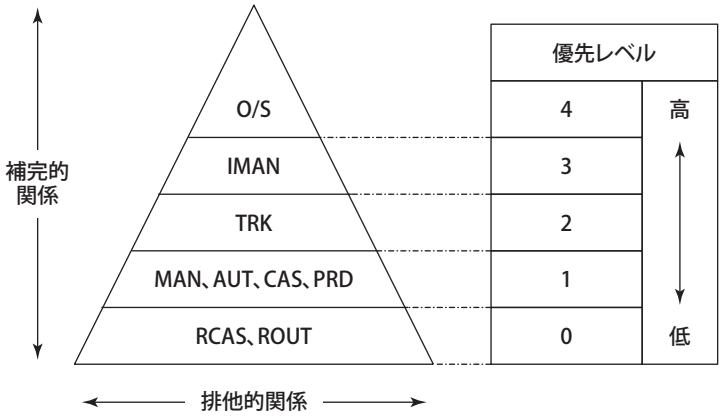


図　基本ブロックモード間の関係

参照 ブロックモードの詳細については、以下を参照してください。
機能ブロック共通機能リファレンス（IM 33J15A20-01JA）「6.1 ブロックモード」

ユーザカスタムブロックのブロックモードは内部的には 32 ビットの整数で表されます。各ブロックモードのマスクパターンが、システム定義インクルードファイル libucadef.h に #define 定義されています。ユーザカスタムアルゴリズム作成用ライブラリの引数には、以下のラベル（UCAMASK_MODE_OS など）を使用します。

参照 ラベルの詳細については、以下を参照してください。
「Appendix C.2 ブロックモード」

```
.....
/＊ブロックモード＊/
#define UCAMASK_MODE_OS          0x80000000   /＊ O/S ＊/
#define UCAMASK_MODE_IMAN        0x08000000   /＊ IMAN ＊/
#define UCAMASK_MODE_TRK         0x04000000   /＊ TRK ＊/
#define UCAMASK_MODE_MAN         0x00800000   /＊ MAN ＊/
#define UCAMASK_MODE_AUT         0x00400000   /＊ AUT ＊/
#define UCAMASK_MODE_CAS         0x00200000   /＊ CAS ＊/
#define UCAMASK_MODE_PRD         0x00100000   /＊ PRD ＊/
#define UCAMASK_MODE_RCAS        0x00080000   /＊ RCAS ＊/
#define UCAMASK_MODE_ROUT        0x00040000   /＊ ROUT ＊/
.....
```

020104J.ai

図　libucadef.hの基本ブロックモードのマスクパターン

O/S（UCAMASK_MODE_OS）と AUT（UCAMASK_MODE_AUT）は、CSTM-C と CSTM-A で共通です。それ以外のブロックモードは、CSTM-C のみで成立します。
たとえば、基本ブロックモード TRK と AUT が成立している場合 (*1)、ブロックモードの内部表現は以下のとおりです。

*1： 基本ブロックモード TRK と AUT が成立している状態を TRK（AUT）と表記します。

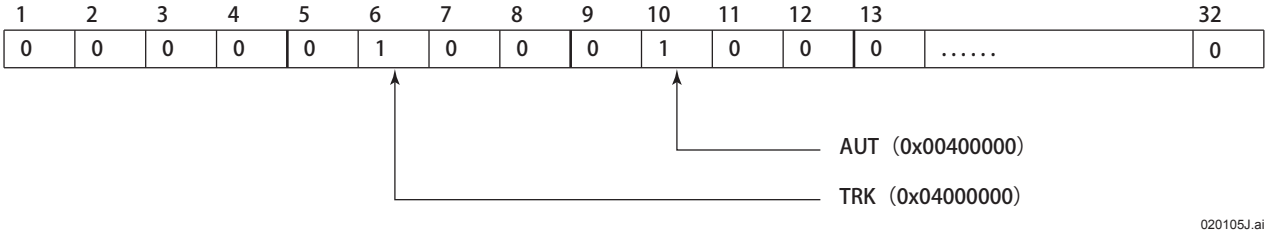


図　ブロックモードのビット位置

2.1.1 UcaModeSet

UcaModeSetは、自ユーザカスタムブロックのブロックモードを設定します。

■ 関数名

I32 UcaModeSet

■ 引数

● blockContext

データ型： UcaBlockContext
タイプ： IN
説明： ブロックコンテキスト

● mode

データ型： U32
タイプ： IN
説明： ブロックモード

■ 戻り値

- SUCCEED： ブロックモード設定に成功
- UCAERR_MODE_INVALID： 正しくないブロックモードを指定した
- UCAERR_MODE_OS： ブロックモード設定禁止状態
- UCAERR_PARA_PTRNULL： 引数のポインタが NULL

■ 引数説明

● mode

以下の中から 1 つだけ指定することができます。

- UCAMASK_MODE_MAN
- UCAMASK_MODE_AUT
- UCAMASK_MODE_CAS
- UCAMASK_MODE_PRD
- UCAMASK_MODE_RCAS
- UCAMASK_MODE_ROUT

■ 詳細説明

UcaModeSet の機能は以下のとおりです。

- 本関数で O/S（UCAMASK_MODE_OS）、IMAN（UCAMASK_MODE_IMAN）、TRK（UCAMASK_MODE_TRK）を設定することはできません。
- ブロックモードが O/S の場合、本関数を使用することはできません。ブロックモードを O/S から O/S 以外に設定することはできません。
- CSTM-A の場合、AUT（UCAMASK_MODE_AUT）のみ指定できます。
- CSTM-C の同一優先レベルのブロックモードでは、1 つの基本ブロックモードのみが成立できます。たとえば、MAN モードのときに AUT（UCAMASK_MODE_AUT）を設定すると MAN は不成立になります。つまり、MAN、AUT、CAS、PRD のどれか 1 つの基本ブロックモードのみが成立します。
- CSTM-C の RCAS と ROUT の優先レベルは同じでどちらか一方のみが成立します。たとえば RCAS モードのときに、ROUT（UCAMASK_MODE_ROUT）を設定すると RCAS は不成立になります。
- CSTM-C で IMAN または TRK が成立しているときに、基本ブロックモード MAN、AUT、CAS、PRD、RCAS、ROUT のどれかを設定しても、IMAN または TRK は成立したまま変化しません。たとえば、複合ブロックモード IMAN（AUT）のとき MAN を設定すると複合ブロックモード IMAN（MAN）になります（IMAN の状態は変換しません）。
- CSTM-C でリモートモード（RCAS または ROUT）が成立しているときに、基本ブロックモード MAN、AUT、CAS、PRD のどれかを設定すると、リモートモードは不成立になります。たとえば、複合ブロックモード RCAS（AUT）のとき MAN を設定すると RCAS は不成立になりブロックモードは MAN（のみ）になります（複合ブロックモード RCAS（MAN）にはなりません）。

2.1.2 UcaModeGet

UcaModeGetは、自ユーザカスタムブロックのブロックモードを32ビット整数として取得します。

■ 関数名

I32 UcaModeGet

■ 引数

● blockContext

データ型： UcaBlockContext
タイプ： IN
説明： ブロックコンテキスト

● modePtr

データ型： U32 *
タイプ： OUT
説明： ブロックモードポインタ

● option

データ型： U32
タイプ： IN
説明： オプション

■ 戻り値

・SUCCEED： ブロックモード取得に成功
・UCAERR_PARA_PTRNULL： 引数のポインタが NULL
・UCAERR_OPT_INVALID： 誤ったオプションを指定した

■ 処理内容説明

現在のアクティブモードを取得します。
基本ブロックモード IMAN が成立していない場合でも、初期化手動動作が必要な場合（IMAN 状態）には、基本ブロックモード IMAN が成立している場合と同じ動作をします。
同じ優先レベルのモードが同時に成立している場合は（通常ありえませんので）、引数 modePtr に 0 を格納します。関数はエラーリターンします。
オプションが正常でない場合は、引数 modePtr に 0 を格納し、関数はエラーリターンします。

■ 引数説明

● option

以下の中から 1 つだけ指定することができます。

- UCAOPT_MODEGET_DOMINANT：
  もっとも優先レベルの高い基本ブロックモードを取得
- UCAOPT_MODEGET_COMPLEX：複合ブロックモードを取得

● modePtr

オプションに UCAOPT_MODEGET_DOMINANT を指定した場合は、以下の中から 1 つのモードへのポインタを取得します。オプションに UCAOPT_MODEGET_COMPLET を指定した場合は、以下の中の 1 つ以上をビット OR したものへのポインタを取得します。

- UCAMASK_MODE_OS
- UCAMASK_MODE_IMAN
- UCAMASK_MODE_TRK
- UCAMASK_MODE_MAN
- UCAMASK_MODE_AUT
- UCAMASK_MODE_CAS
- UCAMASK_MODE_PRD
- UCAMASK_MODE_RCAS
- UCAMASK_MODE_ROUT

■ 詳細説明

本関数は、オプションの指定により、「優先レベルがもっとも高い基本ブロックモード 1 つ（UCAOPT_MODEGET_DOMINANT）」か、または、「成立している複合ブロックモードの各ビットを OR したもの（UCAOPT_MODEGET_COMPLEX）」のどちらかを取得することができます。

複合ブロックモードが IMAN（AUT（RCAS））の場合を例に説明します。複合ブロックモード IMAN（AUT（RCAS））のビット位置を以下に示します。

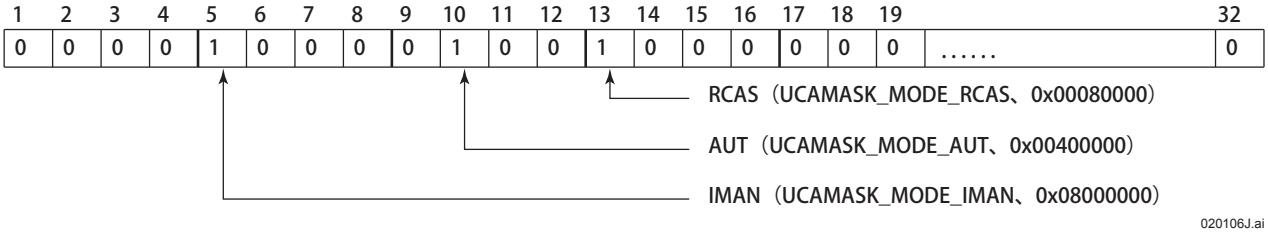


図　複合ブロックモードIMAN（AUT（RCAS））のビット位置

重要 基本ブロックモード IMAN、AUT と RCAS が成立している状態を IMAN（AUT（RCAS））と表記します。

UCAOPT_MODEGET_DOMINANTオプション

UCAOPT_MODEGET_DOMINANT オプションを指定すると優先レベルがもっとも高い基本ブロックモード 1 つを取得します。複合ブロックモードが、IMAN（AUT（RCAS））であれば、引数 modePtr に、UCAMASK_MODE_IMAN（0x08000000）を設定します。IMAN、AUT、RCAS では、IMAN の優先レベルがもっとも高いからです。ブロックモードの状態を取得して処理を分ける場合には、UCAOPT_MODEGET_DOMINANT オプションを指定して「優先レベルがもっとも高い基本ブロックモード 1 つ」を取得します。

UCAOPT_MODEGET_COMPLEXオプション

UCAOPT_MODEGET_COMPLEX オプションを指定すると複合ブロックモードを取得します。複合ブロックモードが、IMAN（AUT（RCAS））であれば、引数 modePtr に、UCAMASK_MODE_IMAN ｜ UCAMASK_MODE_AUT ｜ UCAMASK_MODE_RCAS（0x08480000）を格納します（「｜」は C 言語の OR 演算子です）。つまり、成立している基本ブロックモードをすべて OR した値を格納します。

CSTM-C ブロックの場合、UcaModeGet は IMAN 状態時にも基本ブロックモード IMAN が成立している場合と同じ動作をします。初期化手動処理が必要な以下の条件が成立した場合を IMAN 状態と呼びます。IMAN 状態時はブロックモードは IMAN ではありませんが、IMAN モード時と同等の動作を行います。

- ブロックモードが O/S から AUT、MAN などの通常のブロックモードに復帰するとき
- ブロックモードが IMAN モードから非 IMAN モードに復帰した最初の実効スキャン

2.1.3 UcaModeGetPrev

UcaModeGetPrevは、前回ユーザ定義関数実行時のブロックモードを取得します。

■ 関数名

I32 UcaModeGetPrev

■ 引数

● blockContext

- データ型： UcaBlockContext
- タイプ： IN
- 説明： ブロックコンテキスト

● lastModePtr

- データ型： U32 *
- タイプ： OUT
- 説明： 前回のブロックモードポインタ

● option

- データ型： U32
- タイプ： IN
- 説明： オプション

■ 戻り値

- SUCCEED： 前回のブロックモード取得に成功
- UCAERR_PARA_PTRNULL： 引数のポインタが NULL
- UCAERR_OPT_INVALID： 誤ったオプションを指定した

■ 処理内容説明

前回実行時のブロックモードを取得する。

■ 引数説明

● option

以下の中から 1 つだけ指定することができます。

- UCAOPT_MODEGET_DOMINANT：
  もっとも優先レベルの高い基本ブロックモードを取得
- UCAOPT_MODEGET_COMPLEX：複合ブロックモードを取得

● lastModePtr

option に UCAOPT_MODEGET_DOMINANT を指定した場合は、以下の中から 1 つのブロックモードへのポインタを取得します。option に UCAOPT_MODEGET_COMPLEX を指定した場合は、以下の中の 1 つ以上をビット OR したものへのポインタを取得します。

- UCAMASK_MODE_OS
- UCAMASK_MODE_IMAN
- UCAMASK_MODE_TRK
- UCAMASK_MODE_MAN
- UCAMASK_MODE_AUT
- UCAMASK_MODE_CAS
- UCAMASK_MODE_PRD
- UCAMASK_MODE_RCAS
- UCAMASK_MODE_ROUT

■ 詳細説明

UcaModeGetPrev は、前回ユーザ定義関数が実行を完了したときのブロックモードを引数 lastModePtr に格納します。前回実行の対象になるのは、機能ブロック定周期処理、または、機能ブロックワンショット起動処理です。たとえば、機能ブロック定周期処理を実効スキャン周期ごとに実行しているだけのユーザカスタムブロックであれば、直前の実効スキャン周期の実行を完了したときのブロックモードとなります。
UCAOPT_MODEGET_DOMINANT オプション（もっとも優先レベルの高い基本ブロックモードを取得）と UCAOPT_MODEGET_COMPLEX オプション（複合ブロックモードを取得）の使い分けについては、UcaModeGet と同じです。

2.2 ブロックステータス

ブロックステータスは、機能ブロックの運転状態を表します。本節では、ブロックステータスについて説明します。

■ ブロックステータス

ブロックステータスは、ユーザカスタムブロックのデータアイテム BSTS で参照することができます。
ユーザカスタムブロックのブロックステータスは、ユーザがユーザ定義ステータス文字列ビルダのブロックステータス定義で、USER8 に定義します。

参照 定義方法については、以下を参照してください。
APCS ユーザカスタムブロック プログラミングガイド（IM 33J15U21-01JA）「4.3.1 ブロックステータス」
APCS ユーザカスタムブロック プログラミングガイド（IM 33J15U21-01JA）「4.3.3 ユーザ定義インクルードファイル usrstatus.h」

ブロックステータスは 32 ビットの整数です。ブロックステータスの使い方には、以下の 2 種類があります。

- 各ブロックステータスが排他的に成立します。つまり、同時には 1 つのブロックステータスのみが成立します。内部的には、ブロックステータスに割り付いている各ビットのうちどれか 1 つのビットのみが 1 で、他のビットはすべて 0 です。
- 各ブロックステータスが独立しています。つまり、同時に複数のブロックステータスが成立します。内部的には、ブロックステータスに割り付いている各ビットのうち、ブロックステータスとして成立しているビットが 1 になり、不成立のビットが 0 になります。複数のブロックステータスが成立していれば、複数のビットが 1 になります。この場合、優先度の高いブロックステータスをビット番号の小さい方に割り付けてください。

両者を比較すると前者の排他的なブロックステータスの管理は、単純でプログラミングもやさしいです。後者のブロックステータスをビットごとに独立して管理する方法は、高度な管理でありプログラミングも複雑になります。通常、ユーザカスタムブロックでは、排他的な管理を行うことを推奨します。排他的な管理でも、機能ブロックの運転状態をデータアイテム BSTS としてチューニング画面や計器図に表示するような目的は、十分に満たすことができます。それぞれの管理方法に合わせた関数が用意されています。

表 ブロックステータス操作の関数

用途	関数名	機能
排他的管理	UcaBstsSetExclusive	指定されたブロックステータスに対応するビットをセット（1 に）し、それ以外のブロックステータスをクリア（0 に）します
ブロックステータスごとに独立した管理	UcaBstsSet	指定されたブロックステータスに対応するビットをセット（1 に）します。それ以外のブロックステータスは操作しません
	UcsBstsClear	指定されたブロックステータスに対応するビットをクリア（0 に）します。それ以外のブロックステータスは操作しません

参照 ブロックステータスを排他的に管理する場合のプログラム例については、以下を参照してください。
APCS ユーザカスタムブロック プログラミングガイド（IM 33J15U21-01JA）「4.3.4 ブロックステータスの操作」

2.2.1 UcaBstsSetExclusive

UcaBstsSetExclusiveは、自ブロックの指定されたブロックステータスをセットし、それ以外のブロックステータスをクリアします。

■ 関数名

I32 UcaBstsSetExclusive

■ 引数

● blockContext

- データ型： UcaBlockContext
- タイプ： IN
- 説明： ブロックコンテキスト

● bsts

- データ型： U32
- タイプ： IN
- 説明： ブロックステータス

■ 戻り値

- SUCCEED： ブロックステータス設定に成功
- UCAERR_PARA_PTRNULL： 引数のポインタが NULL

■ 処理内容説明

自ブロックのステータスの、指定したビットをセットします。指定された以外のビットはクリアします。

■ 詳細説明

UcaBstsSetExclusive は､引数 bsts に指定されたブロックステータスをセット（1 に）します。引数 bsts に指定された以外のブロックステータスはクリア（0 に）します。ユーザカスタムブロックで、ブロックステータスを排他的に使用する場合には、本関数を使用してください。

UcaBstsSetExclusive は、1 つのブロックステータスのみを成立させる場合に使用します。ただし、引数 bsts に複数のブロックステータス（つまり複数のビットが 1）を指定してもエラーにはならず、指定された複数のブロックステータスをセット（1 に）します。指定された以外のブロックステータスはクリア（0 に）します。

参照 UcaBstsSetExclusive の引数に指定するラベルはユーザが定義します。定義方法については、以下を参照してください。

APCS ユーザカスタムブロック プログラミングガイド（IM 33J15U21-01JA）「4.3.1 ブロックステータス」

APCS ユーザカスタムブロック プログラミングガイド（IM 33J15U21-01JA）「4.3.3 ユーザ定義インクルードファイル usrstatus.h」

2.2.2 UcaBstsSet

UcaBstsSetは、自ブロックの指定されたブロックステータスをセットします。それ以外のブロックステータスは操作しません。

■ 関数名

I32 UcaBstsSet

■ 引数

● blockContext

- データ型： UcaBlockContext
- タイプ： IN
- 説明： ブロックコンテキスト

● bsts

- データ型： U32
- タイプ： IN
- 説明： ブロックステータス

■ 戻り値

- SUCCEED： ブロックステータス設定に成功
- UCAERR_PARA_PTRNULL： 引数のポインタが NULL

■ 処理内容説明

自ブロックのステータスの、指定したビットをセットします。

■ 詳細説明

UcaBstsSet は、引数 bsts に指定されたブロックステータスをセット（1 に）します。指定された以外のブロックステータスは操作しません。引数 bsts に複数のブロックステータスを指定すると複数のブロックステータスをセット（1 に）します。

参照 UcaBstsSet の引数に指定するラベルはユーザが定義します。定義方法については、以下を参照してください。
APCS ユーザカスタムブロック プログラミングガイド（IM 33J15U21-01JA）「4.3.1 ブロックステータス」
APCS ユーザカスタムブロック プログラミングガイド（IM 33J15U21-01JA）「4.3.3 ユーザ定義インクルードファイル usrstatus.h」

2.2.3 UcaBstsClear

UcaBstsClearは、自ブロックの指定されたブロックステータスをクリアします。それ以外のブロックステータスは操作しません。

■ 関数名

I32 UcaBstsClear

■ 引数

● blockContext

- データ型： UcaBlockContext
- タイプ： IN
- 説明： ブロックコンテキスト

● bsts

- データ型： U32
- タイプ： IN
- 説明： ブロックステータス

■ 戻り値

- SUCCEED： ブロックステータスクリアに成功
- UCAERR_PARA_PTRNULL： 引数のポインタが NULL

■ 処理内容説明

自ブロックのブロックステータスの、指定したビットをクリアします。

■ 詳細説明

UcaBstsClear は、引数 bsts に指定されたブロックステータスをクリア（0 に）します。指定された以外のブロックステータスは操作しません。引数 bsts に複数のブロックステータスを指定すると複数のブロックステータスをクリア（0 に）します。

参照 UcaBstsClear が返すブロックステータスに対応するラベルはユーザが定義します。定義方法については、以下を参照してください。

APCS ユーザカスタムブロック プログラミングガイド（IM 33J15U21-01JA）「4.3.1 ブロックステータス」

APCS ユーザカスタムブロック プログラミングガイド（IM 33J15U21-01JA）「4.3.3 ユーザ定義インクルードファイル usrstatus.h」

2.2.4 UcaBstsGet

UcaBstsGetは、自ブロックのブロックステータスを32ビット整数として取得します。

■ 関数名

I32 UcaBstsGet

■ 引数

● blockContext

- データ型： UcaBlockContext
- タイプ： IN
- 説明： ブロックコンテキスト

● bsts

- データ型： U32 *
- タイプ： OUT
- 説明： ブロックステータス

■ 戻り値

- SUCCEED： ブロックステータス取得に成功
- UCAERR_PARA_PTRNULL： 引数のポインタが NULL

■ 処理内容説明

現在のブロックステータスを、32 ビット整数で取得します。

■ 詳細説明

UcaBstsGet は、自ブロックのブロックステータスを 32 ビット整数として引数 bsts に格納します。成立しているブロックステータスに対応するビットは 1、成立していないブロックステータスに対応するビットは 0 です。

2.3 アラームステータス

アラームステータスは、機能ブロックが検出したプロセスのアラーム状態を表す情報です。本節では、アラームステータスについて説明します。

■ アラームステータス

ユーザカスタムブロックのアラームステータスはデータアイテム ALRM で参照することができます。
ユーザカスタムブロックのアラームステータスは、システムにより決められているアラームステータスと、ユーザ定義のアラームステータスがあります。ユーザ定義のアラームステータスは、ユーザがユーザ定義ステータス文字列ビルダのアラームステータス定義で、USER16 に定義します。

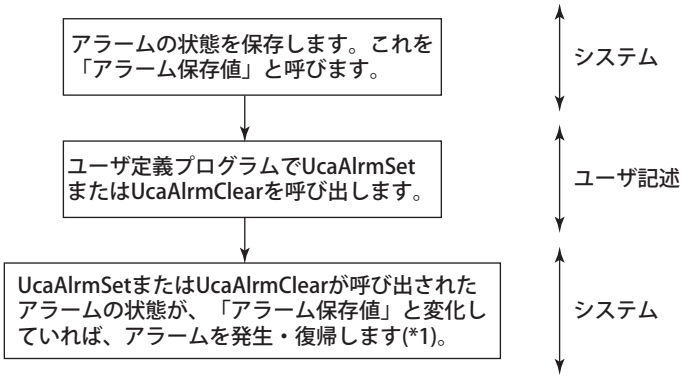
参照 ユーザカスタムブロックのアラームステータスとユーザ定義アラームステータスの定義方法については、以下を参照してください。

APCS ユーザカスタムブロック プログラミングガイド（IM 33J15U21-01JA）「4.3.2 アラームステータス」

APCS ユーザカスタムブロック プログラミングガイド（IM 33J15U21-01JA）「4.3.3 ユーザ定義インクルードファイル usrstatus.h」

ユーザカスタムブロックのアラーム発生の設定は UcaAlrmSet、アラーム復帰の設定は UcaAlrmClear で行います。UcaAlrmSet と UcaAlrmClear による設定は、各アラームステータスごとに、あとに設定したアラームステータスが優先されます。つまり、ユーザ定義関数（たとえば、機能ブロック定周期処理なら UcaBlockPeriodical）の先頭から関数の処理が終了するまでの間で、最後に呼び出された UcaAlrmSet または UcaAlrmClear による設定が有効となります。

実際にアラームの発生・復帰メッセージを出力するのは、ユーザカスタムブロック実行管理部です。ユーザ定義関数の処理から戻ったあと、ユーザカスタムブロック実行管理部は、最後に呼び出された UcaAlrmSet または UcaAlrmClear の設定に従いアラーム発生・復帰処理を行います。ユーザカスタムブロック実行管理部の動作を下図に示します。



*1： アラーム保存値が、「復帰」状態で、UcaAlrmSet により「発生」が設定されていれば、アラームを発生しプロセスアラームメッセージを出力します。アラーム保存値が、「発生」状態で、UcaAlrmClear により「復帰」が設定されていれば、アラームを復帰しプロセスアラーム復帰メッセージを出力します。
UcaAlrmSet が呼び出されていてもアラーム保存値が「発生」状態であれば、プロセスアラームメッセージは出力されません。同様に UcaAlrmClear が呼び出されていてもアラーム保存値が「復帰」状態であれば、プロセスアラーム復帰メッセージは出力されません。

図　ユーザカスタムブロック実行管理部のアラーム発生・復帰処理

重要 ユーザカスタムブロック実行管理部がアラーム処理を行うのは、機能ブロック定周期処理と機能ブロックワンショット起動処理の前後だけです。

参照 アラーム処理のプログラミング説明とプログラム例については、以下を参照してください。
APCS ユーザカスタムブロック プログラミングガイド（IM 33J15U21-01JA）「5.2.2 演算異常 ERRC アラームの発生と復帰」
APCS ユーザカスタムブロック プログラミングガイド（IM 33J15U21-01JA）「6.1.2 ユーザ定義アラームの発生と復帰」

2.3.1 UcaAlrmSet

UcaAlrmSetは、自ブロックにアラーム発生を設定します。

■ 関数名

I32 UcaAlrmSet

■ 引数

● blockContext

- データ型： UcaBlockContext
- タイプ： IN
- 説明： ブロックコンテキスト

● alrm

- データ型： U32
- タイプ： IN
- 説明： アラームステータス

■ 戻り値

- SUCCEED： アラームステータス設定に成功
- UCAERR_ALRM_INVALID： 正しくないアラームステータスを指定した
- UCAERR_PARA_PTRNULL： 引数のポインタが NULL

■ 処理内容説明

自ブロックのアラームステータスの、指定したビットをセットします。

■ 詳細説明

UcaAlrmSet は、引数 alrm に指定されたアラームステータスの発生を設定します。複数のアラームステータス（つまり引数 alrm の複数のビットが 1）を指定することもできます。UcaAlrmSet によるアラーム発生の設定と UcaAlrmClear によるアラーム復帰の設定は、各アラームステータス（のビット）ごとに後優先となります。実際にアラームが発生・復帰するのは、UcaAlrmSet（または UcaAlrmClear）が呼び出されたタイミングではなく、ユーザ定義関数の処理が完了したあとで、アラームの発生・復帰はユーザカスタムブロック実行管理部（システム）が行います。

引数 alrm の 1 ～ 9 ビットが 1 の場合はエラーとなり、戻り値に UCAERR_ALRM_INVALID が返ります。アラームステータスの 1 ～ 8 ビットはシステム予約ですので、UcaAlrmSet でアラームの発生を設定することはできません。また、NR（9 ビット）を指定することはできません。アラームを NR にするには、UcaAlrmClear で発生しているすべてのアラームをクリアしてください。

図　アラームのビット位置

参照 UcaSetAlrm の引数に指定するラベルについては、以下を参照してください。

APCS ユーザカスタムブロック プログラミングガイド（IM 33J15U21-01JA）「4.3.3 ユーザ定義インクルードファイル usrstatus.h」

また、デフォルトのシステムアラームのラベルは、システム定義インクルードファイル libucadef.h に定義されています。デフォルトのアラームステータスのラベル一覧については、以下を参照してください。

「Appendix C.4 アラームステータス」

2.3.2 UcaAlrmClear

UcaAlrmClearは、自ブロックにアラーム復帰を設定します。

■ 関数名

I32 UcaAlrmClear

■ 引数

● blockContext

- データ型： UcaBlockContext
- タイプ： IN
- 説明： ブロックコンテキスト

● alrm

- データ型： U32
- タイプ： IN
- 説明： アラームステータス

■ 戻り値

- SUCCEED： アラームステータスクリアに成功
- UCAERR_ALRM_INVALID： 正しくないアラームステータスを指定した
- UCAERR_PARA_PTRNULL： 引数のポインタが NULL

■ 処理内容説明

自ブロックのアラームステータスの、指定したビットをクリアします。

■ 詳細説明

UcaAlrmClear は、引数 alrm に指定されたアラームステータスの復帰を設定します。複数のアラームステータス（つまり引数 alrm の複数のビットが 1）を指定することもできます。UcaAlrmSet によるアラーム発生の設定と UcaAlrmClear によるアラーム復帰の設定は、各アラームステータス（のビット）ごとに後優先となります。実際にアラームが発生・復帰するのは、UcaAlrmClear（または UcaAlrmSet）が呼び出されたタイミングではなく、ユーザ定義関数の処理が完了したあとで、アラームの発生・復帰はユーザカスタムブロック実行管理部（システム）が行います。

引数 alrm の 1 ～ 9 ビットが 1 の場合はエラーとなり、戻り値に UCAERR_ALRM_INVALID が返ります。アラームステータスの 1 ～ 8 ビットはシステム予約ですので、UcaAlrmClear でアラームの復帰を設定することはできません。また、NR（9 ビット）を指定することはできません。アラームを NR にするには、UcaAlrmClear で発生しているすべてのアラームをクリアしてください。

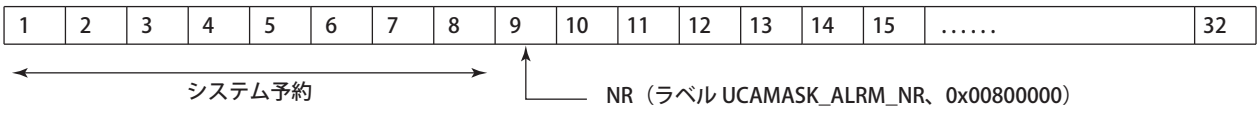


図　アラームのビット位置

参照 UcaAlrmClear の引数に指定するラベルについては、以下を参照してください。
APCS ユーザカスタムブロック プログラミングガイド（IM 33J15U21-01JA）「4.3.3 ユーザ定義インクルードファイル usrstatus.h」

また、デフォルトのシステムアラームのラベルは、システム定義インクルードファイル libucadef.h に定義されています。デフォルトのアラームステータスのラベル一覧については、以下を参照してください。
「Appendix C.4 アラームステータス」

2.3.3 UcaAlrmGet

UcaAlrmGetは、自ブロックのアラーム状態を32ビット整数として取得します。

■ 関数名

I32 UcaAlrmGet

■ 引数

● blockContext

- データ型： UcaBlockContext
- タイプ： IN
- 説明： ブロックコンテキスト

● alrm

- データ型： U32 *
- タイプ： OUT
- 説明： アラームステータス

● option

- データ型： U32
- タイプ： IN
- 説明： オプション

■ 戻り値

- SUCCEED： アラームステータスの取得に成功
- UCAERR_PARA_PTRNULL： 引数のポインタが NULL
- UCAERR_OPT_INVALID： 誤ったオプションを指定した

■ 処理内容説明

現在のアラームステータスを、32 ビット整数で取得します。

■ 引数説明

● option

以下の中から 1 つだけ指定することができます。

- UCAOPT_ALRM_LAST：
  前回までのアラームステータスを取得します。
- UCAOPT_ALRM_THISTIME：
  今回の発生または復帰の設定結果を取得します。
- UCAOPT_ALRM_CURRENT：
  今回発生または復帰が設定されたビットはその設定結果を、そうでないビットは前回までの状態を取得します。

■ 詳細説明

UcaAlrmGet は、自ブロックのアラーム状態を 32 ビット整数として取得します。下図は、ユーザカスタムブロック実行管理部のアラーム発生と復帰に関する処理です。

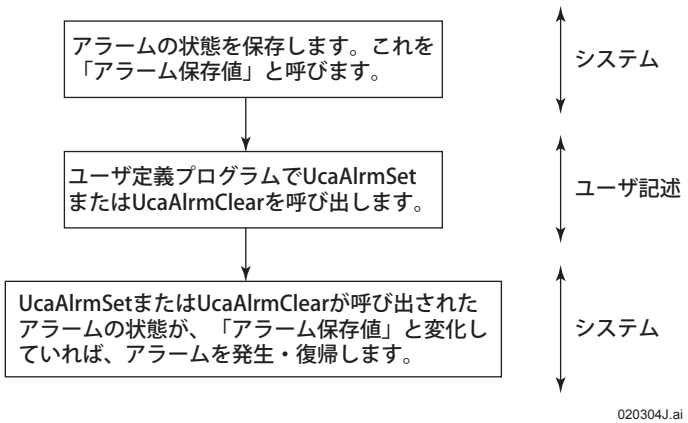


図　ユーザカスタムブロック実行管理部のアラーム発生・復帰処理

UcaAlrmGet は、オプションの指定により、以下の 3 種類のどれか 1 つを引数 alrm に格納します。

● UCAOPT_ALRM_LASTオプション：前回までのアラームステータス

ユーザカスタムブロック実行管理部は、ユーザ定義関数（UcaBlockPeriodical または UcaBlockOneshot）を呼び出す前に、アラームの状態を「アラーム保存値」として保存します。UCAOPT_ALRM_LAST オプションを指定すると、この「アラーム保存値」を引数 alrm に格納します。

● UCAOPT_ALRM_THISTIMEオプション：今回の発生または復帰の設定結果

ユーザ定義関数（UcaBlockPeriodical または UcaBlockOneshot）の入り口から UcaAlrmGet が呼び出されるまでの間に、UcaAlrmSet と UcaAlrmClear によるアラームの発生または復帰の設定結果を引数 alrm に格納します。各アラームごとに UcaAlrmSet と UcaAlrmClear は、後優先になります。

- 最後に UcaAlrmSet により発生が設定されているアラームに対応するビットは 1 です。
- 最後に UcaAlrmClear により復帰が設定されているアラームに対応するビットは 0 です。
- UcaAlrmSet または UcaAlrmClear で操作されていないアラームに対応するビットは 0 です。
- UcaAlrmGet 呼び出し時に現在のアラームステータスにおいて、発生が設定されているビットが 1 つもなければ、NR（0x00800000）を引数 alrm に格納します。

● UCAOPT_ALRM_CURRENTオプション：現在のアラームの状態

UcaAlrmGet が呼び出された時点でのアラームの状態を引数 alrm に格納します。ユーザ定義関数（UcaBlockPeriodical または UcaBlockOneshot）の入り口から UcaAlrmGet が呼び出されるまでの間に、UcaAlrmSet と UcaAlrmClear によりアラームの発生または復帰が設定されているアラーム（のビット）は、UcaAlrmSet と UcaAlrmClear による設定結果となります。UcaAlrmSet または UcaAlrmClear で操作されていないアラームの各ビットは、「アラーム保存値」の値となります。

参照 UcagetAlrm が引数 alrm に格納するアラームステータスに対応したラベルについては、以下を参照してください。

APCS ユーザカスタムブロック プログラミングガイド（IM 33J15U21-01JA）「4.3.3 ユーザ定義インクルードファイル usrstatus.h」

また、デフォルトのシステムアラームのラベルは、システム定義インクルードファイル libucadef.h に定義されています。デフォルトのアラームステータスのラベル一覧については、以下を参照してください。

「Appendix C.4 アラームステータス」

2.4 データステータス

データステータスは、データの品質を表す情報です。本節では、データステータスについて説明します。

■ データステータス

データステータスは、入出力動作によって機能ブロックから機能ブロックへと伝達されます。プロセス入出力の異常や演算の異常などによって、さまざまな例外事象が発生した時の制御動作の判定に利用されます。
データステータスは、内部的には32ビットの整数です。1ビットごとに1つのデータステータスが対応しています（下図）。データステータスが0（全ビット0）であれば、データが正常であることを示します。あるデータステータスが成立すると、対応するビットが1になります。

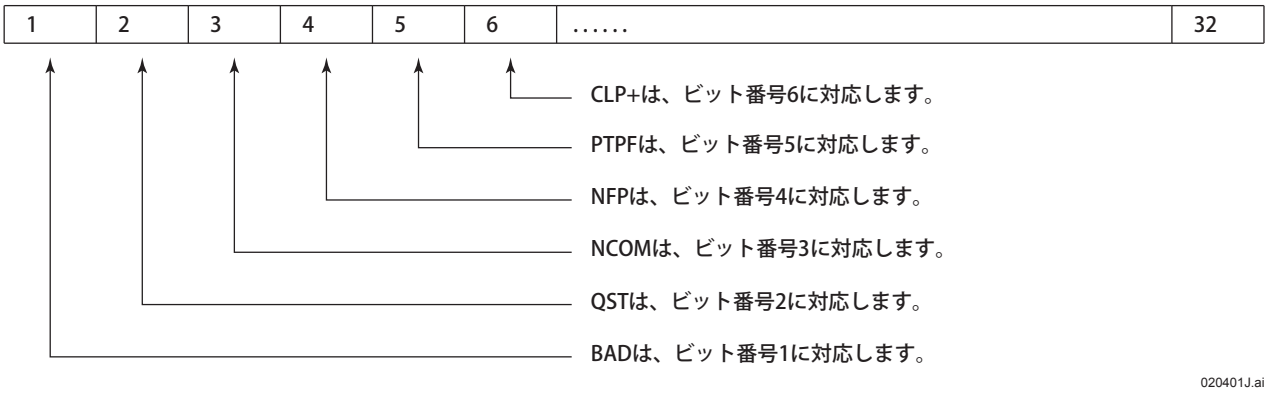


図　ビット番号とデータステータスの対応

ビットごとのマスクパターンがシステム定義インクルードファイル libucadef.h に用意されています。

```
/* データステータス */
#define UCAMASK_DSTS_BAD        0x80000000          /* データ値不良 */
#define UCAMASK_DSTS_QST        0x40000000          /* データ値疑問 */
#define UCAMASK_DSTS_NCOM               0x20000000          /* 通信不良 */
#define UCAMASK_DSTS_NFP        0x10000000          /* 非プロセス起源 */
#define UCAMASK_DSTS_PTPF               0x08000000          /* 出力フェイル */
#define UCAMASK_DSTS_CLPP               0x04000000          /* 上限クランプ */
#define UCAMASK_DSTS_CLPM               0x02000000          /* 下限クランプ */
......
```

データステータスとビット番号の対応を以下に示します。

表　データステータスとビット番号

ビット番号	シンボル	名称	libucadef.hのラベル	マスクパターン
1	BAD	データ値不良（BAD value）	UCAMASK_DSTS_BAD	0x80000000
2	QST	データ値疑問（QueSTionable value）	UCAMASK_DSTS_QST	0x40000000
3	NCOM	通信不良（No COMmunication）	UCAMASK_DSTS_NCOM	0x20000000
4	NFP	非プロセス起源（Not From Process）	UCAMASK_DSTS_NFP	0x10000000
5	PTPF	出力フェイル（Path To Process Failed）	UCAMASK_DSTS_PTPF	0x08000000
6	CLP+	上限クランプ（CLamP high）	UCAMASK_DSTS_CLPP	0x04000000
7	CLP-	下限クランプ（CLamP low）	UCAMASK_DSTS_CLPM	0x02000000
8	CND	コンディショナル（ConNDitional）	UCAMASK_DSTS_CND	0x01000000
9	CAL	キャリブレーション（CALibration）	UCAMASK_DSTS_CAL	0x00800000
10	NEFV	無効データ（Not EFfectiVe）	UCAMASK_DSTS_NEFV	0x00400000
11	O/S	サービスオフ（Out of Service）	UCAMASK_DSTS_OS	0x00200000
12	MNT	メンテナンス（MaiNTenance）	UCAMASK_DSTS_MNT	0x00100000
17	IOP+	上限入力オープン（Input OPen high）	UCAMASK_DSTS_IOPP	0x00008000
18	IOP-	下限入力オープン（Input OPen low）	UCAMASK_DSTS_IOPM	0x00004000
19	OOP	出力オープン（Output OPen）	UCAMASK_DSTS_OOP	0x00002000
20	NRDY	PI/O ノットレディ（PI/O Not ReaDY）	UCAMASK_DSTS_NRDY	0x00001000
21	PFAL	PI/O 無応答（PI/O Power FAiLure）	UCAMASK_DSTS_PFAL	0x00000800
22	LPFL	PI/O 長時間無応答 （PI/O Long Power FaiLure）	UCAMASK_DSTS_LPFL	0x00000400
25	MINT	マスタ初期化（Master INiTialize）	UCAMASK_DSTS_MINT	0x00000080
26	SINT	スレーブ初期化（Slave INiTialize）	UCAMASK_DSTS_SINT	0x00000040
27	SVPB	SV プッシュバック（SV PushBack）	UCAMASK_DSTS_SVPB	0x00000020

補足 データステータス BAD（データ値異常）と QST（データ値疑問）は、プロセスのデータを使用するとき、もっともよく利用されるデータステータスです。

参照 データステータスの意味については、以下を参照してください。
機能ブロック共通機能リファレンス（IM 33J15A20-01JA）「6.4 データステータス」

ユーザカスタムアルゴリズムのプログラミングでは、F64S、I32S などのデータ値とデータステータスを 1 つの変数として扱うデータ型を使用します。ユーザカスタムアルゴリズム作成用ライブラリには、データステータスを検査、設定、消去する関数が用意されています。

- データステータスの検査とは、対応するビットが 1 か 0 かを判定します。
- データステータスの設定とは、対応するビットを 1 にします。
- データステータスの消去とは、対応するビットを 0 にします。

データステータスを操作する関数の使い方を下図に示します（単に関数の呼び出し方を説明するだけですので、処理内容に意味はありません）。

```
UCAUSER_API I32 UCAAPI UcaBlockPeriodical(
UcaBlockContext bc              /* (IN/OUT) : ブロックコンテキスト */
)
{
   F64S p01;
   F64S p02;
   I32 rtnCode;

   rtnCode = UcaDataGetPn(bc, &p01, 1, 1, NOOPTION);
   rtnCode = UcaDataGetPn(bc, &p02, 2, 1, NOOPTION);

   /* P01 が BAD なら P02 も BAD にする */
   if (UcaDstsIsBad(bc, p01.status)) {
      rtnCode = UcaDstsSetBad(bc, &p02.status);
   }

   /* P01 が QST でなければ、P02 の QST を消去する */
   if (!UcaDstsIsQst(bc, p01.status)) {
      rtnCode = UcaDstsClearQst(bc, &p02.status);
   }
   ......

   return SUCCEED
}
```

■ データステータス成立の検査と設定

データステータス成立の検査と設定例は、以下の部分です。

```
/* P01 が BAD なら P02 も BAD にする */
if (UcaDstsIsBad(bc, p01.status)) {
   rtnCode = UcaDstsSetBad(bc, &p02.status);
}
```

UcaDstsIsBad は、p01. status（データアイテム P01 のデータステータス）が BAD か否かを検査しています。UcaDstsIsBad が検査するのは、BAD に対応する 1 ビットだけです。他のビットの状態は、UcaDstsIsBad の結果には影響しません。UcaDstsIsBad は、BAD が成立していれば（ビットが 1）TRUE を返し、BAD が成立していなければ（ビットが 0）FALSE を返します。
P01（変数 p01. status）が BAD であれば、UcaDstsSetBad が呼び出されます。UcaDstsSetBad は、引数に指定された P02 のデータステータス（&p02. status）の BAD のビットを 1 にします（BAD と QST は同時には成立しない規則なので、QST が成立していれば QST のビットを 1 から 0 にします）。

■ データステータス不成立の検査と消去

データステータス不成立の検査と設定例は、以下の部分です。

```
/* P01 が QST でなければ、P02 の QST を消去する */
if (!UcaDstsIsQst(bc, p01.status)) {
   rtnCode = UcaDstsClearQst(bc, &p02.status);
}
```

if (!UcaDstsIsQst (pc, p01. status)) の「!」は、否定演算子（真偽を反転します）です。「!」を「Not」と置き換えて「if Not QST」と読んでください。つまり、この if 文は、「P01 のデータステータス（p01. status）の QST が成立していなければ」という意味です。
P01（変数 p01. status）が QST でなければ、UcaDstsClearQst が呼び出されます。UcaDstsClearQst は、引数に指定された P02 のデータステータス（&p02. status）の QST のビットのみを 0 にします。

2.4.1 UcaDstsIsGood

UcaDstsIsGoodは、データステータスが正常か否かを判定します。

■ 関数名

BOOL UcaDstsIsGood

■ 引数

● blockContext

- データ型： UcaBlockContext
- タイプ： IN
- 説明： ブロックコンテキスト

● dsts

- データ型： U32
- タイプ： IN
- 説明： データステータス

■ 戻り値

- TRUE： データステータスが正常（データステータスが BAD でも QST でもない）
- FALSE： データステータスが BAD または QST

■ 詳細説明

UcaDstsIsGood は、引数に指定されたデータステータス dsts の BAD または QST が成立していると、FALSE を返します。それ以外では、TRUE を返します。

表 UcaDstsIsGoodの戻り値（BADとQST以外のデータステータスは、戻り値と無関係です）

BADのビット	QSTのビット	UcaDstsIsGoodの戻り値
1（成立）	1（成立）	FALSE (*1)
1（成立）	0（不成立）	FALSE
0（不成立）	1（成立）	FALSE
0（不成立）	0（不成立）	TRUE

*1： データステータス BAD と QST が同時に成立しない規則ですが、UcaDstsIsGood は FALSE を返します。

2.4.2 UcaDstsIsBad

UcaDstsIsBadは、データステータスBADが成立しているか否かを判定します。

■ 関数名

BOOL UcaDstsIsBad

■ 引数

● blockContext

- データ型： UcaBlockContext
- タイプ： IN
- 説明： ブロックコンテキスト

● dsts

- データ型： U32
- タイプ： IN
- 説明： データステータス

■ 戻り値

- TRUE： データステータスが BAD
- FALSE： データステータスが BAD でない

■ 詳細説明

UcaDstsIsBad は、引数 dsts に指定されたデータステータスの BAD に対応するビットが 1 なら TRUE、0 なら FALSE を返します。BAD 以外のデータステータス（ビット）は、UcaDstsIsBad の処理結果に影響しません。

2.4.3 UcaDstsIsQst

UcaDstsIsQstは、データステータスQSTが成立しているか否かを判定します。

■ 関数名

BOOL UcaDstsIsQst

■ 引数

● blockContext

- データ型： UcaBlockContext
- タイプ： IN
- 説明： ブロックコンテキスト

● dsts

- データ型： U32
- タイプ： IN
- 説明： データステータス

■ 戻り値

- TRUE： データステータスが QST
- FALSE： データステータスが QST でない

■ 詳細説明

UcaDstsIsQst は、引数 dsts に指定されたデータステータスの QST に対応するビットが 1 なら TRUE、0 なら FALSE を返します。QST 以外のデータステータス（ビット）は、UcaDstsIsQst の処理結果に影響しません。

2.4.4 UcaDstsIsPtpf

UcaDstsIsPtpfは、データステータスPTPFが成立しているか否かを判定します。

■ 関数名

BOOL UcaDstsIsPtpf

■ 引数

● blockContext

- データ型： UcaBlockContext
- タイプ： IN
- 説明： ブロックコンテキスト

● dsts

- データ型： U32
- タイプ： IN
- 説明： データステータス

■ 戻り値

- TRUE： データステータスが PTPF
- FALSE： データステータスが PTPF でない

■ 詳細説明

UcaDstsIsPtpf は、引数 dsts に指定されたデータステータスの PTPF に対応するビットが 1 なら TRUE、0 なら FALSE を返します。PTPF 以外のデータステータス（ビット）は、UcaDstsIsPtpf の処理結果に影響しません。

2.4.5 UcaDstsIsCnd

UcaDstsIsCndは、データステータスCNDが成立しているか否かを判定します。

■ 関数名

BOOL UcaDstsIsCnd

■ 引数

● blockContext

- データ型： UcaBlockContext
- タイプ： IN
- 説明： ブロックコンテキスト

● dsts

- データ型： U32
- タイプ： IN
- 説明： データステータス

■ 戻り値

- TRUE： データステータスがコンディショナル
- FALSE： データステータスがコンディショナルでない

■ 詳細説明

UcaDstsIsCnd は、引数 dsts に指定されたデータステータスの CND に対応するビットが 1 なら TRUE、0 なら FALSE を返します。CND 以外のデータステータス（ビット）は、UcaDstsIsCnd の処理結果に影響しません。

2.4.6 UcaDstsIsCal

UcaDstsIsCalは、データステータスCALが成立しているか否かを判定します。

■ 関数名

BOOL UcaDstsIsCal

■ 引数

● blockContext

- データ型： UcaBlockContext
- タイプ： IN
- 説明： ブロックコンテキスト

● dsts

- データ型： U32
- タイプ： IN
- 説明： データステータス

■ 戻り値

- TRUE： データステータスが CAL
- FALSE： データステータスが CAL でない

■ 詳細説明

UcaDstsIsCal は、引数 dsts に指定されたデータステータスの CAL に対応するビットが 1 なら TRUE、0 なら FALSE を返します。CAL 以外のデータステータス（ビット）は、UcaDstsIsCal の処理結果に影響しません。

2.4.7 UcaDstsSetBad

UcaDstsSetBadは、データステータスにBADを設定します。

■ 関数名

I32 UcaDstsSetBad

■ 引数

● blockContext

- データ型： UcaBlockContext
- タイプ： IN
- 説明： ブロックコンテキスト

● dsts

- データ型： U32 *
- タイプ： OUT
- 説明： データステータスポインタ

■ 戻り値

- SUCCEED： 正常終了
- UCAERR_PARA_PTRNULL： 引数のポインタが NULL

■ 詳細説明

UcaDstsSetBad は、引数 dsts が指すデータステータスの BAD のビットに 1 を設定します。もとのデータステータスの QST が成立していれば、QST のビットは 1 から 0 にします。これは、データステータス BAD と QST は同時に成立しない規則になっているためです。BAD と QST 以外のデータステータスのビットは操作しません。

2.4.8 UcaDstsClearBad

UcaDstsClearBadは、データステータスQSTを消去します。

■ 関数名

I32 UcaDstsClearBad

■ 引数

● blockContext

- データ型： UcaBlockContext
- タイプ： IN
- 説明： ブロックコンテキスト

● dsts

- データ型： U32 *
- タイプ： OUT
- 説明： データステータスポインタ

■ 戻り値

- SUCCEED： 正常終了
- UCAERR_PARA_PTRNULL： 引数のポインタが NULL

■ 詳細説明

UcaDstsClearBad は、引数 dsts が指すデータステータスの BAD を消去します。つまり、データステータスの BAD のビットに 0 を設定します。他のデータステータスのビットは操作しません。

2.4.9 UcaDstsSetQst

UcaDstsSetQstは、データステータスにQSTを設定します。

■ 関数名

I32 UcaDstsSetQst

■ 引数

● blockContext

- データ型： UcaBlockContext
- タイプ： IN
- 説明： ブロックコンテキスト

● dsts

- データ型： U32 *
- タイプ： OUT
- 説明： データステータスポインタ

■ 戻り値

- SUCCEED： 正常終了
- UCAERR_PARA_PTRNULL： 引数のポインタが NULL

■ 詳細説明

UcaDstsSetQst は、引数 dsts が指すデータステータスの QST のビットに 1 を設定します。
もとのデータステータスの BAD が成立していれば、BAD のビットは 1 から 0 にします。
これは、データステータス BAD と QST は同時に成立しない規則になっているためです。
BAD と QST 以外のデータステータスのビットは操作しません。

2.4.10 UcaDstsClearQst

UcaDstsClearQstは、データステータスQSTを消去します。

■ 関数名

I32 UcaDstsClearQst

■ 引数

● blockContext

- データ型： UcaBlockContext
- タイプ： IN
- 説明： ブロックコンテキスト

● dsts

- データ型： U32 *
- タイプ： OUT
- 説明： データステータスポインタ

■ 戻り値

- SUCCEED： 正常終了
- UCAERR_PARA_PTRNULL： 引数のポインタが NULL

■ 詳細説明

UcaDstsClearQst は、引数 dsts が指すデータステータスの QST を消去します。つまり、データステータスの QST のビットに 0 を設定します。他のデータステータスのビットは操作しません。

2.4.11 UcaDstsSetCnd

UcaDstsSetCndは、データステータスにCNDを設定します。

■ 関数名

I32 UcaDstsSetCnd

■ 引数

● blockContext

- データ型： UcaBlockContext
- タイプ： IN
- 説明： ブロックコンテキスト

● dsts

- データ型： U32 *
- タイプ： OUT
- 説明： データステータスポインタ

■ 戻り値

- SUCCEED： 正常終了
- UCAERR_PARA_PTRNULL： 引数のポインタが NULL

■ 詳細説明

UcaDstsSetCnd は、引数 dsts が指すデータステータスに CND を設定します。つまり、データステータスの CND のビットに 1 を設定します。他のデータステータスのビットは操作しません。

2.4.12 UcaDstsClearCnd

UcaDstsClearCndは、データステータスCNDを消去します。

■ 関数名

I32 UcaDstsClearCnd

■ 引数

● blockContext

- データ型： UcaBlockContext
- タイプ： IN
- 説明： ブロックコンテキスト

● dsts

- データ型： U32 *
- タイプ： OUT
- 説明： データステータスポインタ

■ 戻り値

- SUCCEED： 正常終了
- UCAERR_PARA_PTRNULL： 引数のポインタが NULL

■ 詳細説明

UcaDstsClearCnd は、引数 dsts が指すデータステータスの CND を消去します。つまり、データステータスの CND のビットに 0 を設定します。他のデータステータスのビットは操作しません。

2.4.13 UcaDstsChooseBadOrQst

UcaDstsChooseBadOrQstは、指定されたデータステータスより、BADまたはQSTを抽出します。

■ 関数名

I32 UcaDstsChooseBadOrQst

■ 引数

● bc

- データ型： UcaBlockContext
- タイプ： IN
- 説明： ブロックコンテキスト

● status

- データ型： U32 *
- タイプ： IN/OUT
- 説明： データステータスを指すポインタ

■ 戻り値

- 0（データステータス正常）：
  引数 status の指すデータステータスは BAD でも QST でもない
- UCAMASK_DSTS_BAD：引数 status の指すデータステータスは BAD である
- UCAMASK_DSTS_QST：
  引数 status の指すデータステータスは BAD ではないが QST である

■ 詳細説明

引数 status が指すデータステータスを検査し、BAD または QST だけのデータステータスを作り、status が指す領域に設定します。
動作は以下のとおりです。

- BAD なら、BAD（他のデータステータスのビットは 0）を設定します。
- BAD でなく QST なら、QST（他のデータステータスのビットは 0）を設定します。
- BAD でなく QST でもなければ、正常（32 ビットすべて 0）を設定します。

表　UcaDstsChooseBadOrQstの結果（BADとQST以外のデータステータスは、結果と無関係です）

BADのビット	QSTのビット	UcaDstsChooseBadOrQstが引数statusに格納する結果（UcaDstsChooseBadOrQstの戻り値も結果と同じ値です）
1（成立）	1（成立）	BAD（libucadef.h のマスクパターン、UCAMASK_DSTS_BAD）(*1)
1（成立）	0（不成立）	BAD（libucadef.h のマスクパターン、UCAMASK_DSTS_BAD）
0（不成立）	1（成立）	QST（libucadef.h のマスクパターン、UCAMASK_DSTS_QST）
0（不成立）	0（不成立）	0（データステータス正常）

*1： データステータス BAD と QST が同時に成立しない規則ですが、UcaDstsChooseBadOrQst は UCAMASK_DSTS_BAD を返します。

3. 入出力結合

入出力結合関数は、端子からデータを読み書きする処理を行います。本章では、入出力結合に関する説明をします。

■ 入出力結合関数

入出力結合関数には、高水準入出力関数と低水準入出力関数の 2 種類があります。高水準入出力関数は IN 端子と OUT 端子からデータの読み書きを行い、データステータス処理や信号変換処理などを行います。高水準入出力関数には、SUB 端子、TSI 端子、TIN 端子、BIN 端子、RL1 端子、RL2 端子、INT 端子のための関数も用意されています。低水準入出力関数は、Q01 ～ Q32 端子と J01 ～ J16 端子からデータの読み書きを行います。高水準入出力関数と低水準入出力関数の機能の相違点を以下に示します。

表　入力端子読み込み処理における相違点

関数	UcaRWReadIn	UcaRWRead
端子	IN 端子	Q01 ～ Q32 端子
低水準／高水準入出力関数	高水準入出力関数	低水準入出力関数
入力信号変換	する	しない
出力オープン警報（IOP）	常時検出	オプション指定時のみ検出 UCAOPT_RW_SETIOP
結合状態不良警報（CNF）	常時検出	常時検出

表　出力端子書き込み処理における相違点

関数	UcaRWWriteMv	UcaRWWriteCpvToOutSub（UcaRWWriteCpv）(*1)	UcaRWWrite
端子	OUT 端子	OUT 端子	J01 ～ J16 端子
低水準／高水準入出力関数	高水準入出力関数	高水準入出力関数	低水準入出力関数
出力信号変換	する	する	しない
出力リミッタ	する	しない	しない
出力変化率リミッタ	する	しない	しない
出力クランプ	する	しない	しない
プリセット操作出力	する	しない	しない
プリセット操作出力時制御初期化設定	オプション指定時にしない UCAOPT_RW_NOSETINIT	しない	しない
出力値トラッキング	する	する	しない
RMVイコライズ	オプション指定時にしない UCAOPT_RW_NOEQUALIZERMV	しない	しない
出力オープン警報（OOP）	検出する	オプション指定時のみ検出する UCAOPT_RW_SETOOP	オプション指定時のみ検出する UCAOPT_RW_SETOOP
結合状態不良警報（CNF）	検出する	オプション指定時のみ検出する UCAOPT_RW_SETOOP UCAOPT_RW_SETCNF	オプション指定時のみ検出する UCAOPT_RW_SETOOP UCAOPT_RW_SETCNF
出力上下限警報（MHI/MLO）	検出する	検出しない	オプション指定時に検出しない UCAOPT_RW_NOMHML

*1： 通常は、UcaRWWriteCpvToOutSub を使用します。UcaRWWriteCpvToOutSub は、OUT 端子と SUB 端子への出力をひとまとめにした関数で、処理内で UcaRWWriteCpv を呼び出しています。

表　出力端子出力読み返し処理における相違点

関数	UcaRWReadbackMv	UcaRWWriteCpvToOutSub（UcaRWReadbackCpv）(*1)	UcaRWReadback
端子	OUT 端子	OUT 端子	J01 ～ J16 端子
低水準／高水準入出力関数	高水準入出力関数	高水準入出力関数	低水準入出力関数
出力レンジトラッキング	する	しない	しない
出力オープン警報（OOP）	常時検出	オプション指定時のみ検出 UCAOPT_RW_SETOOP	オプション指定時のみ検出 UCAOPT_RW_SETOOP
結合状態不良警報（CNF）	常時検出	常時検出	オプション指定時のみ検出 UCAOPT_RW_SETOOP UCAOPT_RW_SETCNF

*1： 通常は、UcaRWWriteCpvToOutSub を使用します。UcaRWWriteCpvToOutSub は、OUT 端子と SUB 端子への出力をひとまとめにした関数で、処理内で UcaRWReadbackCpv を呼び出しています。

補足 ユーザカスタムブロックのビルダ定義項目の各アラームに対して［警報なし］を指定した場合、入出力結合関数の引数に指定するオプションに［アラームを検出］を指定しても、入出力結合関数は、対応するアラームを検出しません。つまり、関数のオプション指定より、ビルダ定義項目が優先されます。

3.1 低水準入出力

低水準入出力関数は、Q01～Q32端子やJ01～J16端子からデータの読み書きを行います。

3.1.1 UcaRWRead

UcaRWReadは、Q01～Q32端子からデータを読み込み、データアイテムRV01～RV32に格納します。

■ 関数名

I32 UcaRWRead

■ 引数

● blockContext

- データ型： UcaBlockContext
- タイプ： IN
- 説明： ブロックコンテキスト

● i

- データ型： I32
- タイプ： IN
- 説明： 入力端子番号（0 ～ 32）

● option

- データ型： U32
- タイプ： IN
- 説明： オプション

■ 戻り値

- SUCCEED： 入力成功
- UCAERR_RW_NOCONNECT： 結合未定義
- UCAERR_RW_LOOPCONN： 端子間入力結合
- UCAERR_RW_CONNFAIL： データアクセスエラー
- UCAERR_RW_MNT： 結合先がオンラインメンテナンス中
- UCAERR_RW_OS： 結合先機能ブロックのブロックモードが O/S
- UCAERR_RW_SWOPEN： 切換スイッチがオープン
- UCAERR_RW_IOPHI： 入力オープン（上限）
- UCAERR_RW_IOPLO： 入力オープン（下限）または PI/O 機器異常
- UCAERR_RW_BADVALUE： 結合先データのデータステータスが BAD
- UCAERR_RW_INVALIDNO： 端子番号が違う
- UCAERR_PARA_PTRNULL： ポインタ引数が NULL
- UCAERR_OPT_INVALID： 正しくないオプションを指定した

■ 処理内容説明

指定した入力端子から、数値データを入力します。
入力に成功（戻り値がSUCCEED）すると、入力結果はプロセスデータRVnnに格納されます。
入力に失敗（戻り値がSUCCEED以外）したときは、プロセスデータRVnnのデータステータスのみ更新し、データ値は保持します。

■ 引数説明

● i

結合端子を指定します。Qnn（nnはiに対応）端子を使用します。
i=0のときはIN端子を使用し、入力結果はRVに格納されます。i=0の時も低水準入出力関数の動作をします。

● option

以下の中から指定することができます。

- UCAOPT_RW_SETIOP：
  入力に失敗したときに、IOPアラーム（UCAERR_RW_CONNFAIL、UCAERR_RW_MNT、UCAERR_RW_OS、UCAERR_RW_IOPHI、UCAERR_RW_IOPLO（PI/O機器異常）、UCAERR_RW_BADVALUEのとき）またはIOP-アラーム（UCAERR_RW_IOPLO（入力オープン（下限））のとき）を検出します。
- NOOPTION

■ 詳細説明

UcaRWReadのデータの流れは以下のとおりです。

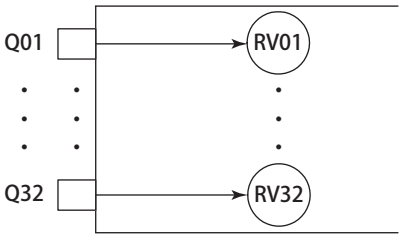


図　UcaRWReadのデータの流れ

UcaRWReadは、以下のアラームを検出します。

表　UcaRWReadのアラーム検出

オプション / アラーム	NOOPTION	UCAOPT_RW_SETIOP
IOP	×	○
CNF	×	○

○：検出する
×：検出しない

Qnn 端子に関するアラームに対するビルダ定義項目はありません。アラームの検出有無の指定は、すべて引数 option で行ってください。引数 option に UCAOPT_RW_SETIOP を指定した場合、IOP アラームと CNF アラームを検出します。
引数 i を 0 に指定した場合、IN 端子からデータを読み込むことができます。しかし高水準入出力関数を使用した場合と比較すると、入力信号変換などの機能が異なります。通常 IN 端子からデータを読み込む場合は、UcaRWRead を使用せず高水準入出力関数を使用してください。
結合を定義してある Qnn 端子のみ UcaRWRead で読みこみ処理をしてください。入力結合を定義していない Qnn 端子に対して UcaRWRead でデータを読み込むと、RVnn のデータステータスは QST になります。

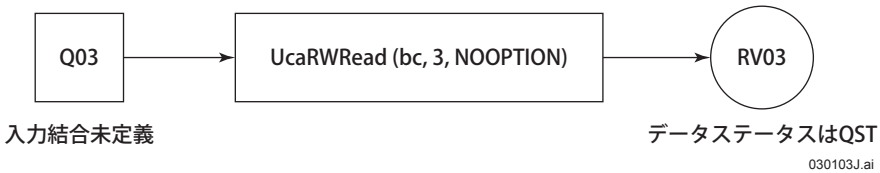


図　入力結合未定義のQnn端子からの読み込み

3.1.2 UcaRWWrite

UcaRWWriteは、J01～J16端子へデータを書きこみます。

■ 関数名

I32 UcaRWWrite

■ 引数

● blockContext

- データ型： UcaBlockContext
- タイプ： IN
- 説明： ブロックコンテキスト

● i

- データ型： I32
- タイプ： IN
- 説明： 出力端子番号（0 ～ 16）

● option

- データ型： U32
- タイプ： IN
- 説明： オプション

■ 戻り値

- SUCCEED： 出力成功
- UCAERR_RW_NOCONNECT： 結合未定義
- UCAERR_RW_LOOPCONN： 端子間入力結合
- UCAERR_RW_CONNFAIL： データアクセスエラー
- UCAERR_RW_MNT： 結合先がオンラインメンテナンス中
- UCAERR_RW_OS： 結合先機能ブロックのブロックモードが O/S
- UCAERR_RW_SWOPEN： 切換スイッチがオープン
- UCAERR_RW_OOP： 出力オープン、PI/O 機器異常
- UCAERR_RW_PTPF： 出力異常
- UCAERR_RW_BADVALUE： 結合先データのデータステータスが BAD
- UCAERR_RW_INVALIDNO： 端子番号が違う
- UCAERR_PARA_PTRNULL： ポインタ引数が NULL
- UCAERR_OPT_INVALID： 正しくないオプションを指定した
- UCAERR_NOTANUMBER： 出力値が非数（Not a Number）である
- UCAERR_INFINITY： 出力値の浮動小数データが無限大である

■ 処理内容説明

指定した出力端子から、連続制御形ユーザカスタムブロックの場合プロセスデータ MVnn（nn は i に対応）を、汎用演算形ユーザカスタムブロックの場合はプロセスデータ CPVnn（nn は i に対応）を出力します。低水準入出力関数の動作をします。

■ 引数説明

● i

結合端子番号を指定します。i=0 のときは、OUT 端子から、MV または CPV を出力し、低水準入出力の動作をします。

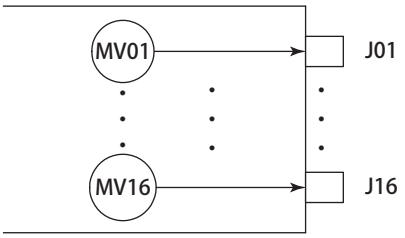
● option

以下の中から複数指定することができます。複数指定する場合は、それぞれを‘|’（or）で結んでください。

- UCAOPT_RW_SETOOP：
  出力に失敗したときに、OOP アラーム（UCAERR_RW_CONNFAIL、 UCAERR_RW_MNT、UCAERR_RW_OS、UCAERR_RW_OOP、 UCAERR_RW_PTPF）を検出します。
- UCAOPT_RW_SETCNF：
  出力に失敗したときに、CNF アラーム（UCAERR_RW_CONNFAIL、 UCAERR_RW_MNT、UCAERR_RW_OS、UCAERR_RW_OOP、 UCAERR_RW_PTPF）を検出します。
- NOOPTION

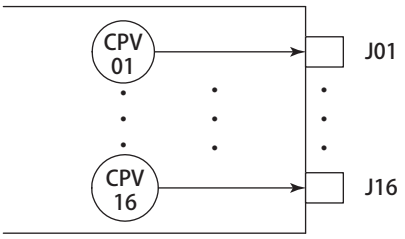
■ 詳細説明

UcaRWWrite は、CSTM-C ブロックの場合はデータアイテム MVnn の値を Jnn 端子へ出力します。CSTM-A ブロックの場合は、データアイテム CPVnn の値を Jnn 端子へ出力します。



030104J.ai

図　UcaRWWriteのデータの流れ（CSTM-Cブロックの場合）



030105J.ai

図　UcaRWWriteのデータの流れ（CSTM-Aブロックの場合）

UcaRWWrite は、以下のアラームを検出します。

表　UcaRWWriteのアラーム検出

オプション / アラーム	NOOPTION	UCAOPT_RW_SETCNF	UCAOPT_RW_SETOOP
OOP	×	×	○
CNF	×	○	○

○： 検出する
×： 検出しない

Jnn 端子に関するアラームに対するビルダ定義項目はありません。アラームの検出有無の指定は、すべて引数 option で行ってください。引数 option に UCAOPT_RW_SETOOP を指定した場合、OOP アラームと CNF アラームを検出します。引数 option に UCAOPT_RW_SETCNF と UCAOPT_RW_SETOOP を両方指定した場合も、OOP アラームと CNF アラームを検出します。
引数 i に 0 を指定した場合、OUT 端子へデータを出力することができます。しかし高水準入出力関数を使用した場合と比較すると、出力信号変換などの機能が異なります。通常 OUT 端子へデータを出力する場合は、UcaRWWrite を使用せず高水準入出力関数を使用してください。

3.1.3 UcaRWReadback

UcaRWReadbackは、J01～J16端子からデータを読み返します。

■ 関数名

I32 UcaRWReadback

■ 引数

● blockContext

- データ型： UcaBlockContext
- タイプ： IN
- 説明： ブロックコンテキスト

● i

- データ型： I32
- タイプ： IN
- 説明： 出力端子番号（0 ～ 16）

● valPtr

- データ型： F64S *
- タイプ： OUT
- 説明： 出力読み返し値ポインタ

● option

- データ型： U32
- タイプ： IN
- 説明： オプション

■ 戻り値

- SUCCEED： 入力成功
- UCAERR_RW_NOCONNECT： 結合未定義
- UCAERR_RW_LOOPCONN： 端子間入力結合
- UCAERR_RW_CONNFAIL： データアクセスエラー
- UCAERR_RW_MNT： 結合先がオンラインメンテナンス中
- UCAERR_RW_OS： 結合先機能ブロックのブロックモードが O/S
- UCAERR_RW_SWOPEN： 切換スイッチがオープン
- UCAERR_RW_OOP： 出力オープン、PI/O 機器異常
- UCAERR_RW_PTPF： 出力異常
- UCAERR_RW_BADVALUE： 結合先データのデータステータスが BAD
- UCAERR_RW_INVALIDNO： 端子番号が違う
- UCAERR_PARA_PTRNULL： ポインタ引数が NULL
- UCAERR_OPT_INVALID： 正しくないオプションを指定した

■ 処理内容説明

指定した出力端子から、データを読み返します。
入力に成功（戻り値が SUCCEED）すると、入力結果は、F64S データ型に変換され、引数 valPtr で指定された場所に格納されます。入力に失敗（戻り値が SUCCEED 以外）したときは、valPtr で指定された領域のデータステータスを更新し、データ値は保持します。

■ 引数説明

● i

結合端子番号を指定します。i=0 のときは、OUT 端子を使用し、低水準入出力関数の動作をします。

● valPtr

入力値を格納する領域へのポインタです。

● option

以下の中から複数指定することができます。複数指定する場合は、それぞれを‘|’（or）で結んでください。

- UCAOPT_RW_SETOOP：
  出力に失敗したときに、OOP アラーム（UCAERR_RW_CONNFAIL、 UCAERR_RW_MNT、UCAERR_RW_OS、UCAERR_RW_OOP、 UCAERR_RW_PTPF）を検出します。
- UCAOPT_RW_SETCNF：
  出力に失敗したときに、CNF アラーム（UCAERR_RW_CONNFAIL、 UCAERR_RW_MNT、UCAERR_RW_OS、UCAERR_RW_OOP、 UCAERR_RW_PTPF）を検出します。
- NOOPTION

■ 詳細説明

UcaRWReadback のデータの流れは以下のとおりです。

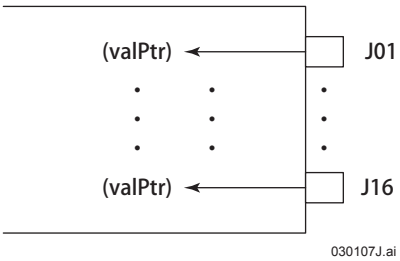


図　UcaRWReadbackのデータの流れ

UcaRWReadback は、以下のアラームを検出します。

表　UcaRWReadbackのアラーム検出

オプション / アラーム	NOOPTION	UCAOPT_RW_SETCNF	UCAOPT_RW_SETOOP
OOP	×	×	○
CNF	×	○	○

○： 検出する
×： 検出しない

Jnn 端子に関するアラームに対するビルダ定義項目はありません。アラームの検出有無の指定は、すべて引数 option で行ってください。引数 option に UCAOPT_RW_SETOOP を指定した場合、OOP アラームと CNF アラームを検出します。
引数 option に UCAOPT_RW_SETCNF と UCAOPT_RW_SETOOP を両方指定した場合も、OOP アラームと CNF アラームを検出します。

3.1.4 UcaRWReadString

UcaRWReadStringは、Q01～Q32端子から文字列データを読み込みます。

■ 関数名

I32 UcaRWReadString

■ 引数

● blockContext

- データ型： UcaBlockContext
- タイプ： IN
- 説明： ブロックコンテキスト

● i

- データ型： I32
- タイプ： IN
- 説明： 入力端子番号（0 ～ 32）

● string

- データ型： BYTE *
- タイプ： OUT
- 説明： 入力値

● option

- データ型： U32
- タイプ： IN
- 説明： オプション

■ 戻り値

- SUCCEED： 入力成功
- UCAERR_RW_NOCONNECT： 結合未定義
- UCAERR_RW_LOOPCONN： 端子間入力結合
- UCAERR_RW_CONNFAIL： データアクセスエラー
- UCAERR_RW_MNT： 結合先がオンラインメンテナンス中
- UCAERR_RW_OS： 結合先機能ブロックのブロックモードが O/S
- UCAERR_RW_SWOPEN： 切換スイッチがオープン
- UCAERR_RW_INVALIDNO： 端子番号が違う
- UCAERR_PARA_PTRNULL： ポインタ引数が NULL
- UCAERR_OPT_INVALID： 正しくないオプションを指定した

■ 処理内容説明

指定した入力端子から、文字列データを入力します。
入力に成功（戻り値が SUCCEED）すると、入力結果は引数 string で指定された場所に格納されます。入力に失敗（戻り値が SUCCEED 以外）したときは、string で指定された場所を更新しません。
入力に失敗しても、IOP、IOP- アラームは検出しません。
引数 string の文字列の終端には '¥0' が付加されます。

重要 文字列を読み込む領域には、機能ブロックデータアイテムの文字列の最大サイズ 16 バイトに、'¥0' の分 1 バイトを加え、17 バイト以上を確保してください。

■ 引数説明

● i

結合端子番号を指定します。i=0 のときは、IN 端子を使用しますが、通常は Q01 ～ Q32 端子を使用してください。

● string

入力値を格納する領域へのポインタです。

● option

NOOPTION を指定してください。

■ 詳細説明

UcaRWReadString のデータの流れは以下のとおりです。

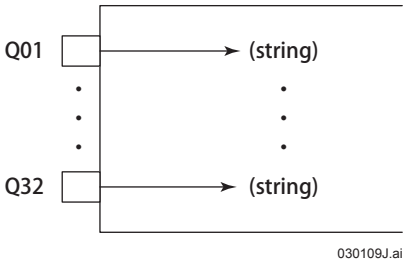


図　UcaRWReadStringのデータの流れ

3.1.5 UcaRWWriteString

UcaRWWriteStringは、J01～J16端子へ文字列データを書きこみます。

■ 関数名

I32 UcaRWWriteString

■ 引数

● blockContext

- データ型： UcaBlockContext
- タイプ： IN
- 説明： ブロックコンテキスト

● i

- データ型： I32
- タイプ： IN
- 説明： 出力端子番号（0 ～ 16）

● string

- データ型： BYTE *
- タイプ： IN
- 説明： 出力値

● option

- データ型： U32
- タイプ： IN
- 説明： オプション

■ 戻り値

- SUCCEED： 出力成功
- UCAERR_RW_NOCONNECT： 結合未定義
- UCAERR_RW_LOOPCONN： 端子間入力結合
- UCAERR_RW_CONNFAIL： データアクセスエラー
- UCAERR_RW_MNT： 結合先がオンラインメンテナンス中
- UCAERR_RW_OS： 結合先機能ブロックのブロックモードが O/S
- UCAERR_RW_SWOPEN： 切換スイッチがオープン
- UCAERR_RW_INVALIDNO： 端子番号が違う
- UCAERR_PARA_PTRNULL： ポインタ引数が NULL
- UCAERR_OPT_INVALID： 正しくないオプションを指定した

■ 処理内容説明

指定した出力端子から、文字列データを出力します。
出力に失敗しても、OOP アラームは検出しません。

■ 引数説明

● i

結合端子番号を指定します。i=0 のときは、OUT 端子を使用しますが、通常は J01 ～ J16 端子を使用してください。

● string

出力値を格納した領域へのポインタです。

● option

NOOPTION を指定してください。

■ 詳細説明

UcaRWWriteString のデータの流れは以下のとおりです。

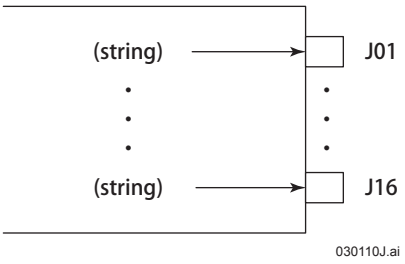


図　UcaRWWriteStringのデータの流れ

3.1.6 UcaRWWritePvToJnSub

UcaRWWritePvToJnSubは、PVをJ01～J16端子へ書きこみ、SUB端子出力を行います。

■ 関数名

I32 UcaRWWritePvToJnSub

■ 引数

● blockContext

- データ型： UcaBlockContext
- タイプ： IN
- 説明： ブロックコンテキスト

● i

- データ型： I32
- タイプ： IN
- 説明： 出力端子番号（0 ～ 16）

● option

- データ型： U32
- タイプ： IN
- 説明： オプション

■ 戻り値

SUCCEED：	正常終了
UCAERR_RW_NOCONNECT：	結合未定義
UCAERR_RW_LOOPCONN：	端子間入力結合
UCAERR_RW_CONNFAIL：	データアクセスエラー
UCAERR_RW_MNT：	結合先がオンラインメンテナンス中
UCAERR_RW_OS：	結合先機能ブロックのブロックモードが O/S
UCAERR_RW_SWOPEN：	切換スイッチがオープン
UCAERR_RW_OOP：	出力オープン、PI/O 機器異常
UCAERR_RW_PTPF：	出力異常
UCAERR_RW_BADVALUE：	結合先データのデータステータスが BAD
UCAERR_RW_INVALIDNO：	端子番号が違う
UCAERR_PARA_PTRNULL：	ポインタ引数が NULL
UCAERR_OPT_INVALID：	正しくないオプションを指定した
UCAERR_NOTANUMBER：	出力値が非数（Not a Number）である
UCAERR_INFINITY：	出力値の浮動小数データが無限大である

■ 処理内容説明

PV を F64S → F32S 型変換し、MVnn（nn は i に対応）に格納します。
型変換に失敗した場合、MVnn のデータ値は変化しません。データステータスは BAD にします。
Jnn（nn は i に対応）端子から、MVnn（nn は i に対応）を出力します。
SUB 端子出力を行います（ビルダ定義項目は PV、DPV のみ有効です）。

■ 引数説明

● i

結合端子番号を指定します。i=0 のときは、OUT 端子を使用し、低水準入出力関数の動作をします。

● option

以下の中から複数指定することができます。複数指定する場合は、それぞれを‘|’(or)で結んでください。

- UCAOPT_RW_SETCNF： 出力失敗時に CNF アラームを検出します。
- UCAOPT_RW_SETOOP： 出力失敗時に OOP アラームを検出します。
- NOOPTION

■ 詳細説明

UcaRWWritePvToJnSub のデータの流れは以下のとおりです。

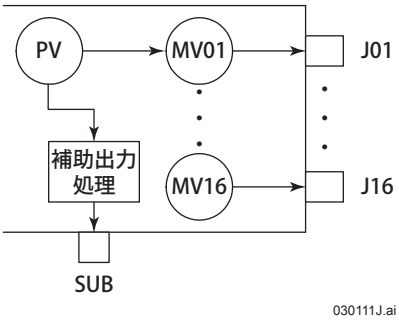


図　UcaRWWritePvToJnSubのデータの流れ

UcaRWWritePvToJnSub は、以下のアラームを検出します。

表　UcaRWWritePvToJnSubのアラーム検出

オプション / アラーム	NOOPTION	UCAOPT_RW_SETCNF	UCAOPT_RW_SETOOP
OOP	×	×	○
CNF	×	○	○

○： 検出する
×： 検出しない

Jnn 端子に関するアラームに対するビルダ定義項目はありません。アラームの検出有無の指定は、すべて引数 option で行ってください。引数 option に UCAOPT_RW_SETOOP を指定した場合、OOP アラームと CNF アラームを検出します。引数 option に UCAOPT_RW_SETCNF と UCAOPT_RW_SETOOP を両方指定した場合も、OOP アラームと CNF アラームを検出します。

UcaRWWritePvToJnSub は、UcaRWSetPv がデータアイテム PV に設定した PV 値を入力とします。UcaRWWritePvToJnSub を呼ぶ前に、あらかじめ UcaRWSetPv を呼び出し PV を設定してください。

UcaRWWritePvToJnSub は、出力タブシートのビルダ定義項目［補助出力］－［出力値］を PV または ΔPV に指定した場合のみ補助出力処理を行います。ビルダ定義項目［補助出力］－［出力値］を MV または ΔMV に指定した場合は、補助出力端子から何も出力しません。

3.2 高水準入出力

高水準入出力関数はIN端子とOUT端子からデータの読み書きを行います。データステータス処理や信号変換処理などを行います。

3.2.1 UcaRWReadIn

UcaRWReadInは、IN端子からデータを読み込み、データアイテムRVに格納します。

■ 関数名

I32 UcaRWReadIn

■ 引数

● blockContext

- データ型： UcaBlockContext
- タイプ： IN
- 説明： ブロックコンテキスト

● option

- データ型： U32
- タイプ： IN
- 説明： オプション

■ 戻り値

SUCCEED：	入力成功
UCAERR_RW_NOCONNECT：	結合未定義
UCAERR_RW_LOOPCONN：	端子間入力結合
UCAERR_RW_CONNFAIL：	データアクセスエラー
UCAERR_RW_MNT：	結合先がオンラインメンテナンス中
UCAERR_RW_OS：	結合先機能ブロックのブロックモードが O/S
UCAERR_RW_SWOPEN：	切換スイッチがオープン
UCAERR_RW_IOPHI：	入力オープン（上限）
UCAERR_RW_IOPLO：	入力オープン（下限）、PI/O 機器異常
UCAERR_RW_BADVALUE：	結合先データのデータステータスが BAD
UCAERR_PARA_PTRNULL：	ポインタ引数が NULL
UCAERR_OPT_INVALID：	正しくないオプションを指定した

■ 処理内容説明

IN 端子から、データを入力します。
結果は、データアイテム RV に格納されます。

■ 引数説明

● option

NOOPTION を指定してください。

■ 詳細説明

UcaRWReadIn のデータの流れは以下のとおりです。

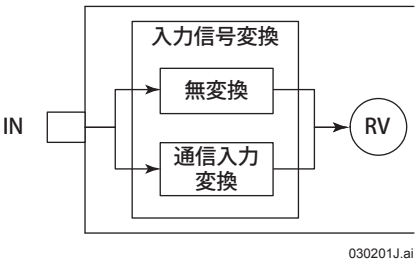


図　UcaRWReadInのデータの流れ

UcaRWReadIn は、以下のアラームを検出します。

表　UcaRWReadInのアラーム検出

オプション / アラーム	NOOPTION
IOP	○
CNF	○

○： 検出する
×： 検出しない

警報タブシートのビルダ定義項目［入力オープン警報］や［結合状態不良警報］に警報なしを指定した場合、対応するアラームは検出しません。
UcaRWReadIn は、基本タブシートのビルダ定義項目［入力信号変換］の指定に基づき入力信号変換を行います。

3.2.2 UcaRWSetPv

UcaRWSetPvは、測定値をPVに格納します。

■ 関数名

I32 UcaRWSetPv

■ 引数

● blockContext

- データ型： UcaBlockContext
- タイプ： IN
- 説明： ブロックコンテキスト

● pvPtr

- データ型： F64S *
- タイプ： IN
- 説明： 測定入力値

● err

- データ型： I32
- タイプ： IN
- 説明： UcaRWReadIn( ) の戻り値

● option

- データ型： U32
- タイプ： IN
- 説明： オプション

■ 戻り値

- SUCCEED： 成功
- UCAERR_PARA_PTRNULL： ポインタ引数が NULL
- UCAERR_BLKTYPE_INVALID： 正しくないブロック形を指定した
- UCAERR_OPT_INVALID： 正しくないオプションを指定した

■ 処理内容説明

入力信号に関する、以下の処理をまとめて行います。

PV データステータス決定
デジタルフィルタ処理
積算処理（SUM を更新します）
入力上下限警報
入力上上下下限警報
入力変化率警報

結果は、PV に格納されます。

■ 引数説明

● pvPtr

測定入力値へのポインタです。NULL を渡すと、プロセスデータ RV を入力結果として使用します。

● err

UcaRWReadIn( ) の戻り値を入れます。UcaRWReadIn( ) を呼ばずに RV を設定した場合は、SUCCEED を指定してください。

● option

以下の中から複数指定することができます。複数指定する場合は、それぞれを‘|’(or) で結んでください。

- UCAOPT_RW_NOPVSIS：
  演算入力値異常検出指定による PV データステータスの作成を行いません。
- UCAOPT_RW_NOFILTER： デジタルフィルタ処理を行いません。
- UCAOPT_RW_NOSUM： 積算処理を行いません。SUM は更新されません。
- UCAOPT_RW_NOHILOALRM： 入力上下限警報の検査を行いません。
- UCAOPT_RW_NOHHLLALRM： 入力上上下下限警報の検査を行いません。
- UCAOPT_RW_NOVELALRM： 入力変化率警報の検査を行いません。
- NOOPTION

■ 詳細説明

UcaRWSetPv は、RV と RVnn と引数 pvPtr を元に演算入力値異常検出を行い、PV のデータステータスを作成します。PV 振り切り、デジタルフィルタ、積算の入力信号処理を行い、PV に格納します。

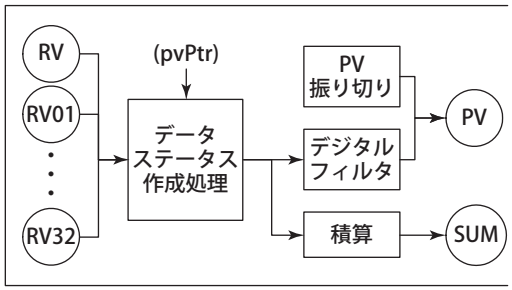


図 UcaRWSetPvのデータの流れ

UcaRWSetPv は、引数 pvPtr の値が非数（Not a Number）または無限大であると、データ値を保存せずデータステータスに BAD を設定しますが、処理は中断しないでデジタルフィルタ処理や積算処理を行い、SUCCEED を返します。

UcaRWSetPv は、以下のアラームを検出します。

表　UcaRWSetPvのアラーム検出

オプション / アラーム	NOOPTION	UCAOPT_RW_NOHHLLALRM	UCAOPT_RW_NOHILOALRM	UCAOPT_RW_NOVELALRM
IOP	○	○	○	○
HH/LL	○	×	○	○
HI/LO	○	○	×	○
VEL+/VEL- (*1)	○	○	○	×

○： 検出する
×： 検出しない
*1： 機能ブロックワンショット処理から呼び出された場合、検出しません。

警報タブシートのビルダ定義項目［入力上下限警報］や［入力変化率警報］に警報なしを指定した場合、引数 option によらず対応するアラームは検出しません。
機能ブロックワンショット処理から呼び出された場合、UcaRWSetPv は積算処理を行わず、また入力変化率警報を検出しません。
UcaRWSetPv は、以下の表に基づき PV のデータステータスを作成します。

表　PVデータステータス作成

ビルダ定義項目［演算入力値異常検出］の指定	引数pvPtrのデータステータス	RVのデータステータス	RVnnのデータステータス	作成されるPVのデータステータス
全検出形	BAD	任意	任意	BAD
	任意	BAD	任意	BAD
	任意	任意	BAD	BAD
	QST	not BAD	not BAD	QST
	not BAD	QST	not BAD	QST
	not BAD	not BAD	QST	QST
	not BAD nor QST	not BAD nor QST	not BAD nor QST	0（正常）
補正演算形	BAD	任意	任意	BAD
	任意	BAD	任意	BAD
	任意	任意	BAD	QST
	QST	not BAD	not BAD	QST
	not BAD	QST	not BAD	QST
	not BAD	not BAD	QST	QST
	not BAD nor QST	not BAD nor QST	not BAD nor QST	0（正常）
非検出形	BAD	任意	任意	BAD
	QST	任意	任意	QST
	not BAD nor QST	任意	任意	0（正常）
オプション UCAOPT_RW_NOPVSTS を指定した場合（[演算入力値異常検出］の指定は無視されます）	BAD	任意	任意	BAD
	QST	任意	任意	QST
	not BAD nor QST	任意	任意	0（正常）

任意：任意のデータステータス

PV のデータステータスの作り方は、警報タブシートのビルダ定義項目［演算入力値異常検出］の指定と UcaRWSetPv のオプション UCAOPT_RW_NOPVSTS の指定により決まります。データステータスの作り方が決まると、上表の各行の条件を上から下に評価して最初に該当する行に従って PV のデータステータスが作成されます。
ビルダ定義項目［演算入力値異常検出］に全検出形または補正演算形を指定した場合、PV データステータス作成処理で RV や RVn のデータステータスを参照します。また、RV のデータ値を元に入力信号変換を行います。UcaRWSetPv を呼ぶ前に、あらかじめ RV と RVn の値を設定してください。

重要

UcaRWSetPv は、1 回の処理タイミングにおいて必ず一度だけ呼び出してください。機能ブロック定周期処理なら、UcaBlockPeriodical の入り口から return までの間で、UcaRWSetPv を一度だけ呼び出してください。
PV 値を毎処理タイミングで設定しないと、以下の問題が発生します。

- 積算値をデータアイテム SUM に正しく設定できません。
- 入力変化率アラーム（VEL+/VEL-）を正しく検出できません。

UcaRWSetPv を複数回呼び出さないでください。複数回呼び出すと、以下の問題が発生します。

- SUB 端子から出力可能な ΔPV（前回 PV と今回 PV の差分）が正常に作成できなくなります。

ΔPV は、UcaRWSetPv が呼び出された間の PV の差分です。UcaRWSetPv を 1 回のユーザ定義関数の処理内で 2 回呼び出すと、ΔPV は同じ処理内の 2 回の UcaRWSetPv 呼び出しに指定した PV 値の差分となってしまいます。

3.2.3 UcaRWSetCpv

UcaRWSetCpvは、演算入力値をCPVに格納します。

■ 関数名

I32 UcaRWSetCpv

■ 引数

● blockContext

- データ型： UcaBlockContext
- タイプ： IN
- 説明： ブロックコンテキスト

● cpvPtr

- データ型： F64S *
- タイプ： IN
- 説明： 演算入力値

● err

- データ型： I32
- タイプ： IN
- 説明： UcaRWReadIn( ) の戻り値

● option

- データ型： U32
- タイプ： IN
- 説明： オプション

■ 戻り値

- SUCCEED： 成功
- UCAERR_PARA_PTRNULL： ポインタ引数が NULL
- UCAERR_BLKTYPE_INVALID： 正しくないブロック形を指定した
- UCAERR_OPT_INVALID： 正しくないオプションを指定した

■ 処理内容説明

入力信号に関する、以下の処理をまとめて行います。

CPV データステータス決定

デジタルフィルタ処理

積算処理（SUM を更新します）

結果は、CPV に格納されます。

■ 引数説明

● cpvPtr

演算入力値へのポインタです。NULL を渡すと、プロセスデータ RV を入力結果として使用します。

● err

UcaRWReadIn( ) の戻り値を入れます。UcaRWReadIn( ) を呼ばずに RV を設定した場合は、SUCCEED を指定してください。

● option

以下の中から指定することができます。複数指定する場合は、それぞれを‘|’（or）で結んでください。

- UCAOPT_RW_NOPVSTS：
  演算入力値異常検出指定による CPV データステータスの作成を行いません。
- UCAOPT_RW_NOFILTER：デジタルフィルタ処理を行いません。
- UCAOPT_RW_NOSUM： 積算処理を行いません。SUM は更新されません。
- NOOPTION

■ 詳細説明

UcaRWSetCpv は、RV と RVnn と引数 cpvPtr を元に演算入力値異常検出を行い、CPV のデータステータスを作成します。CPV振り切り、デジタルフィルタ、積算の入力信号処理を行い、CPV に格納します。

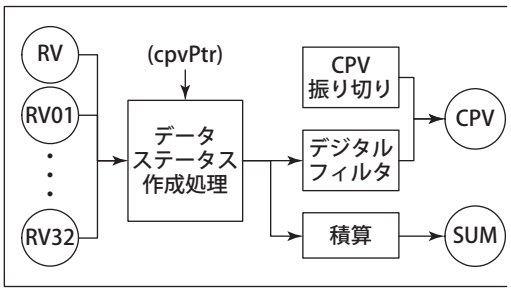


図　UcaRWSetCpvのデータの流れ

UcaRWSetCpv は、引数 cpvPtr の値が非数（Not a Number）または無限大であると、データ値を保存せずデータステータスに BAD を設定しますが、処理は中断しないでデジタルフィルタ処理や積算処理を行い、SUCCEED を返します。
機能ブロックワンショット処理から呼び出された場合、UcaRWSetCpv は積算処理を行いません。
UcaRWSetCpv は、以下のアラームを検出します。

表　UcaRWSetCpvのアラーム検出

オプション / アラーム	NOOPTION
IOP	○

○： 検出する

警報タブシートのビルダ定義項目［入力オープン警報］に警報なしを指定した場合、IOP アラームは検出しません。
UcaRWSetCpv は、以下の表に基づき CPV のデータステータスを作成します。

表　CPVデータステータス作成

ビルダ定義項目［演算入力値異常検出］の指定	引数cpvPtrのデータステータス	RVのデータステータス	RVnnのデータステータス	作成されるCPVのデータステータス
全検出形	BAD	任意	任意	BAD
	任意	BAD	任意	BAD
	任意	任意	BAD	BAD
	QST	not BAD	not BAD	QST
	not BAD	QST	not BAD	QST
	not BAD	not BAD	QST	QST
	not BAD nor QST	not BAD nor QST	not BAD nor QST	0（正常）
補正演算形	BAD	任意	任意	BAD
	任意	BAD	任意	BAD
	任意	任意	BAD	QST
	QST	not BAD	not BAD	QST
	not BAD	QST	not BAD	QST
	not BAD	not BAD	QST	QST
	not BAD nor QST	not BAD nor QST	not BAD nor QST	0（正常）
非検出形	BAD	任意	任意	BAD
	QST	任意	任意	QST
	not BAD nor QST	任意	任意	0（正常）
オプション UCAOPT_RW_NOPVSTS を指定した場合（［演算入力値異常検出］の指定は無視されます）	BAD	任意	任意	BAD
	QST	任意	任意	QST
	not BAD nor QST	任意	任意	0（正常）

任意：任意のデータステータス

CPV のデータステータスの作り方は、警報タブシートのビルダ定義項目［演算入力値異常検出］の指定と UcaRWSetCpv のオプション UCAOPT_RW_NOPVSTS の指定により決まります。データステータスの作り方が決まると、上表の各行の条件を上から下に評価して最初に該当する行に従って CPV のデータステータスが作成されます。
ビルダ定義項目［演算入力値異常検出］に全検出形または補正演算形を指定した場合、CPV データステータス作成処理で RV や RVn のデータステータスを参照します。また、RV のデータ値を元に入力信号変換を行います。UcaRWSetCpv を呼ぶ前に、あらかじめ RV と RVn の値を設定してください。

重要 1 回のユーザ定義関数の処理内では、UcaRWSetCpv は一度だけ呼び出してください。機能ブロック定周期処理なら、UcaBlockPeriodical の入り口から return までの間で、UcaRWSetCpv を一度だけ呼び出してください。
UcaRWSetCpv は、処理タイミングごとに呼び出してください。CPV 値を処理タイミングごとに設定しないと、以下の問題が発生します。

- 積算値をデータアイテム SUM に正しく設定できません。

UcaRWSetCpv を複数回呼び出さないでください。複数回呼び出すと、以下の問題が発生します。

- SUB 端子から出力可能な Δ CPV（前回 CPV と今回 CPV の差分）が正常に作成できなくなります。

ΔCPV は、UcaRWSetCpv が呼び出された間の CPV の差分です。UcaRWSetCpv を 1 回のユーザ定義関数の処理内で 2 回呼び出すと、Δ CPV は同じ処理内の 2 回の UcaRWSetCpv 呼び出しに指定した CPV 値の差分となってしまいます。

3.2.4 UcaRWWriteMvToOutSub

UcaRWWriteMvToOutSubは、内部変数MvblをOUT端子から出力します。また、SUB端子への出力も行います。

■ 関数名

I32 UcaRWWriteMvToOutSub

■ 引数

● blockContext

- データ型： UcaBlockContext
- タイプ： IN
- 説明： ブロックコンテキスト

● option

- データ型： U32
- タイプ： IN
- 説明： オプション

■ 戻り値

SUCCEED：	正常終了
UCAERR_RW_NOCONNECT：	結合未定義
UCAERR_RW_LOOPCONN：	端子間入力結合
UCAERR_RW_CONNFAIL：	データアクセスエラー
UCAERR_RW_MNT：	結合先がオンラインメンテナンス中
UCAERR_RW_OS：	結合先機能ブロックのブロックモードが O/S
UCAERR_RW_SWOPEN：	切換スイッチがオープン
UCAERR_RW_OOP：	出力オープン、PI/O 機器異常
UCAERR_RW_PTPF：	出力異常
UCAERR_RW_BADVALUE：	結合先データのデータステータスが BAD
UCAERR_NOTANUMBER：	出力値が非数（Not a Number）である
UCAERR_INFINITY：	出力値の浮動小数データが無限大である
UCAERR_PARA_PTRNULL：	ポインタ引数が NULL
UCAERR_OPT_INVALID：	正しくないオプションを指定した
UCAERR_BLKTYPE_INVALID：	正しくないブロック形を指定した

■ 処理内容説明

内部変数 Mvbl から OUT 端子へ出力を行います。
SUB 端子出力を行います。

■ 引数説明

● option

以下の中から複数指定することができます。複数指定する場合は、それぞれを‘|’(or)で結んでください。

- UCAOPT_RW_HOLD： 出力をホールドします。
- UCAOPT_RW_NOMHML： 出力上下限警報の検査を行いません。
- UCAOPT_RW_NOSETINIT： 制御初期化フラグを操作しません。
- UCAOPT_RW_NOEQUALIZERMV： RMV をイコライズしません。
- UCAOPT_RW_CTRLTIMING： 制御周期時のみ出力します。
- NOOPTION

重要 OUT 端子出力失敗時には SUCCEED を返します。あらかじめ関数 UcaRWReadbackMv( ) でエラーを検査してください。SUB 端子出力失敗時にはエラーコードを返します。

■ 詳細説明

UcaRWWriteMvToOutSub は、出力リミット前出力値(内部変数 Mvbl)を元に出力リミッタ、出力変化率リミッタの出力信号処理を行います。プリセット操作出力処理を行い、MV へデータを格納します。MV を RMV に代入し、MV と RMV をイコライズします。MV に対して出力信号変換を行い、OUT 端子へ出力します。補助出力処理を行い、補助出力値を決定します。補助出力値を SUB 端子へ出力します。

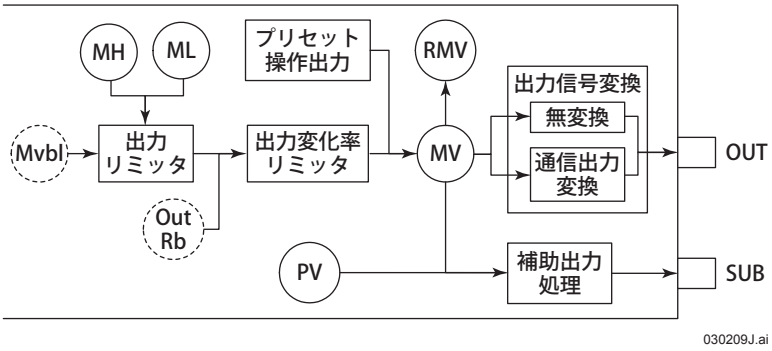


図 UcaRWWriteMvToOutSubのデータの流れ

UcaRWWriteMvToOutSub は、UcaCtrlHandler の出力値作成処理が作成した出力値に対し、出力リミッタ、出力変化率リミッタの処理を行います。

UcaRWWriteMvToOutSub は、以下のアラームを検出します。

表　UcaRWWriteMvToOutSubのアラーム検出

オプション / アラーム	NOOPTION	UCAOPT_RW_NOMHML
OOP	○	○
CNF	○	○
MH/ML	○	×

○： 検出する
×： 検出しない

警報タブシートのビルダ定義項目［出力オープン警報］や［出力上下限警報］に警報なしを指定した場合、引数 option によらず対応するアラームは検出しません。
UcaRWWriteMvToOutSub は、基本タブシートのビルダ定義項目［出力信号変換］の指定に基づき出力信号変換を行います。

3.2.5 UcaRWWriteMv

UcaRWWriteMvは、出力リミット前出力値に対して出力信号処理を行い、OUT端子へ出力します。

■ 関数名

I32 UcaRWWriteMv

■ 引数

● blockContext

- データ型： UcaBlockContext
- タイプ： IN
- 説明： ブロックコンテキスト

● option

- データ型： U32
- タイプ： IN
- 説明： オプション

■ 戻り値

- SUCCEED： 出力成功
- UCAERR_NOTANUMBER： 出力値が非数（Not a Number）である
- UCAERR_INFINITY： 出力値の浮動小数データが無限大である
- UCAERR_PARA_PTRNULL： ポインタ引数が NULL
- UCAERR_OPT_INVALID： 正しくないオプションを指定した
- UCAERR_BLKTYPE_INVALID： 正しくないブロック形を指定した

■ 処理内容説明

連続制御形ユーザカスタムブロックで、OUT 端子から出力するための関数です。

■ 引数説明

● option

以下の中から複数指定することができます。複数指定する場合は、それぞれを‘|’(or)で結んでください。

- UCAOPT_RW_HOLD： 出力をホールドします。
- UCAOPT_RW_NOMHML： 出力上下限警報の検査を行いません。
- UCAOPT_RW_NOSETINIT： 制御初期化フラグを操作しません。
- UCAOPT_RW_NOEQUALIZERMV： RMV をイコライズしません。
- NOOPTION

重要 この関数を使用するためには、あらかじめ関数 UcaRWReadbackMv( ) を呼んでおく必要があります。
この関数は、毎スキャン呼んでください。
ブロックモードが AUT で制御演算をしなかった（制御周期でない、または制御ホールドした）ときは、option に UCAOPT_RW_HOLD を指定してください。
出力失敗時には SUCCEED を返します。あらかじめ関数 UcaRWReadbackMv( ) でエラーを検査してください。

■ 詳細説明

UcaRWWriteMv は、出力リミット前出力値（内部変数 Mvbl）を元に出力リミッタ、出力変化率リミッタの出力信号処理を行います。プリセット操作出力処理を行い、MV へデータを格納します。MV を RMV に代入し、MV と RMV をイコライズします。MV に対して出力信号変換を行い、OUT 端子へ出力します。

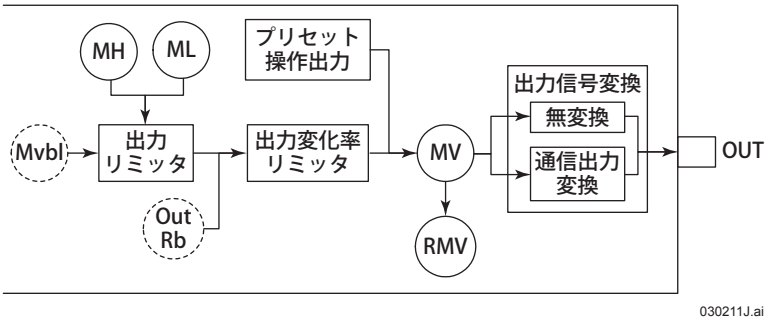


図　UcaRWWriteMvのデータの流れ

UcaRWWriteMv は、UcaCtrlHandler の出力値作成処理が作成した出力値に対し、出力リミッタ、出力変化率リミッタの処理を行います。
UcaRWWriteMv は、以下のアラームを検出します。

表　UcaRWWriteMvのアラーム検出

オプション / アラーム	NOOPTION	UCAOPT_RW_NOMHML
OOP	○	○
CNF	○	○
MH/ML	○	×

○： 検出する
×： 検出しない

警報タブシートのビルダ定義項目［出力オープン警報］や［出力上下限警報］に警報なしを指定した場合、引数 option によらず対応するアラームは検出しません。
UcaRWWriteMv は、基本タブシートのビルダ定義項目［出力信号変換］の指定に基づき出力信号変換を行います。

3.2.6 UcaRWReadbackMv

UcaRWReadbackMvは、OUT端子の出力読み返し処理を行い、内部変数OutRbへ格納します。

■ 関数名

I32 UcaRWReadbackMv

■ 引数

● blockContext

- データ型： UcaBlockContext
- タイプ： IN
- 説明： ブロックコンテキスト

● option

- データ型： U32
- タイプ： IN
- 説明： オプション

■ 戻り値

SUCCEED：	入力成功
UCAERR_RW_NOCONNECT：	結合未定義
UCAERR_RW_LOOPCONN：	端子間入力結合
UCAERR_RW_CONNFAIL：	データアクセスエラー
UCAERR_RW_MNT：	結合先がオンラインメンテナンス中
UCAERR_RW_OS：	結合先機能ブロックのブロックモードが O/S
UCAERR_RW_SWOPEN：	切換スイッチがオープン
UCAERR_RW_OOP：	出力オープン、PI/O 機器異常
UCAERR_RW_PTPF：	出力異常
UCAERR_RW_BADVALUE：	結合先データのデータステータスが BAD
UCAERR_PARA_PTRNULL：	ポインタ引数が NULL
UCAERR_OPT_INVALID：	正しくないオプションを指定した
UCAERR_BLKTYPE_INVALID：	正しくないブロック形を指定した

■ 引数説明

● option

以下の中から指定することができます。

- UCAOPT_RW_NORANGETRACK：出力レンジトラッキングを行いません。
- NOOPTION

■ 詳細説明

UcaRWReadbackMv は、OUT 端子の出力読み返し処理を行い、内部変数 OutRb へ格納します。操作出力（MV）スケール上下限値が出力先データのスケール上下限値と異なる場合、出力レンジトラッキングを行います。出力レンジトラッキングは、OUT 端子の結合がカスケード結合で、結合先が SET 端子のときにのみ動作します。

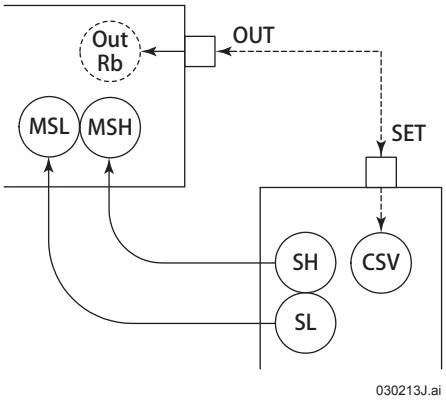


図　UcaRWReadbackMvのデータの流れ

UcaRWReadbackMv は、以下のアラームを検出します。

表　UcaRWReadbackMvのアラーム検出

オプション / アラーム	NOOPTION
OOP	○
CNF	○

○： 検出する
×： 検出しない

警報タブシートのビルダ定義項目［出力オープン警報］や［結合状態不良警報］に警報なしを指定した場合、対応するアラームは検出しません。

3.2.7 UcaRWWriteCpvToOutSub

UcaRWWriteCpvToOutSubは、CPVをOUT端子から出力します。また、SUB端子への出力も行います。

■ 関数名

I32 UcaRWWriteCpvToOutSub

■ 引数

● blockContext

- データ型： UcaBlockContext
- タイプ： IN
- 説明： ブロックコンテキスト

● option

- データ型： U32
- タイプ： IN
- 説明： オプション

■ 戻り値

SUCCEED：	正常終了
UCAERR_RW_NOCONNECT：	結合未定義
UCAERR_RW_LOOPCONN：	端子間入力結合
UCAERR_RW_CONNFAIL：	データアクセスエラー
UCAERR_RW_MNT：	結合先がオンラインメンテナンス中
UCAERR_RW_OS：	結合先機能ブロックのブロックモードが O/S
UCAERR_RW_SWOPEN：	切換スイッチがオープン
UCAERR_RW_OOP：	出力オープン、PI/O 機器異常
UCAERR_RW_PTPF：	出力異常
UCAERR_RW_BADVALUE：	結合先データのデータステータスが BAD
UCAERR_NOTANUMBER：	出力値が非数（Not a Number）である
UCAERR_INFINITY：	出力値の浮動小数データが無限大である
UCAERR_PARA_PTRNULL：	ポインタ引数が NULL
UCAERR_OPT_INVALID：	正しくないオプションを指定した
UCAERR_BLKTYPE_INVALID：	正しくないブロック形を指定した

■ 処理内容説明

OUT 端子読み返し処理を行います。
CPV から OUT 端子へ出力を行います。
SUB 端子出力を行います。

■ 引数説明

● option

以下の中から指定することができます。

- UCAOPT_RW_SETOOP：出力失敗時に OOP アラームを検出します。
- NOOPTION

■ 詳細説明

UcaRWWriteCpvToOutSub は、OUT 端子からの出力の前に、OUT 端子から内部変数 OutRb に出力値を読み返します。

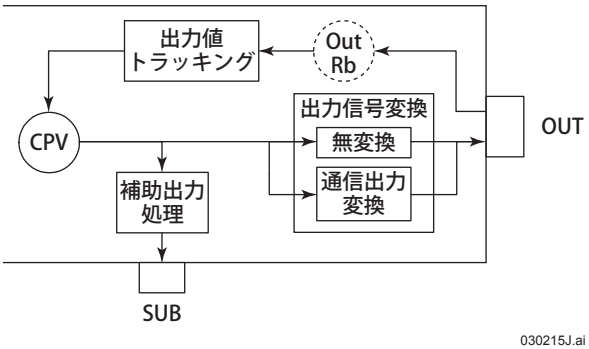


図　UcaRWWriteCpvToOutSubのデータの流れ

出力タブシートのビルダ定義項目［出力値トラッキング］を［あり］に指定した場合、設定先のデータステータスが CND 状態の時に、内部変数 OutRb を CPV に格納し出力値トラッキングを行います。
UcaRWWriteCpvToOutSub は、以下のアラームを検出します。

表　UcaRWWriteCpvToOutSubのアラーム検出

オプション / アラーム	NOOPTION	UCAOPT_RW_SETOOP
OOP	×	○
CNF	○	○

○： 検出する
×： 検出しない

警報タブシートのビルダ定義項目［結合状態不良警報］に警報なしを指定した場合、引数 option によらず CNF アラームは検出しません。引数 option に NOOPTION を指定した場合、CNF アラームのみを検出し OOP アラームを検出しません。
引数 option に UCAOPT_RW_SETOOP を指定した場合、OOP アラームと CNF アラームを検出します。
UcaRWWriteCpvToOutSub は、UcaRWSetCpv により CPV に設定された演算結果を出力します。UcaRWWriteCpvToOutSub を呼ぶ前に、あらかじめ UcaRWSetCpv を呼び CPV を設定してください。

3.2.8 UcaRWWriteCpv

UcaRWWriteCpvは、CPVに出力信号処理を行い、OUT端子へ出力します。

■ 関数名

I32 UcaRWWriteCpv

■ 引数

● blockContext

- データ型： UcaBlockContext
- タイプ： IN
- 説明： ブロックコンテキスト

● option

- データ型： U32
- タイプ： IN
- 説明： オプション

■ 戻り値

- SUCCEED： 出力成功
- UCAERR_NOTANUMBER： 出力値が非数（Not a Number）である
- UCAERR_INFINITY： 出力値の浮動小数データが無限大である
- UCAERR_PARA_PTRNULL： ポインタ引数が NULL
- UCAERR_OPT_INVALID： 正しくないオプションを指定した
- UCAERR_BLKTYPE_INVALID： 正しくないブロック形を指定した

■ 処理内容説明

汎用演算形ユーザカスタムブロックで、OUT 端子から CPV を出力するための関数です。
あらかじめ、プロセスデータ CPV を設定しておいてください。

■ 引数説明

● option

以下の中から指定することができます。

- UCAOPT_RW_SETOOP：出力失敗時に OOP アラームを検出します。
- NOOPTION

重要 この関数を使用するためには、あらかじめ関数 UcaRWReadbackCpv( ) を呼んでおく必要があります。
出力失敗時には SUCCEED を返します。あらかじめ関数 UcaRWReadbackCpv( ) でエラーを検査してください。

■ 詳細説明

設定先のデータステータスが CND 状態の場合は、内部変数 OutRb を CPV に格納し出力値トラッキングを行います。

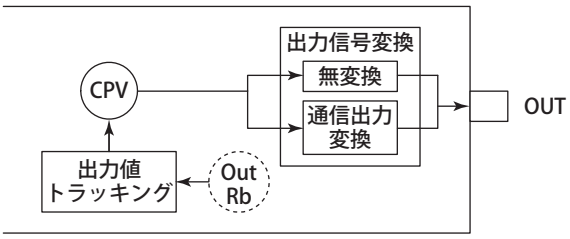


図　UcaRWWriteCpvのデータの流れ

出力タブシートのビルダ定義項目［出力値トラッキング］を［あり］に指定した場合、設定先のデータステータスが CND 状態の時に、内部変数 OutRb を CPV に格納し出力値トラッキングを行います。
UcaRWWriteCpv を呼ぶ前に UcaRWReadbackCpv を呼び出し、出力トラッキング機能で使用する出力読み返し値（内部変数 OutRb）をあらかじめ設定してください。
UcaRWWriteCpv と UcaRWReadbackCpv には、同じオプションを指定してください。
UcaRWReadbackCpv が出力の異常を検出した場合、UcaRWWriteCpv は新たに検出しません。
UcaRWWriteCpv は、以下のアラームを検出します。

表　UcaRWWriteCpvのアラーム検出

オプション / アラーム	NOOPTION	UCAOPT_RW_SETOOP
OOP	×	○
CNF	○	○

○： 検出する
×： 検出しない

警報タブシートのビルダ定義項目［結合状態不良警報］に［なし］を指定した場合、引数 option によらず CNF アラームは検出しません。引数 option に NOOPTION を指定した場合 CNF アラームのみを検出し OOP アラームを検出しませんが、UcaRWWriteCpv を呼ぶ前に OOP アラームが設定されている場合には、OOP アラーム設定状態を保持します。
引数 option に UCAOPT_RW_SETOOP を指定した場合、OOP アラームと CNF アラームを検出します。
UcaRWWriteCpv は、出力タブシートのビルダ定義項目［出力信号変換］の指定に基づき出力信号変換を行います。

3.2.9 UcaRWReadbackCpv

UcaRWReadbackCpvは、OUT端子の出力読み返し処理を行い、内部変数OutRbへ格納します。

■ 関数名

I32 UcaRWReadbackCpv

■ 引数

● blockContext

- データ型： UcaBlockContext
- タイプ： IN
- 説明： ブロックコンテキスト

● option

- データ型： U32
- タイプ： IN
- 説明： オプション

■ 戻り値

SUCCEED：	入力成功
UCAERR_RW_NOCONNECT：	結合未定義
UCAERR_RW_LOOPCPNN：	端子間入力結合
UCAERR_RW_CONNFAIL：	データアクセスエラー
UCAERR_RW_MNT：	結合先がオンラインメンテナンス中
UCAERR_RW_OS：	結合先機能ブロックのブロックモードが O/S
UCAERR_RW_SWOPEN：	切換スイッチがオープン
UCAERR_RW_OOP：	出力オープン、PI/O 機器異常
UCAERR_RW_PTPF：	出力異常
UCAERR_RW_BADVALUE：	結合先データのデータステータスが BAD
UCAERR_PARA_PTRNULL：	ポインタ引数が NULL
UCAERR_OPT_INVALID：	正しくないオプションを指定した
UCAERR_BLKTYPE_INVALID：	正しくないブロック形を指定した

■ 処理内容説明

UcaRWWriteCpv( ) より前に呼び出してください。

■ 引数説明

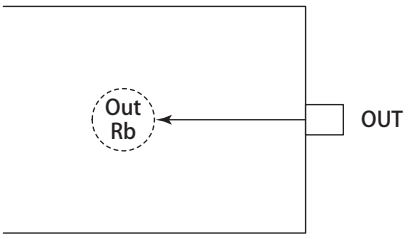
● option

以下の中から指定することができます。

- UCAOPT_RW_SETOOP：出力失敗時に OOP アラームを検出します。
- NOOPTION

■ 詳細説明

UcaRWReadbackCpv のデータの流れは以下のとおりです。



030219J.ai

図　UcaRWReadbackCpvのデータの流れ

UcaRWReadbackCpv は、以下のアラームを検出します。

表　UcaRWReadbackCpvのアラーム検出

オプション / アラーム	NOOPTION	UCAOPT_RW_SETOOP
OOP	×	○
CNF	○	○

○： 検出する
×： 検出しない

警報タブシートのビルダ定義項目［結合状態不良警報］に［なし］を指定した場合、引数 option によらず CNF アラームは検出しません。引数 option に NOOPTION を指定した場合 CNF アラームのみ検出し OOP アラームを検出しませんが、UcaRWReadbackCpv を呼ぶ前に OOP アラームが設定されている場合には、OOP アラーム設定状態を保持します。引数 option に UCAOPT_RW_SETOOP を指定した場合、OOP アラームと CNF アラームを検出します。UcaRWWriteCpv と UcaRWReadbackCpv には、同じオプションを指定してください。UcaRWReadbackCpv が出力の異常を検出している場合、UcaRWWriteCpv は同じ異常を検出しません。

3.2.10 UcaRWCheckOutputCondition

UcaRWCheckOutputConditionは、出力状態を確認します。

■ 関数名

I32 UcaRWCheckOutputCondition

■ 引数

● blockContext

- データ型： UcaBlockContext
- タイプ： IN
- 説明： ブロックコンテキスト

● result

- データ型： BOOL *
- タイプ： OUT
- 説明： 出力トラッキングかどうか

■ 戻り値

- SUCCEED： 正常終了
- UCAERR_PARA_PTRNULL： 引数のポインタが NULL
- UCAERR_BLKTYPE_INVALID： 正しくないブロック形を指定した

■ 処理内容説明

出力状態を確認します。

■ 詳細説明

UcaRWCheckOutputCondition は、出力トラッキング状態かどうかを確認します。UcaRWCheckOutputCondition は、以下の条件が成立した場合、引数 result に TRUE（0 以外）を格納します。条件が成立しない場合は、引数 result に FALSE(0) を格納します。

- 出力タブシートのビルダ定義項目［出力値トラッキング］が［あり］かつ出力読み返し値のデータステータスが CND

補足 CSTM-A ブロックにおいて出力トラッキング動作を行うユーザカスタムアルゴリズムの詳細については、以下の位置にあるサンプルワークスペース _SMPL_ALGO_TRK を参照してください。

<CENTUM VP インストール先 >¥UcaEnv¥Sample¥UcaWork¥UcaSamples¥_SMPL_ALGO_TRK

3.2.11 UcaRWWriteSub

UcaRWWriteSubは、補助出力値をSUB端子へ出力します。

■ 関数名

I32 UcaRWWriteSub

■ 引数

● blockContext

- データ型： UcaBlockContext
- タイプ： IN
- 説明： ブロックコンテキスト

● option

- データ型： U32
- タイプ： IN
- 説明： オプション

■ 戻り値

SUCCEED：	出力成功
UCAERR_RW_NOCONNECT：	結合未定義
UCAERR_RW_LOOPCONN：	端子間入力結合
UCAERR_RW_CONNFAIL：	データアクセスエラー
UCAERR_RW_MNT：	結合先がオンラインメンテナンス中
UCAERR_RW_OS：	結合先機能ブロックのブロックモードが O/S
UCAERR_RW_SWOPEN：	切換スイッチがオープン
UCAERR_RW_OOP：	出力オープン、PI/O 機器異常
UCAERR_RW_PTPF：	出力異常
UCAERR_RW_BADVALUE：	結合先データのデータステータスが BAD
UCAERR_NOTANUMBER：	出力値が非数（Not a Number）である
UCAERR_INFINITY：	出力値の浮動小数データが無限大である
UCAERR_PARA_PTRNULL：	ポインタ引数が NULL
UCAERR_OPT_INVALID：	正しくないオプションを指定した
UCAERR_BLKTYPE_INVALID：	正しくないブロック形を指定した

■ 処理内容説明

SUB 端子から出力を行います。

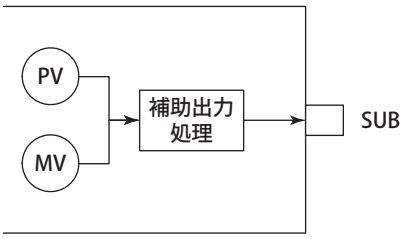
■ 引数説明

● option

NOOPTION を指定してください。

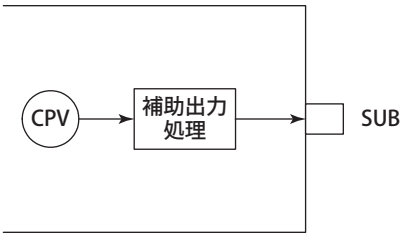
■ 詳細説明

UcaRWWriteSub は、補助出力処理を行い、補助出力値を決定します。補助出力値を SUB 端子へ出力します。



030221J.ai

図　UcaRWWriteSubのデータの流れ（CSTM-Cブロックの場合）



030222J.ai

図　UcaRWWriteSubのデータの流れ（CSTM-Aブロックの場合）

UcaRWWriteSub は、出力タブシートのビルダ定義項目［補助出力］に基づいて、補助出力処理を行います。
UcaRWWriteSubは、アラームを検出しません。補助出力が出力オープンになっても、アラームステータスは OOP（出力オープン警報）になりません。ビルダ定義項目［補助出力］－［出力値］で ΔMV を指定した場合の出力値は、出力補償前出力値（内部変数 Mvbb）です。
CSTM-C ブロックの場合、ビルダ定義項目［補助出力］－［出力値］に PV または ΔPV を指定したときは、UcaRWWriteSub を呼ぶ前にあらかじめ UcaRWSetPv を呼び PV を設定してください。ビルダ定義項目［補助出力］－［出力値］に MV または ΔMV を指定したときは、あらかじめ UcaRWWriteMv を呼び MV を設定してください。
CSTM-A ブロックの場合、UcaRWWriteSub を呼ぶ前にあらかじめ UcaRWSetCpv を呼び CPV を設定してください。

3.2.12 UcaRWReadTrack

UcaRWReadTrackは、TSI端子とTIN端子からデータを読み込み、データアイテムTSWと内部変数Trkに格納します。

■ 関数名

I32 UcaRWReadTrack

■ 引数

● blockContext

- データ型： UcaBlockContext
- タイプ： IN
- 説明： ブロックコンテキスト

● option

- データ型： U32
- タイプ： IN
- 説明： オプション

■ 戻り値

SUCCEED：	入力成功
UCAERR_RW_NOCONNECT：	結合未定義
UCAERR_RW_LOOPCONN：	端子間入力結合
UCAERR_RW_CONNFAIL：	データアクセスエラー
UCAERR_RW_MNT：	結合先がオンラインメンテナンス中
UCAERR_RW_OS：	結合先機能ブロックのブロックモードが O/S
UCAERR_RW_SWOPEN：	切換スイッチがオープン
UCAERR_RW_OOP：	出力オープン、PI/O 機器異常
UCAERR_RW_PTPF：	出力異常
UCAERR_RW_BADVALUE：	結合先データのデータステータスが BAD
UCAERR_PARA_PTRNULL：	ポインタ引数が NULL
UCAERR_OPT_INVALID：	正しくないオプションを指定した
UCAERR_BLKTYPE_INVALID：	正しくないブロック形を指定した

■ 処理内容説明

TSI、TIN 端子の処理を行います。

■ 引数説明

● option

NOOPTION を指定してください。

重要 この関数は、UcaRWReadbackMv( ) よりも前に、呼んでください。

■ 詳細説明

UcaRWReadTrack のデータの流れは以下のとおりです。

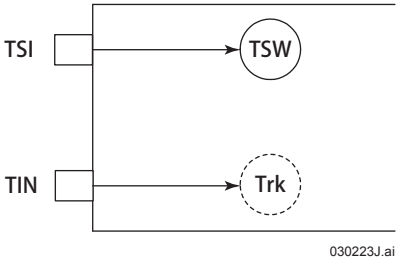


図　UcaRWReadTrackのデータの流れ

UcaRWReadTrack は、アラームを検出しません。
UcaRWReadTrack は、TSI 端子入力失敗時にエラーコードを返します。

3.2.13 UcaRWReadBin

UcaRWReadBinは、BIN端子からデータを読み込み、データアイテムVNに格納します。

■ 関数名

I32 UcaRWReadBin

■ 引数

● blockContext

- データ型： UcaBlockContext
- タイプ： IN
- 説明： ブロックコンテキスト

● option

- データ型： U32
- タイプ： IN
- 説明： オプション

■ 戻り値

SUCCEED：	入力成功
UCAERR_RW_NOCONNECT：	結合未定義
UCAERR_RW_LOOPCONN：	端子間入力結合
UCAERR_RW_CONNFAIL：	データアクセスエラー
UCAERR_RW_MNT：	結合先がオンラインメンテナンス中
UCAERR_RW_OS：	結合先機能ブロックのブロックモードが O/S
UCAERR_RW_SWOPEN：	切換スイッチがオープン
UCAERR_RW_IOPHI：	入力オープン（上限）
UCAERR_RW_IOPLO：	入力オープン（下限）または PI/O 機器異常
UCAERR_RW_BADVALUE：	結合先データのデータステータスが BAD
UCAERR_PARA_PTRNULL：	ポインタ引数が NULL
UCAERR_OPT_INVALID：	正しくないオプションを指定した
UCAERR_BLKTYPE_INVALID：	正しくないブロック形を指定した

■ 処理内容説明

BIN 端子の処理を行います。
入力結果は、プロセスデータ VN に入ります。

■ 引数説明

● option

NOOPTION を指定してください。

■ 詳細説明

UcaRWReadBin のデータの流れは以下のとおりです。

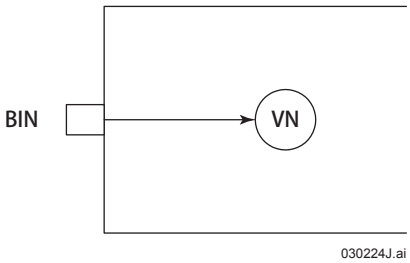


図　UcaRWReadBinのデータの流れ

UcaRWReadBin は、UcaCtrlCompensation の中で自動的に実行されます。入力処理で呼ぶ必要はありません。
VN のデータステータスが BAD のときに MAN フォールバックする場合は、入力処理で UcaRWReadBin を呼び出し、UcaCtrlCompensation のオプションに NOOPTION を指定してください。
UcaRWReadBin は、アラームを検出しません。

3.2.14 UcaRWReadRl

UcaRWReadRlは、RL1端子とRL2端子からデータを読み込み、データアイテムRLV1とデータアイテムRLV2に格納します。

■ 関数名

I32 UcaRWReadRl

■ 引数

● blockContext

- データ型： UcaBlockContext
- タイプ： IN
- 説明： ブロックコンテキスト

● option

- データ型： U32
- タイプ： IN
- 説明： オプション

■ 戻り値

SUCCEED：	入力成功
UCAERR_RW_NOCONNECT：	結合未定義
UCAERR_RW_LOOPCONN：	端子間入力結合
UCAERR_RW_CONNFAIL：	データアクセスエラー
UCAERR_RW_MNT：	結合先がオンラインメンテナンス中
UCAERR_RW_OS：	結合先機能ブロックのブロックモードが O/S
UCAERR_RW_SWOPEN：	切換スイッチがオープン
UCAERR_RW_IOPHI：	入力オープン（上限）
UCAERR_RW_IOPLO：	入力オープン（下限）または PI/O 機器異常
UCAERR_RW_BADVALUE：	結合先データのデータステータスが BAD
UCAERR_PARA_PTRNULL：	ポインタ引数が NULL
UCAERR_OPT_INVALID：	正しくないオプションを指定した
UCAERR_BLKTYPE_INVALID：	正しくないブロック形を指定した

■ 処理内容説明

RL1、RL2 端子の処理を行います。

■ 引数説明

● option

NOOPTION を指定してください。

■ 詳細説明

UcaRWReadRl のデータの流れは以下のとおりです。

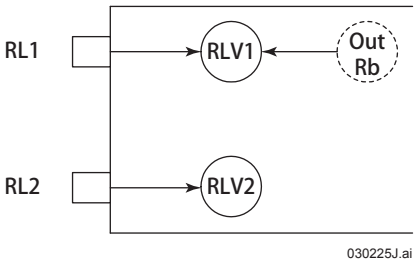


図　UcaRWReadRlのデータの流れ

RL1 端子に結合が定義されていない場合、データアイテム RLV1 に OUT 端子読み返し値（内部変数 OutRb）を格納します。RL2 端子に結合が定義されていない場合、データアイテム RLV2 に 0 を格納します。
UcaRWReadRl は、UcaCtrlHandler の中で自動的に実行されます。入力処理で呼ぶ必要はありません。UcaCtrlHandler を使用しない場合、UcaRWReadRl を呼ぶ前にあらかじめ UcaRWReadbackMv を呼び、出力読み返し値（内部変数 OutRb）を設定してください。
UcaRWReadRl は、アラームを検出しません。
UcaRWReadRl は、RL2 端子入力失敗時にエラーコードを返します。

3.2.15 UcaRWReadInt

UcaRWReadIntは、INT端子からデータを読み込み、引数intPtrに格納します。

■ 関数名

I32 UcaRWReadInt

■ 引数

● blockContext

- データ型： UcaBlockContext
- タイプ： IN
- 説明： ブロックコンテキスト

● intPtr

- データ型： I32S *
- タイプ： OUT
- 説明： 入力結果

● option

- データ型： U32
- タイプ： IN
- 説明： オプション

■ 戻り値

SUCCEED：	入力成功
UCAERR_RW_NOCONNECT：	結合未定義
UCAERR_RW_LOOPCONN：	端子間入力結合
UCAERR_RW_CONNFAIL：	データアクセスエラー
UCAERR_RW_MNT：	結合先がオンラインメンテナンス中
UCAERR_RW_OS：	結合先機能ブロックのブロックモードが O/S
UCAERR_RW_SWOPEN：	切換スイッチがオープン
UCAERR_RW_IOPHI：	入力オープン（上限）
UCAERR_RW_IOPLO：	入力オープン（下限）または PI/O 機器異常
UCAERR_RW_BADVALUE：	結合先データのデータステータスが BAD
UCAERR_PARA_PTRNULL：	ポインタ引数が NULL
UCAERR_OPT_INVALID：	正しくないオプションを指定した
UCAERR_BLKTYPE_INVALID：	正しくないブロック形を指定した

■ 処理内容説明

INT 端子の処理を行います。
入力結果は、intPtr で指された場所に格納します。

■ 引数説明

● option

NOOPTION を指定してください。

■ 詳細説明

UcaRWReadInt のデータの流れは以下のとおりです。

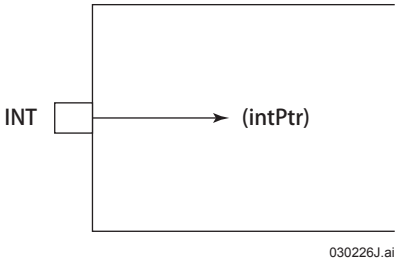


図　UcaRWReadIntのデータの流れ

システムは、UcaCtrlHandler の前処理の中で UcaRWReadInt を実行します。ユーザが記述する入力処理で呼び出す必要はありません。
UcaRWReadInt は、アラームを検出しません。

3.3 シーケンス入出力

ユーザカスタムブロックでは、Qnn端子とJnn端子にシーケンス結合を定義することができます。シーケンス結合は、機能ブロックの入出力端子の結合先として、データを持つ各種素子を指定する結合方式です。入力端子にはデータの状態を判定する条件式を、出力端子には素子の状態操作を行うためのデータを指定します。

参照 シーケンス結合については、以下を参照してください。
機能ブロック共通機能リファレンス（IM 33J15A20-01JA）「2.3 シーケンス結合」

3.3.1 UcaRWSeqCond

UcaRWSeqCondは、Q01～Q32端子からシーケンスデータを読みこみます。

■ 関数名

I32 UcaRWSeqCond

■ 引数

● blockContext

- データ型： UcaBlockContext
- タイプ： IN
- 説明： ブロックコンテキスト

● termNo

- データ型： I32
- タイプ： IN
- 説明： 入力端子番号（0 ～ 32）

● option

- データ型： U32
- タイプ： IN
- 説明： オプション

■ 戻り値

- SUCCEED： 正常終了
- UCAERR_PARA_PTRNULL： ポインタ引数が NULL
- UCAERR_OPT_INVALID： 正しくないオプションを指定した

■ 引数説明

● termNo

結合端子番号を指定します。termNo=0 のときは IN 端子を使用しますが、通常は Q01 ～ Q32 端子を使用してください。

● option

NOOPTION を指定してください。

■ 詳細説明

UcaRWSeqCond のデータの流れは以下のとおりです。

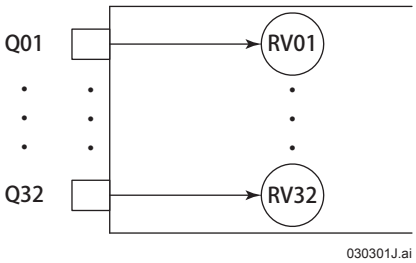


図　UcaRWSeqCondのデータの流れ

UcaRWSeqCond は、Qnn 端子から入力した条件判定結果を RVn に格納します。条件が成立なら 1.0、不成立なら 0.0 を Rvn に格納します。条件判定では、結合先のデータに対して Qnn 端子に指定されている条件式に基づいた判定を行い、条件式の成立または不成立を示す論理値を条件判定結果とします。Qnn 端子が結合未定義の場合、UcaRWSeqCond は Rvn を前回値保持し、SUCCEED を返します。
UcaRWSeqCond は、アラームを検出しません。

3.3.2 UcaRWSeqOprt

UcaRWSeqOprtは、J01～J16端子へシーケンスデータを出力します。

■ 関数名

I32 UcaRWSeqOprt

■ 引数

● blockContext

- データ型： UcaBlockContext
- タイプ： IN
- 説明： ブロックコンテキスト

● termNo

- データ型： I32
- タイプ： IN
- 説明： 出力端子番号（0 ～ 16）

● option

- データ型： U32
- タイプ： IN
- 説明： オプション

■ 戻り値

- SUCCEED： 正常終了
- UCAERR_PARA_PTRNULL： ポインタ引数が NULL
- UCAERR_OPT_INVALID： 正しくないオプションを指定した

■ 引数説明

● termNo

結合端子番号を指定します。termNo=0 のときは OUT 端子を使用しますが、通常は J01 ～ J16 端子を使用してください。

● option

NOOPTION を指定してください。

■ 詳細説明

UcaRWSeqOprt のデータの流れは以下のとおりです。

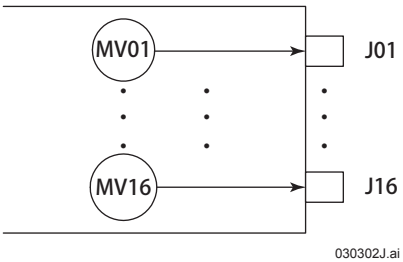


図　UcaRWSeqOprtのデータの流れ（CSTM-Cブロックの場合）

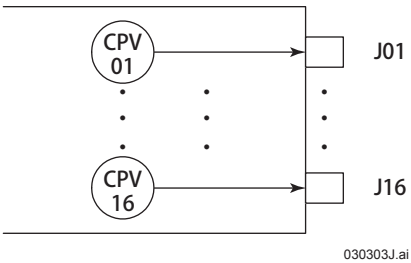


図　UcaRWSeqOprtのデータの流れ（CSTM-Aブロックの場合）

UcaRWSeqOprt は、データアイテム MVn または CPVn のデータ値に従って、Jnn 端子に指定されている結合先状態操作を実行し、結合先の状態を変更します。MVn または CPVn の値が 0.0 でない場合は論理値として真と扱い、状態操作を実行します。MVn または CPVn の値が 0.0 の場合は論理値として偽と扱い、状態操作を実行しません。
UcaRWSeqOprt は、アラームを検出しません。

3.3.3 UcaRWIsSeqCon

UcaRWIsSeqConは、Q01～Q32端子とJ01～J16端子がシーケンス結合かどうかを判定します。

■ 関数名

I32 UcaRWIsSeqCon

■ 引数

● blockContext

- データ型： UcaBlockContext
- タイプ： IN
- 説明： ブロックコンテキスト

● inOrOut

- データ型： U32
- タイプ： IN
- 説明： 入力か出力か

● i

- データ型： I32
- タイプ： IN
- 説明： 端子番号（0 ～ 32）

● result

- データ型： BOOL *
- タイプ： OUT
- 説明： シーケンス結合かどうか

● option

- データ型： U32
- タイプ： IN
- 説明： オプション

■ 戻り値

- SUCCEED
- UCAERR_PARA_PTRNULL： ポインタ引数が NULL
- UCAERR_OPT_INVALID： 正しくないオプションを指定した

■ 処理内容説明

シーケンス結合かどうかを判定します。

■ 引数説明

● i

結合端子番号を指定します。i=0 のときは IN または OUT 端子を使用しますが、通常は Q01 ～ Q32 または J01 ～ J16 端子を使用してください。

● inOrOut

以下の中から 1 つだけ指定することができます。

- UCAOPT_RW_INPUT： 入力端子
- UCAOPT_RW_OUTPUT：出力端子

● option

NOOPTION を指定してください。

■ 詳細説明

シーケンス結合の場合は、引数 result に TRUE を格納します。データ結合の場合は、引数 result に FALSE を格納します。
引数 i=0 の場合、引数 inOrOut に UCAOPT_RW_INPUT を指定すると、IN 端子がシーケンス結合か否かを判定します。引数 i=0 の場合、引数 inOrOut に UCAOPT_RW_OUTPUT を指定すると、OUT 端子がシーケンス結合か否かを判定します。
引数 inOrOut に UCAOPT_RW_INPUT を指定した場合、引数 i に 0 ～ 32 を指定してください。
引数 inOrOut に UCAOPT_RW_OUTPUT を指定した場合、引数 i に 0 ～ 16 を指定してください。
Qnn 端子または Jnn 端子が結合未定義の場合、UcaRWIsSeqCon は引数 result に FALSE を格納し、SUCCEED を返します。

4. 自ブロックデータ

自ブロックデータには、プロセスデータ、データアイテム、内部変数、タグ名などがあります。この章では、自ブロックデータにアクセスするための関数について説明します。

4.1 プロセスデータ

プロセスデータを処理する関数には、偏差処理関数、レンジ変換関数、内部変数を取得する関数などがあります。

4.1.1 UcaDataMeasureTracking

UcaDataMeasureTrackingは、メジャートラッキング処理を行います。

■ 関数名

I32 UcaDataMeasureTracking

■ 引数

● blockContext

- データ型： UcaBlockContext
- タイプ： IN
- 説明： ブロックコンテキスト

● option

- データ型： U32
- タイプ： IN
- 説明： オプション

■ 戻り値

- SUCCEED： 正常終了
- UCAERR_DATA_RANGEOVER： データが範囲外
- UCAERR_PARA_PTRNULL： 引数のポインタが NULL
- UCAERR_OPT_INVALID： 誤ったオプションを指定した
- UCAERR_BLKTYPE_INVALID： 正しくないブロック形を指定した

■ 処理内容説明

メジャートラッキング処理を行います。

■ 引数説明

● option

以下の中から指定することができます。

- UCAOPT_DATA_FORCEMEASTRACK：
  条件によらず、強制的にメジャートラッキングを行います。
- NOOPTION

■ 詳細説明

UcaDataMeasureTracking は、データアイテム PV を F64S 型から F32S 型にデータ型変換し、データアイテム SV、CSV、RSV に代入します。PV のデータがレンジ外の場合は、SV、CSV、RSV への設定値を SH、SL でクリップします。

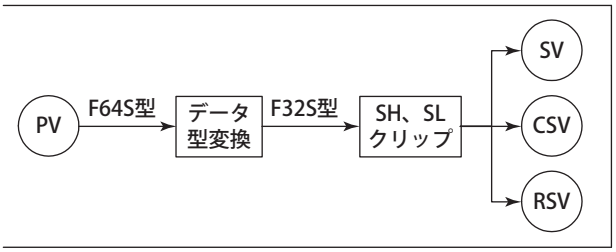


図　UcaDataMeasureTrackingのデータの流れ

UcaDataMeasureTracking は、以下の表の条件が成立する場合に、メジャートラッキングを行います。

表　ビルダ定義項目とブロックモードの関係

ビルダ指定［メジャートラッキング］		ブロックモード
MAN 時	あり	MAN または IMAN MAN
AUT かつ CND 時	あり	IMAN AUT
CAS かつ CND 時	あり	IMAN CAS

UcaDataMeasureTracking は、ビルダ指定によらず、ブロックモードが PRD から AUT、CAS、RCAS に切り換わった直後にも、メジャートラッキングを行います。

参照 動作の詳細については、以下を参照してください。
機能ブロックリファレンス Vol.1（IM 33J15A30-01JA）「1.4 調節ブロックに共通の制御演算処理」の「■測定値トラッキング（メジャートラッキング）」

4.1.2 UcaDataSvPushback

UcaDataSvPushbackは、設定値プッシュバック処理を行います。

■ 関数名

I32 UcaDataSvPushback

■ 引数

● blockContext

- データ型： UcaBlockContext
- タイプ： IN
- 説明： ブロックコンテキスト

● option

- データ型： U32
- タイプ： IN
- 説明： オプション

■ 戻り値

- SUCCEED： 正常終了
- UCAERR_DATA_RANGEOVER： データが範囲外
- UCAERR_PARA_PTRNULL： 引数のポインタが NULL
- UCAERR_OPT_INVALID： 誤ったオプションを指定した
- UCAERR_BLKTYPE_INVALID： 正しくないブロック形を指定した

■ 処理内容説明

設定値プッシュバック処理を行います。

■ 引数説明

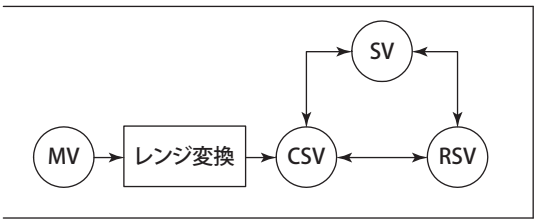
● option

以下の中から 1 つだけ指定することができます。

- UCAOPT_DATA_FORCEPUSHBACK_SV：
  ブロックモードによらず、SV → CSV、RSV プッシュバックします。
- UCAOPT_DATA_FORCEPUSHBACK_CSV：
  ブロックモードによらず、CSV → SV、RSV プッシュバックします。
- UCAOPT_DATA_FORCEPUSHBACK_RSV：
  ブロックモードによらず、RSV → SV、CSV プッシュバックします。
- NOOPTION：
  ブロックモードに応じて設定値プッシュバックします。

■ 詳細説明

UcaDataSvPushback のデータの流れは以下のとおりです。



040103J.ai

図　UcaDataSvPushbackのデータの流れ

UcaDataSvPushback は、ブロックモードやオプション指定に応じて以下の処理を行います。

- SV、CSV、RSV のイコライズ
- SV、CSV、RSV のデータステータス CND をセット／リセット
- 制御演算初期化要求を設定
- MV → SV のデータプッシュバック処理

ブロックモード、オプション指定と各処理の関係を、以下に示します。

表　設定値プッシュバック処理

オプション	ブロックモード	イコライズ	データステータスCND			制御初期化要求の設定条件
			SV	RSV	CSV	
NOOPTION	IMAN、TRK、MAN、ROUT	SV を CSV、RSV に設定	○	○	○	―（制御初期化要求なし）
	AUT	SV を CSV、RSV に設定	×	○	○	SV データステータスが SINT
	CAS	CSV を SV、RSV に設定	○	×	○	CSV データステータスが SINT
	RCAS	RSV を SV、CSV に設定	○	○	×	RSV データステータスが SINT
UCAOPT_DATA_FORCEPUSHBACK_SV	―	SV を CSV、RSV に設定	×	○	○	―（制御初期化要求なし）
UCAOPT_DATA_FORCEPUSHBACK_CSV	―	CSV を SV、RSV に設定	○	×	○	―（制御初期化要求なし）
UCAOPT_DATA_FORCEPUSHBACK_RSV	―	RSV を SV、CSV に設定	○	○	×	―（制御初期化要求なし）

○： セット
×： リセット

オプションに UCAOPT_DATA_FORCEPUSHBACK_SV を指定すると、ブロックモードによらず SV を CSV、RSV に設定します。
オプションに UCAOPT_DATA_FORCEPUSHBACK_CSV を指定すると、ブロックモードによらず CSV を SV、RSV に設定します。
同様に、オプションに UCAOPT_DATA_FORCEPUSHBACK_RSV を指定すると、ブロックモードによらず RSV を SV、CSV に設定します。

UcaDataSvPushback は、以下の場合 MV から SV へのプッシュバック処理を行います。
- ブロックモードが PRD 以外から PRD に変更された場合
- ブロックモードが PRD IMAN または PRD TRK の場合

MV から SV へのプッシュバック処理において以下の条件が成立した場合、UcaDataSvPushback は戻り値 UCAERR_DATA_RANGEOVER を返します。
- データアイテム MSH がデータアイテム MSL よりも小さい
- データアイテム SH がデータアイテム SL よりも小さい

参照 動作の詳細については、以下を参照してください。
機能ブロックリファレンス Vol.1（IM 33J15A30-01JA）「1.4 調節ブロックに共通の制御演算処理」の「■設定値プッシュバック」

4.1.3 UcaDataCheckDv

UcaDataCheckDvは、制御偏差値（データアイテムDV）を作成し、偏差警報処理を行います。

■ 関数名

I32 UcaDataCheckDv

■ 引数

● blockContext

- データ型： UcaBlockContext
- タイプ： IN
- 説明： ブロックコンテキスト

● option

- データ型： U32
- タイプ： IN
- 説明： オプション

■ 戻り値

- SUCCEED： 正常終了
- UCAERR_DATA_RANGEOVER： データが範囲外
- UCAERR_PARA_PTRNULL： 引数のポインタが NULL
- UCAERR_OPT_INVALID： 誤ったオプションを指定した
- UCAERR_BLKTYPE_INVALID： 正しくないブロック形を指定した

■ 処理内容説明

偏差作成処理と偏差警報処理を行います。

■ 引数説明

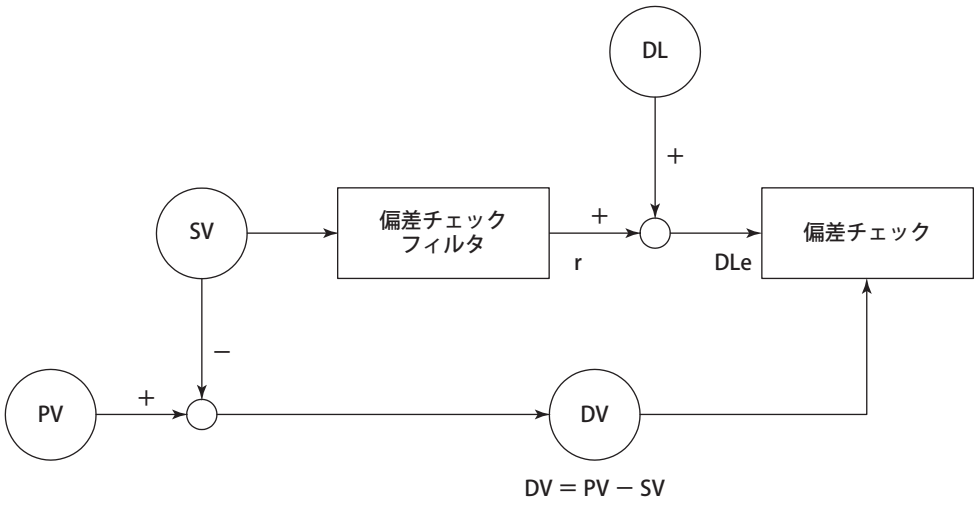
● option

以下の中から指定することができます。

- UCAOPT_DATA_NOCALCDV：DV の計算を行いません。
- NOOPTION

■ 詳細説明

偏差警報処理を行い、偏差アラーム（DV+/DV-）を設定します。



注： UCAOPT_DATA_NOCALCDV オプションを指定すると DV の計算（設定）をしません。

図　UcaDataCheckDvのデータの流れ

UcaDataCheckDv は、測定値（データアイテム PV）と設定値（データアイテム SV）の偏差（DV=PV − SV）を計算し、データアイテム DV に格納します。
UcaDataCheckDv は、偏差警報処理を行います。偏差アラーム設定値（データアイテム DL）を偏差チェックフィルタで補正します。警報タブシートのビルダ定義項目［偏差警報］の指定に従い、以下の表のとおりに偏差警報の検査の方向を決定します。DV が DL を超えている場合、増加方向の変化であれば DV+ アラームの発生を設定します。また減少方向の変化であれば、DV- アラームの発生を設定します。DL が PV スケールスパンと同じ値の場合、偏差アラームの発生を設定しません。DV の絶対値が DL の絶対値からアラームヒステリシス値（HYS）を差し引いた値以下になると、偏差アラームの復帰を設定します。

表　ビルダ定義項目と偏差警報検査方向の関係

ビルダ指定	検査の方向
両方向の検出	正方向と負方向の両方の偏差を監視します
片方向の検出	正方向か負方向のどちらか一方の偏差のみ監視します。 検出したい方向の符号（＋または−）を偏差アラーム設定値（DL）の工業単位記号データに付加します
警報なし	検出を行いません

UcaDataCheckDv は、以下の表の各行の条件を上から下に評価して、最初に該当する行に従って DV のデータステータスを作成します。

表　DVデータステータス作成

PVのデータステータス	SVのデータステータス	作成されるDVのデータステータス
BAD	任意	BAD
任意	BAD	BAD
QST	not BAD	QST
not BAD	QST	QST
not BAD or QST	not BAD or QST	0（正常）

参照 動作の詳細については、以下を参照してください。
機能ブロック共通機能リファレンス（IM 33J15A20-01JA）「5.6 偏差アラームチェック」

オプションに UCAOPT_DATA_NOCALCDV を指定すると、UcaDataCheckDv は制御偏差値の計算とデータステータスの作成を行わず、データアイテム DV を変更しません。ユーザカスタムアルゴリズムにおいて偏差計算処理を記述する場合には、オプションに UCAOPT_DATA_NOCALCDV を指定してください。その場合、UcaDataCheckDv を呼び出す前に UcaDataStoreDv を呼び出し、偏差計算結果をデータアイテム DV にあらかじめ保存してください。

4.1.4 UcaDataConvertRange

UcaDataConvertRangeは、実量化処理を行います。

■ 関数名

I32 UcaDataConvertRange

■ 引数

● blockContext

- データ型： UcaBlockContext
- タイプ： IN
- 説明： ブロックコンテキスト

● dataPvScale

- データ型： F64
- タイプ： IN
- 説明： PV スケール換算データ値

● dataMvScale

- データ型： F64 *
- タイプ： OUT
- 説明： MV スケール換算データ値ポインタ

● option

- データ型： U32
- タイプ： IN
- 説明： オプション

■ 戻り値

- SUCCEED： 正常終了
- UCAERR_RANGE_INVALIDSCALE： スケール上限値／下限値が正しくない
- UCAERR_PARA_PTRNULL： 引数のポインタが NULL
- UCAERR_OPT_INVALID： 誤ったオプションを指定した
- UCAERR_BLKTYPE_INVALID： 正しくないブロック形を指定した

■ 処理内容説明

実量化処理を行います。

■ 引数説明

● dataPvScale

PV 工業単位記号換算出力値

● dataMvScale

MV 工業単位記号換算出力値

● option

以下の中から 1 つだけ指定することができます。

- UCAOPT_RANGE_FULLVALUE：フルバリューのデータを変換します。
- UCAOPT_RANGE_DIFFVALUE：差分値のデータ（ΔMV）を変換します。

■ 詳細説明

UcaDataConvertRange は、引数 dataPvScale に実量化処理を行い、結果を引数 dataMvScale に格納します。

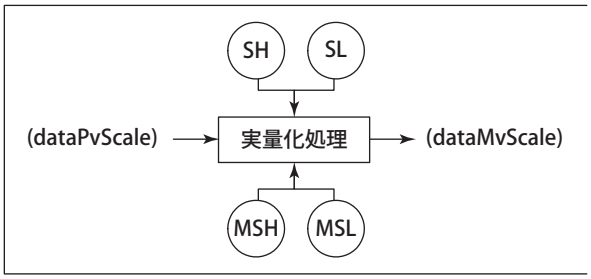


図　UcaDataConvertRangeのデータの流れ

UcaDataConvertRange は、データベースから PV スケール上限値（データアイテム SH）、PV スケール下限値（データアイテム SL）、MV スケール上限値（データアイテム MSH）、MV スケール下限値（データアイテム MSL）を取得します。引数 dataPvScale（PV スケール換算データ値）に対し PV から MV への工業単位記号間の変換を行い、引数 dataMvScale（MV スケール換算データ値）に格納します。
オプションに UCAOPT_RANGE_FULLVALUE を指定すると、オフセットを考慮しフルバリューのデータをレンジ変換します。引数 dataPvScale、データアイテム SH、SL、MSH、MSL から以下のようにゲイン変換計算を行い、引数 dataMvScale に格納します。

*dataMvScale = (dataPvScale − SL) × (MSH − MSL) ／ (SH − SL) + MSL

オプションに UCAOPT_RANGE_DIFFVALUE を指定すると、差分値のデータをレンジ変換します。引数 dataPvScale、データアイテム SH、SL、MSH、MSL から以下のようにゲイン変換計算を行い、引数 dataMvScale に格納します。

*dataMvScale = dataPvScale × (MSH − MSL) ／ (SH − SL)

4.1.5 UcaDataConvertRange_p

UcaDataConvertRange_pは、レンジ変換計算を行います。

■ 関数名

I32 UcaDataConvertRange_p

■ 引数

● blockContext

- データ型： UcaBlockContext
- タイプ： IN
- 説明： ブロックコンテキスト

● preHi

- データ型： F32
- タイプ： IN
- 説明： 変換前スケール上限値

● preLo

- データ型： F32
- タイプ： IN
- 説明： 変換前スケール下限値

● postHi

- データ型： F32
- タイプ： IN
- 説明： 変換後スケール上限値

● postLo

- データ型： F32
- タイプ： IN
- 説明： 変換後スケール下限値

● preData

- データ型： F64
- タイプ： IN
- 説明： 変換前データ値

● postDataPtr

- データ型： F64 *
- タイプ： OUT
- 説明： 変換後データ値ポインタ

● option

- データ型： U32
- タイプ： IN
- 説明： オプション

■ 戻り値

- SUCCEED： 正常終了
- UCAERR_RANGE_INVALIDSCALE： スケール上限値／下限値が正しくない
- UCAERR_PARA_PTRNULL： 引数のポインタが NULL
- UCAERR_OPT_INVALID： 誤ったオプションを指定した

■ 処理内容説明

レンジ変換計算を行います。

■ 引数説明

● preHi

変換前スケール上限値

● preLo

変換前スケール下限値

● postHi

変換後スケール上限値

● postLo

変換後スケール下限値

● preData

変換前データ値

● postDataPtr

変換後データ値ポインタ

● option

以下の中から 1 つだけ指定することができます。

- UCAOPT_RANGE_FULLVALUE：フルバリューのデータを変換します。
- UCAOPT_RANGE_DIFFVALUE：差分値のデータ（ΔMV）を変換します。

■ 詳細説明

UcaDataConvertRange_p のデータの流れは以下のとおりです。

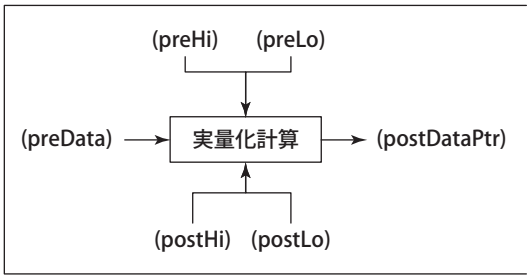


図　UcaDataConvertRange_pのデータの流れ

引数 preHi（変換前の上限値）、preLo（変換前の下限値）、postHi（変換後の上限値）、postLo（変換後の下限値）に基づき、引数 preData に対してレンジ変換計算し、引数 postDataPtr に格納します。
オプションに UCAOPT_RANGE_FULLVALUE を指定すると、オフセットを考慮しフルバリューのデータをレンジ変換します。引数 preData、preHi、preLo、postHi、postLo から以下のようにゲイン変換計算を行い、引数 postDataPtr に格納します。

*postDataPtr = (preData − preLo) × (postHi − postLo) ／ (preHi − preLo) + postLo

オプションに UCAOPT_RANGE_DIFFVALUE を指定すると、差分値のデータをレンジ変換します。引数 preData、preHi、preLo、postHi、postLo から以下のようにゲイン変換計算を行い、引数 postDataPtr に格納します。

*postDataPtr = preData × (postHi − postLo) ／ (preHi − preLo)

4.1.6 UcaDataGetReadback

UcaDataGetReadbackは、出力読み返し値（内部変数OutRb）の値を取得します。

■ 関数名

I32 UcaDataGetReadback

■ 引数

● blockContext

- データ型： UcaBlockContext
- タイプ： IN
- 説明： ブロックコンテキスト

● dataPtr

- データ型： F64S *
- タイプ： OUT
- 説明： 出力読み返し値

● option

- データ型： U32
- タイプ： IN
- 説明： オプション

■ 戻り値

- SUCCEED： 成功
- UCAERR_DATA_INVALIDDATA： データが無効
- UCAERR_PARA_PTRNULL： 引数のポインタが NULL
- UCAERR_OPT_INVALID： 誤ったオプションを指定した

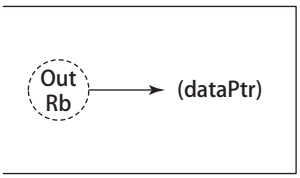
■ 引数説明

● option

NOOPTION を指定してください。

■ 詳細説明

UcaDataGetReadback は、出力読み返し値（内部変数 OutRb）の値を取得し、引数 dataPtr に格納します。



040110J.ai

図　UcaDataGetReadbackのデータの流れ

システムは UcaCtrlHandler の前処理において、OUT 端子からデータを読み返し、内部変数 OutRb に格納します。
UcaCtrlHandler を使用しない場合は、UcaDataGetReadback を呼び出す前に以下の関数を呼び出し、OUT 端子からデータを読み返してください。

- CSTM-C ブロックの場合：UcaRWReadbackMv
- CSTM-A ブロックの場合：UcaRWReadbackCpv

OUT 端子読み返し処理が行われる前に UcaDataGetReadback を呼び出すと、内部変数 OutRb にはまだ値が格納されていないため、データを取得できません。この場合、UcaDataGetReadback は、以下の値を引数 dataPtr に格納し、エラーコード UCAERR_DATA_INVALIDDATA を返します。

- CSTM-C ブロックの場合：データアイテム MV
- CSTM-A ブロックの場合：データアイテム CPV

4.1.7 UcaDataGetTrack

UcaDataGetTrackは、トラッキング入力値（内部変数Trk）の値を取得します。

■ 関数名

I32 UcaDataGetTrack

■ 引数

● blockContext

- データ型： UcaBlockContext
- タイプ： IN
- 説明： ブロックコンテキスト

● dataPtr

- データ型： F64S *
- タイプ： OUT
- 説明： TIN 端子読み返し値

● option

- データ型： U32
- タイプ： IN
- 説明： オプション

■ 戻り値

- SUCCEED： 成功
- UCAERR_DATA_INVALIDDATA： データが無効
- UCAERR_PARA_PTRNULL： 引数のポインタが NULL
- UCAERR_OPT_INVALID： 誤ったオプションを指定した
- UCAERR_BLKTYPE_INVALID： 正しくないブロック形を指定した

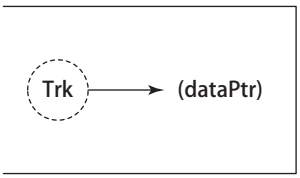
■ 引数説明

● option

NOOPTION を指定してください。

■ 詳細説明

UcaDataGetTrack は、トラッキング入力値（内部変数 Trk）の値を取得し、引数 dataPtr に格納します。



040111J.ai

図　UcaDataGetTrackのデータの流れ

システムは UcaCtrlHandler の前処理において、TIN 端子からデータを読み込み、内部変数 Trk に格納します。
UcaCtrlHandler を使用しない場合は、UcaDataGetTrack を呼び出す前に UcaRWReadTrack 関数を呼び出し、TIN 端子読み込み処理を行ってください。
TIN 端子読み込み処理が行われる前に UcaDataGetTrack を呼び出すと、内部変数 Trk にはまだ値が格納されていないため、データを取得できません。この場合、UcaDataGetTrack はデータアイテム MV の値を引数 dataPtr に格納し、エラーコード UCAERR_DATA_INVALIDDATA を返します。

4.1.8 UcaDataGetMvbb

UcaDataGetMvbbは、出力補償前出力値（内部変数Mvbb）の値を取得します。

■ 関数名

I32 UcaDataGetMvbb

■ 引数

● blockContext

- データ型： UcaBlockContext
- タイプ： IN
- 説明： ブロックコンテキスト

● dataPtr

- データ型： F64 *
- タイプ： OUT
- 説明： 前回出力補償前出力値

● option

- データ型： U32
- タイプ： IN
- 説明： オプション

■ 戻り値

- SUCCEED： 成功
- UCAERR_PARA_PTRNULL： 引数のポインタが NULL
- UCAERR_OPT_INVALID： 誤ったオプションを指定した
- UCAERR_BLKTYPE_INVALID： 正しくないブロック形を指定した

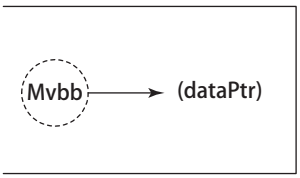
■ 引数説明

● option

NOOPTION を指定してください。

■ 詳細説明

UcaDataGetMvbb は、出力補償前出力値（内部変数 Mvbb）の値を取得し、引数 dataPtr に格納します。



040112J.ai

図　UcaDataGetMvbbのデータの流れ

システムは UcaCtrlHandler の出力値作成処理において、ユーザ記述の制御演算処理関数が引数に返した出力補償前出力値を元に作成した出力値を、内部変数 Mvbb に格納します。

参照 UcaCtrlHandler の出力値作成処理については、以下を参照してください。
APCS ユーザカスタムブロック プログラミング ガイド（IM 33J15U21-01JA）「6.2.6 出力値作成処理（UcaCtrlHandler の内部で処理）」

UcaCtrlHandler の出力値作成処理が行われる前に UcaDataGetMvbb を呼び出すと、引数 dataPtr には前回の出力補償前出力値（内部変数 Mvbb）を格納します。

4.1.9 UcaDataGetMvbl

UcaDataGetMvblは、出力リミット前出力値（内部変数Mvbl）の値を取得します。

■ 関数名

I32 UcaDataGetMvbl

■ 引数

● blockContext

- データ型： UcaBlockContext
- タイプ： IN
- 説明： ブロックコンテキスト

● dataPtr

- データ型： F64 *
- タイプ： OUT
- 説明： 前回出力補償後出力値

● option

- データ型： U32
- タイプ： IN
- 説明： オプション

■ 戻り値

- SUCCEED： 成功
- UCAERR_PARA_PTRNULL： 引数のポインタが NULL
- UCAERR_OPT_INVALID： 誤ったオプションを指定した
- UCAERR_BLKTYPE_INVALID： 正しくないブロック形を指定した

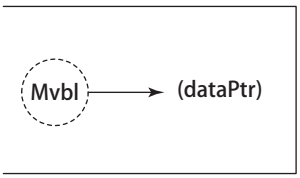
■ 引数説明

● option

NOOPTION を指定してください。

■ 詳細説明

UcaDataGetMvbl は、出力リミット前出力値（内部変数 Mvbl）の値を取得し、引数 dataPtr に格納します。



040113J.ai

図　UcaDataGetMvblのデータの流れ

システムは UcaCtrlHandler の出力値作成処理において、ユーザ記述の制御演算処理関数が引数に返した出力リミット前出力値を元に作成した出力値を、内部変数 Mvbl に格納します。

参照 UcaCtrlHandler の出力値作成処理については、以下を参照してください。
APCS ユーザカスタムブロック プログラミングガイド（IM 33J15U21-01JA）「6.2.6 出力値作成処理（UcaCtrlHandler の内部で処理）」

UcaCtrlHandler の出力値作成処理が行われる前に UcaDataGetMvbl を呼び出すと、引数 dataPtr には前回の出力リミット前出力値（内部変数 Mvbl）を格納します。

4.2 その他

その他の関数には、タグ名を取得する関数、エラーコードを設定する関数があります。

4.2.1 UcaDataGetTagName

UcaDataGetTagNameは、自ブロックのタグ名を取得します。

■ 関数名

I32 UcaDataGetTagName

■ 引数

● blockContext

- データ型： UcaBlockContext
- タイプ： IN
- 説明： ブロックコンテキスト

● dataPtr

- データ型： BYTE *
- タイプ： OUT
- 説明： タグ名

● option

- データ型： U32
- タイプ： IN
- 説明： オプション

■ 戻り値

- SUCCEED： 成功
- UCAERR_PARA_PTRNULL： 引数のポインタが NULL
- UCAERR_OPT_INVALID： 誤ったオプションを指定した

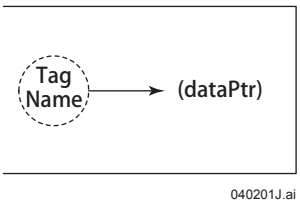
■ 引数説明

● option

NOOPTION を指定してください。

■ 詳細説明

UcaDataGetTagName は、自ブロックのタグ名を dataPtr が指す領域に格納します。



040201J.ai

図　UcaDataGetTagNameのデータの流れ

重要 UcaDataGetTagName はタグ名の末尾に、'¥0' を付加します。タグ名を読み込む領域には、タグ名の最大サイズ 16 バイトに、'¥0' の分 1 バイトを加え、17 バイト以上を確保してください。

4.2.2 UcaDataStoreErrorNumber

UcaDataStoreErrorNumberは、データアイテムERRC、ERRLに値を設定します。

■ 関数名

I32 UcaDataStoreErrorNumber

■ 引数

● blockContext

- データ型： UcaBlockContext
- タイプ： IN
- 説明： ブロックコンテキスト

● errc

- データ型： I32
- タイプ： IN
- 説明： ERRC

● errl

- データ型： I32
- タイプ： IN
- 説明： ERRL

● option

- データ型： U32
- タイプ： IN
- 説明： オプション

■ 戻り値

- SUCCEED： 正常終了
- UCAERR_PARA_PTRNULL： 引数のポインタが NULL
- UCAERR_OPT_INVALID： 誤ったオプションを指定した

■ 処理内容説明

ERRC、ERRL を格納します。

■ 引数説明

● option

NOOPTION を指定してください。

■ 詳細説明

UcaDataStoreErrorNumber は、引数 errc の値をデータアイテム ERRC に設定します。引数 errl の値をデータアイテム ERRL に設定します。

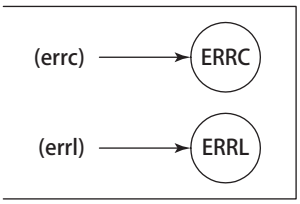


図　UcaDataStoreErrorNumberのデータの流れ

データアイテム ERRC、ERRL は、ユーザカスタムアルゴリズムのプログラマが、障害解析のために設定し使用します。

参照 使用方法の詳細については、以下を参照してください。
APCS ユーザカスタムブロック プログラミングガイド（IM 33J15U21-01JA）「5.2.1 入力処理と UcaFpuExpCheck による演算エラーの検出」

4.3 自ブロックデータアイテム

自ブロックのデータアイテムを読み書きするための関数が用意されています。自ブロックのデータアイテムは、PV、MV、CPVなどの単一形と、RV01～RV32やS01～S16などの番号付きに分けることができます。

4.3.1 UcaDataGet***/UcaDataStore***（単一形）

PV、MV、CPVなどの単一形データアイテムの参照／保存関数は、以下のようなインタフェースを持ちます。

- 関数名のデータ名の部分は、データアイテム名の先頭1文字だけ大文字であとは小文字です。
- 第2引数のデータ型は、データアイテムのデータ型に一致します。

■ 関数名

I32 UcaDataGet データアイテム名

■ 引数

● blockContext

- データ型： UcaBlockContext
- タイプ： IN
- 説明： ブロックコンテキスト

● data

- データ型： データ型 *
- タイプ： OUT
- 説明： データ格納ポインタ

● option

- データ型： U32
- タイプ： IN
- 説明： オプション

■ 戻り値

- SUCCEED： 正常終了
- UCAERR_PARA_PTRNULL： 引数のポインタが NULL
- UCAERR_OPT_INVALID： 誤ったオプションを指定した
- UCAERR_BLKTYPE_INVALID： 正しくないブロック形を指定した

■ 処理内容説明

データアイテムからプロセスデータを取得します。

■ 引数説明

● option

NOOPTION を指定してください。

■ 詳細説明

UcaDataGet*** は、データアイテムの値を引数 data に格納します。

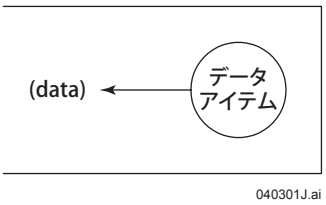


図　UcaDataGet*のデータの流れ**

UcaDataGet*** は、データ型が文字列の場合、文字列の末尾に '¥0' を付加します。該当するデータアイテムは PRGN のみです。

■ 関数名

I32 UcaDataStore データアイテム名

■ 引数

● blockContext

- データ型：　UcaBlockContext
- タイプ：　IN
- 説明：　ブロックコンテキスト

● data

- データ型：　データ型 *
- タイプ：　IN
- 説明：　データ格納ポインタ

● option

- データ型：　U32
- タイプ：　IN
- 説明：　オプション

■ 戻り値

- SUCCEED：　正常終了
- UCAERR_PARA_PTRNULL：　引数のポインタが NULL
- UCAERR_OPT_INVALID：　誤ったオプションを指定した
- UCAERR_BLKTYPE_INVALID：　正しくないブロック形を指定した
- UCAERR_NOTANUMBER：
  引数の浮動小数データが非数（Not a Number）である
- UCAERR_INFINITY：　引数の浮動小数データが無限大である
- UCAERR_DATA_RANGEOVER：　データが範囲外

■ 処理内容説明

プロセスデータをデータアイテムへ格納します。

■ 引数説明

● option

以下の中から複数指定することができます。複数指定する場合は、それぞれを‘|’(or)で結んでください。

- UCAOPT_DATA_LIMIT：SH/SL でリミットします（PV のみ有効）。
- UCAOPT_DATA_NOSTATUS：
  引数のデータステータスを、データアイテムに反映しません（データステータス付きデータのみ有効）。
- NOOPTION

■ 詳細説明

UcaDataStore*** は、引数 data の値をデータアイテムに格納します。

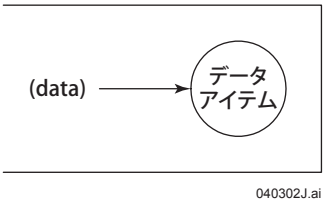


図　UcaDataStore*のデータの流れ**

UcaDataStore*** は、データ型が浮動小数点データ型（F32、F64、F32S、F64S、F32SR）の場合、引数 data の値が非数（Not a Number）または無限大であると、データ値を保存せず、データステータスに BAD を設定します。非数（Not a Number）の場合エラーコード UCAERR_NOTANUMBER を、無限大の場合エラーコード UCAERR_INFINITY を戻り値として返します。

4.3.2 UcaDataGet***n/UcaDataStore***n（番号付き）

RV01～RV32などの番号付きデータの参照／保存関数は、以下のようなインタフェースを持ちます。

- 関数名のデータ名の部分は、データアイテム名の先頭1文字だけ大文字であとは小文字にしたものに、小文字の'n'を付加します。
  例：
  RV01 ～ RV32 参照関数は UcaDataGetRvn( )、P01 ～ P32 保存関数は UcaDataStorePn( )
- 第2引数のデータ型は、データアイテムのデータ型に一致します。
- 第3引数で指定したデータ番号から、第4引数で指定したデータ個数だけ、第2引数で指示された場所に格納します。第2引数には、データ個数分の配列を指定してください。

■ 関数名

I32 UcaDataGet データアイテム名 n

■ 引数

● blockContext

- データ型： UcaBlockContext
- タイプ： IN
- 説明： ブロックコンテキスト

● data

- データ型： データ型 *
- タイプ： OUT
- 説明： データ格納ポインタ

● termNo

- データ型： I32
- タイプ： IN
- 説明： 開始データ番号

● dataNo

- データ型： I32
- タイプ： IN
- 説明： データ数

● option

- データ型： U32
- タイプ： IN
- 説明： オプション

■ 戻り値

- SUCCEED： 正常終了
- UCAERR_PARA_PTRNULL： 引数のポインタが NULL
- UCAERR_OPT_INVALID： 誤ったオプションを指定した
- UCAERR_BLKTYPE_INVALID： 正しくないブロック形を指定した
- UCAERR_DATA_INVALIDNO： データ番号／データ数が正しくない

■ 処理内容説明

データアイテムからプロセスデータを取得します。

■ 引数説明

● option

NOOPTION を指定してください。

■ 詳細説明

UcaDataGet***n は、データアイテムの値を、引数 termNo の番号から引数 dataNo の個数だけ、引数 data に格納します。

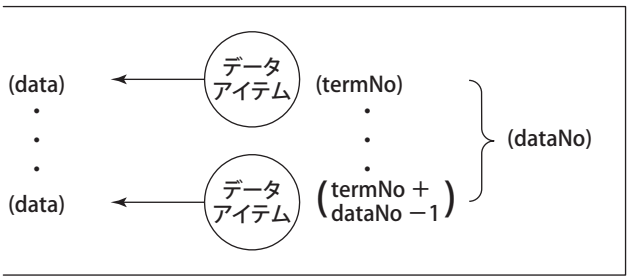


図　UcaDataGet*nのデータの流れ**

■ 関数名

I32 UcaDataStore データアイテム名 n

■ 引数

● blockContext

- データ型： UcaBlockContext
- タイプ： IN
- 説明： ブロックコンテキスト

● rvnn

- データ型： データ型 *
- タイプ： IN
- 説明： データ格納ポインタ

● termNo

- データ型： I32
- タイプ： IN
- 説明： 開始データ番号

● dataNo

- データ型： I32
- タイプ： IN
- 説明： データ数

● option

- データ型： U32
- タイプ： IN
- 説明： オプション

■ 戻り値

- SUCCEED： 正常終了
- UCAERR_PARA_PTRNULL： 引数のポインタが NULL
- UCAERR_OPT_INVALID： 誤ったオプションを指定した
- UCAERR_BLKTYPE_INVALID： 正しくないブロック形を指定した
- UCAERR_NOTANUMBER：
  引数の浮動小数データが非数（Not a Number）である
- UCAERR_INFINITY： 引数の浮動小数データが無限大である
- UCAERR_DATA_INVALIDNO： データ番号／データ数が正しくない

■ 処理内容説明

プロセスデータをデータアイテムへ格納します。

■ 引数説明

● option

NOOPTION を指定してください。

■ 詳細説明

UcaDataStore***n は、引数 data の値を、引数 termNo の番号から引数 dataNo の個数だけ、データアイテムに格納します。

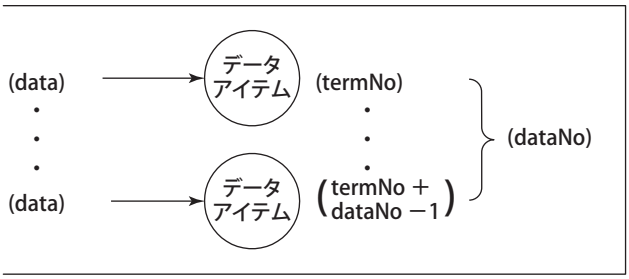


図　UcaDataStore*nのデータの流れ**

UcaDataStore***n は、データ型が浮動小数点データ型（F32、F64、F32S、F64S、F32SR）の場合、引数 data の値が非数（Not a Number）または無限大であると、データ値を保存せず、データステータスに BAD を設定します。引数 dataNo に 1 以上の値を指定し複数のデータを保存する場合、非数または無限大のデータのみ保存を行わず、正常値のデータは保存を行います。1 つでも非数または無限大のデータが存在すると、非数（Not a Number）の場合エラーコード UCAERR_NOTANUMBER を、無限大の場合エラーコード UCAERR_INFINITY を戻り値として返します。非数と無限大のデータの両方が存在する場合は、データ番号の大きいデータのエラーコードを戻り値として返します。

4.3.3 データアイテム参照／保存関数一覧

本項では、データアイテム参照／保存関数一覧を説明します。

■ データアイテム参照／保存関数一覧

以下に、データアイテム参照／保存関数の一覧を示します。

- Get の欄に書いてあるのが、UcaDataGet*** 関数名です。
- Store の欄に書いてあるのが、UcaDataStore*** 関数名です。
- 番号付の欄に○がついていないものは、単一形のインタフェースが適用されます。
- 番号付の欄に○がついているものは、番号付きのインタフェースが適用されます。
- 関数名が、UcaDataGet***、UcaDataStore*** 以外の関数は、各データアイテム専用です。

表　自ブロックデータ参照／保存関数（1/4）

シンボル	データ名	データ型	番号付	専用	Get	Store	CSTM-C	CSTM-A
MODE	ブロックモード	U32		○	UcaModeGet	UcaModeSet	○	○
ALRM	アラームステータス	U32		○	UcaAlrmGet	UcaAlrmSet	○	○
AFLS	アラームフラッシングステータス	U32			UcaDataGetAfls	なし	○	○
AF	アラーム検出指定	U32			UcaDataGetAf	なし	○	○
AOFS	アラーム抑制指定	U32			UcaDataGetAofs	なし	○	○
PV	測定値	F64S			UcaDataGetPv	UcaDataStorePv UcaRWSetPv	○	
RAW	生入力データ	IND			なし	なし	○	○
PVP	変化率警報サンプリング最旧値	F32			UcaDataGetPvp	なし	○	
CPV	演算出力値	F64S			UcaDataGetCpv	UcaDataStoreCpv UcaRWSetCpv		○
SUM	積算値	F64S			UcaDataGetSum	UcaDataStoreSum	○	○
CPV01 ～ CPV16	演算出力値 1 ～ 16	F64S	○		UcaDataGetCpvn	UcaDataStoreCpvn		○
RSV	リモート設定値	F32S			UcaDataGetRsv	UcaDataStoreRsv	○	
DV	制御偏差値	F32S			UcaDataGetDv	UcaDataStoreDv	○	
VN	入出力補償値	F32S			UcaDataGetVn	UcaDataStoreVn	○	
MV	操作出力値	F32S			UcaDataGetMv	UcaDataStoreMv	○	
RMV	リモート操作出力値	F32S			UcaDataGetRmv	UcaDataStoreRmv	○	
RLV1	リセット信号値 1	F32S			UcaDataGetRlv1	UcaDataStoreRlv1	○	
RLV2	リセット信号値 2	F32S			UcaDataGetRlv2	UcaDataStoreRlv2	○	
HH	上上限警報設定値	F32			UcaDataGetHh	UcaDataStoreHn	○	
LL	下下限警報設定値	F32			UcaDataGetLl	UcaDataStoreLl	○	
PH	上限警報設定値	F32			UcaDataGetPh	UcaDataStorePh	○	
PL	下限警報設定値	F32			UcaDataGetPl	UcaDataStorePl	○	
VL	変化率警報設定値	F32			UcaDataGetVl	UcaDataStoreVl	○	
DL	偏差警報設定値	F32			UcaDataGetDl	UcaDataStoreDl	○	
MH	操作出力上限設定値	F32			UcaDataGetMh	UcaDataStoreMh	○	
ML	操作出力下限設定値	F32			UcaDataGetMl	UcaDataStoreNl	○	
SVH	設定上限設定値	F32			UcaDataGetSvh	UcaDataStoreSvh	○	
シンボル	**データ名**	**データ型**	**番号付**	**専用**	**Get**	**Store**	**CSTM-C**	**CSTM-A**

表　自ブロックデータ参照／保存関数（2/4）

シンボル	データ名	データ型	番号付	専用	Get	Store	CSTM-C	CSTM-A
SVL	設定下限設定値	F32			UcaDataGetSvl	UcaDataStoreSvl	○	
P	比例帯	F32			UcaDataGetP	UcaDataStoreP	○	
I	積分時間	F32			UcaDataGetI	UcaDataStoreI	○	
D	微分時間	F32			UcaDataGetD	UcaDataStoreD	○	
GW	ギャップ幅	F32			UcaDataGetGw	UcaDataStoreGw	○	
DB	不感帯	F32			UcaDataGetDb	UcaDataStoreDb	○	
CK	補償ゲイン	F32			UcaDataGetCk	UcaDataStoreCk	○	
CB	補償バイアス	F32			UcaDataGetCb	UcaDataStoreCb	○	
PMV	プリセット操作出力値	F32			UcaDataGetPmv	UcaDataStorePmv	○	
TSW	トラッキングスイッチ	I16S			UcaDataGetTsw	UcaDataStoreTsw	○	
CSW	制御スイッチ	I16			UcaDataGetCsw	UcaDataStoreCsw	○	
PSW	プリセット MV スイッチ	I16			UcaDataGetPsw	UcaDataStorePsw	○	
RSW	パルスリセットスイッチ	I16			UcaDataGetRsw	UcaDataStoreRsw	○	
BSW	バックアップスイッチ	I16			UcaDataGetBsw	UcaDataStoreBsw	○	
OPHI	出力上限置針	F32			UcaDataGetOphi	UcaDataStoreOphi	○	
OPLO	出力下限置針	F32			UcaDataGetOplo	UcaDataStoreOplo	○	
OPMK	オペマーク	I16			UcaDataGetOpmk	UcaDataStoreOpmk	○	○
SAID	システムアプリケーション ID	I16			UcaDataGetSaid	UcaDataStoreSaid	○	○
UAID	ユーザアプリケーション ID	I16			UcaDataGetUaid	UcaDataStoreUaid	○	○
TYPE	機能ブロック形名	U16			UcaDataGetType	なし	○	○
OMOD	ブロックモード（最低優先順位）	U32			なし	なし	—	—
CMOD	ブロックモード（最高優先順位）	U32			なし	なし	—	—
SH	スケール上限値	F32			UcaDataGetSh	なし	○	○
SL	スケール下限値	F32			UcaDataGetSl	なし	○	○
MSH	MV スケール上限値	F32			UcaDataGetMsh	なし	○	
MSL	MV スケール下限値	F32			UcaDataGetMsl	なし	○	
DR	制御動作方向	I16			UcaDataGetDr	なし	○	
RAWRL1	生データ（RL1）	IND			なし	なし	○	
RAWRL2	生データ（RL2）	IND			なし	なし	○	
RAWBIN	生データ（BIN）	IND			なし	なし	○	
RAWTIN	生データ（TIN）	IND			なし	なし	○	
RAWTSI	生データ（TSI）	IND			なし	なし	○	
RAWINT	生データ（INT）	IND			なし	なし	○	
BSTS	ブロックステータス	U32		○	UcaBstsGet	UcaBstsSet	○	○
RV	入力値	F64S			UcaDataGetRv	UcaDataStoreRv	○	○
RV01 ～ RV32	副入力データ	F64S	○		UcaDataGetRvn	UcaDataStoreRvn	○	○
RAW01 ～ RAW32	生入力データ	IND			なし	なし	—	—
MV01 ～ MV16	副出力データ	F32S	○		UcaDataGetMvn	UcaDataStoreF64SToMvn UcaDataStoreMvn	○	
シンボル	データ名	データ型	番号付	専用	Get	Store	CSTM-C	CSTM-A

表　自ブロックデータ参照／保存関数（3/4）

シンボル	データ名	データ型	番号付	専用	Get	Store	CSTM-C	CSTM-A
P01 ～ P32	パラメータ	F64S	○		UcaDataGetPn	UcaDataStorePn	○	○
S01 ～ S16	文字列パラメータ	CHR		○	UcaDataGetEachSn	UcaDataStoreEachSn	○	○
I01 ～ I08	整数パラメータ	I32	○		UcaDataGetIn	UcaDataStoreIn	○	○
CALC	演算結果	F64S			UcaDataGetCalc	UcaDataStoreCalc	○	
ERRL	エラー発生個所	I32		○	UcaDataGetErrl	UcaDataStoreError	○	○
ERRC	エラーコード	I32		○	UcaDataGetErrc	Number	○	○
ERRA	エラーコード（自動設定）	I32			UcaDataGetErra	なし	○	○
TRSW	実行時間リセット	I16			なし	なし	—	—
TSC	実効スキャン周期	I32			UcaDataGetTsc	なし	—	—
PRGN (*1)	プログラム名称	CHR			UcaDataGetPrgn	なし	—	—
VERS	プログラムバージョン番号	CHR			なし	なし	—	—
PRGS	実行管理情報	CHR			なし	なし	—	—
ICON	初期化処理起動回数	I32			なし	なし	—	—
ICER	初期化処理起動エラー回数	I32			なし	なし	—	—
ILER	初期化処理起動前回エラーコード	I32			なし	なし	—	—
ITIM	初期化処理起動前回実行時間	I32			なし	なし	—	—
IMIN	初期化処理最小実行時間	I32			なし	なし	—	—
IMAX	初期化処理最大実行時間	I32			なし	なし	—	—
IAVE	初期化処理平均実行時間	I32			なし	なし	—	—
ITOT	初期化処理通算実行時間	I32			なし	なし	—	—
FCON	終了処理起動回数	I32			なし	なし	—	—
FCER	終了処理起動エラー回数	I32			なし	なし	—	—
FLER	終了処理起動前回エラーコード	I32			なし	なし	—	—
FTIM	終了処理前回実行時間	I32			なし	なし	—	—
FMIN	終了処理最小実行時間	I32			なし	なし	—	—
FMAX	終了処理最大実行時間	I32			なし	なし	—	—
FAVE	終了処理平均実行時間	I32			なし	なし	—	—
FTOT	終了処理通算実行時間	I32			なし	なし	—	—
PCON	定周期起動回数	I32			なし	なし	—	—
PCER	定周期処理エラー回数	I32			なし	なし	—	—
PLER	定周期起動前回エラーコード	I32			なし	なし	—	—
シンボル	**データ名**	**データ型**	**番号付**	**専用**	**Get**	**Store**	**CSTM-C**	**CSTM-A**

*1： UcaDataGetPrgn はデータアイテム PRGN に保持されているユーザカスタムアルゴリズムのプログラム名称を返します。UcaDataGetPrgn はプログラム名称の末尾に、'¥0' を付加します。

表　自ブロックデータ参照／保存関数（4/4）

シンボル	データ名	データ型	番号付	専用	Get	Store	CSTM-C	CSTM-A
PTIM	定周期起動前回実行時間	I32			なし	なし	—	—
PMIN	定周期起動最小実行時間	I32			なし	なし	—	—
PMAX	定周期起動最大実行時間	I32			なし	なし	—	—
PAVE	定周期起動平均実行時間	I32			なし	なし	—	—
PTOT	定周期起動通算実行時間	I32			なし	なし	—	—
OCON	ワンショット起動回数	I32			なし	なし	—	—
OCER	ワンショット起動エラー回数	I32			なし	なし	—	—
OLER	ワンショット起動前回エラーコード	I32			なし	なし	—	—
OTIM	ワンショット起動前回実行時間	I32			なし	なし	—	—
OMIN	ワンショット起動最小実行時間	I32			なし	なし	—	—
OMAX	ワンショット起動最大実行時間	I32			なし	なし	—	—
OAVE	ワンショット起動平均実行時間	I32			なし	なし	—	—
OTOT	ワンショット起動通算実行時間	I32			なし	なし	—	—
DCON	データ設定時特殊処理回数	I32			なし	なし	—	—
DCER	データ設定時特殊処理エラー回数	I32			なし	なし	—	—
DLER	データ設定時特殊処理前回エラーコード	I32			なし	なし	—	—
DTIM	データ設定時特殊処理前回実行時間	I32			なし	なし	—	—
DMIN	データ設定時特殊処理最小実行時間	I32			なし	なし	—	—
DMAX	データ設定時特殊処理最大実行時間	I32			なし	なし	—	—
DAVE	データ設定時特殊処理平均実行時間	I32			なし	なし	—	—
DTOT	データ設定時特殊処理通算実行時間	I32			なし	なし	—	—

重要 プログラム名称を読み込む領域には、プログラム名称の最大サイズ 16 バイトに、'¥0' の分 1 バイトを加え、17 バイト以上を確保してください。

4.3.4 UcaDataGetEachSn

データアイテムS01～S16は、文字列型です。このため数値型の番号付きデータ参照関数UcaDataGet*nと異なるインタフェースを持つ専用の関数UcaDataGetEachSnを用意しました。**

■ 関数名

I32 UcaDataGetEachSn

■ 引数

● blockContext

- データ型： UcaBlockContext
- タイプ： IN
- 説明： ブロックコンテキスト

● string

- データ型： BYTE *
- タイプ： OUT
- 説明： データ格納ポインタ

● termNo

- データ型： I32
- タイプ： IN
- 説明： S01 ～ S16 の番号（1 ～ 16）

● option

- データ型： U32
- タイプ： IN
- 説明： オプション

■ 戻り値

- SUCCEED： 正常終了
- UCAERR_PARA_PTRNULL： 引数のポインタが NULL
- UCAERR_OPT_INVALID： 誤ったオプションを指定した
- UCAERR_DATA_INVALIDNO： データ番号／データ数が正しくない

■ 処理内容説明

データアイテム S01 ～ S16 からプロセスデータを取得します。

■ 引数説明

● option

NOOPTION を指定してください。

■ 詳細説明

UcaDataGetEachSn は、string が指す領域に、自ブロックのデータアイテム S01 ～ S16（termNo に指定された番号のデータアイテム 1 つ）の値を返します。

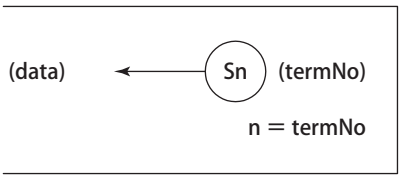


図　UcaDataGetEachSnのデータの流れ

重要 UcaDataGetEachSn は文字列の末尾に、'¥0' を付加します。データアイテム Sn を読み込む領域には、データアイテム Sn の最大サイズ 16 バイトに、'¥0' の分 1 バイトを加え、17 バイト以上を確保してください。

4.3.5 UcaDataStoreEachSn

データアイテムS01～S16は、文字列型です。このため数値型の番号付きデータ保存関数UcaDataGet*nと異なるインタフェースを持つ専用の関数UcaDataStoreEachSnを用意しました。**

■ 関数名

I32 UcaDataStoreEachSn

■ 引数

● blockContext

- データ型： UcaBlockContext
- タイプ： IN
- 説明： ブロックコンテキスト

● string

- データ型： BYTE *
- タイプ： IN
- 説明： データ格納ポインタ

● termNo

- データ型： I32
- タイプ： IN
- 説明： S01 ～ S16 の番号（1 ～ 16）

● option

- データ型： U32
- タイプ： IN
- 説明： オプション

■ 戻り値

- SUCCEED： 正常終了
- UCAERR_PARA_PTRNULL： 引数のポインタが NULL
- UCAERR_OPT_INVALID： 誤ったオプションを指定した
- UCAERR_DATA_INVALIDNO： データ番号／データ数が正しくない

■ 処理内容説明

データ格納ポインタ string が指す領域からデータアイテム S01 ～ S16 へデータをコピーします。

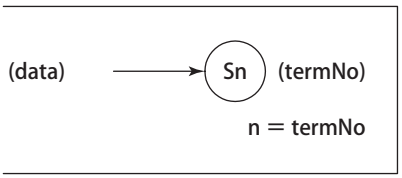
■ 引数説明

● option

NOOPTION を指定してください。

■ 詳細説明

UcaDataStoreEachSn は、引数データ格納ポインタ string が指す領域からデータアイテム S01 ～ S16 へデータをコピーします。



040310J.ai

図　UcaDataStoreEachSnのデータの流れ

引数 string の先が 16 バイトを越える場合は、16 バイトまでをコピーします。15 バイト以下の場合は、16 バイトまで '¥0'（NULL コード）を詰めます。

4.3.6 UcaDataStoreF64SToMvn

UcaDataStoreF64SToMvnは、引数をデータ型変換し、Mvnに格納します。

■ 関数名

I32 UcaDataStoreF64SToMvn

■ 引数

● blockContext

- データ型： UcaBlockContext
- タイプ： IN
- 説明： ブロックコンテキスト

● valPtr

- データ型： F64S
- タイプ： IN
- 説明： データ値ポインタ

● i

- データ型： I32
- タイプ： IN
- 説明： 格納 MVn 番号（0 ～ 16）

● option

- データ型： U32
- タイプ： IN
- 説明： オプション

■ 戻り値

- SUCCEED： 正常終了
- UCAERR_PARA_PTRNULL： 引数のポインタが NULL
- UCAERR_BLKTYPE_INVALID： 正しくないブロック形を指定した
- UCAERR_NOTANUMBER：
  引数の浮動小数データが非数（Not a Number）である
- UCAERR_INFINITY： 引数の浮動小数データが無限大である
- UCAERR_DATA_INVALIDNO： データ番号／データ数が正しくない

■ 処理内容説明

引数 valPtr を F64S → F32S 型変換し、MVnn（nn は i に対応）に格納します。

■ 引数説明

● option

NOOPTION を指定してください。NOOPTION 以外を指定しても SUCCEED を返します。

■ 詳細説明

UcaDataStoreF64SToMvn は、引数 valPtr の値を F64S 型から F32S 型にデータ型変換し、Mvnn（nn は i に対応）に格納します。

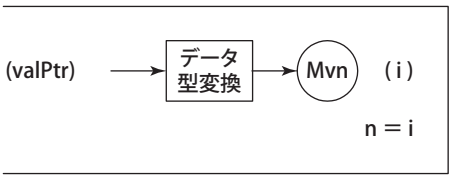


図　UcaDataStoreF64SToMvnのデータの流れ

UcaDataStoreF64SToMvn は、型変換に失敗した場合、MVnn のデータ値を変更しません。データステータスに BAD を設定します。

5. ビルダ定義項目

ユーザカスタムアルゴリズム作成用ライブラリには、ビルダ定義項目の指定内容を取得する関数があります。

参照 ユーザカスタムアルゴリズム作成用ライブラリとビルダ定義項目の関係については、以下を参照してください。

APCS ユーザカスタムブロック プログラミングガイド（IM 33J15U21-01JA）「4.5 ビルダ定義項目の決定」

5.1 ビルダ定義項目

ビルダ定義項目の指定内容を取得する関数のインタフェースについて説明します。

5.1.1 UcaConfigNonlinearGain

UcaConfigNonlinearGainは、ビルダ定義項目［非線形ゲイン］の指定を取得します。

■ 関数名

I32 UcaConfigNonlinearGain

■ 引数

● blockContext

- データ型： UcaBlockContext
- タイプ： IN
- 説明： ブロックコンテキスト

● config

- データ型： I16 *
- タイプ： OUT
- 説明： ビルダ定義項目ポインタ

■ 戻り値

- SUCCEED： 正常終了
- UCAERR_BLKTYPE_INVALID： 正しくないブロック形を指定した
- UCAERR_PARA_PTRNULL： 引数のポインタが NULL

■ 引数説明

● config

- UCACONFIG_NONLINEAR_NON： なし
- UCACONFIG_NONLINEAR_GAPACT： ギャップ動作
- UCACONFIG_NONLINEAR_SQRDVACT：偏差二乗動作

■ 詳細説明

UcaConfigNonlinearGain は、制御演算タブシートのビルダ定義項目［非線形ゲイン］の指定を、引数 config が指す領域に設定します。
UcaConfigNonlinearGain が引数 config の指す領域に設定する値を以下に示します。

表 ［非線形ゲイン］の指定

指定	ラベル
なし	UCACONFIG_NONLINEAR_NON
ギャップ動作	UCACONFIG_NONLINEAR_GAPACT
偏差二乗動作	UCACONFIG_NONLINEAR_SQRDVACT

参照 ラベルの値については、以下を参照してください。
「Appendix C. ラベル一覧」

5.1.2 UcaConfigCompensation

UcaConfigCompensationは、ビルダ定義項目［入出力補償］の指定を取得します。

■ 関数名

I32 UcaConfigCompensation

■ 引数

● blockContext

- データ型： UcaBlockContext
- タイプ： IN
- 説明： ブロックコンテキスト

● config

- データ型： I16 *
- タイプ： OUT
- 説明： ビルダ定義項目ポインタ

■ 戻り値

- SUCCEED： 正常終了
- UCAERR_BLKTYPE_INVALID： 正しくないブロック形を指定した
- UCAERR_PARA_PTRNULL： 引数のポインタが NULL

■ 引数説明

● config

- UCACONFIG_COMP_NON： なし
- UCACONFIG_COMP_INPUT： 入力補償
- UCACONFIG_COMP_OUTPUT： 出力補償

■ 詳細説明

UcaConfigCompensation は、制御演算タブシートのビルダ定義項目［入出力補償］の指定を、引数 config が指す領域に設定します。
UcaConfigCompensation が引数 config の指す領域に設定する値を以下に示します。

表 ［入出力補償］の指定

指定	ラベル
なし	UCACONFIG_COMP_NON
入力補償	UCACONFIG_COMP_INPUT
出力補償	UCACONFIG_COMP_OUTPUT

参照 ラベルの値については、以下を参照してください。
「Appendix C. ラベル一覧」

5.1.3 UcaConfigDirection

UcaConfigDirectionは、ビルダ定義項目［制御動作方向］の指定を取得します。

■ 関数名

I32 UcaConfigDirection

■ 引数

● blockContext

- データ型： UcaBlockContext
- タイプ： IN
- 説明： ブロックコンテキスト

● config

- データ型： I16 *
- タイプ： OUT
- 説明： ビルダ定義項目ポインタ

■ 戻り値

- SUCCEED： 正常終了
- UCAERR_BLKTYPE_INVALID： 正しくないブロック形を指定した
- UCAERR_PARA_PTRNULL： 引数のポインタが NULL

■ 引数説明

● config

- UCACONFIG_DIR_REV： 逆動作
- UCACONFIG_DIR_DIR： 正動作

■ 詳細説明

UcaConfigDirection は、制御演算タブシートのビルダ定義項目［制御動作方向］の指定を、引数 config が指す領域に設定します。
UcaConfigDirection が引数 config の指す領域に設定する値を以下に示します。

表 ［制御動作方向］の指定

指定	ラベル
逆動作	UCACONFIG_DIR_REV
正動作	UCACONFIG_DIR_DIR

参照 ラベルの値については、以下を参照してください。
「Appendix C. ラベル一覧」

5.1.4 UcaConfigOutput

UcaConfigOutputは、ビルダ定義項目［制御／演算出力動作］の指定を取得します。

■ 関数名

I32 UcaConfigOutput

■ 引数

● blockContext

- データ型： UcaBlockContext
- タイプ： IN
- 説明： ブロックコンテキスト

● config

- データ型： I16 *
- タイプ： OUT
- 説明： ビルダ定義項目ポインタ

■ 戻り値

- SUCCEED： 正常終了
- UCAERR_BLKTYPE_INVALID： 正しくないブロック形を指定した
- UCAERR_PARA_PTRNULL： 引数のポインタが NULL

■ 引数説明

● config

- UCACONFIG_ACT_POS：位置形
- UCACONFIG_ACT_VEL： 速度形

■ 詳細説明

UcaConfigOutput は、制御演算タブシートのビルダ定義項目［制御／演算出力動作］の指定を、引数 config が指す領域に設定します。
UcaConfigOutput が引数 config の指す領域に設定する値を以下に示します。

表 ［制御／演算出力動作］の指定

指定	ラベル
位置形	UCACONFIG_ACT_POS
速度形	UCACONFIG_ACT_VEL

参照 ラベルの値については、以下を参照してください。
「Appendix C. ラベル一覧」

5.1.5 UcaConfigDeadband

UcaConfigDeadbandは、ビルダ定義項目［不感帯動作］の指定を取得します。

■ 関数名

I32 UcaConfigDeadband

■ 引数

● blockContext

- データ型： UcaBlockContext
- タイプ： IN
- 説明： ブロックコンテキスト

● config

- データ型： I16 *
- タイプ： OUT
- 説明： ビルダ定義項目ポインタ

■ 戻り値

- SUCCEED： 正常終了
- UCAERR_BLKTYPE_INVALID： 正しくないブロック形を指定した
- UCAERR_PARA_PTRNULL： 引数のポインタが NULL

■ 引数説明

● config

- UCACONFIG_DEADBAND_FALSE： 不感帯動作なし
- UCACONFIG_DEADBAND_TRUE： 不感帯動作あり

■ 詳細説明

UcaConfigDeadband は、制御演算タブシートのビルダ定義項目［不感帯動作］の指定を、引数 config が指す領域に設定します。
UcaConfigDeadband が引数 config の指す領域に設定する値を以下に示します。

表 ［不感帯動作］の指定

指定	ラベル
なし	UCACONFIG_DEADBAND_FALSE
あり	UCACONFIG_DEADBAND_TRUE

参照 ラベルの値については、以下を参照してください。
「Appendix C. ラベル一覧」

5.1.6 UcaConfigDeadbandHys

UcaConfigDeadbandHysは、ビルダ定義項目の［不感帯動作／ヒステリシス］の指定を取得します。

■ 関数名

I32 UcaConfigDeadbandHys

■ 引数

● blockContext

- データ型： UcaBlockContext
- タイプ： IN
- 説明： ブロックコンテキスト

● config

- データ型： F32 *
- タイプ： OUT
- 説明： ビルダ定義項目ポインタ

■ 戻り値

- SUCCEED： 正常終了
- UCAERR_BLKTYPE_INVALID： 正しくないブロック形を指定した
- UCAERR_PARA_PTRNULL： 引数のポインタが NULL

■ 詳細説明

UcaConfigDeadbandHys は、制御演算タブシートのビルダ定義項目［不感帯動作／ヒステリシス］の値を取得します。
機能ブロック詳細ビルダで指定する［不感帯動作］のヒステリシスの値 hys[%] を、データアイテム SH、SL でレンジ変換します。取得したデータ config、データアイテム SH、SL は、PV と同じ単位のデータです。

config = hys / 100.0 *(SH - SL)

5.1.7 UcaConfigGeneral

UcaConfigGeneralは、ビルダ定義項目［汎用指定項目1～8］の指定を取得します。

■ 関数名

I32 UcaConfigGeneral

■ 引数

● blockContext

- データ型： UcaBlockContext
- タイプ： IN
- 説明： ブロックコンテキスト

● itemNo

- データ型： I32
- タイプ： IN
- 説明： 汎用指定項目番号（1 ～ 8）

● config

- データ型： I16 *
- タイプ： OUT
- 説明： ビルダ定義項目ポインタ

■ 戻り値

- SUCCEED： 正常終了
- UCAERR_PARA_PTRNULL： 引数のポインタが NULL
- UCAERR_NUM_INVALID： 正しくない番号を指定した

■ 詳細説明

UcaConfigGeneral は、その他タブシートのビルダ定義項目［汎用指定項目 1 ～ 8］の指定を、引数 config が指す領域に設定します。引数 itemNo には、汎用指定項目の番号を指定します。たとえば、itemNo に 1 を指定すれば、［汎用指定項目 1］に指定された値を取得します。
［汎用指定項目 1 ～ 8］には、－ 32768 ～ 32767 の整数を指定することができます。

参照 ［汎用指定項目 1 ～ 8］については、以下を参照してください。
APCS ユーザカスタムブロック プログラミングガイド（IM 33J15U21-01JA）「4.7 ユーザカスタムアルゴリズムの共用」

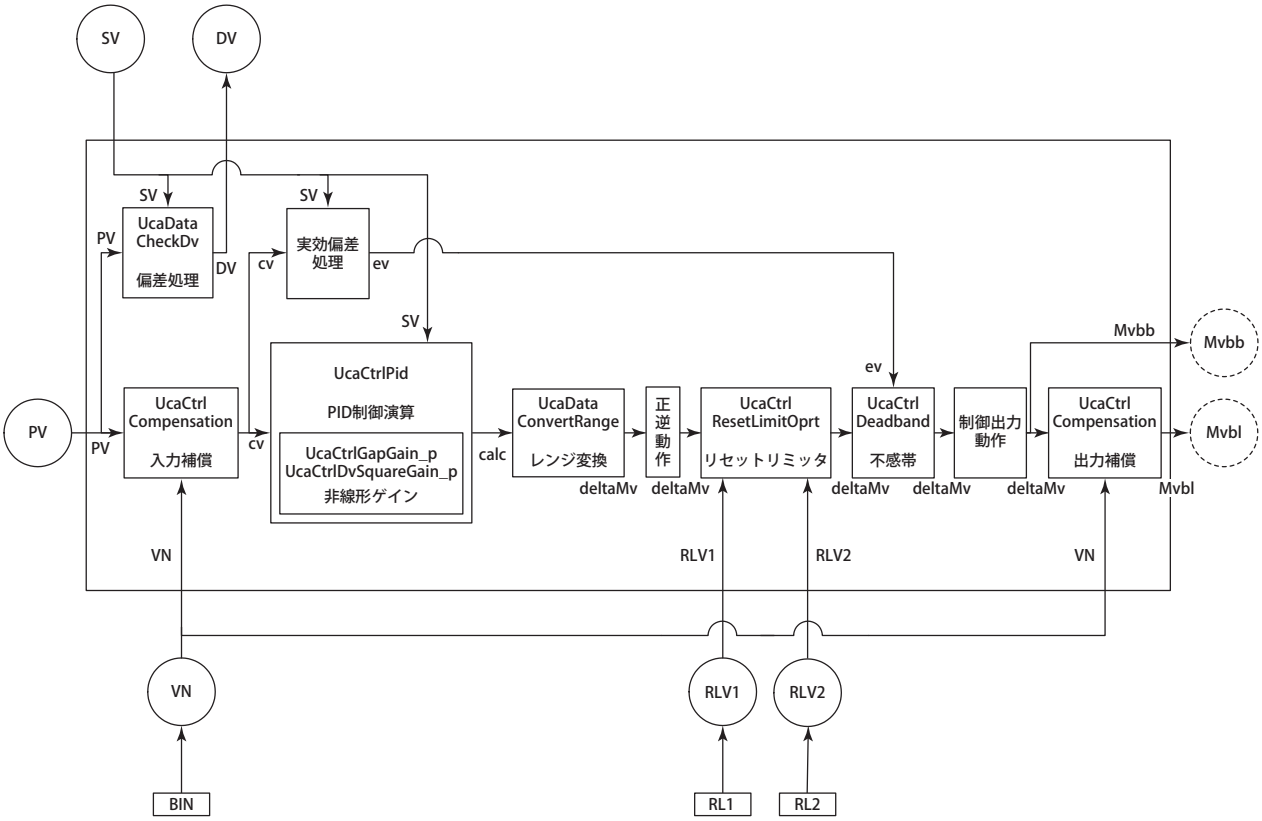
Blank Page

6. 制御演算

制御演算関数は、制御演算に関する処理を行います。制御演算関数には、入出力補償、PID制御演算、レンジ変換、リセットリミッタ、不感帯動作などの処理を行う関数があります。ユーザカスタムアルゴリズムにおいて、必要な処理に関する関数を呼び出し、制御演算処理を記述してください。たとえば、入出力補償処理が必要な場合には、入出力補償関数を呼び出してください。PID制御演算処理以外の制御演算を行う場合には、PID制御演算関数を呼び出さず、制御演算処理を記述してください。

■ 制御演算処理の概要

例として、PID 制御演算の処理の流れを示します。



PV： 測定値
SV： 設定値
DV： 偏差（PVとSVの偏差）
cv： 入力補償後のPV
ev： 実効偏差（cvとSVの偏差）
calc： 制御演算値
deltaMv： 操作出力変更量
VN： 入出力補償値
RLV1： RL1端子からのリセット信号1
RLV2： RL2端子からのリセット信号2
Mvbb： 出力補償前出力値
Mvbl： 出力リミット前出力値

060001J.ai

図 PID制御演算の処理の流れ

制御演算関数には、前図の関数以外に制御演算初期化関数、制御周期処理関数があります。これらの関数の動作は、他の制御演算関数の動作と関係があります。

UcaCtrlPid を呼び出し PID 制御演算を行う場合、制御演算初期化処理関数として UcaCtrlPidInit を使用してください。初期化処理を行う関数は、UcaCtrlPidInit と UcaCtrlFuncInit の 2 つがあります。この 2 つの関数の相違点は、UcaCtrlPid の内部パラメータを初期化するかしないかです。

表　制御演算と制御演算初期化関数

制御演算	PID制御演算を行う	PID制御演算を行わない
UcaCtrlPid の呼び出し	呼び出す	呼び出さない
制御演算初期化処理として呼び出す関数	UcaCtrlPidInit	UcaCtrlFuncInit

制御演算に使用する時間として制御周期を用いる場合には、UcaCtrlPidTiming を呼び出し制御周期処理を行ってください。
UcaCtrlPid、UcaCtrlResetLimitOprt、UcaCtrlResetLimitOprt_p 関数でオプションに NOOPTION を指定すると、制御時間として UcaCtrlTiming で処理する制御周期カウンタの値を使用します。オプションに UCAOPT_CTRL_TSCTIME を指定すると、実効スキャン周期の値を使用します。
UcaRWWriteMvToOutSub 関数でオプションに UCAOPT_RW_CTRLTIMING を指定すると、制御周期時のみ出力を行います。オプションに NOOPTION を指定すると、毎起動タイミングで出力を行います。

表　制御演算に使用する時間と制御周期処理関数

制御演算に使用する制御時間	制御周期		実効スキャン周期
	自動決定、4、8、16、32、64	間欠制御動作	
UcaCtrlPidTiming の呼び出し	呼び出す	呼び出す	呼び出さない
UcaCtrlPid、UcaCtrlResetLimitOprt、UcaCtrlResetLimitOprt_p に指定するオプション	NOOPTION	NOOPTION	UCAOPT_CTRL_TSCTIME
UcaRWWriteMvToOutSub に指定するオプション	UCAOPT_RW_CTRLTIMING	UCAOPT_RW_CTRLTIMING	NOOPTION

参照 制御演算関数の動作は、標準の調節ブロックの各機能と同じです。詳細については、以下を参照してください。
機能ブロックリファレンス Vol.1（IM 33J15A30-01JA）「1.4 調節ブロックに共通の制御演算処理」

6.1 制御演算ハンドラ

制御演算ハンドラ関数は、制御演算を行うユーザカスタムアルゴリズムのメイン処理を行います。制御演算ハンドラ関数は、ユーザ定義の入力処理、制御演算処理、出力処理の関数を呼び出します。また、ブロックモード遷移処理と出力値決定処理を行います。

6.1.1 UcaCtrlHandler

UcaCtrlHandlerは、ユーザ定義の関数を呼び出し、モード遷移処理と出力値決定処理を行います。

■ 関数名

I32 UcaCtrlHandler

■ 引数

● blockContext

- データ型： UcaBlockContext
- タイプ： IN/OUT
- 説明： ブロックコンテキスト

● input

- データ型： UCAFUNC_INPUT
- タイプ： IN
- 説明： 入力処理関数アドレス

● control

- データ型： UCAFUNC_CONTROL
- タイプ： IN
- 説明： 制御演算処理関数アドレス

● output

- データ型： UCAFUNC_OUTPUT
- タイプ： IN
- 説明： 出力処理関数アドレス

● execMode

- データ型： U32
- タイプ： IN
- 説明： 演算処理実行ブロックモード

● option

- データ型： U32
- タイプ： IN
- 説明： オプション

■ 戻り値

- SUCCEED： 正常終了
- UCAERR_EXECMODE_INVALID：
  演算処理実行ブロックモードの指定に誤りあり
- UCAERR_PARA_PTRNULL： 引数のポインタが NULL
- UCAERR_OPT_INVALID： 誤ったオプションを指定した
- UCAERR_BLKTYPE_INVALID： 正しくないブロック形を指定した

■ 処理内容説明

ユーザ定義の関数を呼び出し、モード遷移処理と出力値決定処理を行います。

■ 引数説明

● exeMode

以下の中から 1 つだけ指定することができます。

- UCACTRL_AUTCASRCAS：
  AUT/CAS/RCAS モード時のみ制御演算処理関数（control）を実行します。
- UCACTRL_ALLMODE：
  すべてのモードで制御演算処置関数（control）を実行します。

● option

以下の中から複数指定することができます。複数指定する場合は、それぞれを‘|’（or）で結んでください。

- UCAOPT_CTRL_MANISIMAN：MAN モード時に、IMAN 出力動作を行います。
- UCAOPT_CTRL_NOSVOPRT：
  メジャートラッキング、設定値プッシュバック、偏差警報処理を行いません。
- NOOPTION

■ 詳細説明

UcaCtrlHandler は、制御演算を行うユーザカスタムアルゴリズムのメイン処理を行います。演算処理実行ブロックモードに UCACTRL_AUTCASRCAS を指定したときの、UcaCtrlHandler の処理を以下に示します（太字がユーザ記述の処理です）。

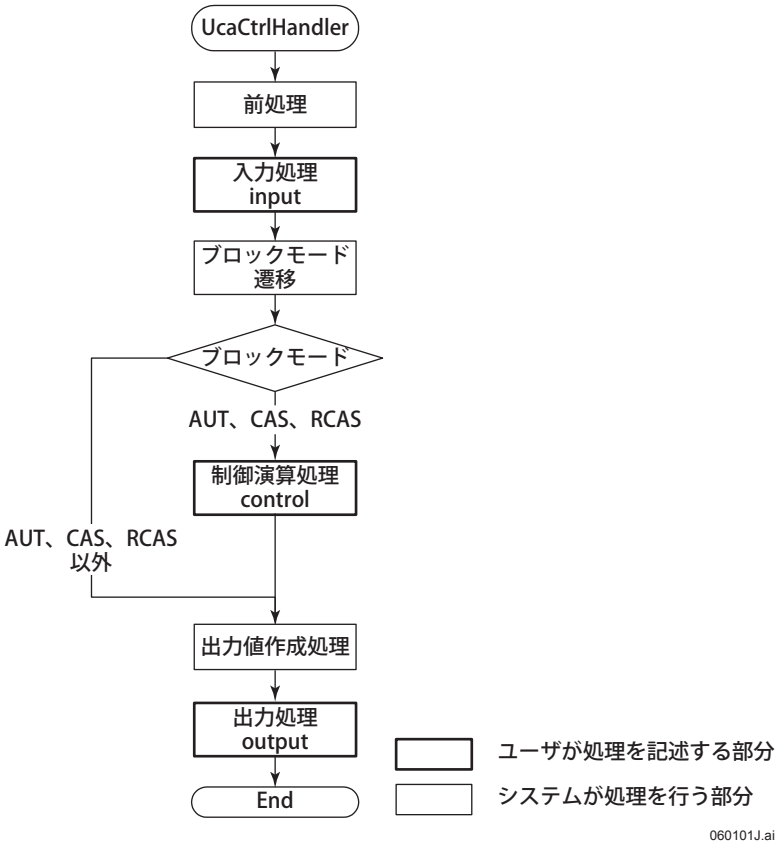


図　UcaCtrlHandlerの処理の流れ

UcaCtrlHandler は、OUT 端子読み返しや TIN 端子読み返しなどの前処理を行います。前処理が完了すると、引数に指定されたユーザ記述の入力処理関数（input）を呼び出します。入力処理関数は測定値 PV を作成します。
入力関数から戻ると、UcaCrtlHandler はブロックモードを遷移します。ブロックモードの遷移が完了すると、メジャートラッキング処理と設定値プッシュバック処理を行います。
UcaCtrlHandler は、引数に指定されたユーザ記述の制御演算関数（control）を呼び出します。制御演算関数は、制御演算を実行し、引数に演算結果を返します。
UcaCtrlHandler は、引数に返された演算結果とブロックモードを基に、内部変数 Mvbl を決定します。UcaCtrlHandler は、ユーザ記述の出力処理関数（output）を呼び出します。
UcaCtrlHandler は、制御ホールドが設定されている場合、ブロックモードが AUT、CAS、RCAS のときに制御演算初期化要求を設定します。

参照 動作の詳細については、以下を参照してください。
APCS ユーザカスタムブロック プログラミングガイド（IM 33J15U21-01JA）「6.2 CSTM-C のブロックモード遷移」

オプションに UCAOPT_CTRL_MANISIMAN を指定すると、ブロックモードが MAN のときに、出力読み返し値（内部変数 outRb）を基に出力処理を行います。オプションに NOOPTION を指定すると、ブロックモードが MAN の時には、データアイテム MV 値を基に出力処理を行います。
オプションに UCAOPT_CTRL_NOSVOPRT を指定すると、メジャートラッキング、設定値プッシュバック、偏差警報処理を行いません。

参照 動作の詳細については、以下を参照してください。
APCS ユーザカスタムブロック プログラミングガイド（IM 33J15U21-01JA）「6.2.4 ブロックモード遷移と設定値の作成（UcaCtrlHandler の内部で処理）」

ユーザカスタムアルゴリズムにおいてユーザが記述する入力処理関数（input）、演算処理関数（control）、出力処理関数（output）の関数インタフェースを示します。ユーザ記述の関数のインタフェースは固定です。
これらの関数を、UcaCtrlHandler の引数に指定してください。
必ず SUCCEED を戻してください。

● 入力処理関数

- 関数名
  I32 input
- 引数
  blockContext
    データ型： UcaBlockContext
    タイプ： IN/OUT
    説明： ブロックコンテキスト
- 戻り値
  SUCCEED：正常終了
- 処理内容説明
  入力処理を記述します。

● 演算処理関数

- 関数名
  I32 control
- 引数
  - blockContext
    データ型： UcaBlockContext
    タイプ： IN/OUT
    説明： ブロックコンテキスト
  - mvbbPtr
    データ型： F64 *
    タイプ： IN/OUT
    説明： 出力補償前出力値ポインタ
  - mvblPtr
    データ型： F64 *
    タイプ： IN/OUT
    説明： 出力リミット前出力値ポインタ
- 戻り値
  SUCCEED：正常終了
- 処理内容説明
  制御演算処理を記述します。

● 出力処理関数

- 関数名
  I32 output
- 引数
  blockContext
  データ型： UcaBlockContext
  タイプ： IN/OUT
  説明： ブロックコンテキスト
- 戻り値
  SUCCEED：正常終了
- 処理内容説明
  出力処理を記述します。

6.2 制御演算初期化

制御演算初期化関数は、制御演算初期化要求の設定、解除、検査を行います。

■ 制御演算初期化処理の概要

制御演算初期化要求処理の流れを以下に示します。

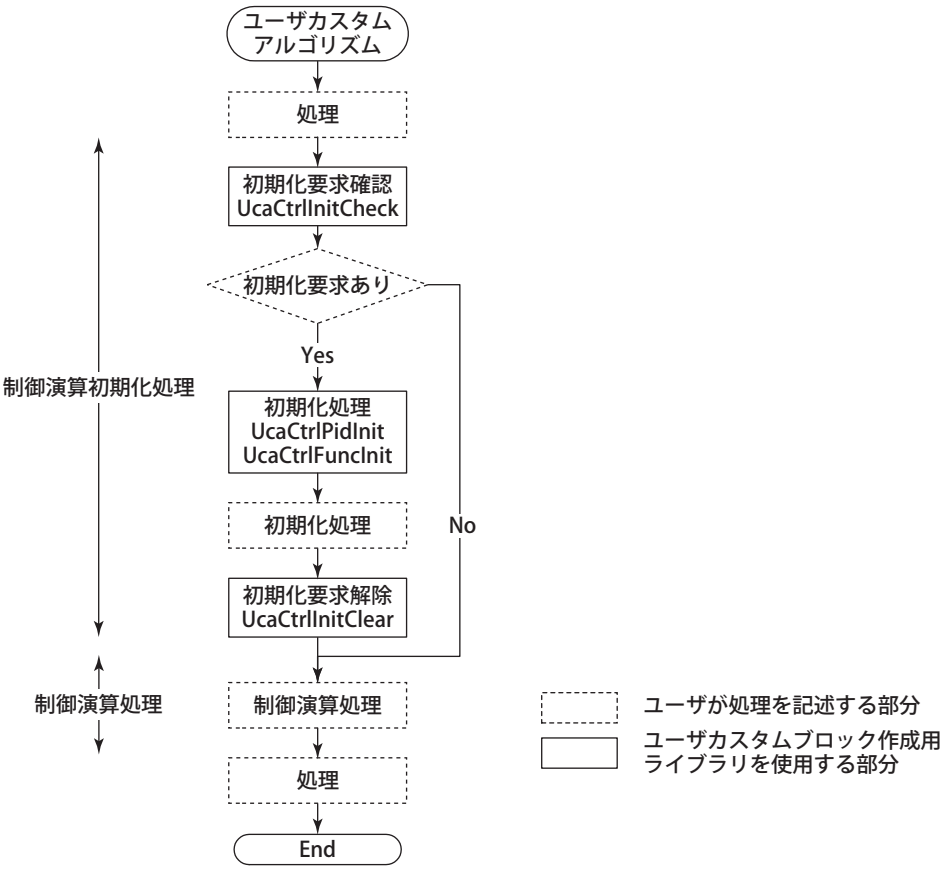


図　制御演算初期化要求の設定、検査、解除の流れ

システムは、制御演算初期化が必要な条件が成立した場合、制御演算初期化要求を設定します。特に、ユーザカスタムアルゴリズムで検出する条件に従い制御演算初期化要求を設定する必要がある場合には、UcaCtrlInitSet を呼び出して制御演算初期化要求を設定してください。

制御演算処理の前に、以下の順序で関数を呼び出し、制御演算初期化処理を行います。

1. UcaCtrlInitCheck を呼び出し、初期化要求があるかどうかを確認します。
2. 初期化要求がある場合には、UcaCtrlPidInit または UcaCtrlFuncInit を呼び出し、制御演算を初期化します。
3. システムが行う初期化処理以外に必要な処理がある場合、初期化処理を記述します。
4. UcaCtrlInitClear を呼び出し制御演算初期化要求を解除します。
   制御演算初期化処理後、制御演算処理を行い、処理を続行します。

6.2.1 UcaCtrlInitCheck

UcaCtrlInitCheckは、制御演算の初期化要求が設定されているかを確認します。

■ 関数名

I32 UcaCtrlInitCheck

■ 引数

● blockContext

- データ型： UcaBlockContext
- タイプ： IN
- 説明： ブロックコンテキスト

● result

- データ型： BOOL *
- タイプ： OUT
- 説明： 初期化設定があるかどうか

■ 戻り値

- SUCCEED： 正常終了
- UCAERR_PARA_PTRNULL： 引数のポインタが NULL

■ 処理内容説明

制御演算の初期化要求が設定されているかを確認します。

■ 詳細説明

制御演算処理の前に、UcaCtrlInitCheck を呼び出し初期化要求があるかどうかを確認します。
UcaCtrlInitCheck は、初期化要求がある場合、引数 result に TRUE を格納します。初期化要求がない場合、引数 result に FALSE を格納します。

6.2.2 UcaCtrlInitClear

UcaCtrlInitClearは、制御演算の初期化要求を解除します。

■ 関数名

I32 UcaCtrlInitClear

■ 引数

● blockContext

- データ型： UcaBlockContext
- タイプ： IN
- 説明： ブロックコンテキスト

■ 戻り値

- SUCCEED： 正常終了
- UCAERR_PARA_PTRNULL： 引数のポインタが NULL

■ 処理内容説明

制御演算の初期化要求を解除します。

■ 詳細説明

制御演算初期化処理が終了したら、UcaCtrlInitClear を呼び出し制御演算初期化要求を解除します。

6.2.3 UcaCtrlInitSet

UcaCtrlInitSetは、制御演算の初期化要求を設定します。

■ 関数名

I32 UcaCtrlInitSet

■ 引数

● blockContext

- データ型： UcaBlockContext
- タイプ： IN
- 説明： ブロックコンテキスト

■ 戻り値

- SUCCEED： 正常終了
- UCAERR_PARA_PTRNULL： 引数のポインタが NULL

■ 処理内容説明

制御演算の初期化要求を設定します。

■ 詳細説明

システムは、制御演算初期化が必要な条件が成立した場合、制御演算初期化要求を設定しますので、通常ユーザカスタムアルゴリズムで UcaCtrlInitSet を呼び出す必要はありません。ユーザカスタムアルゴリズムで検出する条件に従った制御演算初期化要求の設定が必要な場合には、UcaCtrlInitSet を呼び出して制御演算初期化要求を設定します。

6.2.4 UcaCtrlPidInit

UcaCtrlPidInitは、PID演算や制御演算で用いる内部パラメータの値を初期化します。

■ 関数名

I32 UcaCtrlPidInit

■ 引数

● blockContext

- データ型： UcaBlockContext
- タイプ： IN
- 説明： ブロックコンテキスト

● cv

- データ型： F64
- タイプ： IN
- 説明： 補償後入力データ

● option

- データ型： U32
- タイプ： IN
- 説明： オプション

■ 戻り値

- SUCCEED： 正常終了
- UCAERR_PARA_PTRNULL： 引数のポインタが NULL
- UCAERR_OPT_INVALID： 誤ったオプションを指定した
- UCAERR_BLKTYPE_INVALID： 正しくないブロック形を指定した

■ 処理内容説明

PID 演算や制御演算に用いる内部パラメータの値を初期化します。

■ 引数説明

● option

NOOPTION を指定してください。

■ 詳細説明

UcaCtrlPidInit は、PID 演算関数 UcaCtrlPid の内部パラメータを初期化します。ユーザカスタムアルゴリズムにおいて、PID 演算関数 UcaCtrlPid を使用する場合、制御演算初期化関数として UcaCtrlPidInit を呼び出してください。
UcaCtrlPidInit は、以下の PID 演算や制御演算関数で用いる内部パラメータの値を初期化します。

表　UcaCtrlPidInitが初期化する内部パラメータ

初期化する内部パラメータ	パラメータを使用する関数
制御周期に 0 を設定します	UcaCtrlPidTiming
不感帯動作で用いるパラメータを初期化します	UcaCtrlDeadband UcaCtrlDeadband_p
PID 演算で用いるパラメータを初期化します	UcaCtrlPid

6.2.5 UcaCtrlFuncInit

UcaCtrlFuncInitは、制御演算で用いる内部パラメータの値を初期化します。

■ 関数名

I32 UcaCtrlFuncInit

■ 引数

● blockContext

- データ型： UcaBlockContext
- タイプ： IN
- 説明： ブロックコンテキスト

● option

- データ型： U32
- タイプ： IN
- 説明： オプション

■ 戻り値

- SUCCEED： 正常終了
- UCAERR_PARA_PTRNULL： 引数のポインタが NULL
- UCAERR_OPT_INVALID： 誤ったオプションを指定した
- UCAERR_BLKTYPE_INVALID： 正しくないブロック形を指定した

■ 処理内容説明

制御演算に用いる内部パラメータの値を初期化します。

■ 引数説明

● option

NOOPTION を指定してください。

■ 詳細説明

UcaCtrlFuncInit は、制御演算関数で用いる内部パラメータの値を初期化します。ユーザカスタムアルゴリズムにおいて、偏差処理や不感帯処理などの制御演算関数を使用し、PID 演算関数 UcaCtrlPid を使用せずに制御演算処理を記述する場合、制御演算初期化関数として UcaCtrlFuncInit を呼び出してください。UcaCtrlFuncInit は、PID 演算関数 UcaCtrlPid の内部パラメータを初期化しません。
UcaCtrlFuncInit は、以下の制御演算関数で用いる内部パラメータの値を初期化します。

表 UcaCtrlFuncInitが初期化する内部パラメータ

初期化する内部パラメータ	パラメータを使用する関数
制御周期カウンタの値に 0 を設定します	UcaCtrlPidTiming
不感帯動作で用いるパラメータを初期化します	UcaCtrlDeadband UcaCtrlDeadband_p

6.3 PID演算

PID制御演算は、比例（P）、積分（I）、微分（D）の3種類の調節動作が組み合わされた、プロセス制御でもっとも基本的な制御演算です。PID制御演算関数には、PID制御演算や制御周期に関する関数があります。

■ PID演算処理の概要

制御周期処理を行う場合には、UcaCtrlPidTiming を呼び出します。UcaCtrlPidTiming で処理される制御周期カウンタの値を、PID 演算関数 UcaCtrlPid から参照します。

参照 動作の詳細については、以下を参照してください。
機能ブロックリファレンス Vol.1（IM 33J15A30-01JA）「1.5 PID 調節ブロック（PID）」の「■ PID 制御演算」

6.3.1 UcaCtrlPid

UcaCtrlPidは、PID制御演算を行います。

■ 関数名

I32 UcaCtrlPid

■ 引数

● blockContext

- データ型： UcaBlockContext
- タイプ： IN
- 説明： ブロックコンテキスト

● cv

- データ型： F64
- タイプ： IN
- 説明： 補償後入力データ

● calcPtr

- データ型： F64 *
- タイプ： OUT
- 説明： 演算値ポインタ

● option

- データ型： U32
- タイプ： IN
- 説明： オプション

■ 戻り値

- SUCCEED： 正常終了
- UCAERR_CTRL_PIDCALC： PID 演算計算中に浮動小数演算で例外発生
- UCAERR_CTRL_GAPGAIN： ギャップゲイン計算中に浮動小数演算で例外発生
- UCAERR_CTRL_DVSQUAREGAIN：
  偏差二乗ゲイン計算中に浮動小数演算で例外発生
- UCAERR_CTRL_PIDTI： PID 制御計算で積分時間が異常
- UCAERR_PARA_PTRNULL： 引数のポインタが NULL
- UCAERR_OPT_INVALID： 誤ったオプションを指定した
- UCAERR_BLKTYPE_INVALID： 正しくないブロック形を指定した

■ 処理内容説明

入力補償後のデータを用いて、PID 演算を行い、結果を演算値に設定します。

■ 引数説明

● option

以下の中から指定することができます。

- UCAOPT_CTRL_TSCTIME：制御時間に実効スキャン周期を使用します。
- NOOPTION

■ 詳細説明

UcaCtrlPid は、PID 演算処理を行い、結果を引数の演算値ポインタ calcPtr に格納します。

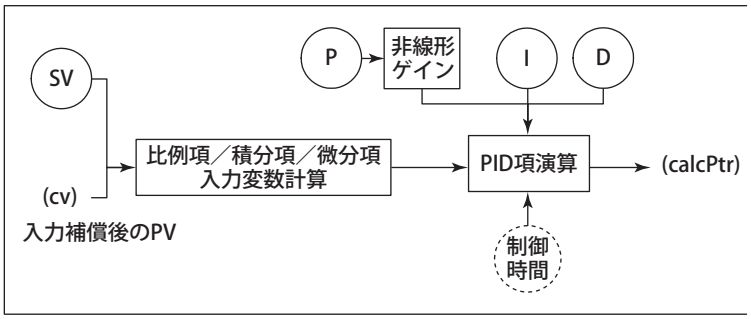


図　UcaCtrlPidのデータの流れ

UcaCtrlPid は、制御演算タブシートのビルダ定義項目［PID 制御アルゴリズム］の値に応じて、引数の補償後入力データ cv を用いて比例項、微分項、積分項の入力値を求めます。UcaCtrlPid は、制御演算タブシートのビルダ定義項目［非線形ゲイン］に応じて、データアイテム P に対してギャップ動作、偏差二乗動作処理を行います。

参照 動作の詳細については、以下を参照してください。
機能ブロックリファレンス Vol.1（IM 33J15A30-01JA）「1.5 PID 調節ブロック（PID）」の「■ PID 制御演算」
機能ブロックリファレンス Vol.1（IM 33J15A30-01JA）「1.5 PID 調節ブロック（PID）」の「■ 制御動作バイパス」

オプションに NOOPTION を指定すると、UcaCtrlPid は PID 項演算で用いる制御時間に制御周期カウンタ値を使用します。オプションに UCAOPT_CTRL_TSCTIME を指定すると、制御時間に実効スキャン周期を使用します。ユーザカスタムアルゴリズムにおいて制御周期を使用しない場合には、オプションに UCAOPT_CTRL_TSCTIME を指定してください。

6.3.2 UcaCtrlPidTiming

UcaCtrlPidTimingは、制御周期処理を行い、制御周期時かどうか判定します。

■ 関数名

I32 UcaCtrlPidTiming

■ 引数

● blockContext

- データ型： UcaBlockContext
- タイプ： IN
- 説明： ブロックコンテキスト

● result

- データ型： BOOL *
- タイプ： OUT
- 説明： 制御周期かどうか

● option

- データ型： U32
- タイプ： IN
- 説明： オプション

■ 戻り値

- SUCCEED： 正常終了
- UCAERR_CTRL_CTRLTIME： 制御周期時間が異常
- UCAERR_PARA_PTRNULL： 引数のポインタが NULL
- UCAERR_OPT_INVALID： 誤ったオプションを指定した
- UCAERR_BLKTYPE_INVALID： 正しくないブロック形を指定した

■ 処理内容説明

制御周期処理を行います。

■ 引数説明

● option

NOOPTION を指定してください。

■ 詳細説明

UcaCtrlPidTiming のデータの流れは以下のとおりです。

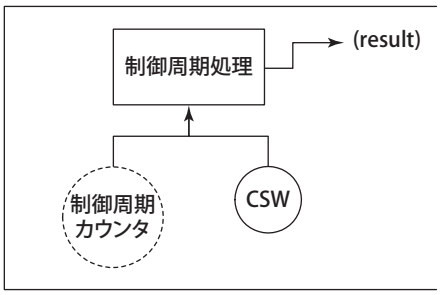


図　UcaCtrlPidTimingのデータの流れ

システムは毎実効スキャン周期に制御周期カウンタのカウントアップを行います。UcaCtrlPidTiming は現在の時間と制御周期カウンタの値を比較し、制御周期時であるか判定します。UcaCtrlPidTiming は、制御周期時には引数 result に TRUE を設定します。非制御周期時には引数 result に FALSE を設定します。
連続制御動作の場合、制御演算タブシートのビルダ定義項目［制御周期］の値に応じて、制御周期時かどうかを判定します。間欠制御動作の場合、制御スイッチ（データアイテム CSW）が ON になったスキャン周期のみ、制御周期時と判定します。
ユーザカスタムアルゴリズムにおいて制御周期を使用する場合、必ず処理タイミングを与えられるごとに UcaCtrlPidTiming を呼び出してください。ユーザカスタムアルゴリズムにおいて一度の処理タイミング内に複数回 UcaCtrlPidTiming を呼び出す場合、UcaCtrlPidTiming は初回のみ制御周期処理を行います。2 回目以降は制御周期処理は行わず、初回呼び出し時の判定結果を返します。
制御周期カウンタの値は、UcaCtrlPidGetTime を用いて取得することができます。制御周期カウンタの分解能は実効スキャン周期で、最大値は 1,000,000（UCA_NUM_MAXCTRLTIME）［sec］です。最大値に達すると 0 に戻りカウントを続けます。

6.3.3 UcaCtrlPidGetTime

UcaCtrlPidGetTimeは、制御周期カウンタの値を取得します。

■ 関数名

I32 UcaCtrlPidGetTime

■ 引数

● blockContext

- データ型： UcaBlockContext
- タイプ： IN
- 説明： ブロックコンテキスト

● cntPtr

- データ型： I32 *
- タイプ： OUT
- 説明： カウンタ値ポインタ

■ 戻り値

- SUCCEED： 正常終了
- UCAERR_PARA_PTRNULL： 引数のポインタが NULL
- UCAERR_BLKTYPE_INVALID： 正しくないブロック形を指定した

■ 処理内容説明

制御周期カウンタの値を取得します。

■ 詳細説明

UcaCtrlPidGetTime は内部変数の制御周期カウンタの値を取得し、引数 cntPtr に格納します。

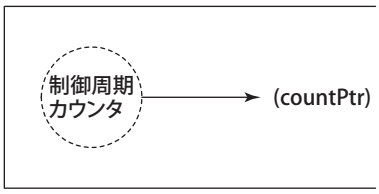


図 UcaCtrlPidGetTimeのデータの流れ

制御周期カウンタの単位は秒［sec］です。制御周期カウンタは、システムが自動的に進めます。分解能は実効スキャン周期で、最大値は 1,000,000（UCA_NUM_MAXCTRLTIME）［sec］です。最大値に達すると 0 に戻りカウントを続けます。

UcaCtrlPidGetTime には、以下の 2 種類の使い方があります。

● 制御周期をUcaCtrlPidTimingで作る場合

UcaCtrlPidTiming は、制御周期時には TRUE を、制御周期でなければ FALSE を引数 result に格納します。制御周期時に UcaCtrlPidGetTime を呼び出し、制御周期カウンタを取得します。

```
UcaCtrlPidTiming を呼び出す
if（引数 result が TRUE）{
  UcaCtrlGetTime を呼び出す
  制御演算を実行
}
```

制御周期が間欠制御動作の場合に有効な方法です。

● 制御周期をUcaCtrlPutTimeとUcaCtrlGetTimeで作る場合

UcaCtrlPidGetTime と UcaCtrlPidPutTime を呼び出し、以下のように制御周期の計算を行います。

```
UcaCtrlPidGetTime を呼び出し、制御周期カウンタを取得
制御周期時かどうか判断
if（制御周期時）{
  制御演算を実行
  UcaCtrlPidPutTime で制御周期カウンタを 0 に戻す
}
```

重要 制御周期を UcaCtrlPutTime と UcaCtrlGetTime で作る場合、UcaCtrlPidTiming を使用しないでください。UcaCtrlPidTiming の内部で制御周期カウンタを操作しているため、正しく制御周期を作ることができません。UcaRWWriteMvToOutSub は内部で UcaCtrlPidGetTiming を呼び出しているため、使用しないでください。UcaRWWriteMvToOutSub の代わりに、UcaRWWriteMv と UcaRWWriteSub を呼び出してください。

6.3.4 UcaCtrlPidPutTime

UcaCtrlPidPutTimeは、制御周期カウンタを設定します。

■ 関数名

I32 UcaCtrlPidPutTime

■ 引数

● blockContext

- データ型： UcaBlockContext
- タイプ： IN
- 説明： ブロックコンテキスト

● cntVal

- データ型： I32
- タイプ： IN
- 説明： 制御周期カウンタ［s］

■ 戻り値

- SUCCEED： 成功
- UCAERR_CTRL_CTRLTIME： 制御周期カウンタ値が設定可能範囲外
- UCAERR_PARA_PTRNULL： 引数のポインタが NULL
- UCAERR_BLKTYPE_INVALID： 正しくないブロック形を指定した

■ 処理内容説明

制御周期時間を設定します。

■ 詳細説明

UcaCtrlPidPutTime は、制御周期カウンタに引数 cntVal の値を設定します。

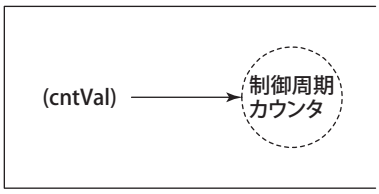


図 UcaCtrlPidPutTimeのデータの流れ

制御周期カウンタの単位は秒［sec］です。制御周期カウンタの設定可能範囲は、0 ～ 1、000、000（UCA_NUM_MAXCTRLTIME）［sec］です。通常は制御周期カウンタを 0 にリセットするのに使いますが、PG-L13 ブロックのように特定の値まで進めることもできます。ユーザカスタムアルゴリズムにおいて一度の処理タイミング内に複数回 UcaCtrlPidTiming を呼び出す場合、UcaCtrlPidTiming は初回のみ制御周期処理を行います。2 回目以降は制御周期処理は行わず、初回呼び出し時の判定結果を返します。

6.4 制御ホールド

制御ホールド機能は、ブロックモードを現状維持したまま、一時的に制御動作を中断させる異常処理の1つです。

■ 制御ホールド処理の概要

システムは UcaRWReadIn において以下のように制御ホールド要求を設定／解除します。

- IN 端子の結合先がオープンの（切り換えスイッチなどで選択されていない）ときに制御ホールド要求を設定します。
- 上記以外の場合、制御ホールド要求を解除します。

通常、ユーザカスタムアルゴリズムで制御ホールドの設定／解除をする必要はありません。特に、制御ホールド要求の設定／解除が必要な場合に限り、UcaCtrlHoldSet/UcaCtrlHoldClear を呼び出し制御ホールドを設定／解除してください。

以下の順序で関数を呼び出し、制御ホールド処理を行います。

1. UcaCtrlHoldCheck を呼び出し、制御ホールド要求が設定されているか確認します。
2. 制御ホールド要求が設定されていない場合のみ、制御演算処理を記述します。

参照 動作の詳細については、以下を参照してください。
機能ブロックリファレンス Vol.1（IM 33J15A30-01JA）「1.4 調節ブロックに共通の制御演算処理」の「■制御ホールド」

6.4.1 UcaCtrlHoldCheck

UcaCtrlHoldCheckは、制御演算がホールド要求が設定されているかを確認します。

■ 関数名

I32 UcaCtrlHoldCheck

■ 引数

● blockContext

- データ型： UcaBlockContext
- タイプ： IN
- 説明： ブロックコンテキスト

● result

- データ型： BOOL *
- タイプ： OUT
- 説明： 制御ホールド中かどうか

■ 戻り値

- SUCCEED： 正常終了
- UCAERR_PARA_PTRNULL： 引数のポインタが NULL
- UCAERR_BLKTYPE_INVALID： 正しくないブロック形を指定した

■ 処理内容説明

制御ホールド中かどうか確認します。

■ 詳細説明

ユーザカスタムアルゴリズムにおいて、制御演算処理の前に UcaCtrlHoldCheck を呼び出し、制御ホールド要求が設定されているか確認します。
UcaCtrlHoldCheck は、制御ホールド要求がある場合、引数 result に TRUE を格納します。
制御ホールド要求がない場合、引数 result に FALSE を格納します。

6.4.2 UcaCtrlHoldClear

UcaCtrlHoldClearは、制御演算のホールド要求を解除します。

■ 関数名

I32 UcaCtrlHoldClear

■ 引数

● blockContext

- データ型： UcaBlockContext
- タイプ： IN
- 説明： ブロックコンテキスト

■ 戻り値

- SUCCEED： 正常終了
- UCAERR_PARA_PTRNULL： 引数のポインタが NULL
- UCAERR_BLKTYPE_INVALID： 正しくないブロック形を指定した

■ 処理内容説明

制御ホールドを解除します。

■ 詳細説明

システムは UcaRWSetPv において以下の条件が成立した場合、制御ホールド要求を解除します。

・ IN 端子の結合先がオープンの（切り換えスイッチなどで選択されていない）とき以外

通常は、ユーザカスタムアルゴリズムにおいて UcaCtrlHoldClear を呼び出す必要はありません。特に制御ホールド要求の解除が必要な場合に限り、UcaCtrlHoldClear を呼び出し制御ホールドを解除します。

6.4.3 UcaCtrlHoldSet

UcaCtrlHoldSetは、制御演算のホールド要求を設定します。

■ 関数名

I32 UcaCtrlHoldSet

■ 引数

● blockContext

- データ型： UcaBlockContext
- タイプ： IN
- 説明： ブロックコンテキスト

■ 戻り値

- SUCCEED： 正常終了
- UCAERR_PARA_PTRNULL： 引数のポインタが NULL
- UCAERR_BLKTYPE_INVALID： 正しくないブロック形を指定した

■ 処理内容説明

制御ホールドを設定します。

■ 詳細説明

システムは UcaRWReadIn において以下の条件が成立した場合、制御ホールド要求を設定します。

・ IN 端子の結合先がオープンの（切り換えスイッチなどで選択されていない）とき

通常は、ユーザカスタムアルゴリズムにおいて UcaCtrlHoldSet を呼び出す必要はありません。特に、制御ホールド要求の設定が必要な場合に限り、UcaCtrlHoldSet を呼び出し制御ホールドを設定します。

6.5 非線形ゲイン

非線形ゲイン機能は、比例ゲインを偏差の大きさに従って変化させ、偏差と制御出力変更量の関係を非線形とする機能です。非線形ゲイン演算には次の2種類があります。

- **ギャップ動作**
- **偏差二乗動作**

ユーザカスタムブロックでは、非線形ゲイン［ギャップ動作］、［偏差二乗動作］、［なし］のいずれかをビルダ定義項目で指定することができます。デフォルトは［なし］です。

■ 非線形ゲイン処理の概要

ユーザカスタムアルゴリズムにおいて以下の順序で関数を呼び出し、ギャップ動作処理を行います。

1. UcaConfigNonlinearGain を呼び出し、ビルダ定義項目を取得します。
2. ビルダ定義項目の非線形ゲインが［ギャップ動作］の場合、UcaCtrlGetGapGainCoef_p を呼び出し、ギャップゲイン値を取得します。
3. ギャップゲイン値を引数として UcaCtrlGapGain を呼び出し、ギャップ動作処理を行います。

ユーザカスタムアルゴリズムにおいて以下の順序で関数を呼び出し、偏差二乗動作処理を行います。

1. UcaConfigNonlinearGain を呼び出し、ビルダ定義項目を取得します。
2. ビルダ定義項目の非線形ゲインが［偏差二乗動作］の場合、UcaCtrlDvSquareGain_p を呼び出し、偏差二乗動作処理を行います。

参照 動作の詳細については、以下を参照してください。
機能ブロックリファレンス Vol.1（IM 33J15A30-01JA）「1.4 調節ブロックに共通の制御演算処理」の「■ 非線形ゲイン」

6.5.1 UcaCtrlGetGapGainCoef_p

UcaCtrlGetGapGainCoef_pは、ギャップゲインの値を取得します。

■ 関数名

I32 UcaCtrlGetGapGainCoef_p

■ 引数

● blockContext

- データ型： UcaBlockContext
- タイプ： IN
- 説明： ブロックコンテキスト

● coefPtr

- データ型： F64 *
- タイプ： OUT
- 説明： ギャップゲイン値ポインタ

● option

- データ型： U32
- タイプ： IN
- 説明： オプション

■ 戻り値

- SUCCEED： 正常終了
- UCAERR_PARA_PTRNULL： 引数のポインタが NULL
- UCAERR_OPT_INVALID： 誤ったオプションを指定した
- UCAERR_BLKTYPE_INVALID： 正しくないブロック形を指定した

■ 処理内容説明

ギャップゲイン（ビルダ定義項目）の値を取得する。

■ 引数説明

● option

NOOPTION を指定してください。

■ 詳細説明

UcaCtrlGetGapGainCoef_p は、制御演算タブシートのビルダ定義項目［ギャップゲイン］の値を取得し、引数 coefPtr に格納します。

6.5.2 UcaCtrlGapGain_p

UcaCtrlGapGain_pは、ギャップゲインの計算を行います。

■ 関数名

I32 UcaCtrlGapGain_p

■ 引数

● blockContext

- データ型： UcaBlockContext
- タイプ： IN
- 説明： ブロックコンテキスト

● knl

- データ型： F64
- タイプ： IN
- 説明： ギャップゲイン係数

● gw

- データ型： F64
- タイプ： IN
- 説明： ギャップ幅

● absEv

- データ型： F64
- タイプ： IN
- 説明： 実効偏差 EV の絶対値

● kpePtr

- データ型： F64 *
- タイプ： IN/OUT
- 説明： 実効比例ゲイン

● option

- データ型： U32
- タイプ： IN
- 説明： オプション

■ 戻り値

- SUCCEED： 正常終了
- UCAERR_CTRL_GAPGAIN： ギャップゲイン計算中に浮動小数演算で例外発生
- UCAERR_PARA_PTRNULL： 引数のポインタが NULL
- UCAERR_OPT_INVALID： 誤ったオプションを指定した
- UCAERR_BLKTYPE_INVALID： 正しくないブロック形を指定した

■ 処理内容説明

ギャップゲインの計算を行います。

■ 引数説明

● knl

ギャップゲイン係数
ギャップゲイン（ビルダ定義項目）の値。UcaCtrlGetGapGainCoef_p 関数で取得します。

● gw

ギャップ幅（データアイテム GW）

● absEv

実効偏差 EV の絶対値

● kpePtr

- （IN）比例ゲイン
- （OUT）実効比例ゲイン

● option

NOOPTION を指定してください。

■ 詳細説明

UcaCtrlGapGain_p は、非線形ゲインのギャップ動作の計算を行い、結果を引数 kpePtr に格納します。

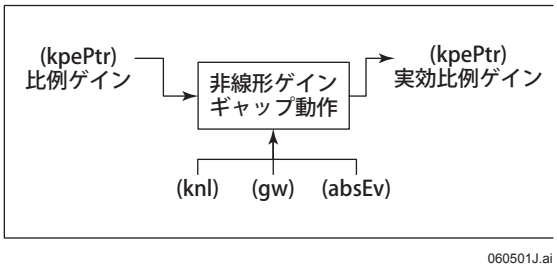


図　UcaCtrlGapGain_pのデータの流れ

ギャップ動作は、偏差があらかじめ設定したギャップ幅（データアイテム GW）の範囲内にある場合に、比例ゲインを小さくして調節作用をゆるやかにする機能です。

参照 動作の詳細については、以下を参照してください。
機能ブロックリファレンス Vol.1（IM 33J15A30-01JA）「1.4 調節ブロックに共通の制御演算処理」の「■ ギャップ動作」

6.5.3 UcaCtrlDvSquareGain_p

UcaCtrlDvSquareGain_pは、偏差二乗ゲインの計算を行います。

■ 関数名

I32 UcaCtrlDvSquareGain_p

■ 引数

● blockContext

- データ型： UcaBlockContext
- タイプ： IN
- 説明： ブロックコンテキスト

● gw

- データ型： F64
- タイプ： IN
- 説明： ギャップ幅

● absEv

- データ型： F64
- タイプ： IN
- 説明： 実効偏差 EV の絶対値

● kpePtr

- データ型： F64 *
- タイプ： IN/OUT
- 説明： 実効比例ゲイン

● option

- データ型： U32
- タイプ： IN
- 説明： オプション

■ 戻り値

- SUCCEED： 正常終了
- UCAERR_CTRL_DVSQUAREGAIN：
  偏差二乗ゲイン計算中に浮動小数演算で例外発生
- UCAERR_PARA_PTRNULL： 引数のポインタが NULL
- UCAERR_OPT_INVALID： 誤ったオプションを指定した
- UCAERR_BLKTYPE_INVALID： 正しくないブロック形を指定した

■ 処理内容説明

偏差二乗ゲインの計算を行います。

■ 引数説明

● gw

ギャップ幅（データアイテム GW）

● absEv

実効偏差 EV の絶対値

● kpePtr

- （IN）比例ゲイン
- （OUT）実効比例ゲイン

● option

NOOPTION を指定してください。

■ 詳細説明

UcaCtrlDvSquareGain_p は、非線形ゲインの偏差二乗動作の計算を行い、結果を引数 kpePtr に格納します。

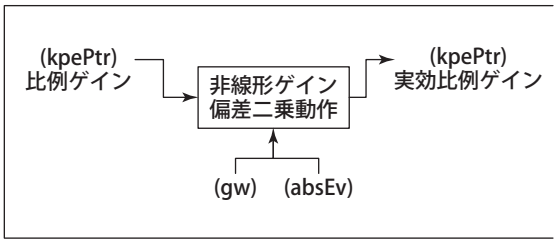


図　UcaCtrlDvSquareGain_pのデータの流れ

偏差二乗動作は、偏差があらかじめ設定したギャップ幅（データアイテム GW）の範囲内にある場合に、実効偏差の大きさに比例して比例ゲインを変化させる機能です。

参照 動作の詳細については、以下を参照してください。
機能ブロックリファレンス Vol.1（IM 33J15A30-01JA）「1.4 調節ブロックに共通の制御演算処理」の「■偏差二乗動作」

6.6 入出力補償

入出力補償は、PID制御演算の入力信号または出力信号に外部からの入出力補償値（VN）を加算する機能です。入出力補償には以下に示す2種類の制御動作があります。

- **入力補償**
- **出力補償**

ユーザカスタムブロックでは、入出力補償の［入力補償］、［出力補償］、［なし］のいずれかをビルダ定義項目で指定することができます。デフォルトは［なし］です。

■ 入出力補償処理の概要

ユーザカスタムアルゴリズムにおいて以下の順序で関数を呼び出し、入力補償処理を行います。

1. UcaConfigCompensation を呼び出し、ビルダ定義項目を取得します。
2. ビルダ定義項目の入出力補償が［入力補償］の場合、UcaCtrlCompensation を呼び出し、入力補償処理を行います。

ユーザカスタムアルゴリズムにおいて以下の順序で関数を呼び出し、出力補償処理を行います。

1. UcaConfigCompensation を呼び出し、ビルダ定義項目を取得します。
2. ビルダ定義項目の入出力補償が［出力補償］の場合、UcaCtrlCompensation を呼び出し、出力補償処理を行います。

参照 動作の詳細については、以下を参照してください。

機能ブロックリファレンス Vol.1（IM 33J15A30-01JA）「1.4 調節ブロックに共通の制御演算処理」の「■ 入出力補償」

機能ブロックリファレンス Vol.1（IM 33J15A30-01JA）「1.4 調節ブロックに共通の制御演算処理」の「■ 入力補償」

機能ブロックリファレンス Vol.1（IM 33J15A30-01JA）「1.4 調節ブロックに共通の制御演算処理」の「■ 出力補償」

6.6.1 UcaCtrlCompensation

UcaCtrlCompensationは、入出力補償処理を行います。

■ 関数名

I32 UcaCtrlCompensation

■ 引数

● blockContext

- データ型： UcaBlockContext
- タイプ： IN
- 説明： ブロックコンテキスト

● compPtr

- データ型： F64 *
- タイプ： IN/OUT
- 説明： 被補償データ

● option

- データ型： U32
- タイプ： IN
- 説明： オプション

■ 戻り値

- SUCCEED： 正常終了
- UCAERR_PARA_PTRNULL： 引数のポインタが NULL
- UCAERR_OPT_INVALID： 誤ったオプションを指定した
- UCAERR_BLKTYPE_INVALID： 正しくないブロック形を指定した

■ 処理内容説明

入出力補償処理を行います。

■ 引数説明

● compPtr

- （IN）
  入力補償の場合：測定値 PV
  出力補償の場合：出力補償前の操作出力（出力補償前出力値 Mvbb）
- （OUT）
  入力補償の場合：CV（入力補償後の PV）
  出力補償の場合：出力補償後の操作出力（出力リミット前出力値 Mvbl）

● option

以下の中から指定することができます。

- UCAOPT_CTRL_BIN：BIN 端子の入力処理を行います。
- NOOPTION

■ 詳細説明

UcaCtrlCompensation は、入出力補償処理を行います。入出力補償には以下に示す 2 種類の制御動作があります。

- 入力補償
- 出力補償

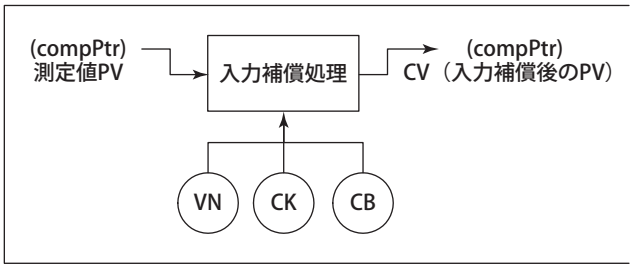


図　入力補償処理のデータの流れ

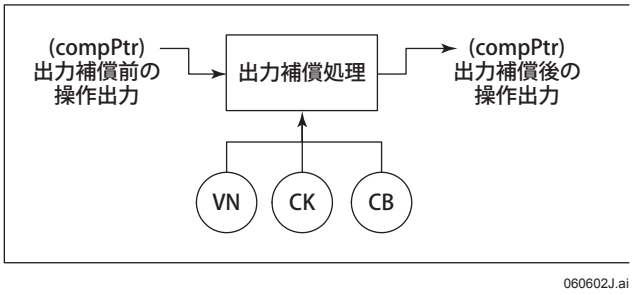


図　出力補償処理のデータの流れ

UcaCtrlCompensation は、引数 compPtr の値に対して入出力補償処理を行い、結果を引数 compPtr に格納します。入出力補償処理後に、入出力補償値（データアイテム VN）に 0.0 を設定します。

参照 動作の詳細については、以下を参照してください。

機能ブロックリファレンス Vol.1（IM 33J15A30-01JA）「1.4 調節ブロックに共通の制御演算処理」の「■ 入出力補償」

機能ブロックリファレンス Vol.1（IM 33J15A30-01JA）「1.4 調節ブロックに共通の制御演算処理」の「■ 入力補償」

機能ブロックリファレンス Vol.1（IM 33J15A30-01JA）「1.4 調節ブロックに共通の制御演算処理」の「■ 出力補償」

UcaCtrlCompensation は、オプションに NOOPTION を指定した場合、BIN 端子の入力処理を行いません。システムが UcaCtrlHandler において BIN 端子入力処理を行います。オプションに UCAOPT_CTRL_BIN を指定した場合、UcaCtrlCompensation は BIN 端子からデータを読み込み、データアイテム VN に格納します。ユーザカスタムアルゴリズムにおいて UcaCtrlHandler を使用しない場合には、オプションに UCAOPT_CTRL_BIN を指定し、BIN 端子の入力処理を行ってください。

6.6.2 UcaCtrlCompensation_p

UcaCtrlCompensation_pは、入出力補償計算を行います。

■ 関数名

I32 UcaCtrlCompensation_p

■ 引数

● blockContext

- データ型： UcaBlockContext
- タイプ： IN
- 説明： ブロックコンテキスト

● vnPtr

- データ型： F32S *
- タイプ： IN/OUT
- 説明： 入出力補償値ポインタ

● ck

- データ型： F32
- タイプ： IN
- 説明： 入出力補償ゲイン

● cb

- データ型： F32
- タイプ： IN
- 説明： 入出力補償バイアス

● compPtr

- データ型： F64 *
- タイプ： IN/OUT
- 説明： 被補償データ

● option

- データ型： U32
- タイプ： IN
- 説明： オプション

■ 戻り値

- SUCCEED： 正常終了
- UCAERR_PARA_PTRNULL： 引数のポインタが NULL
- UCAERR_OPT_INVALID： 誤ったオプションを指定した
- UCAERR_BLKTYPE_INVALID： 正しくないブロック形を指定した

■ 処理内容説明

入出力補償計算を行います。

■ 引数説明

● vnPtr

入出力補償値ポインタ（バイアス信号）

● ck

入出力補償ゲイン

● cb

入出力補償バイアス（内部バイアス）

● compPtr

- （IN）
  入力補償の場合：測定値 PV
  出力補償の場合：出力補償前の操作出力（補償前出力値 Mvbb）
- （OUT）
  入力補償の場合：CV（入力補償後の PV）
  出力補償の場合：出力補償後の操作出力（リミット前出力値 Mvbl）

● option

NOOPTION を指定してください。

■ 詳細説明

UcaCtrlCompensation_p は、引数 compPtr に対して入出力補償計算を行い、結果を引数 compPtr に格納します。入出力補償計算後に、入出力補償値（引数 vnPtr）に 0.0 を設定します。

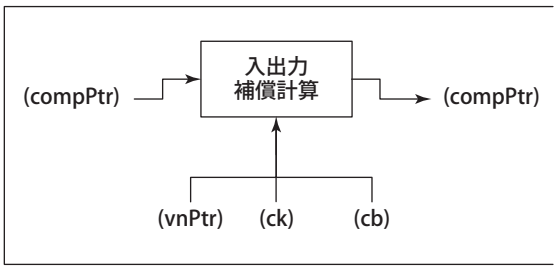


図　UcaCtrlCompensation_pのデータの流れ

参照 動作の詳細については、以下を参照してください。

機能ブロックリファレンス Vol.1（IM 33J15A30-01JA）「1.4 調節ブロックに共通の制御演算処理」の「■入出力補償」

機能ブロックリファレンス Vol.1（IM 33J15A30-01JA）「1.4 調節ブロックに共通の制御演算処理」の「■入力補償」

機能ブロックリファレンス Vol.1（IM 33J15A30-01JA）「1.4 調節ブロックに共通の制御演算処理」の「■出力補償」

6.7 不感帯動作

不感帯動作は、偏差があらかじめ設定された不感帯幅内のときは、制御を停止させる非線形制御です。ユーザカスタムブロックでは、不感帯動作の［あり］、［なし］のいずれかをビルダ定義項目で指定することができます。デフォルトは［なし］です。

■ 不感帯動作処理の概要

ユーザカスタムアルゴリズムにおいて以下の順序で関数を呼び出し、不感帯動作処理を行います。

1. UcaConfigDeadband を呼び出し、ビルダ定義項目を取得します。
2. ビルダ定義項目の不感帯動作が［あり］の場合、UcaCtrlDeadband を呼び出し、不感帯動作処理を行います。

参照 ビルダ定義項目取得関数については、以下を参照してください。
「5.1.5 UcaConfigDeadband」

不感帯動作の詳細については、以下を参照してください。
機能ブロックリファレンス Vol.1（IM 33J15A30-01JA）「1.4 調節ブロックに共通の制御演算処理」の「■ 不感帯動作」

6.7.1 UcaCtrlDeadband

UcaCtrlDeadbandは、不感帯動作処理を行います。

■ 関数名

I32 UcaCtrlDeadband

■ 引数

● blockContext

- データ型： UcaBlockContext
- タイプ： IN
- 説明： ブロックコンテキスト

● ev

- データ型： F64
- タイプ： IN
- 説明： 実効偏差

● deltaMvPtr

- データ型： F64 *
- タイプ： IN/OUT
- 説明： 操作出力変更量

● option

- データ型： U32
- タイプ： IN
- 説明： オプション

■ 戻り値

- SUCCEED： 正常終了
- UCAERR_PARA_PTRNULL： 引数のポインタが NULL
- UCAERR_OPT_INVALID： 誤ったオプションを指定した
- UCAERR_BLKTYPE_INVALID： 正しくないブロック形を指定した

■ 処理内容説明

不感帯動作処理を行います。

■ 引数説明

● ev

実効偏差（PV と同じ単位のデータ）

● deltaMvPtr

- （IN）： 不感帯動作前の操作出力変更量
- （OUT）： 不感帯動作後の操作出力変更量

● option

NOOPTION を指定してください。

■ 詳細説明

UcaCtrlDeadband は、引数の実効偏差 ev の絶対値がデータアイテムの不感帯幅（データアイテム DB）内にあれば、出力値（引数 deltaMvPtr）に 0.0 を設定します。不感帯にはじめて入ったときには不感帯幅（データアイテム DB）にビルダ定義項目のヒステリシスを加えます。

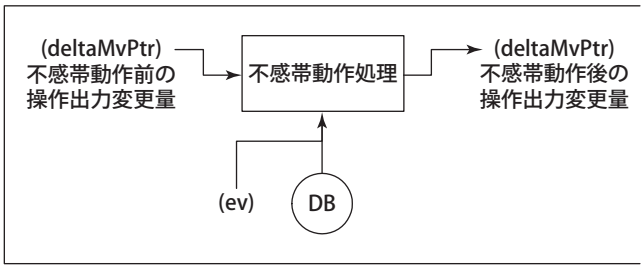


図 UcaCtrlDeadbandのデータの流れ

参照 動作の詳細については、以下を参照してください。
機能ブロックリファレンス Vol.1（IM 33J15A30-01JA）「1.4 調節ブロックに共通の制御演算処理」の「■ 不感帯動作」

6.7.2 UcaCtrlDeadband_p

UcaCtrlDeadband_pは不感帯の計算を行います。

■ 関数名

I32 UcaCtrlDeadband_p

■ 引数

● blockContext

- データ型： UcaBlockContext
- タイプ： IN
- 説明： ブロックコンテキスト

● deadBand

- データ型： F32
- タイプ： IN
- 説明： 不感帯幅

● dbandHys

- データ型： F32
- タイプ： IN
- 説明： 不感帯ヒステリシス

● ev

- データ型： F64
- タイプ： IN
- 説明： 実効偏差

● deltaMvPtr

- データ型： F64 *
- タイプ： IN/OUT
- 説明： 操作出力変更量

● lastCondPtr

- データ型： I32 *
- タイプ： IN/OUT
- 説明： 前回運転状態ポインタ

● option

- データ型： U32
- タイプ： IN
- 説明： オプション

■ 戻り値

- SUCCEED： 正常終了
- UCAERR_PARA_PTRNULL： 引数のポインタが NULL
- UCAERR_OPT_INVALID： 誤ったオプションを指定した
- UCAERR_BLKTYPE_INVALID： 正しくないブロック形を指定した

■ 処理内容説明

不感帯動作の出力値を計算します。

■ 引数説明

● deadBand

不感帯幅

● dbandHys

不感帯動作のヒステリシス（PV と同じ単位のデータ）

● ev

実効偏差（PV と同じ単位のデータ）

● deltaMvPtr

- （IN）： 不感帯動作前の操作出力変更量
- （OUT）： 不感帯動作後の動作出力変更量

● lastCondPtr

前回運転状態ポインタ

- UCA_DEADBAND_SMALL： 偏差小
- UCA_DEADBAND_LARGEPLUS： 正方向の偏差大
- UCA_DEADBAND_LARGEMINUS： 負方向の偏差大

■ 引数説明

● option

NOOPTION を指定してください。

■ 詳細説明

UcaCtrlDeadband_p は、引数の実効偏差 ev の値が引数の不感帯（引数 deadBand）内にあれば、出力値（引数 deltaMvPtr）に 0.0 を設定します。不感帯にはじめて入ったときには、不感帯幅にヒステリシスを加えます。

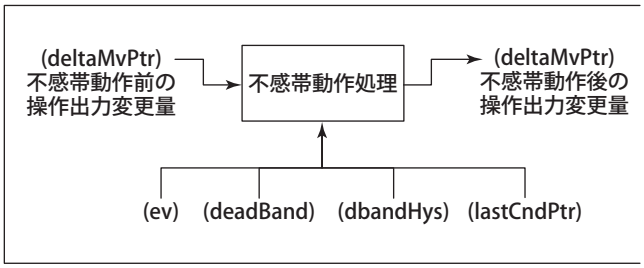


図　UcaCtrlDeadband_pのデータの流れ

引数 lastCondPtr には、前回 UcaCtrlDeadband_p を呼び出し時に下図のどの状態にあったかを示す前回運転状態を指定してください。UcaCtrlDeadband_p は、引数 lastCondPtr に今回の運転状態を格納します。UcaCtrlDeadband_p 呼び出し後、引数 lastCondPtr の値を、外部変数またはデータアイテム Pn に保存してください。次回 UcaCtrlDeadband_p を呼び出し時に、その格納しておいた運転状態の値を、引数 lastCondPtr に指定してください。UcaCtrlDeadband_p を初回呼び出すときや演算初期化時には、初回運転状態として UCA_DEADBAND_SMALL（偏差小）を指定してください。

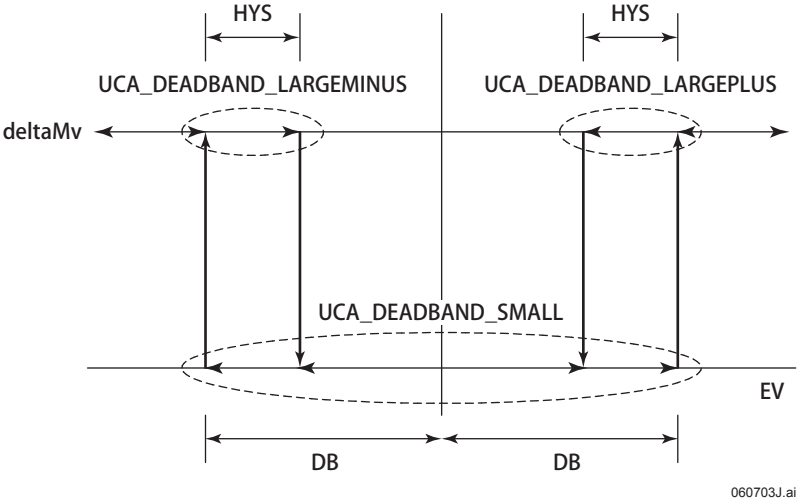


図　運転状態

引数 lastCondPtr に NULL を指定した場合、システムが前回運転状態を管理します。運転状態を外部変数やデータアイテムに保存しておく必要はありません。

参照 動作の詳細については、以下を参照してください。
機能ブロックリファレンス Vol.1（IM 33J15A30-01JA）「1.4 調節ブロックに共通の制御演算処理」の「■不感帯動作」

6.8 リセットリミット

リセットリミット機能は、積分動作による出力上昇分、または下降分に制限値を設けるアンチリセットワインドアップ（積分項の飽和）機能です。

参照 動作の詳細については、以下を参照してください。
機能ブロックリファレンス Vol.1（IM 33J15A30-01JA）「1.4 調節ブロックに共通の制御演算処理」の「■リセットリミット機能」

6.8.1 UcaCtrlResetLimitOprt

UcaCtrlResetLimitOprtは、リセットリミット処理を行います。

■ 関数名

I32 UcaCtrlResetLimitOprt

■ 引数

● blockContext

- データ型： UcaBlockContext
- タイプ： IN
- 説明： ブロックコンテキスト

● deltaMvPtr

- データ型： F64 *
- タイプ： IN/OUT
- 説明： 演算出力差分値

● option

- データ型： U32
- タイプ： IN
- 説明： オプション

■ 戻り値

- SUCCEED： 正常終了
- UCAERR_CTRL_RESETLIMITTI： リセットリミット計算で積分時間が異常
- UCAERR_PARA_PTRNULL： 引数のポインタが NULL
- UCAERR_OPT_INVALID： 誤ったオプションを指定した
- UCAERR_BLKTYPE_INVALID： 正しくないブロック形を指定した

■ 処理内容説明

リセットリミットの処理を行います。

■ 引数説明

● deltaMvPtr

- （IN）： 補正前操作出力変更量
- （OUT）： 補正後操作出力変更量

● option

以下の中から複数指定することができます。複数指定する場合は、それぞれを‘|’（or）で結んでください。

- UCAOPT_RL_RLCON： RL1、RL2 端子の入力処理を行います。
- UCAOPT_CTRL_TSCTIME： 制御時間に実効スキャン周期を使用します。
- NOOPTION

■ 詳細説明

UcaCtrlResetLimitOprt は、リセットリミッタ処理を行い、結果を引数 deltaMvPtr に格納します。

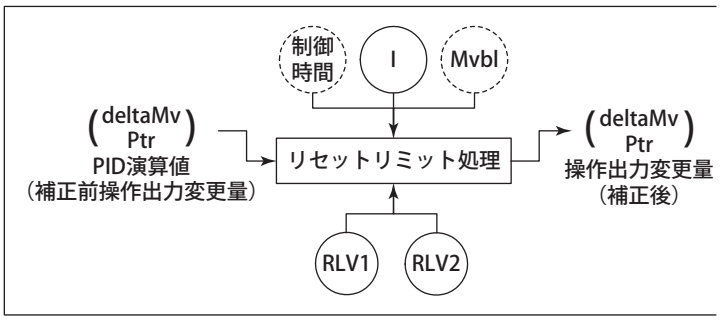


図　UcaCtrlResetLimitOprtのデータの流れ

UcaCtrlResetLimitOprt は、制御演算の出力値（引数 deltaMvPtr）に対してリセット信号（データアイテム RLV1、RLV2）を使って補正演算を行い、出力値（引数 deltaMvPtr）の積分項を制限します。

参照 動作の詳細については、以下を参照してください。
機能ブロックリファレンス Vol.1（IM 33J15A30-01JA）「1.4 調節ブロックに共通の制御演算処理」の「■ リセットリミット機能」

UcaCtrlResetLimitOprt は、オプションに UCAOPT_CTRL_TSCTIME を指定しなかった場合、制御時間に制御周期カウンタ値を使用します。オプションに UCAOPT_CTRL_TSCTIME を指定した場合、制御時間に実効スキャン周期を使用します。ユーザカスタムアルゴリズムにおいて制御周期カウンタを使用しない場合には、オプションに UCAOPT_CTRL_TSCTIME を指定してください。
UcaCtrlResetLimitOprt は、オプションに UCAOPT_RL_RLCON を指定しなかった場合、RL1/RL2 端子からデータを読み込みません。通常は、システムが関数 UcaCtrlHandler 内で RL1/RL2 端子の入力処理を行います。
UcaCtrlrResetLimitOprt は、オプションに UCAOPT_CTRL_RL を指定した場合、RL1/RL2 端子からデータを読み込み、データアイテム RLV1、RLV2 に格納します。ユーザカスタムアルゴリズムで関数 UcaCtrlHandler を使用しない場合には、オプションに UCAOPT_RW_RLCON を指定して、RL1、RL2 端子の入力処理を行ってください。

6.8.2 UcaCtrlResetLimitOprt_p

UcaCtrlResetLimitOprt_pは、リセットリミットの計算を行います。

■ 関数名

I32 UcaCtrlResetLimitOprt_p

■ 引数

● blockContext

- データ型： UcaBlockContext
- タイプ： IN
- 説明： ブロックコンテキスト

● deltaMvPtr

- データ型： F64 *
- タイプ： IN/OUT
- 説明： 演算出力差分値

● rlv1Ptr

- データ型： F32S *
- タイプ： IN
- 説明： リセット信号 1

● rlv2Ptr

- データ型： F32S *
- タイプ： IN
- 説明： リセット信号 2

● ti

- データ型： F32
- タイプ： IN
- 説明： 積分時間

● lastMvbl

- データ型： F64
- タイプ： IN
- 説明： 制限前前回値出力

● option

- データ型： U32
- タイプ： IN
- 説明： オプション

■ 戻り値

- SUCCEED： 正常終了
- UCAERR_CTRL_RESETLIMITTI： リセットリミット計算で積分時間が異常
- UCAERR_PARA_PTRNULL： 引数のポインタが NULL
- UCAERR_OPT_INVALID： 誤ったオプションを指定した
- UCAERR_BLKTYPE_INVALID： 正しくないブロック形を指定した

■ 処理内容説明

リセットリミットの計算を行います。

■ 引数説明

● deltaMvPtr

- （IN）： 補正前操作出力変更量
- （OUT）： 補正後操作出力変更量

● rlv1Ptr

リセット信号 1

● rlv2Ptr

リセット信号 2

● ti

積分時間

● lastMvbl

前回出力リミット前出力値（内部変数 Mvbl）

● option

以下の中から指定することができます。

- UCAOPT_CTRL_TSCTIME：制御時間に実効スキャン周期を使用します。
- NOOPTION

■ 詳細説明

UcaCtrlResetLimitOprt_p は、リセットリミットの計算を行い、結果を引数 deltaMvPtr に格納します。

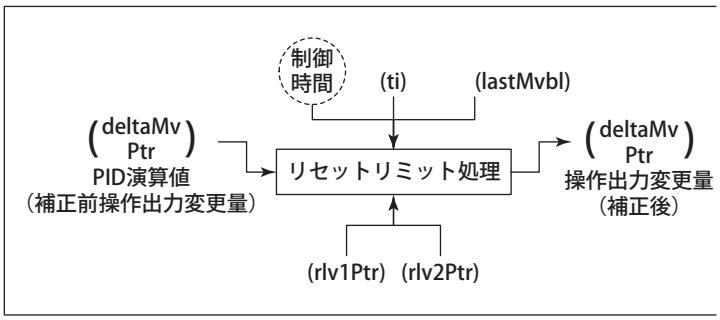


図　UcaCtrlResetLimitOprt_pのデータの流れ

UcaCtrlResetLimitOprt_p は、制御演算の出力値 deltaMvPtr に対してリセット信号（rlv1Ptr、rlv2Ptr）を使って補正演算を行います。

参照 動作の詳細については、以下を参照してください。
機能ブロックリファレンス Vol.1（IM 33J15A30-01JA）「1.4 調節ブロックに共通の制御演算処理」の「■ リセットリミット機能」

UcaCtrlResetLimitOprt_p は、オプションに NOOPTION を指定した場合、制御時間に制御周期カウンタ値を使用します。オプションに UCAOPT_CTRL_TSCTIME を指定した場合、制御時間に実効スキャン周期を使用します。ユーザカスタムアルゴリズムにおいて制御周期カウンタを使用しない場合には、オプションに UCAOPT_CTRL_TSCTIME を指定してください。

7. 機能ブロック

機能ブロック関数には、機能ブロックからデータを取得する関数と、機能ブロックへデータを設定する関数があります。

■ 機能ブロック関数

取得／設定するデータ型や、機能ブロックの存在するステーションにより以下のように分類できます。

表　タグ名を指定したデータ入出力の関数

ステーション / データ型／読み書き	自ステーション（APCS内）	他ステーション（APCS外）
数値データ読み込み（副入力 RVnn 設定付き）	UcaTagReadToRvnF64S	UcaOtherTagReadToRvnF64S
数値データ読み込み	UcaTagReadF64S	UcaOtherTagReadF64S
	UcaTagReadI32S	UcaOtherTagReadI32S
	UcaTagRead	UcaOtherTagRead
数値データ書き込み	UcaTagReadF64S	UcaOtherTagReadF64S
	UcaTagReadI32S	UcaOtherTagReadI32S
文字列データ読み込み	UcaTagReadString	なし
文字列データ書き込み	UcaTagWriteString	なし
汎用データ型読み込み	UcaTagRead	UcaOtherTagRead
汎用データ型書き込み	UcaTagWrite	UcaOtherTagWrite

7.1 自ステーションタグデータ

本節でまとめた関数はユーザカスタムブロックと同一ステーションに存在する機能ブロックのデータを、取得／設定します。

■ 自ステーションタグデータアクセス関数

以下に APCS 内データアクセスに使用するライブラリを示します。

表　タグ名を指定したデータ入出力の関数（自ステーション）

ステーション / データ型／読み書き	自ステーション（APCS内）
数値データ読み込み	UcaTagReadF64S
	UcaTagReadI32S
	UcaTagRead
数値データ書き込み	UcaTagReadF64S
	UcaTagReadI32S
文字列データ読み込み	UcaTagReadString
文字列データ書き込み	UcaTagWriteString
汎用データ型読み込み	UcaTagRead
汎用データ型書き込み	UcaTagWrite

7.1.1 UcaTagReadF64S

UcaTagReadF64Sは、自ステーションの機能ブロックのデータを参照し、F64S型データを格納します。

■ 関数名

I32 UcaTagReadF64S

■ 引数

● blockContext

- データ型： UcaBlockContext
- タイプ： IN
- 説明： ブロックコンテキスト

● tagName

- データ型： TAGN
- タイプ： IN
- 説明： タグ名

● ditmName

- データ型： DITMN
- タイプ： IN
- 説明： データアイテム名

● array1

- データ型： I32
- タイプ： IN
- 説明： 配列添え字 1

● array2

- データ型： I32
- タイプ： IN
- 説明： 配列添え字 2

● valPtr

- データ型： F64S *
- タイプ： OUT
- 説明： 入力値ポインタ

● option

- データ型： U32
- タイプ： IN
- 説明： オプション

■ 戻り値

- SUCCEED： 成功
- UCAERR_PARA_PTRNULL： 引数のポインタが NULL
- UCAERR_OPT_INVALID： 誤ったオプションを指定した

■ 処理内容説明

タグ名とデータアイテム名を指定して、データを参照します。

■ 引数説明

● tagName

参照先のタグ名

● ditmName

参照先のデータアイテム名

● array1、array2

配列添え字 1、2

● valPtr

入力値ポインタ

● option

以下の中から指定することができます。

- UCAOPT_TAG_SETIOP：データ参照異常時に、IOP アラームを発生します。
- NOOPTION

■ 詳細説明

UcaTagReadF64S は、自ステーションの機能ブロックからデータを参照し、F64S 型の引数 valPtr に格納します。

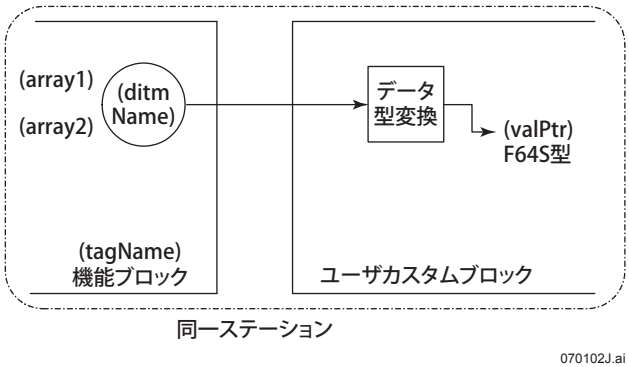


図　UcaTagReadF64Sのデータの流れ

参照先機能ブロックは同一ステーション内に限ります。
引数 tagName、ditmName の英字部分は、大文字で記述してください。
引数 array1、array2 には、ditmName で指示されるデータアイテムが配列型でない場合、0 を指定してください。1 次元配列データの場合、array1 には配列添え字を、array2 には 0 を指定してください。2 次元配列データの場合、array1 には配列添え字 1 を、array2 には配列添え字 2 を指定してください。
UcaTagReadF64S は、機能ブロックのデータ参照に失敗した場合、引数 valPtr のデータステータスに BAD を設定し、データ値に 0 を設定します。
UcaTagReadF64S は、参照先のデータ型を F64S 型に変換し、引数 valPtr に格納します。参照先のデータ型がデータステータスつきであれば、そのデータステータスも同時に取得します。参照先のデータ型がデータステータスなしであれば、引数 valPtr のデータステータスに 0（正常）を設定します。
UcaTagReadF64S は、以下のアラームを検出します。

表　UcaTagReadF64Sのアラーム検出

オプション / アラーム	NOOPTION	UCAOPT_TAG_SETIOP
IOP	×	○
CNF	×	○

○： 検出する
×： 検出しない

オプションに UCAOPT_TAG_SETIOP を指定すると、機能ブロックのデータ参照に失敗した場合、プロセスアラーム IOP と CNF を検出します。参照に成功した場合でも、取得したデータのデータステータスが BAD のときにはプロセスアラーム IOP を設定します。
警報タブシートのビルダ定義項目［入力オープン警報］や［結合状態不良警報］になしを指定した場合、対応するアラームは検出しません。

7.1.2 UcaTagReadI32S

UcaTagReadI32Sは、自ステーションの機能ブロックのデータを参照し、I32S型データを格納します。

■ 関数名

I32 UcaTagReadI32S

■ 引数

● blockContext

- データ型： UcaBlockContext
- タイプ： IN
- 説明： ブロックコンテキスト

● tagName

- データ型： TAGN
- タイプ： IN
- 説明： タグ名

● ditmName

- データ型： DITMN
- タイプ： IN
- 説明： データアイテム名

● array1

- データ型： I32
- タイプ： IN
- 説明： 配列添え字 1

● array2

- データ型： I32
- タイプ： IN
- 説明： 配列添え字 2

● valPtr

- データ型： I32S *
- タイプ： OUT
- 説明： 入力値ポインタ

● option

- データ型： U32
- タイプ： IN
- 説明： オプション

■ 戻り値

- SUCCEED： 成功
- UCAERR_PARA_PTRNULL： 引数のポインタが NULL
- UCAERR_OPT_INVALID： 誤ったオプションを指定した

■ 処理内容説明

タグ名とデータアイテム名を指定して、データを参照します。

■ 引数説明

● tagName

参照先のタグ名

● ditmName

参照先のデータアイテム名

● array1、array2

配列添え字 1、2

● valPtr

入力値ポインタ

● option

以下の中から指定することができます。

- UCAOPT_TAG_SETIOP：データ参照異常時に、IOP アラームを発生します。
- NOOPTION

■ 詳細説明

UcaTagReadI32S は、自ステーションの機能ブロックからデータを参照し、I32S 型の引数 valPtr に格納します。

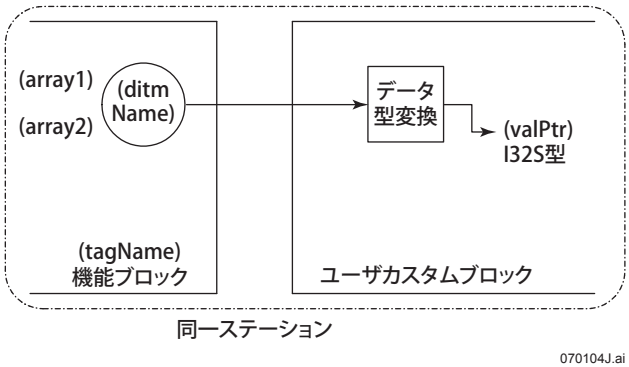


図　UcaTagReadI32Sのデータの流れ

参照先機能ブロックは同一ステーション内に限ります。
引数 tagName、ditmName の英字部分は、大文字で記述してください。
引数 array1、array2 には、ditmName で指示されるデータアイテムが配列型でない場合、0 を指定してください。1 次元配列データの場合、array1 には配列添え字を、array2 には 0 を指定してください。2 次元配列データの場合、array1 には配列添え字 1 を、array2 には配列添え字 2 を指定してください。
UcaTagReadI32S は、機能ブロックのデータ参照に失敗した場合、引数 valPtr のデータステータスに BAD を設定し、データ値に 0 を設定します。
UcaTagReadI32S は、参照先のデータ型を I32S に変換し、引数 valPtr に格納します。参照先のデータ型がデータステータスつきであれば、そのデータステータスも同時に取得します。参照先のデータ型がデータステータスなしであれば、引数 valPtr のデータステータスに 0（正常）を設定します。
UcaTagReadI32S は、以下のアラームを検出します。

表　UcaTagReadI32Sのアラーム検出

オプション / アラーム	NOOPTION	UCAOPT_TAG_SETIOP
IOP	×	○
CNF	×	○

○： 検出する
×： 検出しない

オプションに UCAOPT_TAG_SETIOP を指定すると、機能ブロックのデータ参照に失敗した場合、プロセスアラーム IOP と CNF を検出します。参照に成功した場合でも、取得したデータのデータステータスが BAD のときにはプロセスアラーム IOP を検出します。
警報タブシートのビルダ定義項目［入力オープン警報］や［結合状態不良警報］になしを指定した場合、対応するアラームは検出しません。

7.1.3 UcaTagReadString

UcaTagReadStringは、自ステーションの機能ブロックからデータを参照し、文字列型データに格納します。

■ 関数名

I32 UcaTagReadString

■ 引数

● blockContext

- データ型： UcaBlockContext
- タイプ： IN
- 説明： ブロックコンテキスト

● tagName

- データ型： TAGN
- タイプ： IN
- 説明： タグ名

● ditmName

- データ型： DITMN
- タイプ： IN
- 説明： データアイテム名

● array1

- データ型： I32
- タイプ： IN
- 説明： 配列添え字 1

● array2

- データ型： I32
- タイプ： IN
- 説明： 配列添え字 2

● valPtr

- データ型： BYTE *
- タイプ： OUT
- 説明： 入力値ポインタ

● option

- データ型： U32
- タイプ： IN
- 説明： オプション

■ 戻り値

- SUCCEED： 成功
- UCAERR_TAG_NOSTRING： 文字列データでない
- UCAERR_PARA_PTRNULL： 引数のポインタが NULL
- UCAERR_OPT_INVALID： 誤ったオプションを指定した

■ 処理内容説明

タグ名とデータアイテム名を指定して、データを参照します。

■ 引数説明

● tagName

参照先のタグ名

● ditmName

参照先のデータアイテム名

● array1、array2

配列添え字 1、2

● valPtr

入力値ポインタ

● option

NOOPTION を指定してください。

■ 詳細説明

UcaTagReadString は、自ステーションの機能ブロックからデータを参照し、文字列型の引数 valPtr に格納します。

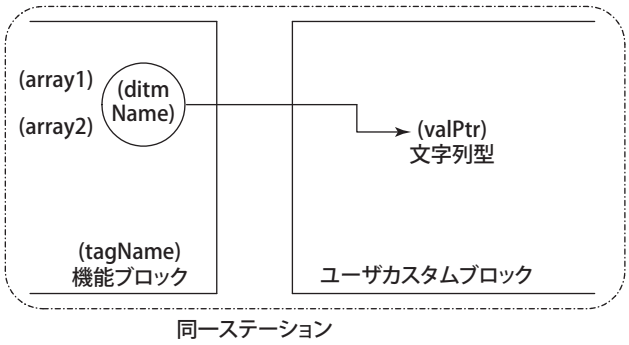


図　UcaTagReadStringのデータの流れ

参照先機能ブロックは同一ステーション内に限ります。
引数 tagName、ditmName の英字部分は、大文字で記述してください。
引数 array1、array2 には、ditmName で指示されるデータアイテムが配列型でない場合、array1、array2 には 0 を指定してください。1 次元配列データの場合、array1 には配列添え字を、array2 には 0 を指定してください。2 次元配列データの場合、array1 には配列添え字 1 を、array2 には配列添え字 2 を指定してください。

重要 UcaTagReadString は、参照先データを引数 valPtr に格納します。入力文字列の最大長は 16 バイトです。UcaTagReadString は、文字列の終端に '¥0' を付加します。機能ブロックの文字列データを読み込む領域（引数 valPtr の指す領域）には、文字列のデータアイテムの最大サイズ 16 バイトに、'¥0' の分 1 バイトを加え、17 バイト以上を確保してください。

7.1.4 UcaTagRead

UcaTagReadは、自ステーションの機能ブロックからデータを参照し、UcaUnivType型データに格納します。

■ 関数名

I32 UcaTagRead

■ 引数

● blockContext

- データ型： UcaBlockContext
- タイプ： IN
- 説明： ブロックコンテキスト

● tagName

- データ型： TAGN
- タイプ： IN
- 説明： タグ名

● ditmName

- データ型： DITMN
- タイプ： IN
- 説明： データアイテム名

● array1

- データ型： I32
- タイプ： IN
- 説明： 配列添え字 1

● array2

- データ型： I32
- タイプ： IN
- 説明： 配列添え字 2

● valPtr

- データ型： UcaUnivType *
- タイプ： OUT
- 説明： 入力値ポインタ

● option

- データ型： U32
- タイプ： IN
- 説明： オプション

■ 戻り値

- SUCCEED： 成功
- UCAERR_TAG_INVALIDPARA： 引数が正しくない
- UCAERR_PARA_PTRNULL： 引数のポインタが NULL
- UCAERR_OPT_INVALID： 誤ったオプションを指定した

■ 処理内容説明

タグ名とデータアイテム名を指定して、データを参照します。

■ 引数説明

● tagName

参照先のタグ名

● ditmName

参照先のデータアイテム名

● array1、array2

配列添え字 1、2

● valPtr

入力値ポインタ

● option

NOOPTION を指定してください。

■ 詳細説明

UcaTagRead は、自ステーションの機能ブロックからデータを参照し、UcaUnivType 型の引数 valPtr に格納します。

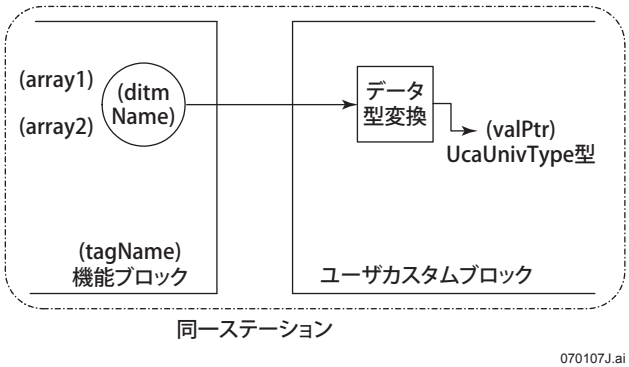


図　UcaTagReadのデータの流れ

参照先機能ブロックは同一ステーション内に限ります。
引数 tagName、ditmName の英字部分は、大文字で記述してください。
引数 array1、array2 には、ditmName で指示されるデータアイテムが配列型でない場合、0 を指定してください。1 次元配列データの場合、array1 には配列添え字を、array2 には 0 を指定してください。2 次元配列データの場合、array1 には配列添え字 1 を、array2 には配列添え字 2 を指定してください。
UcaTagRead は、参照先データを引数 valPtr に格納します。データ値を valPtr->dataValue に、データ型を valPtr->dType に格納します。UcaTagRead を呼び出す前に、あらかじめ valPtr->indOpt、valPtr->rqstType を 0 に初期化しておいてください。valPtr->indOpt と valPtr->rqstType が 0 以外の場合、UcaTagRead はエラーコード UCAERR_TAG_INVALIDPARA を返します。

参照 UcaUnivType 型のデータは、UcaDataConvertType により、特定のデータ型に変換することができます。UcaDataConvertType については、以下を参照してください。
「9.1.1 UcaDataConvertType」

7.1.5 UcaTagWriteF64S

UcaTagWriteF64Sは、自ステーションの機能ブロックにF64S型データを設定します。

■ 関数名

I32 UcaTagWriteF64S

■ 引数

● blockContext

- データ型： UcaBlockContext
- タイプ： IN
- 説明： ブロックコンテキスト

● tagName

- データ型： TAGN
- タイプ： IN
- 説明： タグ名

● ditmName

- データ型： DITMN
- タイプ： IN
- 説明： データアイテム名

● array1

- データ型： I32
- タイプ： IN
- 説明： 配列添え字 1

● array2

- データ型： I32
- タイプ： IN
- 説明： 配列添え字 2

● valPtr

- データ型： F64S *
- タイプ： IN
- 説明： 出力値ポインタ

● option

- データ型： U32
- タイプ： IN
- 説明： オプション

■ 戻り値

- SUCCEED： 成功
- UCAERR_NOTANUMBER：
  引数の浮動小数データが非数（Not a Number）である
- UCAERR_INFINITY： 引数の浮動小数データが無限大である
- UCAERR_PARA_PTRNULL： 引数のポインタが NULL
- UCAERR_OPT_INVALID： 誤ったオプションを指定した

■ 処理内容説明

タグ名とデータアイテム名を指定して、データを設定します。

■ 引数説明

● tagName

設定先のタグ名

● ditmName

設定先のデータアイテム名

● array1、array2

配列添え字 1、2

● valPtr

出力値ポインタ

● option

NOOPTION を指定してください。

■ 詳細説明

UcaTagWriteF64S のデータの流れは以下のとおりです。

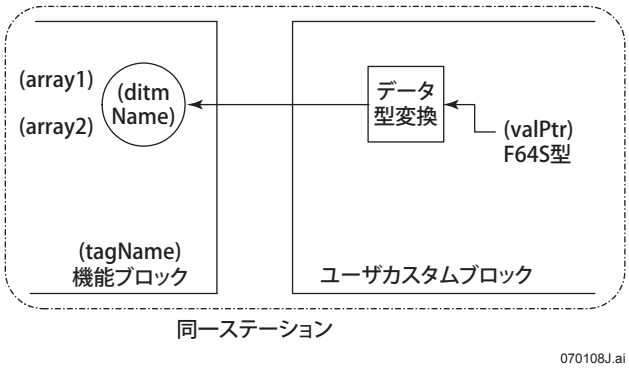


図　UcaTagWriteF64Sのデータの流れ

設定先機能ブロックは同一ステーション内に限ります。
引数 tagName、ditmName の英字部分は、大文字で記述してください。
引数 array1、array2 には、ditmName で指示されるデータアイテムが配列型でない場合、0 を指定してください。1 次元配列データの場合、array1 には配列添え字を、array2 には 0 を指定してください。2 次元配列データの場合、array1 には配列添え字 1 を、array2 には配列添え字 2 を指定してください。
引数 valPtr には、F64S 型でデータを格納してください。UcaTagWriteF64S は、出力先に書き込む時に出力先のデータ型に変換します。
UcaTagWriteF64S は、引数 valPtr のデータステータスのうち、BAD、QST のみを設定先に伝えます。それ以外のデータステータスは無視します。正常データであることを伝えるためには、引数 valPtr のデータステータスに 0 を設定してください。
UcaTagWriteF64S は、引数 valPtr のデータ値が非数または無限大の場合、機能ブロックへの出力は行いません。引数 valPtr の値が非数の場合はエラーコード UCAERR_NOTANUMBER、無限大の場合はエラーコード UCAERR_INFINITY を返します。

7.1.6 UcaTagWriteI32S

UcaTagWriteI32Sは、自ステーションの機能ブロックにI32S型データを設定します。

■ 関数名

I32 UcaTagWriteI32S

■ 引数

● blockContext

- データ型： UcaBlockContext
- タイプ： IN
- 説明： ブロックコンテキスト

● tagName

- データ型： TAGN
- タイプ： IN
- 説明： タグ名

● ditmName

- データ型： DITMN
- タイプ： IN
- 説明： データアイテム名

● array1

- データ型： I32
- タイプ： IN
- 説明： 配列添え字 1

● array2

- データ型： I32
- タイプ： IN
- 説明： 配列添え字 2

● valPtr

- データ型： I32S *
- タイプ： IN
- 説明： 出力値ポインタ

● option

- データ型： U32
- タイプ： IN
- 説明： オプション

■ 戻り値

- SUCCEED： 成功
- UCAERR_PARA_PTRNULL： 引数のポインタが NULL
- UCAERR_OPT_INVALID： 誤ったオプションを指定した

■ 処理内容説明

タグ名とデータアイテム名を指定して、データを設定します。

■ 引数説明

● tagName

設定先のタグ名

● ditmName

設定先のデータアイテム名

● array1、array2

配列添え字 1、2

● valPtr

出力値ポインタ

● option

NOOPTION を指定してください。

■ 詳細説明

UcaTagWriteI32S のデータの流れは以下のとおりです。

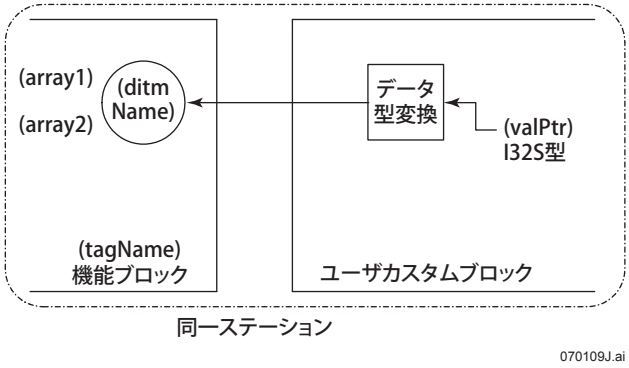


図　UcaTagWriteI32Sのデータの流れ

設定先機能ブロックは同一ステーション内に限ります。
引数 tagName、ditmName の英字部分は、大文字で記述してください。
引数 array1、array2 には、ditmName で指示されるデータアイテムが配列型でない場合、0 を指定してください。1 次元配列データの場合、array1 には配列添え字を、array2 には 0 を指定してください。2 次元配列データの場合、array1 には配列添え字 1 を、array2 には配列添え字 2 を指定してください。
引数 valPtr には、I32S 型でデータを格納してください。UcaTagWriteI32S は、出力先に書き込むときに出力先のデータ型に変換します。
UcaTagWriteI32S は、引数 valPtr のデータステータスのうち、BAD、QST のみを設定先に伝えます。それ以外のデータステータスは無視します。正常データであることを伝えるためには、引数 valPtr のデータステータスに 0 を設定してください。

7.1.7 UcaTagWriteString

UcaTagWriteString は、自ステーションの機能ブロックに文字列型データを設定します。

■ 関数名

I32 UcaTagWriteString

■ 引数

● blockContext

- データ型： UcaBlockContext
- タイプ： IN
- 説明： ブロックコンテキスト

● tagName

- データ型： TAGN
- タイプ： IN
- 説明： タグ名

● ditmName

- データ型： DITMN
- タイプ： IN
- 説明： データアイテム名

● array1

- データ型： I32
- タイプ： IN
- 説明： 配列添え字 1

● array2

- データ型： I32
- タイプ： IN
- 説明： 配列添え字 2

● valPtr

- データ型： BYTE *
- タイプ： IN
- 説明： 出力文字列ポインタ

● option

- データ型： U32
- タイプ： IN
- 説明： オプション

■ 戻り値

- SUCCEED： 成功
- UCAERR_TEXT_CONVFAIL： 文字コード変換失敗
- UCAERR_PARA_PTRNULL： 引数のポインタが NULL
- UCAERR_OPT_INVALID： 誤ったオプションを指定した

■ 処理内容説明

タグ名とデータアイテム名を指定して、データを設定します。

■ 引数説明

● tagName

設定先のタグ名

● ditmName

設定先のデータアイテム名

● array1、array2

配列添え字 1、2

● valPtr

出力文字列ポインタ

● option

NOOPTION を指定してください。

■ 詳細説明

UcaTagWriteString のデータの流れは以下のとおりです。

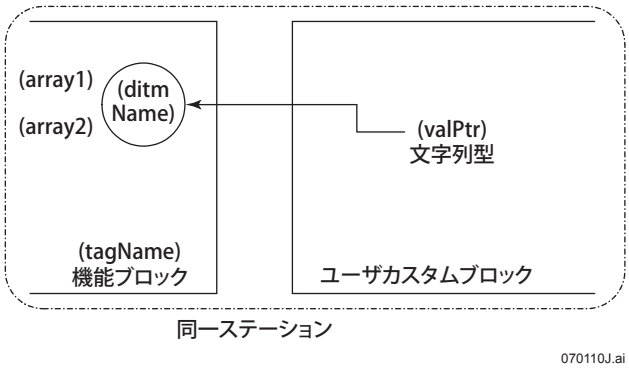


図　UcaTagWriteStringのデータの流れ

設定先機能ブロックは同一ステーション内に限ります。
引数 tagName、ditmName の英字部分は、大文字で記述してください。
引数 array1、array2 には、ditmName で指示されるデータアイテムが配列型でない場合、0 を指定してください。1 次元配列データの場合、array1 には配列添え字を、array2 には 0 を指定してください。2 次元配列データの場合、array1 には配列添え字 1 を、array2 には配列添え字 2 を指定してください。

重要 引数 valPtr には出力文字列を指定します。出力文字列の最大長は 16 バイトです。文字列長が 15 バイト以下の場合、文字列を NULL（0）で終端します。16 バイトを超える場合は、先頭の 16 バイトだけ書き込み、文字列を NULL（0）で終端しません。

7.1.8 UcaTagWrite

UcaTagWriteは、自ステーションの機能ブロックにデータを設定します。

■ 関数名

I32 UcaTagWrite

■ 引数

● blockContext

- データ型： UcaBlockContext
- タイプ： IN
- 説明： ブロックコンテキスト

● tagName

- データ型： TAGN
- タイプ： IN
- 説明： タグ名

● ditmName

- データ型： DITMN
- タイプ： IN
- 説明： データアイテム名

● array1

- データ型： I32
- タイプ： IN
- 説明： 配列添え字 1

● array2

- データ型： I32
- タイプ： IN
- 説明： 配列添え字 2

● valPtr

- データ型： UcaUnivType *
- タイプ： IN
- 説明： 出力値ポインタ

● option

- データ型： U32
- タイプ： IN
- 説明： オプション

■ 戻り値

- SUCCEED： 成功
- UCAERR_TEXT_CONVFAIL： 文字コード変換失敗
- UCAERR_NOTANUMBER：
  引数の浮動小数データが非数（Not a Number）である
- UCAERR_INFINITY： 引数の浮動小数データが無限大である
- UCAERR_TAG_INVALIDDTYPE： データ型が正しくない
- UCAERR_PARA_PTRNULL： 引数のポインタが NULL
- UCAERR_OPT_INVALID： 誤ったオプションを指定した

■ 処理内容説明

タグ名とデータアイテム名を指定して、データを設定します。

■ 引数説明

● tagName

設定先のタグ名

● ditmName

設定先のデータアイテム名

● array1、array2

配列添え字 1、2

● valPtr

出力値ポインタ

● option

NOOPTION を指定してください。

■ 詳細説明

UcaTagWrite は、自ステーションの機能ブロックに、データ型 UcaUnivType の引数 valPtr の値を設定します。

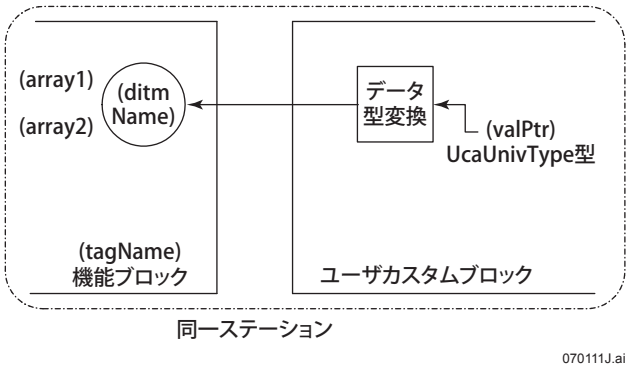


図　UcaTagWriteのデータの流れ

設定先機能ブロックは同一ステーション内に限ります。
引数 tagName、ditmName の英字部分は、大文字で記述してください。
引数 array1、array2 には、ditmName で指示されるデータアイテムが配列型でない場合、0 を指定してください。1 次元配列データの場合、array1 には配列添え字を、array2 には 0 を指定してください。2 次元配列データの場合、array1 には配列添え字 1 を、array2 には配列添え字 2 を指定してください。
引数 valPtr には、出力したいデータ型でデータを引数 valPtr->dataValue に格納してください。格納したデータ型は valPtr->dType で指示してください。UcaTagWrite は、出力先に書き込む時に出力先のデータ型に変換します。
引数 valPtr にデータステータス付きのデータ型を指定した場合、データステータスのうち BAD、QST のみを設定先に伝えます。それ以外のデータステータスは無視します。正常データであることを伝えるためには、引数 valPtr->dataValue のデータステータスに 0 を設定してください。
UcaTagWrite は、引数 valPtr のデータ型が浮動小数データ（F32、F32S、F64、F64S、F32SR）でデータ値が非数または無限大の場合、機能ブロックへの出力は行いません。引数 valPtr のデータ値が非数の場合はエラーコード UCAERR_NOTANUMBER、無限大の場合はエラーコード UCAERR_INFINITY を返します。

7.2 ワンショット起動

ワンショット起動関数は、自ブロックと同一ステーションに存在する機能ブロックを、ワンショット起動します。ユーザカスタムブロックからワンショット起動された機能ブロックは、呼び出しに応じて起動して1回だけ処理を実行します。ユーザカスタムブロック処理から呼び出しがあった時点で、呼び出し元の処理に割り込んで起動し処理を実行し、処理終了後は呼び出し元の処理を再開します。

参照 動作の詳細については、以下を参照してください。
機能ブロック共通機能リファレンス（IM 33J15A20-01JA）「7. 処理タイミング」

7.2.1 UcaTagOneshot

UcaTagOneshotは、自ステーションの機能ブロックをワンショット起動します。

■ 関数名

I32 UcaTagOneshot

■ 引数

● blockContext

- データ型： UcaBlockContext
- タイプ： IN
- 説明： ブロックコンテキスト

● tagName

- データ型： TAGN
- タイプ： IN
- 説明： タグ名

● parameter

- データ型： I32
- タイプ： IN
- 説明： パラメータ

● result

- データ型： U32 *
- タイプ： OUT
- 説明： 実行結果

■ 戻り値

- SUCCEED： 成功
- UCAERR_PARA_PTRNULL： 引数のポインタが NULL

■ 処理内容説明

タグ名を指定して、機能ブロックをワンショット起動します。

■ 引数説明

● tagName

設定先のタグ名

● parameter

ワンショット起動パラメータ

● result

条件判定結果が返ります。

■ 詳細説明

UcaTagOneshot は、引数 tagName で指定した自ステーションの機能ブロックをワンショット起動し、実行結果を引数 result に格納します。

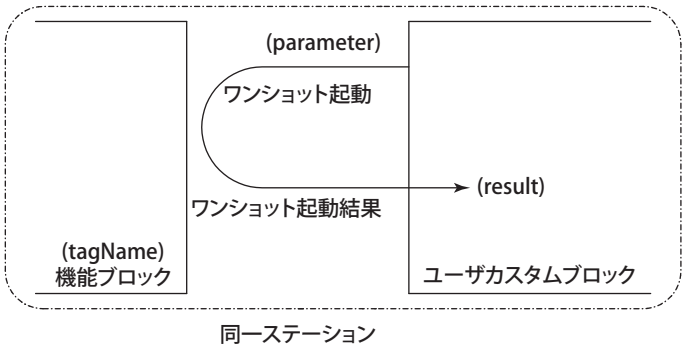


図　UcaTagOneshotのデータの流れ

起動する機能ブロックは同一ステーション内に限ります。
引数 tagName の英字部分は、大文字で記述してください。
UcaTagOneshot は、引数 result にワンショット起動結果を格納します。ワンショット起動ができなかった場合には引数 result に FALSE を格納し、ワンショット起動に成功した場合には引数 result に TRUE を格納します。

7.3 他ステーションタグデータ

ユーザカスタムアルゴリズムから他ステーション（つまりFCS）のデータアクセスは、つぎの手順で行います。

- 1バッチ形数値データ設定ブロック（BDSET-1L）に他ステーションのデータ結合（タグ名＋アイテム名＋（あれば）配列添え字）を定義します。その結果、BDSET-1Lに定義したデータのステーション間結合ブロック（ADL）がシステムにより自動生成されます。
- カスタムアルゴリズムライブラリに「FCSのタグ名＋アイテム名＋（あれば）配列添え字」を指定することによりADL経由で他ステーションデータにアクセスします。

■ 他ステーションタグデータアクセス処理の概要

以下に他ステーションデータアクセス（ADL 経由）に使用するライブラリを示します。

表　タグ名を指定したデータ入出力の関数

ステーション / データ型／読み書き	他ステーション（APCS外）
数値データ読み込み	UcaOtherTagReadF64S
	UcaOtherTagReadI32S
	UcaOtherTagRead
数値データ書き込み	UcaOtherTagWriteF64S
	UcaOtherTagWriteI32S
文字列データ読み込み	なし
文字列データ書き込み	なし
汎用データ型読み込み	UcaOtherTagRead
汎用データ型書き込み	UcaOtherTagWrite

以後、他ステーションデータアクセスライブラリの総称を UcaOtherTagRead/Write と記述します。

重要 ここで説明する他ステーションデータアクセス関数を使用する場合は、BDSET-1L をステーション間結合ブロック（ADL）の自動生成に使用します。BDSET-1L 本来の機能は、各種設定値や制御パラメータを他ステーションに設定することですが、同一の APCS 内では BDSET-1L をバッチデータ設定ブロックの機能としては使用しないでください。バッチデータ設定ブロックの機能が必要な場合には、2 バッチ形数値バッチデータ設定ブロック BDSET-2L を使用してください。BDSET-2L は、BDSET-1L の機能を包含しています。

重要 他ステーションデータアクセス関数のための ADL ブロックは、BDSET-1L で生成してください。BDSET-2L など BDSET-1L 以外のバッチデータ設定ブロックで生成した ADL ブロックは、他ステーションデータアクセス関数の動作に無関係です。

以下に、BDSET-1L の機能ブロック詳細定義ウィンドウを示します。BDSET-1L に機能ブロックデータの結合を定義すると、該当データのステーション間結合ブロック（ADL）が自動的に作成されます。

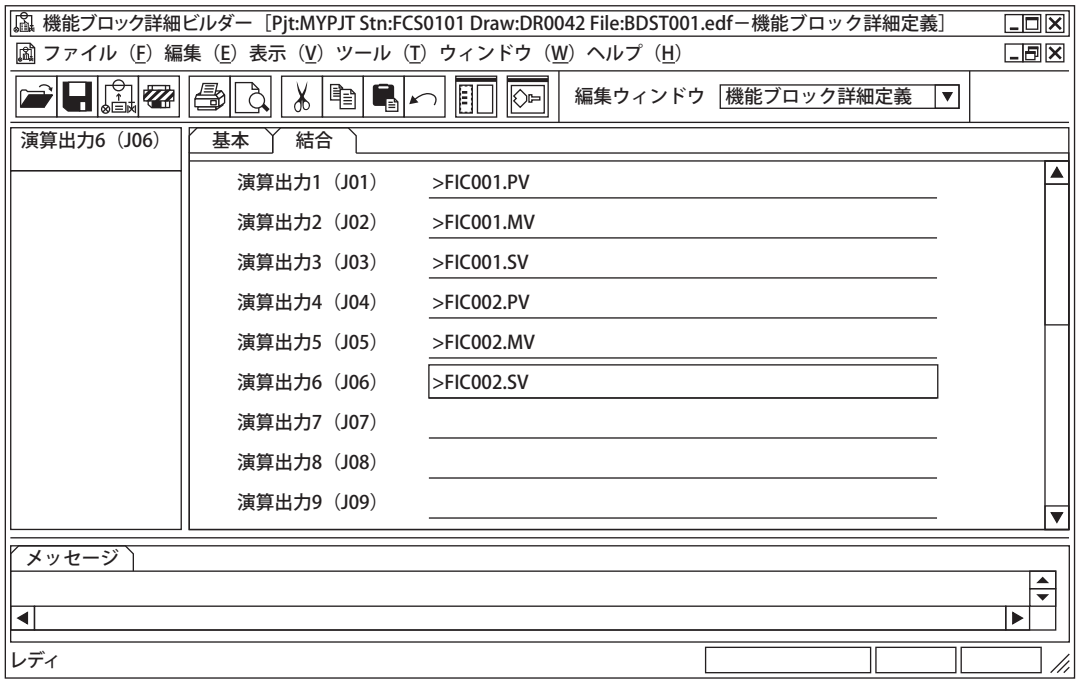


図　BDSET-1Lの機能ブロック詳細定義ウィンドウ

- 1 つの BDSET-1L あたり 16 データ（J01 ～ J16）記述できます。
- ＞ FIC001. MV の「＞」が必要です。「＞」は、配線の場合の「AREAOUT」相当、つまり他ステーションのデータであることを示します。「＞」とタグ名の間には空白は入れないで詰めて記述します。
- データが 1 次元配列要素の場合には、タグ名 . アイテム名 [添え字] のように指定します。
- データが 2 次元配列要素の場合には、タグ名 . アイテム名 [1 次元添え字、2 次元添え字] のように指定します。

例を示します。

単純変数　　＞ FIC001. SV
1 次元配列要素 ＞ UNT001. ARY1[3]
2 次元配列要素 ＞ UNT001. ARY2[3, 4]

以下に APCS 外部のデータに ADL 経由でアクセスする構成を示します。

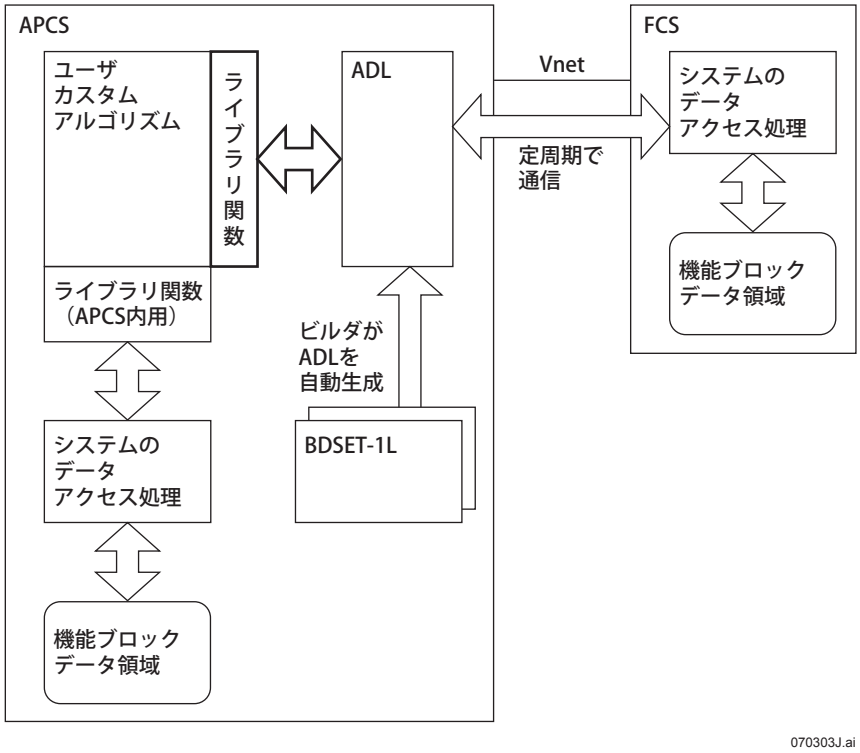


図　ユーザカスタムアルゴリズムからのデータアクセス

- 制御ステーション間結合を行う ADL ブロックは、BDSET-1L を定義することで自動的に作成されます。
- 太枠の部分が UcaOtherTagRead/Write 処理に相当します。

自 APCS 内のデータアクセス用関数 UcaTagRead/Write と、ADL 経由で他ステーションデータにアクセスする UcaOtherTagRead/Write の動作の違いを説明します。自 APCS 内のデータアクセス用関数は、関数が呼び出されたタイミングで指定された機能ブロックデータに直接アクセスします。UcaTagReadF64S などは、呼び出されたタイミングで指定された機能ブロックデータを読み込みます。また、UcaTagWriteF64S などは、呼び出されたタイミングで機能ブロックデータに書き込みます。

```
......
rtnCode = UcaTagWriteF64S(bc, "TAG001", "ITEM", 0, 0, &data, NOOPTION);
/* UcaTagWriteF64S が完了した時点で、データの書きこみは完了しています */
```

これに対し、他ステーションデータアクセスは、ステーション間結合ブロックを中継するデータアクセスとなります。関数 UcaOtherTagRead/Write がアクセスするのはステーション間結合ブロックです。他ステーションの機能ブロックデータにアクセスするのはステーション間結合ブロックです。BDSET-1L の結合は、「データ設定結合」ですから、自動生成されるステーション間結合ブロックは、「データ参照およびデータ設定」となり、ステーション間結合ブロックはデータ結合ごとに「参照データ」と「設定データ」の領域を持ちます。

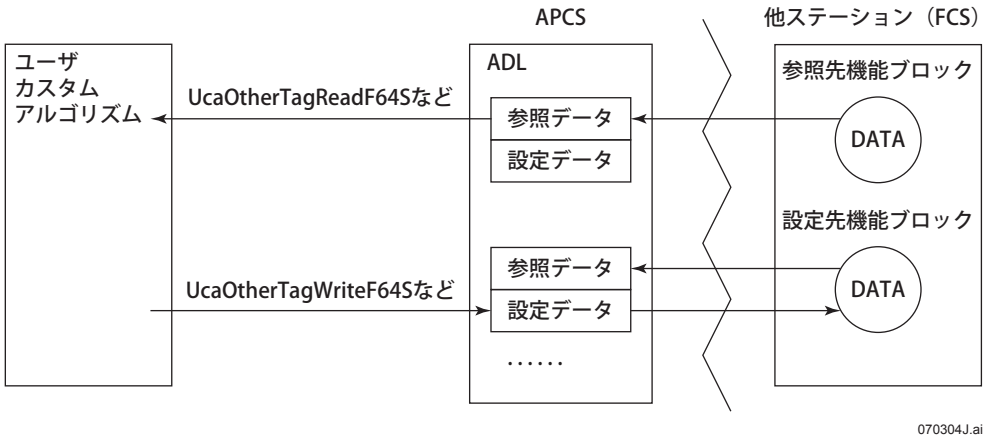


図　ユーザカスタムアルゴリズムからの他ステーションデータアクセス

ステーション間結合ブロックは、通信周期でデータアクセス通信を行います。参照データには、通信周期で機能ブロックデータが取り込まれます。通信周期は、データ結合の定義数により決まる一定の周期です。また、設定データにデータが設定されると次の通信処理で（一度だけ）機能ブロックに出力されます。したがって、UcaOtherTagRead/Write の動作は、以下のようになります。

- UcaOtherTagReadF64S などは、ステーション間結合ブロックの参照データ（もっとも最近に読み込んだデータ）を取得します。
- UcaOtherTagWriteF64S などの出力先はステーション間結合ブロックの「設定データ」に保持されます。当該ステーション間ブロックの次回の通信時に、設定データに保持されている値が実際の機能ブロックデータに（一度だけ）出力されます。

したがって、UcaOtherTagReadF64S などで取得できるデータは関数を実行する「少し前」のデータであり、UcaOtherTagWriteF64S などによる出力が実際に機能ブロックに出力されるのは関数を実行してから「少し後」となります。

参照 ステーション間結合ブロックについては、以下を参照してください。
機能ブロックリファレンス Vol.2（IM 33J15A31-01JA）「1.46 ステーション間結合ブロック（ADL）」
ステーション間結合ブロックの通信周期については、以下を参照してください。
APCS（IM 33J15U10-01JA）「2.5 制御ステーション間結合」

7.3.1 UcaOtherTagReadF64S

UcaOtherTagReadF64Sは、他ステーションの機能ブロックからデータを参照し、F64S型データに格納します。

■ 関数名

I32 UcaOtherTagReadF64S

■ 引数

● blockContext

- データ型： UcaBlockContext
- タイプ： IN
- 説明： ブロックコンテキスト

● tagName

- データ型： TAGN
- タイプ： IN
- 説明： タグ名

● ditmName

- データ型： DITMN
- タイプ： IN
- 説明： データアイテム名

● array1

- データ型： I32
- タイプ： IN
- 説明： 配列添え字 1

● array2

- データ型： I32
- タイプ： IN
- 説明： 配列添え字 2

● valPtr

- データ型： F64S *
- タイプ： OUT
- 説明： 入力値ポインタ

● option

- データ型： U32
- タイプ： IN
- 説明： オプション

■ 戻り値

- SUCCEED： 成功
- UCAERR_TAG_NOADL： BDSET-1L でのステーション間結合未定義
- UCAERR_TAG_INVALIDDB： データベースタイプが正しくない
- UCAERR_PARA_PTRNULL： 引数のポインタが NULL
- UCAERR_OPT_INVALID： 誤ったオプションを指定した

■ 処理内容説明

タグ名とデータアイテム名を指定して、他ステーションのデータを参照します。

■ 引数説明

● tagName

参照先のタグ名

● ditmName

参照先のデータアイテム名

● array1、array2

配列添え字 1、2

● valPtr

入力値ポインタ

● option

以下の中から指定することができます。

- UCAOPT_TAG_SETIOP：データ参照異常時に、IOP アラームを発生します。
- NOOPTION

■ 詳細説明

UcaOtherTagReadF64S は、他ステーションの機能ブロックからデータを参照し、F64S 型の引数 valPtr に格納します。

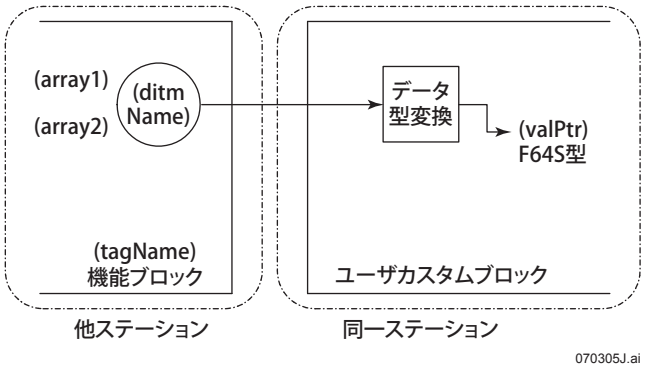


図　UcaOtherTagReadF64Sのデータの流れ

参照先機能ブロックは他ステーションに限ります。あらかじめ、UcaOtherTagReadF64S で参照する機能ブロックデータに対して、BDSET-1L でタグ名を登録しておいてください。タグ名を登録していない場合、UcaOtherTagReadF64S はエラーコード UCAERR_TAG_NOADL を戻り値として返します。
UcaOtherTagReadF64S では、参照に失敗した詳細な理由（相手ステーションがフェイルしているのか、タグ名が間違っているのか、など）は、分かりません。
引数 tagName、ditmName の英字部分は、大文字で記述してください。
引数 array1、array2 には、ditmName で指示されるデータアイテムが配列型でない場合、0 を指定してください。1 次元配列データの場合、array1 には配列添え字を、array2 には 0 を指定してください。2 次元配列データの場合、array1 には配列添え字 1 を、array2 には配列添え字 2 を指定してください。
UcaOtherTagReadF64S は、機能ブロックのデータ参照に失敗したときには、引数 valPtr のデータステータスに BAD を設定します。引数 valPtr のデータ値には、0 を設定します。
UcaOtherTagReadF64S は、参照先のデータ型を F64S に変換して引数 valPtr に格納します。参照先データがデータステータスつきであれば、そのデータステータスも同時に取得します。参照先データがデータステータスなしであれば、引数 valPtr のデータステータスに 0（正常）を格納します。
UcaOtherTagReadF64S は、以下のアラームを検出します。

表　UcaOtherTagReadF64Sのアラーム検出

オプション / アラーム	NOOPTION	UCAOPT_TAG_SETIOP
IOP	×	○
CNF	×	○

○： 検出する
×： 検出しない

オプションに UCAOPT_TAG_SETIOP を指定すると、機能ブロックのデータ参照に失敗した場合、プロセスアラーム IOP と CNF を検出します。参照に成功した場合でも、取得したデータのデータステータスが BAD のときにはプロセスアラーム IOP を検出します。
警報タブシートのビルダ定義項目［入力オープン警報］や［結合状態不良警報］に警報なしを指定した場合、対応するアラームは検出しません。

7.3.2 UcaOtherTagReadI32S

UcaOtherTagReadI32Sは、他ステーションの機能ブロックからデータを参照し、I32S型データに格納します。

■ 関数名

I32 UcaOtherTagReadI32S

■ 引数

● blockContext

- データ型： UcaBlockContext
- タイプ： IN
- 説明： ブロックコンテキスト

● tagName

- データ型： TAGN
- タイプ： IN
- 説明： タグ名

● ditmName

- データ型： DITMN
- タイプ： IN
- 説明： データアイテム名

● array1

- データ型： I32
- タイプ： IN
- 説明： 配列添え字 1

● array2

- データ型： I32
- タイプ： IN
- 説明： 配列添え字 2

● valPtr

- データ型： I32S *
- タイプ： OUT
- 説明： 入力値ポインタ

● option

- データ型： U32
- タイプ： IN
- 説明： オプション

■ 戻り値

- SUCCEED： 成功
- UCAERR_TAG_NOADL： BDSET-1L でのステーション間結合未定義
- UCAERR_TAG_INVALIDDB： データベースタイプが正しくない
- UCAERR_PARA_PTRNULL： 引数のポインタが NULL
- UCAERR_OPT_INVALID： 誤ったオプションを指定した

■ 処理内容説明

タグ名とデータアイテム名を指定して、他ステーションのデータを参照します。

■ 引数説明

● tagName

参照先のタグ名

● ditmName

参照先のデータアイテム名

● array1、array2

配列添え字 1、2

● valPtr

入力値ポインタ

● option

以下の中から指定することができます。

- UCAOPT_TAG_SETIOP：データ参照異常時に、IOP アラームを発生します。
- NOOPTION

■ 詳細説明

UcaOtherTagReadI32S のデータの流れは以下のとおりです。

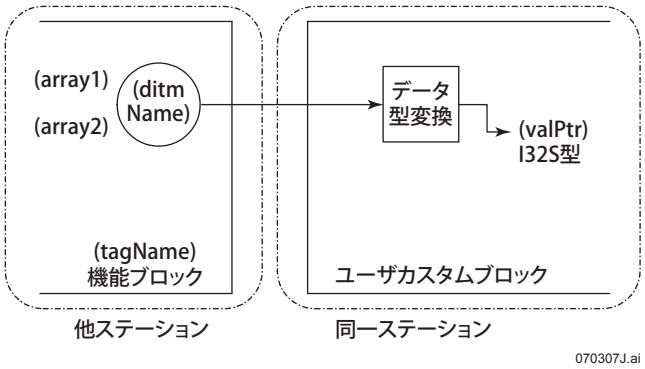


図　UcaOtherTagReadI32Sのデータの流れ

参照先機能ブロックは他ステーションに限ります。あらかじめ、UcaOtherTagReadI32S で参照する機能ブロックデータに対して、BDSET-1L でタグ名を登録しておいてください。タグ名を登録していない場合、UcaOtherTagReadI32S はエラーコード UCAERR_TAG_NOADL を戻り値として返します。
UcaOtherTagReadI32S では、参照に失敗した詳細な理由（相手ステーションがフェイルしているのか、タグ名が間違っているのか、など）は、分かりません。
引数 tagName、ditmName の英字部分は、大文字で記述してください。
引数 array1、array2 には、ditmName で指示されるデータアイテムが配列型でない場合、0 を指定してください。1 次元配列データの場合、array1 には配列添え字を、array2 には 0 を指定してください。2 次元配列データの場合、array1 には配列添え字 1 を、array2 には配列添え字 2 を指定してください。
UcaOtherTagReadI32S は、機能ブロックのデータ参照に失敗したときには、引数 valPtr のデータステータスに BAD を設定します。引数 valPtr のデータ値には、0 を設定します。
UcaOtherTagReadI32S は、参照先のデータ型を I32S に変換して引数 valPtr に格納します。参照先データがデータステータスつきであれば、そのデータステータスも同時に取得します。参照先データがデータステータスなしであれば、引数 valPtr のデータステータスに 0（正常）を格納します。
UcaOtherTagReadI32S は、以下のアラームを検出します。

表　UcaOtherTagReadI32Sのアラーム検出

オプション / アラーム	NOOPTION	UCAOPT_TAG_SETIOP
IOP	×	○
CNF	×	○

○： 検出する
×： 検出しない

オプションに UCAOPT_TAG_SETIOP を指定すると、機能ブロックのデータ参照に失敗した場合、プロセスアラーム IOP と CNF を検出します。参照に成功した場合でも、取得したデータのデータステータスが BAD のときにはプロセスアラーム IOP を検出します。
警報タブシートのビルダ定義項目［入力オープン警報］や［結合状態不良警報］になしを指定した場合、対応するアラームは検出しません。

7.3.3 UcaOtherTagRead

UcaOtherTagReadは、他ステーションの機能ブロックからデータを参照し、UcaUnivType型データに格納します。

■ 関数名

I32 UcaOtherTagRead

■ 引数

● blockContext

- データ型： UcaBlockContext
- タイプ： IN
- 説明： ブロックコンテキスト

● tagName

- データ型： TAGN
- タイプ： IN
- 説明： タグ名

● ditmName

- データ型： DITMN
- タイプ： IN
- 説明： データアイテム名

● array1

- データ型： I32
- タイプ： IN
- 説明： 配列添え字 1

● array2

- データ型： I32
- タイプ： IN
- 説明： 配列添え字 2

● valPtr

- データ型： UcaUnivType *
- タイプ： OUT
- 説明： 入力値

● option

- データ型： U32
- タイプ： IN
- 説明： オプション

■ 戻り値

- SUCCEED： 成功
- UCAERR_TAG_NOADL： BDSET-1L でのステーション間結合未定義
- UCAERR_TAG_INVALIDPARA： 引数が正しくない
- UCAERR_TAG_INVALIDDB： データベースタイプが正しくない
- UCAERR_PARA_PTRNULL： 引数のポインタが NULL
- UCAERR_OPT_INVALID： 誤ったオプションを指定した

■ 処理内容説明

タグ名とデータアイテム名を指定して、他ステーションのデータを参照します。

■ 引数説明

● tagName

参照先のタグ名

● ditmName

参照先のデータアイテム名

● array1、array2

配列添え字 1、2

● valPtr

参照先のデータ型で参照結果が入ります。データ型は valPtr->dType に入ります。

● option

NOOPTION を指定してください。

■ 詳細説明

UcaOtherTagRead のデータの流れは以下のとおりです。

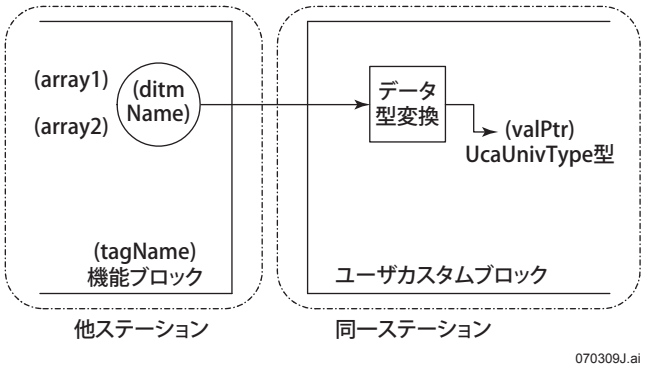


図　UcaOtherTagReadのデータの流れ

参照先機能ブロックは他ステーションに限ります。あらかじめ、UcaOtherTagRead で参照する機能ブロックデータに対して、BDSET-1L でタグ名を登録しておいてください。タグ名を登録していない場合、UcaOtherTagRead はエラーコード UCAERR_TAG_NOADL を戻り値として返します。
UcaOtherTagRead では、参照に失敗した詳細な理由（相手ステーションがフェイルしているのか、タグ名が間違っているのか、など）は、分かりません。
引数 tagName、ditmName の英字部分は、大文字で記述してください。
引数 array1、array2 には、ditmName で指示されるデータアイテムが配列型でない場合、0 を指定してください。1 次元配列データの場合、array1 には配列添え字を、array2 には 0 を指定してください。2 次元配列データの場合、array1 には配列添え字 1 を、array2 には配列添え字 2 を指定してください。
UcaOtherTagRead を呼び出す前に、あらかじめ valPtr->indOpt、valPtr->rqstType を 0 に初期化しておいてください。valPtr->indOpt と valPtr->rqstType が 0 以外の場合、UcaOtherTagRead はエラーコード UCAERR_TAG_INVALIDPARA を返します。
UcaOtherTagRead は、参照先データを引数 valPtr に格納します。データ値を valPtr->dataValue に、データ型を valPtr->dType に格納します。

参照 UcaUnivType 型のデータは、UcaDataConvertType により、特定のデータ型に変換することができます。UcaDataConvertType については、以下を参照してください。
「9.1.1 UcaDataConvertType」

7.3.4 UcaOtherTagWriteF64S

UcaOtherTagWriteF64Sは、他ステーションの機能ブロックにF64S型データを設定します。

■ 関数名

I32 UcaOtherTagWriteF64S

■ 引数

● blockContext

- データ型： UcaBlockContext
- タイプ： IN
- 説明： ブロックコンテキスト

● tagName

- データ型： TAGN
- タイプ： IN
- 説明： タグ名

● ditmName

- データ型： DITMN
- タイプ： IN
- 説明： データアイテム名

● array1

- データ型： I32
- タイプ： IN
- 説明： 配列添え字 1

● array2

- データ型： I32
- タイプ： IN
- 説明： 配列添え字 2

● valPtr

- データ型： F64S *
- タイプ： IN
- 説明： 出力値ポインタ

● option

- データ型： U32
- タイプ： IN
- 説明： オプション

■ 戻り値

- SUCCEED： 成功
- UCAERR_TAG_NOADL： BDSET-1L でのステーション間結合未定義
- UCAERR_NOTANUMBER：
  引数の浮動小数データが非数（Not a Number）である
- UCAERR_INFINITY： 引数の浮動小数データが無限大である
- UCAERR_TAG_INVALIDDB： データベースタイプが正しくない
- UCAERR_PARA_PTRNULL： 引数のポインタが NULL
- UCAERR_OPT_INVALID： 誤ったオプションを指定した

■ 処理内容説明

タグ名とデータアイテム名を指定して、他ステーションのデータを設定します。

■ 引数説明

● tagName

設定先のタグ名

● ditmName

設定先のデータアイテム名

● array1、array2

配列添え字 1、2

● valPtr

出力値ポインタ

● option

NOOPTION を指定してください。

■ 詳細説明

UcaOtherTagWriteF64S のデータの流れは以下のとおりです。

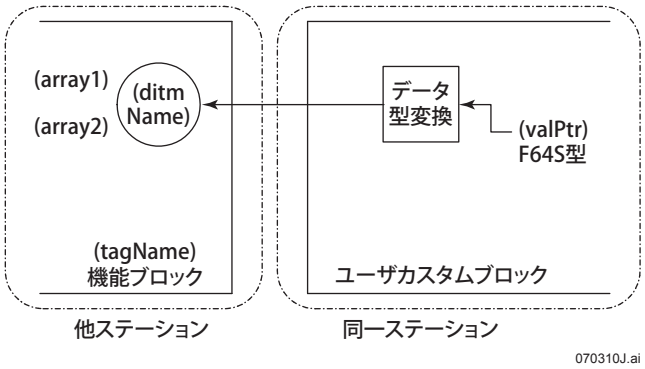


図　UcaOtherTagWriteF64Sのデータの流れ

設定先機能ブロックは他ステーションに限ります。あらかじめ、UcaOtherTagWriteF64S で設定する機能ブロックデータに対して、BDSET-1L でタグ名を登録しておいてください。タグ名を登録していない場合、UcaOtherTagWriteF64S はエラーコード UCAERR_TAG_NOADL を戻り値として返します。
UcaOtherTagWriteF64S では、設定先に正しくデータ設定できたかどうか（通信が成立したか、設定先機能ブロックの状態によって設定が拒絶されていないか、設定先のデータが文字列型、など）は、分かりません。
引数 tagName、ditmName の英字部分は、大文字で記述してください。
引数 array1、array2 には、ditmName で指示されるデータアイテムが配列型でない場合、0 を指定してください。1 次元配列データの場合、array1 には配列添え字を、array2 には 0 を指定してください。2 次元配列データの場合、array1 には配列添え字 1 を、array2 には配列添え字 2 を指定してください。
引数 valPtr には、F64S 型でデータを格納してください。UcaOtherTagWriteF64S は、出力先に書き込むとき出力先のデータ型に変換します。UcaOtherTagWriteF64S は、引数 ValPtr のデータステータスのうち、BAD、QST のみを設定先に伝えます。それ以外のデータステータスは無視します。正常データであることを伝えるためには、引数 valPtr のデータステータスに 0 を設定してください。
UcaOtherTagWriteF64S は、引数 valPtr の値が非数または無限大の場合、機能ブロックへの出力は行いません。引数 valPtr の値が非数の場合はエラーコード UCAERR_NOTANUMBER、無限大の場合はエラーコード UCAERR_INFINITY を返します。

7.3.5 UcaOtherTagWriteI32S

UcaOtherTagWriteI32Sは、他ステーションの機能ブロックにI32S型データを設定します。

■ 関数名

I32 UcaOtherTagWriteI32S

■ 引数

● blockContext

- データ型： UcaBlockContext
- タイプ： IN
- 説明： ブロックコンテキスト

● tagName

- データ型： TAGN
- タイプ： IN
- 説明： タグ名

● ditmName

- データ型： DITMN
- タイプ： IN
- 説明： データアイテム名

● array1

- データ型： I32
- タイプ： IN
- 説明： 配列添え字 1

● array2

- データ型： I32
- タイプ： IN
- 説明： 配列添え字 2

● valPtr

- データ型： I32S *
- タイプ： IN
- 説明： 出力値ポインタ

● option

- データ型： U32
- タイプ： IN
- 説明： オプション

■ 戻り値

- SUCCEED： 成功
- UCAERR_TAG_NOADL： BDSET-1L でのステーション間結合未定義
- UCAERR_TAG_INVALIDDB： データベースタイプが正しくない
- UCAERR_PARA_PTRNULL： 引数のポインタが NULL
- UCAERR_OPT_INVALID： 誤ったオプションを指定した

■ 処理内容説明

タグ名とデータアイテム名を指定して、他ステーションのデータを設定します。

■ 引数説明

● tagName

設定先のタグ名

● ditmName

設定先のデータアイテム名

● array1、array2

配列添え字 1、2

● valPtr

出力値ポインタ

● option

NOOPTION を指定してください。

■ 詳細説明

UcaOtherTagWriteI32S のデータの流れは以下のとおりです。

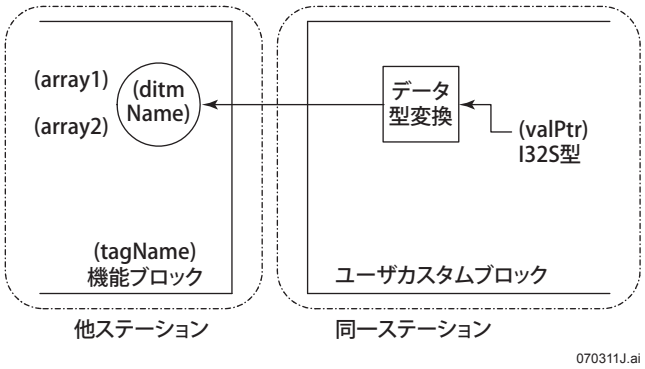


図　UcaOtherTagWriteI32Sのデータの流れ

設定先機能ブロックは他ステーションに限ります。あらかじめ、UcaOtherTagWriteI32S で設定する機能ブロックデータに対して、BDSET-1L でタグ名を登録しておいてください。タグ名を登録していない場合、UcaOtherTagWriteI32S はエラーコード UCAERR_TAG_NOADL を戻り値として返します。
UcaOtherTagWriteI32S では、設定先に正しくデータ設定できたかどうか（通信が成立したか、設定先機能ブロックの状態によって設定が拒絶されていないか、設定先のデータが文字列型、など）は、分かりません。
引数 tagName、ditmName の英字部分は、大文字で記述してください。
引数 array1、array2 には、ditmName で指示されるデータアイテムが配列型でない場合、0 を指定してください。1 次元配列データの場合、array1 には配列添え字を、array2 には 0 を指定してください。2 次元配列データの場合、array1 には配列添え字 1 を、array2 には配列添え字 2 を指定してください。
引数 valPtr には、I32S 型でデータを格納してください。UcaOtherTagWriteI32S は、出力先に書き込むときに出力先のデータ型に変換します。
UcaOtherTagWriteI32S は、引数 valPtr のデータステータスのうち、BAD、QST のみを設定先に伝えます。それ以外のデータステータスは無視します。正常データであることを伝えるためには、引数 valPtr のデータステータスに 0 を設定してください。

7.3.6 UcaOtherTagWrite

UcaOtherTagWriteは、他ステーションの機能ブロックにデータを設定します。

■ 関数名

I32 UcaOtherTagWrite

■ 引数

● blockContext

- データ型： UcaBlockContext
- タイプ： IN
- 説明： ブロックコンテキスト

● tagName

- データ型： TAGN
- タイプ： IN
- 説明： タグ名

● ditmName

- データ型： DITMN
- タイプ： IN
- 説明： データアイテム名

● array1

- データ型： I32
- タイプ： IN
- 説明： 配列添え字 1

● array2

- データ型： I32
- タイプ： IN
- 説明： 配列添え字 2

● valPtr

- データ型： UcaUnivType *
- タイプ： OUT
- 説明： 出力値ポインタ

● option

- データ型： U32
- タイプ： IN
- 説明： オプション

■ 戻り値

- SUCCEED： 成功
- UCAERR_TAG_NOADL： BDSET-1L でのステーション間結合未定義
- UCAERR_NOTANUMBER：
  引数の浮動小数データが非数（Not a Number）である
- UCAERR_INFINITY： 引数の浮動小数データが無限大である
- UCAERR_TAG_INVALIDDTYPE： データ型が正しくない
- UCAERR_TAG_INVALIDDB： データベースタイプが正しくない
- UCAERR_PARA_PTRNULL： 引数のポインタが NULL
- UCAERR_OPT_INVALID： 誤ったオプションを指定した

■ 処理内容説明

タグ名とデータアイテム名を指定して、他ステーションのデータを設定します。

■ 引数説明

● tagName

設定先のタグ名

● ditmName

設定先のデータアイテム名

● array1、array2

配列添え字 1、2

● valPtr

出力値ポインタ

● option

NOOPTION を指定してください。

■ 詳細説明

UcaOtherTagWrite のデータの流れは以下のとおりです。

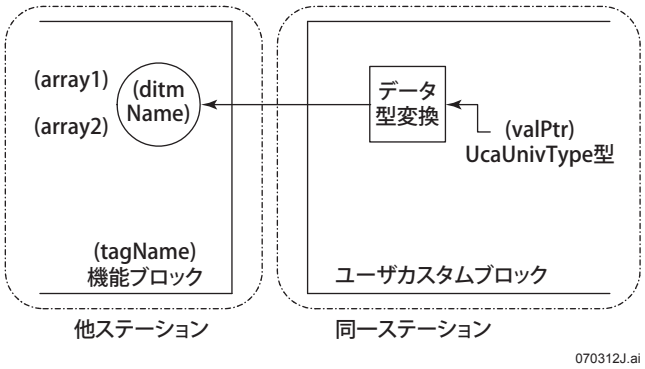


図　UcaOtherTagWriteのデータの流れ

設定先機能ブロックは他ステーションに限ります。あらかじめ、UcaOtherTagWrite で設定する機能ブロックデータに対して、BDSET-1L でタグ名を登録しておいてください。タグ名を登録していない場合、UcaOtherTagWrite はエラーコード UCAERR_TAG_NOADL を戻り値として返します。
UcaOtherTagWrite では、設定先に正しくデータ設定できたかどうか（通信が成立したか、設定先機能ブロックの状態によって設定が拒絶されていないか、など）は、分かりません。
引数 tagName、ditmName の英字部分は、大文字で記述してください。
引数 array1、array2 には、ditmName で指示されるデータアイテムが配列型でない場合には、0 を指定してください。1 次元配列データの場合、array1 には配列添え字を、array2 には 0 を指定してください。2 次元配列データの場合、array1 には配列添え字 1 を、array2 には配列添え字 2 を指定してください。
引数 valPtr には、出力したいデータ型でデータを引数 valPtr->dataValue に格納してください。格納したデータ型は引数 valPtr->dType で指示してください。UcaOtherTagWrite は、出力先に書き込むときに出力先のデータ型に変換します。
UcaOtherTagWrite は、引数 valPtr にデータステータス付きのデータ型を指定した場合、valPtr のデータステータスのうち、BAD、QST のみを設定先に伝えます。それ以外のデータステータスは無視します。正常データであることを伝えるためには、引数 valPtr のデータステータスに 0 を設定してください。
UcaOtherTagWrite は、引数 valPtr のデータ型が浮動小数データ（F32、F32S、F64、F64S、F32SR）でデータ値が非数または無限大の場合、機能ブロックへの出力は行いません。引数 valPtr の値が非数の場合はエラーコード UCAERR_NOTANUMBER、無限大の場合はエラーコード UCAERR_INFINITY を返します。

7.4 タグデータ参照（副入力RVnnへの設定付き）

ユーザカスタムブロックと同一ステーションまたは他ステーションに存在する機能ブロック関数のデータを、取得／設定し、データアイテムRVnに格納します。

■ RVnn設定付きタグデータ参照関数

以下に APCS 内データアクセス（副入力 Rvnn 設定付き）に使用するライブラリを示します。

表　タグ名を指定したデータ入出力の関数（副入力Rvnn設定付き）

ステーション／データ型／読み書き	自ステーション（APCS内）	他ステーション（APCS外）
数値データ読み込み（副入力 RVnn 設定付き）	UcaTagReadToRvnF64S	UcaOtherTagReadToRvnF64S

7.4.1 UcaTagReadToRvnF64S

UcaTagReadToRvnF64Sは、自ステーションの機能ブロックからデータを参照し、F64S型の引数valPtrとデータアイテムRVnに格納します。

■ 関数名

I32 UcaTagReadToRvnF64S

■ 引数

● blockContext

- データ型： UcaBlockContext
- タイプ： IN
- 説明： ブロックコンテキスト

● tagName

- データ型： TAGN
- タイプ： IN
- 説明： タグ名

● ditmName

- データ型： DITMN
- タイプ： IN
- 説明： データアイテム名

● array1

- データ型： I32
- タイプ： IN
- 説明： 配列添え字 1

● array2

- データ型： I32
- タイプ： IN
- 説明： 配列添え字 2

● valPtr

- データ型： F64S *
- タイプ： OUT
- 説明： 入力値ポインタ

● itemNo

- データ型： I32
- タイプ： IN
- 説明： データ番号

● option

- データ型： U32
- タイプ： IN
- 説明： オプション

■ 戻り値

- SUCCEED： 成功
- UCAERR_PARA_PTRNULL： 引数のポインタが NULL
- UCAERR_OPT_INVALID： 誤ったオプションを指定した

■ 処理内容説明

タグ名とデータアイテム名を指定して、データを取得します。そして取得したデータを副入力データアイテム RVnn に設定します。

■ 引数説明

● tagName

参照先のタグ名

● ditmName

参照先のデータアイテム名

● array1、array2

配列添え字 1、2

● valPtr

入力値ポインタ

● itemNo

データ番号

● option

以下の中から指定することができます。

- UCAOPT_TAG_SETIOP：データ参照異常時に、IOP アラームを発生します。
- NOOPTION

■ 詳細説明

UcaTagReadToRvnF64S のデータの流れは以下のとおりです。

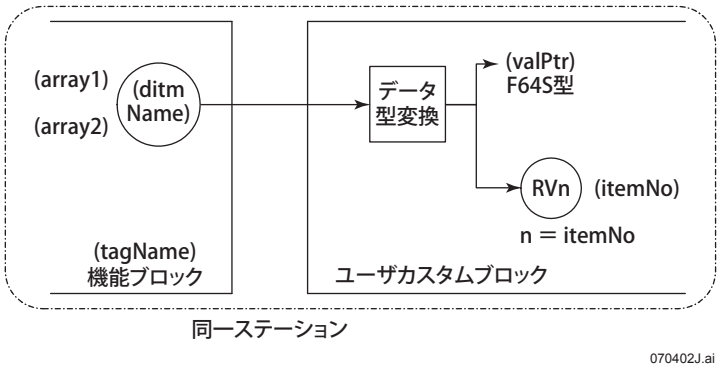


図　UcaTagReadToRvnF64Sのデータの流れ

参照先機能ブロックは同一ステーション内に限ります。
引数 tagName、ditmName の英字部分は、大文字で記述してください。
引数 array1、array2 には、ditmName で指示されるデータアイテムが配列型でない場合、0 を指定してください。1 次元配列データの場合、array1 には配列添え字を、array2 には 0 を指定してください。2 次元配列データの場合、array1 には配列添え字 1 を、array2 には配列添え字 2 を指定してください。
UcaTagReadToRvnF64S は、参照先のデータ型を F64S 型に変換し、引数 valPtr に格納します。参照先データがデータステータスつきであれば、そのデータステータスも同時に取得します。参照先データがデータステータスなしであれば、引数 valPtr のデータステータスに 0（正常）を格納します。エラー時には、データステータスに BAD（BAD のみ真で他のデータステータスは偽）を格納します。
UcaTagReadToRvnF64S は、valPtr に返すデータと同じデータを itemNo 番目の副入力 RVnn に格納します。itemNo が 0 ならデータアイテム RV に設定します。データの取得に失敗した場合には、RVnn（または RV）のデータは変更しませんが、RVnn（または RV）のデータステータスに BAD（BAD のみ真で他のデータステータスは偽）を設定します。
UcaTagReadToRvnF64S は、以下のアラームを検出します。

表　UcaTagReadToRvnF64Sのアラーム検出

オプション / アラーム	NOOPTION	UCAOPT_TAG_SETIOP
IOP	×	○
CNF	×	○

○： 検出する
×： 検出しない

オプションに UCAOPT_TAG_SETIOP を指定すると、機能ブロックのデータ参照に失敗した場合、プロセスアラーム IOP と CNF を検出します。参照に成功した場合でも、取得したデータのデータステータスが BAD の時にはプロセスアラーム IOP を検出します。
警報タブシートのビルダ定義項目［入力オープン警報］や［結合状態不良警報］に警報なしを指定した場合、対応するアラームは検出しません。

7.4.2 UcaOtherTagReadToRvnF64S

UcaOtherTagReadToRvnF64Sは、他ステーションの機能ブロックからデータを参照し、F64S型の引数valPtrとデータアイテムRVnに格納します。

■ 関数名

I32 UcaOtherTagReadToRvnF64S

■ 引数

● blockContext

- データ型： UcaBlockContext
- タイプ： IN
- 説明： ブロックコンテキスト

● tagName

- データ型： TAGN
- タイプ： IN
- 説明： タグ名

● ditmName

- データ型： DITMN
- タイプ： IN
- 説明： データアイテム名

● array1

- データ型： I32
- タイプ： IN
- 説明： 配列添え字 1

● array2

- データ型： I32
- タイプ： IN
- 説明： 配列添え字 2

● valPtr

- データ型： F64S *
- タイプ： OUT
- 説明： 入力値ポインタ

● itemNo

- データ型： I32
- タイプ： IN
- 説明： データ番号

● option

- データ型： U32
- タイプ： IN
- 説明： オプション

■ 戻り値

- SUCCEED： 成功
- UCAERR_TAG_NOADL： BDSET-1L でのステーション間結合未定義
- UCAERR_TAG_INVALIDDB： データベースタイプが正しくない
- UCAERR_PARA_PTRNULL： 引数のポインタが NULL
- UCAERR_OPT_INVALID： 誤ったオプションを指定した

■ 処理内容説明

タグ名とデータアイテム名を指定して、他ステーションのデータを参照します。そして取得したデータを副入力データアイテム RVnn に設定します。

■ 引数説明

● tagName

参照先のタグ名

● ditmName

参照先のデータアイテム名

● array1、array2

配列添え字 1、2

● valPtr

入力値ポインタ

● itemNo

データ番号

● option

以下の中から指定することができます。

- UCAOPT_TAG_SETIOP：データ参照異常時に、IOP アラームを発生します。
- NOOPTION

■ 詳細説明

UcaOtherTagReadToRvnF64S のデータの流れは以下のとおりです。

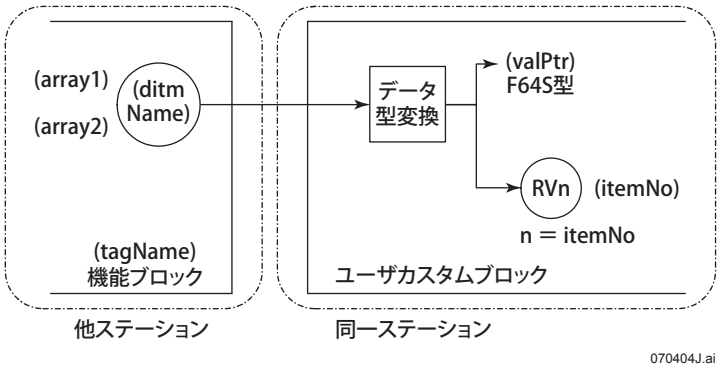


図　UcaOtherTagReadToRvnF64Sのデータの流れ

参照先機能ブロックは他ステーションに限ります。あらかじめ、UcaOtherTagReadToRvnF64S で参照する機能ブロックデータに対して、BDSET-1L でタグ名を登録しておいてください。タグ名を登録していない場合、UcaOtherTagReadToRvnF64S はエラーコード UCAERR_TAG_NOADL を戻り値として返します。
UcaOtherTagReadToRvnF64S では、参照に失敗した詳細な理由（相手ステーションがフェイルしているのか、タグ名が間違っているのか、など）は、分かりません。
引数 tagName、ditmName の英字部分は、大文字で記述してください。
引数 array1、array2 には、ditmName で指示されるデータアイテムが配列型でない場合、0 を指定してください。1 次元配列データの場合、array1 には配列添え字を、array2 には 0 を指定してください。2 次元配列データの場合、array1 には配列添え字 1 を、array2 には配列添え字 2 を指定してください。
UcaOtherTagReadToRvnF64S は、参照先のデータ型を F64S 型に変換し、引数 valPtr に格納します。参照先データがデータステータスつきであれば、そのデータステータスも同時に取得します。参照先データがデータステータスなしであれば、引数 valPtr のデータステータスに 0（正常）を格納します。
UcaOtherTagReadToRvnF64S は、valPtr に返すデータと同じデータを itemNo 番目の副入力 RVnn に設定します。itemNo が 0 ならデータアイテム RV に設定します。データの取得に失敗した場合には、RVnn（または RV）のデータは変更しませんが、RVnn（または RV）のデータステータスに BAD（BAD のみ真で他のデータステータスは偽）を設定します。
UcaOtherTagReadToRvnF64S は、以下のアラームを検出します。

表　UcaOtherTagReadToRvnF64Sのアラーム検出

オプション / アラーム	NOOPTION	UCAOPT_TAG_SETIOP
IOP	×	○
CNF	×	○

○： 検出する
×： 検出しない

オプションに UCAOPT_TAG_SETIOP を指定すると、機能ブロックのデータ参照に失敗した場合、プロセスアラーム IOP と CNF を検出します。参照に成功した場合でも、取得したデータのデータステータスが BAD のときにはプロセスアラーム IOP を検出します。
警報タブシートのビルダ定義項目［入力オープン警報］や［結合状態不良警報］に警報なしを指定した場合、対応するアラームは検出しません。

7.5 タグデータ1次元配列一括アクセス

タグデータ1次元配列一括アクセス関数は、ユーザカスタムブロックと同一ステーションまたは他ステーションに存在する機能ブロック関数の1次元配列データを、一括に取得または設定します。

■ タグデータ1次元配列一括アクセス関数の概要

以下にタグデータアクセス（1 次元配列一括アクセス）に使用するライブラリを示します。

表　タグ名を指定したデータ入出力の関数（1次元配列一括アクセス）

ステーション / データ型／読み書き	自ステーション（APCS内）	他ステーション（APCS外）
数値データ読み込み	UcaTagArray1ReadF64S	UcaOtherTagArray1ReadF64S
	UcaTagArray1ReadI32S	UcaOtherTagArray1ReadI32S
数値データ書き込み	UcaTagArray1WriteF64S	UcaOtherTagArray1WriteF64S
	UcaTagArray1WriteI32S	UcaOtherTagArray1WriteI32S

タグデータ 1 次元配列一括アクセスのサンプルプログラムが用意されています。

表　タグデータ1次元配列一括アクセスサンプルプログラム

ステーション / データ型／読み書き	自ステーション（APCS内）	他ステーション（APCS外）(*2)
制御ドローイング (*1) (*3)	ARRAY1.txt	OARRAY1.txt（APCS 側）
		FARRAY1.txt（FCS 側）
ユーザカスタムアルゴリズム (*3)	_SMPL_ARRAY1	_SMPL_OARRAY1（APCS 側のみ）

*1： 制御ドローイングビルダよりインポートしてください。
*2： APCS のユーザカスタムブロックから FCS のデータにアクセスするサンプルプログラムです。
*3： サンプルは、以下のフォルダにインストールされます。
<CENTUM VP インストール先 >
¥UcaEnv¥Sample¥LearnUca¥Drawings¥< 制御ドローイング >
¥UcaWork¥UcaSamples¥< ユーザカスタムアルゴリズム >

重要 ＜ CENTUM VP インストール先＞のサンプルのユーザカスタムアルゴリズムを直接リビルド、修正などしないでください。コピーしたものをリビルド、修正してください。

7.5.1 UcaTagArray1ReadF64S

UcaTagArray1ReadF64Sは、自ステーションの機能ブロックから1次元配列データを一括で取得し、F64S型の配列に格納します。

■ 関数名

I32 UcaTagArray1ReadF64S

■ 引数

● blockContext

- データ型： UcaBlockContext
- タイプ： IN
- 説明： ブロックコンテキスト

● tagName

- データ型： TAGN
- タイプ： IN
- 説明： タグ名

● ditmName

- データ型： DITMN
- タイプ： IN
- 説明： データアイテム名

● start1

- データ型： I32
- タイプ： IN
- 説明： データアイテム添え字開始（1 始まり）

● data[ ]

- データ型： F64S
- タイプ： OUT
- 説明： 読み込み値配列

● start2

- データ型： I32
- タイプ： IN
- 説明： 読み込み値添え字開始（0 始まり）

● num

- データ型： I32
- タイプ： IN
- 説明： データ数

● allStatus

- データ型： U32 *
- タイプ： OUT
- 説明： データステータス（全データの OR）

● err[ ]

- データ型： I32
- タイプ： OUT
- 説明： エラー配列

● option

- データ型： U32
- タイプ： IN
- 説明： オプション

■ 戻り値

- SUCCEED： 成功（全データ取得成功）
- SUCCEED 以外：
  最初（添え字が小さい）に取得失敗したデータに対するエラーコードです。
  UCAERR_PARA_PTRNULL： 引数のポインタが NULL
  UCAERR_OPT_INVALID： 誤ったオプションを指定した

■ 処理内容説明

タグ名とデータアイテム名を指定して、1 次元配列データを一括で取得します。

■ 引数説明

● tagName

参照先のタグ名

● ditmName

参照先のデータアイテム名

● start1

ditmName で指定した 1 次元配列データアイテムの配列添え字開始番号

● data[ ]

参照先のデータを格納する配列

● start2

data の配列添え字開始番号

● num

処理するデータの数

● allStatus
全データステータス

● err[ ]
対応するデータのエラーコード

● option
以下の中から指定することができます。
- UCAOPT_TAG_SETIOP：データ参照異常時に、IOP アラームを発生します。
- NOOPTION

■ 詳細説明

UcaTagArray1ReadF64S のデータの流れは以下のとおりです。

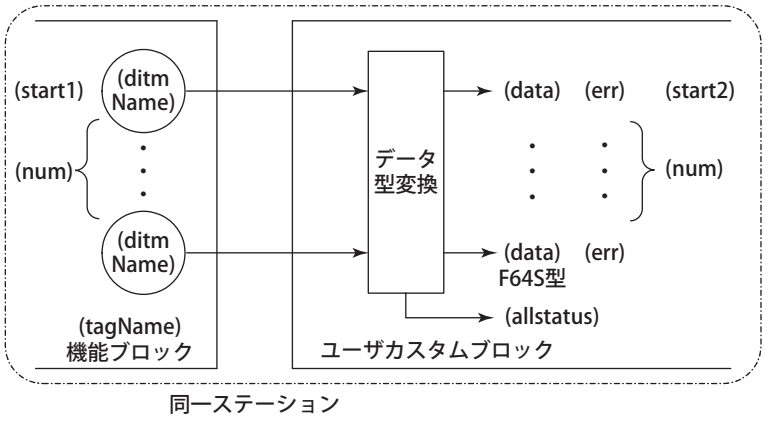


図　UcaTagArray1ReadF64Sのデータの流れ

参照先機能ブロックは同一ステーション内に限ります。
引数 tagName、ditmName の英字部分は、大文字で記述してください。
引数 start1 には、ditmName で指定した 1 次元配列データアイテムの配列添え字開始番号を指定します。データアイテムの配列添え字は 1 始まりです。
引数 data には、参照先のデータを格納する配列を指定します。参照先のデータ型が F64S 型以外の場合 UcaTagArray1ReadF64S は参照先データを F64S 型に変換し、引数 data に格納します。参照先データがデータステータスつきであれば、そのデータステータスも同時に取得します。参照先データがデータステータスなしであれば、対応するデータのデータステータスに 0（正常）を格納します。
引数 start2 には、引数 data の配列添え字開始番号を指定します。C 言語の配列の添え字は 0 始まりです。
UcaTagArray1ReadF64S は、引数 allStatus に、取得したデータの全データステータスを OR した結果を格納します。
UcaTagArray1ReadF64S は、引数 err[0] ～ err[num-1] に、対応するデータのエラーコードまたは 0（正常）を設定します。
UcaTagArray1ReadF64S は、参照に失敗したデータのデータステータスを BAD に設定します。

UcaTagArray1ReadF64S は、以下のアラームを検出します。

表　UcaTagArray1ReadF64Sのアラーム検出

オプション / アラーム	NOOPTION	UCAOPT_TAG_SETIOP
IOP	×	○
CNF	×	○

○： 検出する
×： 検出しない

オプションに UCAOPT_TAG_SETIOP を指定すると、機能ブロックのデータ参照に失敗した場合、プロセスアラーム IOP と CNF を検出します。参照に成功した場合でも、取得したデータのデータステータスが BAD のときにはプロセスアラーム IOP を設定します。
警報タブシートのビルダ定義項目［入力オープン警報］や［結合状態不良警報］に警報なしを指定した場合、対応するアラームは検出しません。

補足 自ステーションの機能ブロックから 1 次元配列データを一括で取得するユーザカスタムアルゴリズムの詳細については、以下の位置にあるサンプルワークスペース _SMPL_ARRAY1 を参照してください（7.5 節の最初に、タグデータ 1 次元配列一括アクセスサンプルプログラムの一覧があります）。

<CENTUM VP インストール先 >¥UcaEnv¥Sample¥UcaWork¥UcaSamples¥_SMPL_ARRAY1

7.5.2 UcaTagArray1ReadI32S

UcaTagArray1ReadI32Sは、自ステーションの機能ブロックから1次元配列データを一括で取得し、I32S型の配列に格納します。

■ 関数名

I32 UcaTagArray1ReadI32S

■ 引数

● blockContext

- データ型： UcaBlockContext
- タイプ： IN
- 説明： ブロックコンテキスト

● tagName

- データ型： TAGN
- タイプ： IN
- 説明： タグ名

● ditmName

- データ型： DITMN
- タイプ： IN
- 説明： データアイテム名

● start1

- データ型： I32
- タイプ： IN
- 説明： データアイテム添え字開始（1 始まり）

● data[ ]

- データ型： I32S
- タイプ： OUT
- 説明： 読み込み値配列

● start2

- データ型： I32
- タイプ： IN
- 説明： 読み込み値添え字開始（0 始まり）

● num

- データ型： I32
- タイプ： IN
- 説明： データ数

● allStatus

- データ型： U32 *
- タイプ： OUT
- 説明： データステータス（全データの OR）

● err[ ]

- データ型： I32
- タイプ： OUT
- 説明： エラー配列

● option

- データ型： U32
- タイプ： IN
- 説明： オプション

■ 戻り値

- SUCCEED： 成功（全データ取得成功）
- SUCCEED 以外：
  最初（添え字が小さい）に取得失敗したデータに対するエラーコードです。
  UCAERR_PARA_PTRNULL： 引数のポインタが NULL
  UCAERR_OPT_INVALID： 誤ったオプションを指定した

■ 処理内容説明

タグ名とデータアイテム名を指定して、1 次元配列データを一括で取得します。

■ 引数説明

● tagName

参照先のタグ名

● ditmName

参照先のデータアイテム名

● start1

ditmName で指定した 1 次元配列データアイテムの配列添え字開始番号

● data[ ]

参照先のデータを格納する配列

● start2

data の配列添え字開始番号

● num

処理するデータの数

● allStatus
全データステータス

● err[ ]
対応するデータのエラーコード

● option
以下の中から指定することができます。
- UCAOPT_TAG_SETIOP：データ参照異常時に、IOP アラームを発生します。
- NOOPTION

■ 詳細説明

UcaTagArray1ReadI32S のデータの流れは以下のとおりです。

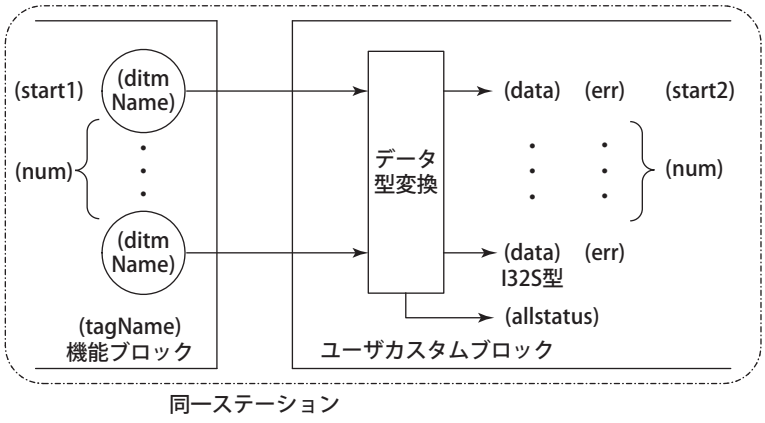


図　UcaTagArray1ReadI32Sのデータの流れ

参照先機能ブロックは同一ステーション内に限ります。
引数 tagName、ditmName の英字部分は、大文字で記述してください。
引数 start1 には、ditmName で指定した 1 次元配列データアイテムの配列添え字開始番号を指定します。データアイテムの配列添え字は 1 始まりです。
引数 data には、参照先のデータを格納する配列を指定します。参照先のデータ型が I32S 型以外の場合 UcaTagArray1ReadI32S は参照先データを I32S 型に変換し、引数 data に格納します。参照先データがデータステータスつきであれば、そのデータステータスも同時に取得します。参照先データがデータステータスなしであれば、対応するデータのデータステータスに 0（正常）を格納します。
引数 start2 には、引数 data の配列添え字開始番号を指定します。C 言語の配列の添え字は 0 始まりです。
UcaTagArray1ReadI32S は、引数 allStatus に、取得したデータの全データステータスを OR した結果を格納します。
UcaTagArray1ReadI32S は、引数 err[0] ～ err[num-1] に、対応するデータのエラーコードまたは 0（正常）を設定します。
UcaTagArray1ReadI32S は、参照に失敗したデータのデータステータスに BAD を設定します。

UcaTagArray1ReadI32S は、以下のアラームを検出します。

表　UcaTagArray1ReadI32Sのアラーム検出

オプション / アラーム	NOOPTION	UCAOPT_TAG_SETIOP
IOP	×	○
CNF	×	○

○： 検出する
×： 検出しない

オプションに UCAOPT_TAG_SETIOP を指定すると、機能ブロックのデータ参照に失敗した場合、プロセスアラーム IOP と CNF を検出します。参照に成功した場合でも、取得したデータのデータステータスが BAD のときにはプロセスアラーム IOP を検出します。
警報タブシートのビルダ定義項目［入力オープン警報］や［結合状態不良警報］に警報なしを指定した場合、対応するアラームは検出しません。

補足 自ステーションの機能ブロックから 1 次元配列データを一括で取得するユーザカスタムアルゴリズムの詳細については、以下の位置にあるサンプルワークスペース _SMPL_ARRAY1 を参照してください（7.5 節の最初に、タグデータ 1 次元配列一括アクセスサンプルプログラムの一覧があります）。

<CENTUM VP インストール先 >¥UcaEnv¥Sample¥UcaWork¥UcaSamples¥_SMPL_ARRAY1

7.5.3 UcaTagArray1WriteF64S

UcaTagArray1WriteF64Sは、F64S型の1次元配列データを自ステーションの機能ブロック1次元配列データアイテムに一括で設定します。

■ 関数名

I32 UcaTagArray1WriteF64S

■ 引数

● blockContext

- データ型： UcaBlockContext
- タイプ： IN
- 説明： ブロックコンテキスト

● tagName

- データ型： TAGN
- タイプ： IN
- 説明： タグ名

● ditmName

- データ型： DITMN
- タイプ： IN
- 説明： データアイテム名

● start1

- データ型： I32
- タイプ： IN
- 説明： データアイテム添え字開始（1 始まり）

● data[ ]

- データ型： F64S
- タイプ： IN
- 説明： 設定値配列

● start2

- データ型： I32
- タイプ： IN
- 説明： 設定値添え字開始（0 始まり）

● num

- データ型： I32
- タイプ： IN
- 説明： データ数

● err[ ]

- データ型： I32
- タイプ： OUT
- 説明： エラー配列

● option

- データ型： U32
- タイプ： IN
- 説明： オプション

■ 戻り値

- SUCCEED： 成功（全データ設定成功）
- SUCCEED 以外：
  最初（添え字が小さい）に設定失敗したデータに対するエラーコードです。
  UCAERR_NOTANUMBER：
  引数の浮動小数データが非数（Not a Number）である
  UCAERR_INFINITY： 引数の浮動小数データが無限大である
  UCAERR_PARA_PTRNULL： 引数のポインタが NULL
  UCAERR_OPT_INVALID： 誤ったオプションを指定した

■ 処理内容説明

タグ名とデータアイテム名を指定して、1 次元配列データを一括で設定します。

■ 引数説明

● tagName

参照先のタグ名

● ditmName

参照先のデータアイテム名

● start1

ditmName で指定した 1 次元配列データアイテムの配列添え字開始番号

● data[ ]

設定先のデータを格納する配列

● start2

data の配列添え字開始番号

● num

処理するデータの数

● err[ ]

対応するデータのエラーコード

● option

NOOPTION を指定してください。

■ 詳細説明

UcaTagArray1WriteF64S のデータの流れは以下のとおりです。

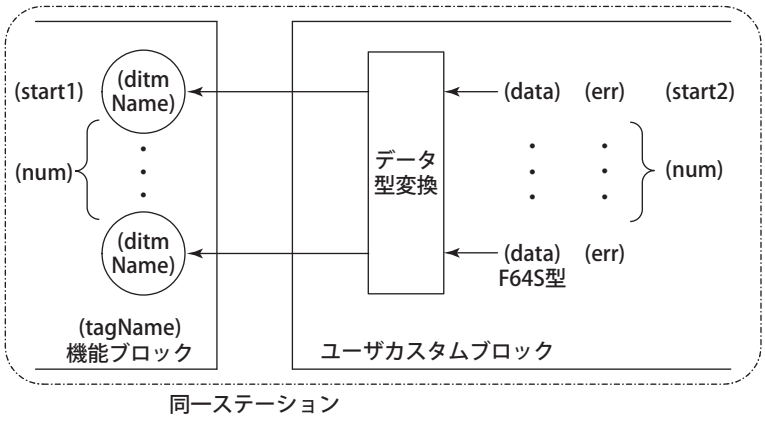


図　UcaTagArray1WriteF64Sのデータの流れ

設定先機能ブロックは同一ステーション内に限ります。
引数 tagName、ditmName の英字部分は、大文字で記述してください。
引数 start1 には、ditmName で指定した 1 次元配列データアイテムの配列添え字開始番号を指定します。データアイテムの配列添え字は 1 始まりです。
引数 data には、設定先のデータを格納する配列を指定します。設定先データアイテムのデータ型が F64S 型以外の場合、UcaTagArray1WriteF64S は引数 data の F64S 型データを設定先データアイテムのデータ型に変換し、設定を行います。
引数 start2 には、引数 data の配列添え字開始番号を指定します。C 言語の配列の添え字は 0 始まりです。
UcaTagArray1WriteF64S は、引数 err[0] ～ err[num-1] に、対応するデータのエラーコードまたは 0（正常）を設定します。

補足 自ステーションの機能ブロックに 1 次元配列データを一括で設定するユーザカスタムアルゴリズムの詳細については、以下の位置にあるサンプルワークスペース _SMPL_ARRAY1 を参照してください（7.5 節の最初に、タグデータ 1 次元配列一括アクセスサンプルプログラムの一覧があります）。

<CENTUM VP インストール先 >¥UcaEnv¥Sample¥UcaWork¥UcaSamples¥_SMPL_ARRAY1

7.5.4 UcaTagArray1WriteI32S

UcaTagArray1WriteI32Sは、I32S型の1次元配列データを自ステーションの機能ブロック1次元配列データアイテムに一括で設定します。

■ 関数名

I32 UcaTagArray1WriteI32S

■ 引数

● blockContext

- データ型： UcaBlockContext
- タイプ： IN
- 説明： ブロックコンテキスト

● tagName

- データ型： TAGN
- タイプ： IN
- 説明： タグ名

● ditmName

- データ型： DITMN
- タイプ： IN
- 説明： データアイテム名

● start1

- データ型： I32
- タイプ： IN
- 説明： データアイテム添え字開始（1 始まり）

● data[ ]

- データ型： I32S
- タイプ： IN
- 説明： 設定値配列

● start2

- データ型： I32
- タイプ： IN
- 説明： 設定値添え字開始（0 始まり）

● num

- データ型： I32
- タイプ： IN
- 説明： データ数

● err[ ]

- データ型： I32
- タイプ： OUT
- 説明： エラー配列

● option

- データ型： U32
- タイプ： IN
- 説明： オプション

■ 戻り値

- SUCCEED： 成功（全データ取得成功）
- SUCCEED 以外：
  最初（添え字が小さい）に設定失敗したデータに対するエラーコードです。
  UCAERR_PARA_PTRNULL： 引数のポインタが NULL
  UCAERR_OPT_INVALID： 誤ったオプションを指定した

■ 処理内容説明

タグ名とデータアイテム名を指定して、1 次元配列データを一括で設定します。

■ 引数説明

● tagName

参照先のタグ名

● ditmName

参照先のデータアイテム名

● start1

ditmName で指定した 1 次元配列データアイテムの配列添え字開始番号

● data[ ]

設定先のデータを格納する配列

● start2

data の配列添え字開始番号

● num

処理するデータの数

● err[ ]

対応するデータのエラーコード

● option

NOOPTION を指定してください。

■ 詳細説明

UcaTagArray1WriteI32S のデータの流れは以下のとおりです。

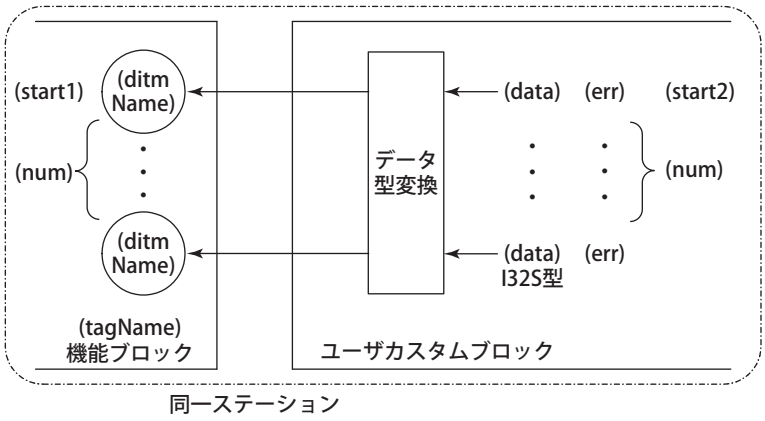


図　UcaTagArray1WriteI32Sのデータの流れ

設定先機能ブロックは同一ステーション内に限ります。
引数 tagName、ditmName の英字部分は、大文字で記述してください。
引数 start1 には、ditmName で指定した 1 次元配列データアイテムの配列添え字開始番号を指定します。データアイテムの配列添え字は 1 始まりです。
引数 data には、設定先のデータを格納する配列を指定します。設定先データアイテムのデータ型が I32S 型以外の場合、UcaTagArray1WriteI32S は引数 data の I32S 型データを設定先データアイテムのデータ型に変換し、設定を行います。
引数 start2 には、引数 data の配列添え字開始番号を指定します。C 言語の配列の添え字は 0 始まりです。
UcaTagArray1WriteI32S は、引数 err[0] ～ err[num-1] に、対応するデータのエラーコードまたは 0（正常）を設定します。

補足 自ステーションの機能ブロックに 1 次元配列データを一括で設定するユーザカスタムアルゴリズムの詳細については、以下の位置にあるサンプルワークスペース _SMPL_ARRAY1 を参照してください（7.5 節の最初に、タグデータ 1 次元配列一括アクセスサンプルプログラムの一覧があります）。

<CENTUM VP インストール先 >¥UcaEnv¥Sample¥UcaWork¥UcaSamples¥_SMPL_ARRAY1

7.5.5 UcaOtherTagArray1ReadF64S

UcaOtherTagArray1ReadF64Sは、他ステーションの機能ブロックから1次元配列データを一括で取得し、F64S型の配列に格納します。

■ 関数名

I32 UcaOtherTagArray1ReadF64S

■ 引数

● blockContext

- データ型： UcaBlockContext
- タイプ： IN
- 説明： ブロックコンテキスト

● tagName

- データ型： TAGN
- タイプ： IN
- 説明： タグ名

● ditmName

- データ型： DITMN
- タイプ： IN
- 説明： データアイテム名

● start1

- データ型： I32
- タイプ： IN
- 説明： データアイテム添え字開始（1 始まり）

● data[ ]

- データ型： F64S
- タイプ： OUT
- 説明： 読み込み値配列

● start2

- データ型： I32
- タイプ： IN
- 説明： 読み込み値添え字開始（0 始まり）

● num

- データ型： I32
- タイプ： IN
- 説明： データ数

● allStatus

- データ型： U32 *
- タイプ： OUT
- 説明： データステータス（全データの OR）

● err[ ]

- データ型： I32
- タイプ： OUT
- 説明： エラー配列

● option

- データ型： U32
- タイプ： IN
- 説明： オプション

■ 戻り値

- SUCCEED： 成功（全データ取得成功）
- SUCCEED 以外：
  最初（添え字が小さい）に取得失敗したデータに対するエラーコードです。
  UCAERR_TAG_NOADL： BDSET-1L でのステーション間結合未定義
  UCAERR_TAG_INVALIDDB： データベースタイプが正しくない
  UCAERR_PARA_PTRNULL： 引数のポインタが NULL
  UCAERR_OPT_INVALID： 誤ったオプションを指定した

■ 処理内容説明

タグ名とデータアイテム名を指定して、他ステーションのデータを参照します。

■ 引数説明

● tagName

参照先のタグ名

● ditmName

参照先のデータアイテム名

● start1

ditmName で指定した 1 次元配列データアイテムの配列添え字開始番号

● data[ ]

参照先のデータを格納する配列

● start2

data の配列添え字開始番号

● num

処理するデータの数

● allStatus
全データステータス

● err[ ]
対応するデータのエラーコード

● option
以下の中から指定することができます。
- UCAOPT_TAG_SETIOP：データ参照異常時に、IOP アラームを発生します。
- NOOPTION

■ 詳細説明

UcaOtherTagArray1ReadF64S のデータの流れは以下のとおりです。

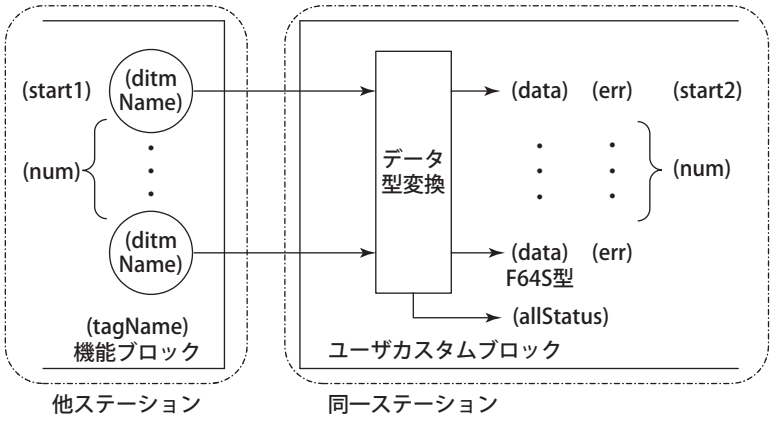


図　UcaOtherTagArray1ReadF64Sのデータの流れ

参照先機能ブロックは他ステーションに限ります。あらかじめ、UcaOtherTagArray1ReadF64S で参照する機能ブロックデータに対して、BDSET-1L でタグ名を登録しておいてください。タグ名を登録していない場合、UcaOtherTagArray1ReadF64S はエラーコード UCAERR_TAG_NOADL を戻り値として返します。
UcaOtherTagArray1ReadF64S では、参照に失敗した詳細な理由（相手ステーションがフェイルしているのか、タグ名が間違っているのか、など）は、分かりません。

引数 tagName、ditmName の英字部分は、大文字で記述してください。
引数 start1 には、ditmName で指定した 1 次元配列データアイテムの配列添え字開始番号を指定します。データアイテムの配列添え字は 1 始まりです。
引数 data には、参照先のデータを格納する配列を指定します。参照先のデータ型が F64S 型以外の場合、UcaOtherTagArray1ReadF64S は、参照先データを F64S 型に変換し、引数 data に格納します。参照先データがデータステータスつきであれば、そのデータステータスも同時に取得します。参照先データがデータステータスなしであれば、対応するデータのデータステータスに 0（正常）を格納します。
引数 start2 には、引数 data の配列添え字開始番号を指定します。C 言語の配列の添え字は 0 始まりです。
UcaOtherTagArray1ReadF64S は、引数 allStatus に、取得したデータの全データステータスを OR した結果を格納します。
UcaOtherTagArray1ReadF64S は、引数 err[0] ～ err[num-1] に、対応するデータのエラーコードまたは 0（正常）を設定します。
UcaOtherTagArray1ReadF64S は、参照に失敗したデータのデータステータスに BAD を設定します。
UcaOtherTagArray1ReadF64S は、以下のアラームを検出します。

表　UcaOtherTagArray1ReadF64Sのアラーム検出

オプション / アラーム	NOOPTION	UCAOPT_TAG_SETIOP
IOP	×	○
CNF	×	○

○： 検出する
×： 検出しない

オプションに UCAOPT_TAG_SETIOP を指定すると、機能ブロックのデータ参照に失敗した場合、プロセスアラーム IOP と CNF を検出します。参照に成功した場合でも、取得したデータのデータステータスが BAD のときにはプロセスアラーム IOP を検出します。
警報タブシートでビルダ定義項目［入力オープン警報］や［結合状態不良警報］に警報なしを指定した場合、対応するアラームは検出しません。

補足 他ステーションの機能ブロックから 1 次元配列データを一括で取得するユーザカスタムアルゴリズムの詳細については、以下の位置にあるサンプルワークスペース _SMPL_OARRAY1 を参照してください（7.5 節の最初に、タグデータ 1 次元配列一括アクセスサンプルプログラムの一覧があります）。

<CENTUM VP インストール先 >¥UcaEnv¥Sample¥UcaWork¥UcaSamples¥_SMPL_OARRAY1

7.5.6 UcaOtherTagArray1ReadI32S

UcaOtherTagArray1ReadI32Sは、他ステーションの機能ブロックから1次元配列データを一括で取得し、I32S型の配列に格納します。

■ 関数名

I32 UcaOtherTagArray1ReadI32S

■ 引数

● blockContext

- データ型： UcaBlockContext
- タイプ： IN
- 説明： ブロックコンテキスト

● tagName

- データ型： TAGN
- タイプ： IN
- 説明： タグ名

● ditmName

- データ型： DITMN
- タイプ： IN
- 説明： データアイテム名

● start1

- データ型： I32
- タイプ： IN
- 説明： データアイテム添え字開始（1 始まり）

● data[ ]

- データ型： I32S
- タイプ： OUT
- 説明： 読み込み値配列

● start2

- データ型： I32
- タイプ： IN
- 説明： 読み込み値添え字開始（0 始まり）

● num

- データ型： I32
- タイプ： IN
- 説明： データ数

● allStatus

- データ型： U32 *
- タイプ： OUT
- 説明： データステータス（全データの OR）

● err[ ]

- データ型： I32
- タイプ： OUT
- 説明： エラー配列

● option

- データ型： U32
- タイプ： IN
- 説明： オプション

■ 戻り値

- SUCCEED： 成功（全データ取得成功）
- SUCCEED 以外：
  最初（添え字が小さい）に取得失敗したデータに対するエラーコードです。
  UCAERR_TAG_NOADL： BDSET-1L でのステーション間結合未定義
  UCAERR_TAG_INVALIDDB： データベースタイプが正しくない
  UCAERR_PARA_PTRNULL： 引数のポインタが NULL
  UCAERR_OPT_INVALID： 誤ったオプションを指定した

■ 処理内容説明

タグ名とデータアイテム名を指定して、他ステーションのデータを参照します。

■ 引数説明

● tagName

参照先のタグ名

● ditmName

参照先のデータアイテム名

● start1

ditmName で指定した 1 次元配列データアイテムの配列添え字開始番号

● data[ ]

参照先のデータを格納する配列

● start2

data の配列添え字開始番号

● num

処理するデータの数

● allStatus
全データステータス

● err[ ]
対応するデータのエラーコード

● option
以下の中から指定することができます。
- UCAOPT_TAG_SETIOP：データ参照異常時に、IOP アラームを発生します。
- NOOPTION

■ 詳細説明

UcaOtherTagArray1ReadI32S のデータの流れは以下のとおりです。

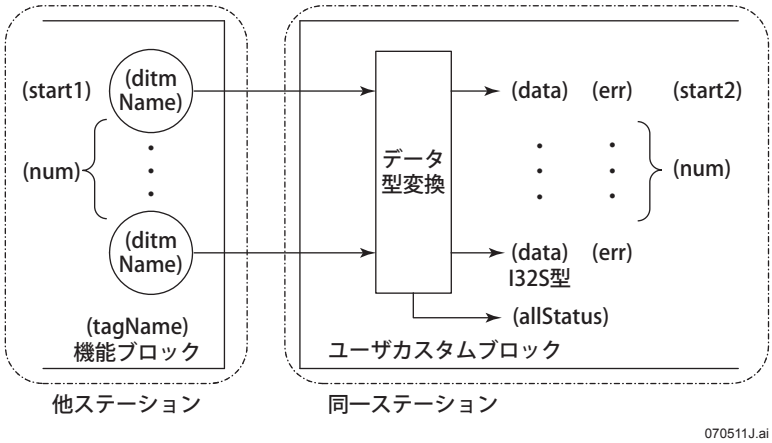


図　UcaOtherTagArray1ReadI32Sのデータの流れ

参照先機能ブロックは他ステーションに限ります。あらかじめ、UcaOtherTagArray1ReadI32S で参照する機能ブロックデータに対して、BDSET-1L でタグ名を登録しておいてください。タグ名を登録していない場合、UcaOtherTagArray1ReadI32S はエラーコード UCAERR_TAG_NOADL を戻り値として返します。
UcaOtherTagArray1ReadI32S では、参照に失敗した詳細な理由（相手ステーションがフェイルしているのか、タグ名が間違っているのか、など）は、分かりません。
引数 tagName、ditmName の英字部分は、大文字で記述してください。
引数 start1 には、ditmName で指定した 1 次元配列データアイテムの配列添え字開始番号を指定します。データアイテムの配列添え字は 1 始まりです。
引数 data には、参照先のデータを格納する配列を指定します。参照先のデータ型が I32S 型以外の場合、UcaOtherTagArray1ReadI32S は、参照先データを I32S 型に変換し、引数データに格納します。参照先データがデータステータスつきであれば、そのデータステータスも同時に取得します。参照先データがデータステータスなしであれば、対応するデータのデータステータスに 0（正常）を格納します。
引数 start2 には、引数 data の配列添え字開始番号を指定します。C 言語の配列の添え字は 0 始まりです。

UcaOtherTagArray1ReadI32S は、引数 allStatus に、取得したデータの全データステータスを OR した結果を格納します。
UcaOtherTagArray1ReadI32S は、引数 err[0] ～ err[num-1] に、対応するデータのエラーコードまたは 0（正常）を設定します。
UcaOtherTagArray1ReadI32S は、参照に失敗したデータのデータステータスに BAD を設定します。
UcaOtherTagArray1ReadI32S は、以下のアラームを検出します。

表　UcaOtherTagArray1ReadI32Sのアラーム検出

オプション / アラーム	NOOPTION	UCAOPT_TAG_SETIOP
IOP	×	○
CNF	×	○

○： 検出する
×： 検出しない

オプションに UCAOPT_TAG_SETIOP を指定すると、機能ブロックのデータ参照に失敗した場合、プロセスアラーム IOP と CNF を検出します。参照に成功した場合でも、取得したデータのデータステータスが BAD のときにはプロセスアラーム IOP を検出します。
警報タブシートでビルダ定義項目［入力オープン警報］や［結合状態不良警報］に警報なしを指定した場合、対応するアラームは検出しません。

補足 他ステーションの機能ブロックから 1 次元配列データを一括で取得するユーザカスタムアルゴリズムの詳細については、以下の位置にあるサンプルワークスペース _SMPL_OARRAY1 を参照してください（7.5 節の最初に、タグデータ 1 次元配列一括アクセスサンプルプログラムの一覧があります）。

<CENTUM VP インストール先 >¥UcaEnv¥Sample¥UcaWork¥UcaSamples¥_SMPL_OARRAY1

7.5.7 UcaOtherTagArray1WriteF64S

UcaOtherTagArray1WriteF64Sは、F64S型の1次元配列データを他ステーションの機能ブロック1次元配列データアイテムに一括で設定します。

■ 関数名

I32 UcaOtherTagArray1WriteF64S

■ 引数

● blockContext

- データ型： UcaBlockContext
- タイプ： IN
- 説明： ブロックコンテキスト

● tagName

- データ型： TAGN
- タイプ： IN
- 説明： タグ名

● ditmName

- データ型： DITMN
- タイプ： IN
- 説明： データアイテム名

● start1

- データ型： I32
- タイプ： IN
- 説明： データアイテム添え字開始（1 始まり）

● data[ ]

- データ型： F64S
- タイプ： IN
- 説明： 設定値配列

● start2

- データ型： I32
- タイプ： IN
- 説明： 設定値添え字開始（0 始まり）

● num

- データ型： I32
- タイプ： IN
- 説明： データ数

● err[ ]

- データ型： I32
- タイプ： OUT
- 説明： エラー配列

● option

- データ型： U32
- タイプ： IN
- 説明： オプション

■ 戻り値

- SUCCEED： 成功（全データ設定成功）
- SUCCEED 以外：
  最初（添え字が小さい）に設定失敗したデータに対するエラーコードです。
  UCAERR_TAG_NOADL： BDSET-1L でのステーション間結合未定義
  UCAERR_NOTANUMBER：
  引数の浮動小数データが非数（Not a Number）である
  UCAERR_INFINITY： 引数の浮動小数データが無限大である
  UCAERR_TAG_INVALIDDB： データベースタイプが正しくない
  UCAERR_PARA_PTRNULL： 引数のポインタが NULL
  UCAERR_OPT_INVALID： 誤ったオプションを指定した

■ 処理内容説明

タグ名とデータアイテム名を指定して、他ステーションのデータを設定します。

■ 引数説明

● tagName

設定先のタグ名

● ditmName

設定先のデータアイテム名

● start1

ditmName で指定した 1 次元配列データアイテムの配列添え字開始番号

● data[ ]

設定先のデータを格納する配列

● start2

data の配列添え字開始番号

● num

処理するデータの数

● err[ ]

対応するデータのエラーコード

● option

NOOPTION を指定してください。

■ 詳細説明

UcaOtherTagArray1WriteF64S のデータの流れは以下のとおりです。

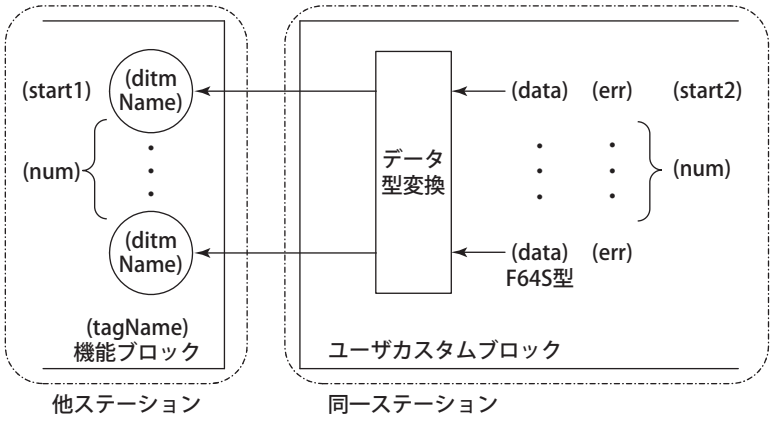


図　UcaOtherTagArray1WriteF64Sのデータの流れ

設定先機能ブロックは他ステーションに限ります。あらかじめ、UcaOtherTagArray1WriteF64S で設定する機能ブロックデータに対して、BDSET-1L でタグ名を登録しておいてください。タグ名を登録していない場合、UcaOtherTagArray1WriteF64S はエラーコード UCAERR_TAG_NOADL を戻り値として返します。

UcaOtherTagArray1WriteF64S では、参照に失敗した詳細な理由（相手ステーションがフェイルしているのか、タグ名が間違っているのか、設定先のデータが文字列型、など）は、分かりません。

引数 tagName、ditmName の英字部分は、大文字で記述してください。

引数 start1 には、ditmName で指定した 1 次元配列データアイテムの配列添え字開始番号を指定します。データアイテムの配列添え字は 1 始まりです。

引数 data には、設定先のデータを格納する配列を F64S 型で指定します。設定先データアイテムのデータ型が F64S 型以外の場合、UcaOtherTagArray1WriteI32S は引数 data の F64S 型データを設定先データアイテムのデータ型に変換し、設定を行います。

引数 start2 には、引数 data の配列添え字開始番号を指定します。C 言語の配列の添え字は 0 始まりです。

UcaOtherTagArray1WriteF64S は、引数 err[0] ～ err[num-1] に、対応するデータのエラーコードまたは 0（正常）を設定します。

補足 他ステーションの機能ブロックに 1 次元配列データを一括で設定するユーザカスタムアルゴリズムの詳細については、以下の位置にあるサンプルワークスペース _SMPL_OARRAY1 を参照してください（7.5 節の最初に、タグデータ 1 次元配列一括アクセスサンプルプログラムの一覧があります）。

<CENTUM VP インストール先 >¥UcaEnv¥Sample¥UcaWork¥UcaSamples¥_SMPL_OARRAY1

7.5.8 UcaOtherTagArray1WriteI32S

UcaOtherTagArray1WriteI32Sは、I32S型の1次元配列データを他ステーションの機能ブロック1次元配列データアイテムに一括で設定します。

■ 関数名

I32 UcaOtherTagArray1WriteI32S

■ 引数

● blockContext

- データ型： UcaBlockContext
- タイプ： IN
- 説明： ブロックコンテキスト

● tagName

- データ型： TAGN
- タイプ： IN
- 説明： タグ名

● ditmName

- データ型： DITMN
- タイプ： IN
- 説明： データアイテム名

● start1

- データ型： I32
- タイプ： IN
- 説明： データアイテム添え字開始（1 始まり）

● data[ ]

- データ型： I32S
- タイプ： IN
- 説明： 設定値配列

● start2

- データ型： I32
- タイプ： IN
- 説明： 設定値添え字開始（0 始まり）

● num

- データ型： I32
- タイプ： IN
- 説明： データ数

● **err[ ]**
- データ型： I32
- タイプ： OUT
- 説明： エラー配列

● **option**
- データ型： U32
- タイプ： IN
- 説明： オプション

■ 戻り値

- SUCCEED： 成功
- SUCCEED 以外：
  最初（添え字が小さい）に設定失敗したデータに対するエラーコードです。
  UCAERR_TAG_NOADL： BDSET-1L でのステーション間結合未定義
  UCAERR_TAG_INVALIDDB： データベースタイプが正しくない
  UCAERR_PARA_PTRNULL： 引数のポインタが NULL
  UCAERR_OPT_INVALID： 誤ったオプションを指定した

■ 処理内容説明

タグ名とデータアイテム名を指定して、他ステーションのデータを設定します。

■ 引数説明

● **tagName**
設定先のタグ名

● **ditmName**
設定先のデータアイテム名

● **start1**
ditmName で指定した 1 次元配列データアイテムの配列添え字開始番号

● **data[ ]**
設定先のデータを格納する配列

● **start2**
data の配列添え字開始番号

● **num**
処理するデータの数

● **err[ ]**
対応するデータのエラーコード

● **option**
NOOPTION を指定してください。

■ 詳細説明

UcaOtherTagArray1WriteI32S のデータの流れは以下のとおりです。

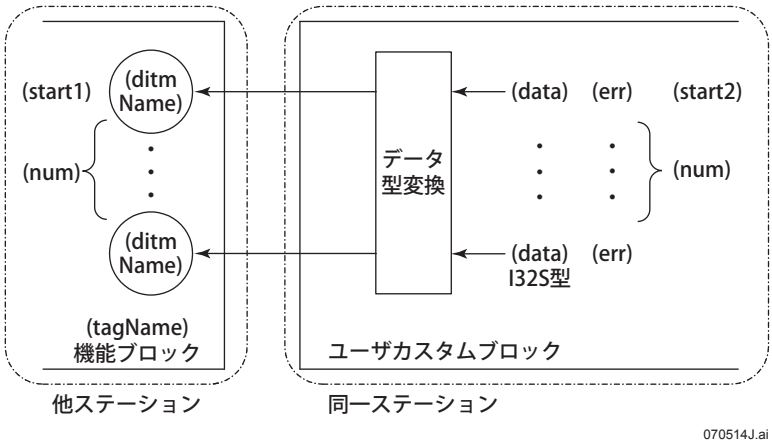


図　UcaOtherTagArray1WriteI32Sのデータの流れ

設定先機能ブロックは他ステーションに限ります。あらかじめ、UcaOtherTagArray1WriteI32S で設定する機能ブロックデータに対して、BDSET-1L でタグ名を登録しておいてください。タグ名を登録していない場合、UcaOtherTagArray1WriteI32S はエラーコード UCAERR_TAG_NOADL を戻り値として返します。

UcaOtherTagArray1ReadI32S では、参照に失敗した詳細な理由（相手ステーションがフェイルしているのか、タグ名が間違っているのか、設定先のデータが文字列型、など）は、分かりません。

引数 tagName、ditmName の英字部分は、大文字で記述してください。

引数 start1 には、ditmName で指定した 1 次元配列データアイテムの配列添え字開始番号を指定します。データアイテムの配列添え字は 1 始まりです。

引数 data には、設定先のデータを格納する配列を I32S 型で指定します。設定先データアイテムのデータ型が I32S 型以外の場合、UcaOtherTagArray1WriteI32S は引数 data の I32S 型データを設定先データアイテムのデータ型に変換し、設定を行います。

引数 start2 には、引数 data の配列添え字開始番号を指定します。C 言語の配列の添え字は 0 始まりです。

UcaOtherTagArray1WriteI32S は、引数 err[0] ～ err[num-1] に、対応するデータのエラーコードまたは 0（正常）を設定します。

補足 他ステーションの機能ブロックに 1 次元配列データを一括で設定するユーザカスタムアルゴリズムの詳細については、以下の位置にあるサンプルワークスペース _SMPL_OARRAY1 を参照してください（7.5 節の最初に、タグデータ 1 次元配列一括アクセスサンプルプログラムの一覧があります）。

<CENTUM VP インストール先 >¥UcaEnv¥Sample¥UcaWork¥UcaSamples¥_SMPL_OARRAY1

8. 温圧補正演算

温圧補正演算関数は、液体流量の補正演算を実行する関数です。温圧補正とASTM補正の2種類の関数群について説明します。

8.1 温圧補正

温圧補正関数は、絞り機能（オリフィス）を使った差圧式流量計の測定流量に対する補正演算を実行する関数です。理想気体に対する補正演算を行います。

8.1.1 UcaCalcTc

UcaCalcTcは、温度補正（温度単位：摂氏）を行います。

■ 関数名

I32 UcaCalcTc

■ 引数

● blockContext

- データ型： UcaBlockContext
- タイプ： IN
- 説明： ブロックコンテキスト

● Fi

- データ型： F64 *
- タイプ： IN/OUT
- 説明： 測定流量

● T

- データ型： F64
- タイプ： IN
- 説明： 測定温度［℃］

● Tb

- データ型： F64
- タイプ： IN
- 説明： 基準温度［℃］

● option

- データ型： U32
- タイプ： IN
- 説明： オプション

■ 戻り値

- SUCCEED： 正常終了
- UCAERR_TC_OVERFLOW： オーバフロー
- UCAERR_TC_ROOTMINUS： ルートの中が負
- UCAERR_TC_ZERODIV： ゼロ割り発生
- UCAERR_PARA_PTRNULL： 引数のポインタが NULL
- UCAERR_OPT_INVALID： 誤ったオプションを指定した

■ 処理内容説明

温度補正を行います（温度単位：摂氏）。

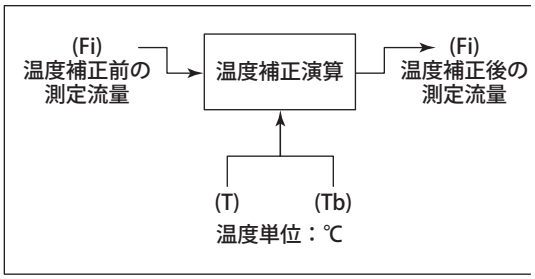
■ 引数説明

● option

NOOPTION を指定してください。

■ 詳細説明

UcaCalcTc は、引数 Fi の測定流量に対して温度補正（温度単位：℃）を行い、結果を引数 Fi に格納します。



080101J.ai

図　UcaCalcTcのデータの流れ

温度補正演算式を以下に示します。

$$TC(Fi、T、Tb) = \sqrt{\frac{Tb + 273.15}{T + 273.15}} \times Fi$$

Fi　：測定流量
T　：測定温度［℃］
Tb　：基準温度［℃］

080102J.ai

8.1.2 UcaCalcPcp

UcaCalcPcpは、圧力補正（圧力単位：Pa）を行います。

■ 関数名

I32 UcaCalcPcp

■ 引数

● blockContext

- データ型： UcaBlockContext
- タイプ： IN
- 説明： ブロックコンテキスト

● Fi

- データ型： F64 *
- タイプ： IN/OUT
- 説明： 測定流量

● P

- データ型： F64
- タイプ： IN
- 説明： 測定圧力［Pa］

● Pb

- データ型： F64
- タイプ： IN
- 説明： 基準圧力［Pa］

● option

- データ型： U32
- タイプ： IN
- 説明： オプション

■ 戻り値

- SUCCEED： 正常終了
- UCAERR_TC_OVERFLOW： オーバフロー
- UCAERR_TC_ROOTMINUS： ルートの中が負
- UCAERR_TC_ZERODIV： ゼロ割り発生
- UCAERR_PARA_PTRNULL： 引数のポインタが NULL
- UCAERR_OPT_INVALID： 誤ったオプションを指定した

■ 処理内容説明

圧力補正を行います（圧力単位：Pa）。

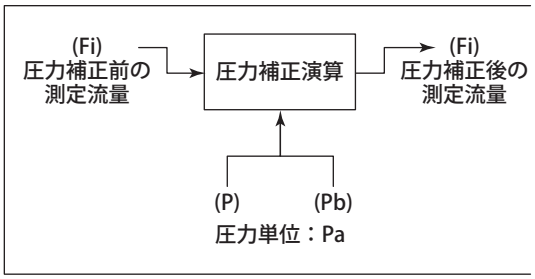
■ 引数説明

● option

NOOPTION を指定してください。

■ 詳細説明

UcaCalcPcp は、引数 Fi の測定流量に対して圧力補正（圧力単位：Pa）を行い、結果を引数 Fi に格納します。



080103J.ai

図　UcaCalcPcpのデータの流れ

圧力補正演算式を以下に示します。

$$PCP（Fi、P、Pb）= \sqrt{\frac{Pb + 1.01325 \times 10^5}{P + 1.01325 \times 10^5}} \times Fi$$

Fi　：測定流量
P　：測定圧力［Pa］
Pb　：基準圧力［Pa］

080104J.ai

8.1.3 UcaCalcPckp

UcaCalcPckpは、圧力補正（圧力単位：kPa）を行います。

■ 関数名

I32 UcaCalcPckp

■ 引数

● blockContext

- データ型： UcaBlockContext
- タイプ： IN
- 説明： ブロックコンテキスト

● Fi

- データ型： F64 *
- タイプ： IN/OUT
- 説明： 測定流量

● P

- データ型： F64
- タイプ： IN
- 説明： 測定圧力［kPa］

● Pb

- データ型： F64
- タイプ： IN
- 説明： 基準圧力［kPa］

● option

- データ型： U32
- タイプ： IN
- 説明： オプション

■ 戻り値

- SUCCEED： 正常終了
- UCAERR_TC_OVERFLOW： オーバフロー
- UCAERR_TC_ROOTMINUS： ルートの中が負
- UCAERR_TC_ZERODIV： ゼロ割り発生
- UCAERR_PARA_PTRNULL： 引数のポインタが NULL
- UCAERR_OPT_INVALID： 誤ったオプションを指定した

■ 処理内容説明

圧力補正を行います（圧力単位：kPa）。

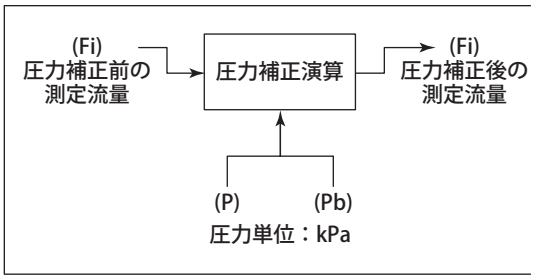
■ 引数説明

● option

NOOPTION を指定してください。

■ 詳細説明

UcaCalcPckp は、引数 Fi の測定流量に対して圧力補正（圧力単位：kPa）を行い、結果を引数 Fi に格納します。



080105J.ai

図　UcaCalcPckpのデータの流れ

圧力補正演算式を以下に示します。

$$PCKP(Fi、P、Pb) = \sqrt{\frac{Pb + 1.01325 \times 10^2}{P + 1.01325 \times 10^2}} \times Fi$$

Fi　：測定流量
P　：測定圧力［kPa］
Pb　：基準圧力［kPa］

080106J.ai

8.1.4 UcaCalcPcmp

UcaCalcPcmpは、圧力補正（圧力単位：MPa）を行います。

■ 関数名

I32 UcaCalcPcmp

■ 引数

● blockContext

- データ型： UcaBlockContext
- タイプ： IN
- 説明： ブロックコンテキスト

● Fi

- データ型： F64 *
- タイプ： IN/OUT
- 説明： 測定流量

● P

- データ型： F64
- タイプ： IN
- 説明： 測定圧力［MPa］

● Pb

- データ型： F64
- タイプ： IN
- 説明： 基準圧力［MPa］

● option

- データ型： U32
- タイプ： IN
- 説明： オプション

■ 戻り値

- SUCCEED： 正常終了
- UCAERR_TC_OVERFLOW： オーバフロー
- UCAERR_TC_ROOTMINUS： ルートの中が負
- UCAERR_TC_ZERODIV： ゼロ割り発生
- UCAERR_PARA_PTRNULL： 引数のポインタが NULL
- UCAERR_OPT_INVALID： 誤ったオプションを指定した

■ 処理内容説明

圧力補正を行います（圧力単位：MPa）。

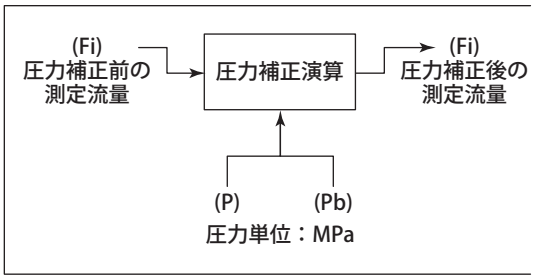
■ 引数説明

● option

NOOPTION を指定してください。

■ 詳細説明

UcaCalcPcmp は、引数 Fi の測定流量に対して圧力補正（圧力単位：MPa）を行い、結果を引数 Fi に格納します。



080107J.ai

図　UcaCalcPcmpのデータの流れ

圧力補正演算式を以下に示します。

$$PCMP（Fi、P、Pb）= \sqrt{\frac{Pb + 1.01325 \times 10^{-1}}{P + 1.01325 \times 10^{-1}}} \times Fi$$

Fi　：測定流量
P　：測定圧力［MPa］
Pb　：基準圧力［MPa］

080108J.ai

8.1.5 UcaCalcTpcp

UcaCalcTpcpは、温圧補正（温度単位：摂氏、圧力単位：Pa）を行います。

■ 関数名

I32 UcaCalcTpcp

■ 引数

● blockContext

- データ型： UcaBlockContext
- タイプ： IN
- 説明： ブロックコンテキスト

● Fi

- データ型： F64 *
- タイプ： IN/OUT
- 説明： 測定流量

● T

- データ型： F64
- タイプ： IN
- 説明： 測定温度［℃］

● P

- データ型： F64
- タイプ： IN
- 説明： 測定圧力［Pa］

● Tb

- データ型： F64
- タイプ： IN
- 説明： 基準温度［℃］

● Pb

- データ型： F64
- タイプ： IN
- 説明： 基準圧力［Pa］

● option

- データ型： U32
- タイプ： IN
- 説明： オプション

■ 戻り値

- SUCCEED： 正常終了
- UCAERR_TC_OVERFLOW： オーバフロー
- UCAERR_TC_ROOTMINUS： ルートの中が負
- UCAERR_TC_ZERODIV： ゼロ割り発生
- UCAERR_PARA_PTRNULL： 引数のポインタが NULL
- UCAERR_OPT_INVALID： 誤ったオプションを指定した

■ 処理内容説明

温圧補正を行います（温度単位：摂氏、圧力単位：Pa）。

■ 引数説明

● option

NOOPTION を指定してください。

■ 詳細説明

UcaCalcTpcp は、引数 Fi の測定流量に対して温圧補正（温度単位：℃、圧力単位：Pa）を行い、結果を引数 Fi に格納します。

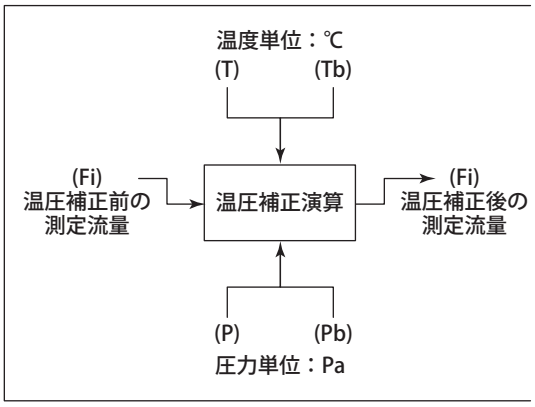


図　UcaCalcTpcpのデータの流れ

温圧補正演算式を以下に示します。

$$TPCP（Fi、T、P、Tb、Pb）= \sqrt{\frac{Pb + 1.01325 \times 10^5}{P + 1.01325 \times 10^5}} \times \frac{Tb + 273.15}{T + 273.15} \times Fi$$

Fi ：測定流量
P ：測定圧力［Pa］
T ：測定温度［℃］
Pb ：基準圧力［Pa］
Tb ：基準温度［℃］

080110J.ai

8.1.6 UcaCalcTpckp

UcaCalcTpckpは、温圧補正（温度単位：摂氏、圧力単位：kPa）を行います。

■ 関数名

I32 UcaCalcTpckp

■ 引数

● blockContext

- データ型： UcaBlockContext
- タイプ： IN
- 説明： ブロックコンテキスト

● Fi

- データ型： F64 *
- タイプ： IN/OUT
- 説明： 測定流量

● T

- データ型： F64
- タイプ： IN
- 説明： 測定温度［℃］

● P

- データ型： F64
- タイプ： IN
- 説明： 測定圧力［kPa］

● Tb

- データ型： F64
- タイプ： IN
- 説明： 基準温度［℃］

● Pb

- データ型： F64
- タイプ： IN
- 説明： 基準圧力［kPa］

● option

- データ型： U32
- タイプ： IN
- 説明： オプション

■ 戻り値

- SUCCEED： 正常終了
- UCAERR_TC_OVERFLOW： オーバフロー
- UCAERR_TC_ROOTMINUS： ルートの中が負
- UCAERR_TC_ZERODIV： ゼロ割り発生
- UCAERR_PARA_PTRNULL： 引数のポインタが NULL
- UCAERR_OPT_INVALID： 誤ったオプションを指定した

■ 処理内容説明

温圧補正を行います（温度単位：摂氏、圧力単位：kPa）。

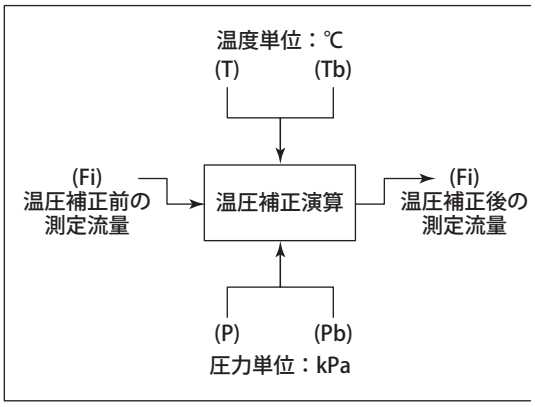
■ 引数説明

● option

NOOPTION を指定してください。

■ 詳細説明

UcaCalcTpckp は、引数 Fi の測定流量に対して温圧補正（温度単位：℃、圧力単位：kPa）を行い、結果を引数 Fi に格納します。



080111J.ai

図　UcaCalcTpckpのデータの流れ

温圧補正演算式を以下に示します。

$$TPCKP(Fi、T、P、Tb、Pb) = \sqrt{\frac{Pb + 1.01325 \times 10^2}{P + 1.01325 \times 10^2}} \times \frac{Tb + 273.15}{T + 273.15} \times Fi$$

Fi ：測定流量
P ：測定圧力［kPa］
T ：測定温度［℃］
Pb ：基準圧力［kPa］
Tb ：基準温度［℃］

080112J.ai

8.1.7 UcaCalcTpcmp

UcaCalcTpcmpは、温圧補正（温度単位：摂氏、圧力単位：MPa）を行います。

■ 関数名

I32 UcaCalcTpcmp

■ 引数

● blockContext

- データ型： UcaBlockContext
- タイプ： IN
- 説明： ブロックコンテキスト

● Fi

- データ型： F64 *
- タイプ： IN/OUT
- 説明： 測定流量

● T

- データ型： F64
- タイプ： IN
- 説明： 測定温度［℃］

● P

- データ型： F64
- タイプ： IN
- 説明： 測定圧力［MPa］

● Tb

- データ型： F64
- タイプ： IN
- 説明： 基準温度［℃］

● Pb

- データ型： F64
- タイプ： IN
- 説明： 基準圧力［MPa］

● option

- データ型： U32
- タイプ： IN
- 説明： オプション

■ 戻り値

- SUCCEED： 正常終了
- UCAERR_TC_OVERFLOW： オーバフロー
- UCAERR_TC_ROOTMINUS： ルートの中が負
- UCAERR_TC_ZERODIV： ゼロ割り発生
- UCAERR_PARA_PTRNULL： 引数のポインタが NULL
- UCAERR_OPT_INVALID： 誤ったオプションを指定した

■ 処理内容説明

温圧補正を行います（温度単位：摂氏、圧力単位：MPa）。

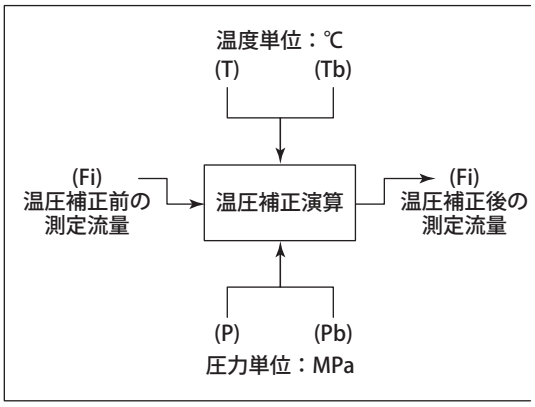
■ 引数説明

● option

NOOPTION を指定してください。

■ 詳細説明

UcaCalcTpcmp は、引数 Fi の測定流量に対して温圧補正（温度単位：℃、圧力単位：MPa）を行い、結果を引数 Fi に格納します。



080113J.ai

図　UcaCalcTpcmpのデータの流れ

温圧補正演算式を以下に示します。

$$TPCMP（Fi、T、P、Tb、Pb）= \sqrt{\frac{Pb + 1.01325 \times 10^{-1}}{P + 1.01325 \times 10^{-1}}} \times \frac{Tb + 273.15}{T + 273.15} \times Fi$$

Fi ：測定流量
P ：測定圧力［MPa］
T ：測定温度［℃］
Pb ：基準圧力［MPa］
Tb ：基準温度［℃］

080114J.ai

8.2 ASTM補正

ASTM補正関数は、流体の流量の補正演算を行います。UcaCalcAstm1関数は、ASTM（旧JIS）補正に基づき補正流量計算を行います。UcaCalcAstm2～4関数は、ASTM補正（新JIS）に基づき補正流量計算を行います。UcaCalcAstm2は原油を、UcaCalcAstm3は燃料油を、UcaCalcAstm4は潤滑油をそれぞれ扱います。

表　ASTM補正

関数名	補正方式	原料
UcaCalcAstm1	ASTM 補正（旧 JIS）	―
UcaCalcAstm2	ASTM 補正（新 JIS）	原油
UcaCalcAstm3		燃料油
UcaCalcAstm4		潤滑油

8.2.1 UcaCalcAstm1

UcaCalcAstm1は、ASTM補正（旧JIS）を行います。

■ 関数名

I32 UcaCalcAstm1

■ 引数

● blockContext

- データ型： UcaBlockContext
- タイプ： IN
- 説明： ブロックコンテキスト

● T

- データ型： F64
- タイプ： IN
- 説明： 測定温度

● F

- データ型： F64 *
- タイプ： IN/OUT
- 説明： 測定流量

● C

- データ型： F64
- タイプ： IN
- 説明： 比重

● option

- データ型： U32
- タイプ： IN
- 説明： オプション

■ 戻り値

- SUCCEED： 正常終了
- UCAERR_TC_OVERFLOW： オーバフロー
- UCAERR_TC_ROOTMINUS： ルートの中が負
- UCAERR_TC_ZERODIV： ゼロ割り発生
- UCAERR_PARA_PTRNULL： 引数のポインタが NULL
- UCAERR_OPT_INVALID： 誤ったオプションを指定した

■ 処理内容説明

ASTM 補正（旧 JIS）を行います。

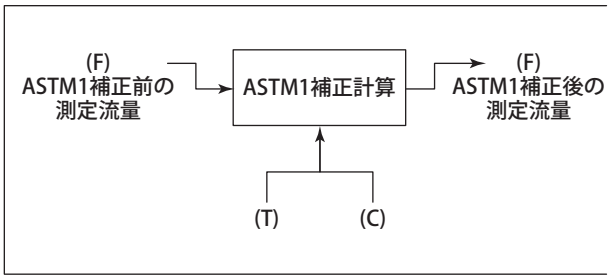
■ 引数説明

● option

NOOPTION を指定してください。

■ 詳細説明

UcaCalcAstm1 は、ASTM（旧 JIS）補正に基づき、引数 F の測定流量に対して補正流量計算を行い、結果を引数 F に格納します。



080202J.ai

図　UcaCalcAstm1のデータの流れ

ASTM1 補正演算式を以下に示します。

$$ASTM1（F、T、C）＝ \{1 + \alpha \times （T - 15） + \beta \times （T - 15）^2\} \times F$$

$$\alpha = \frac{- P_1（T）}{C} \times P_2（T）　\beta = \frac{- P_3（T）}{C} \times P_4（T）$$

F　：測定流量
T　：測定温度［℃］
C　：15/4 ℃比重
P1（T）～P4（T）　：温度で決まるパラメータ

080203J.ai

8.2.2 UcaCalcAstm2

UcaCalcAstm2は、ASTM補正（新JIS：原油）を行います。

■ 関数名

I32 UcaCalcAstm2

■ 引数

● blockContext

- データ型： UcaBlockContext
- タイプ： IN
- 説明： ブロックコンテキスト

● T

- データ型： F64
- タイプ： IN
- 説明： 測定温度

● F

- データ型： F64 *
- タイプ： IN/OUT
- 説明： 測定流量

● R

- データ型： F64
- タイプ： IN
- 説明： 密度

● option

- データ型： U32
- タイプ： IN
- 説明： オプション

■ 戻り値

- SUCCEED： 正常終了
- UCAERR_TC_OVERFLOW： オーバフロー
- UCAERR_TC_ROOTMINUS： ルートの中が負
- UCAERR_TC_ZERODIV： ゼロ割り発生
- UCAERR_PARA_PTRNULL： 引数のポインタが NULL
- UCAERR_OPT_INVALID： 誤ったオプションを指定した

■ 処理内容説明

ASTM 補正（新 JIS：原油）を行います。

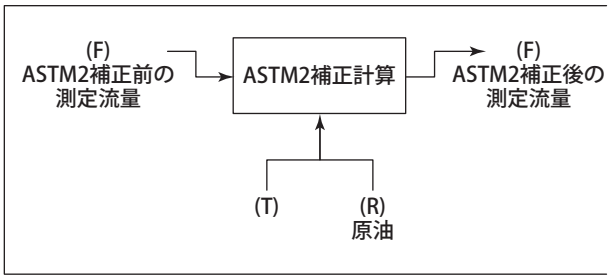
■ 引数説明

● option

NOOPTION を指定してください。

■ 詳細説明

UcaCalcAstm2 は、ASTM 補正（新 JIS）に基づき、引数 F の測定流量に対して補正流量計算を行い、結果を引数 F に格納します。原油を扱うときに使用します。



080204J.ai

図　UcaCalcAstm2のデータの流れ

ASTM2 補正演算式を以下に示します。

$$ASTM2(F、T、R) = \exp \times \{-\alpha \times (T-15) + 0.8\alpha^2 \times (T-15)^2\} \times F$$

$$\alpha = \frac{K_0}{R^2} \times \frac{K_1}{R}$$

F　　：測定流量
T　　：測定温度［℃］
C　　：15 ℃密度［kg/m^3］
K_0、K_1 ：油種ごとの定数（下表参照）

080205J.ai

表　油種ごとの定数（原油）

油種	15［℃］での密度Rの範囲［kg/m^3］	定数	
		K_0	K_1
原油	610.5 ≦ R ≦ 1075.0	613.9723	0.0

8.2.3 UcaCalcAstm3

UcaCalcAstm3は、ASTM補正（新JIS：燃料油）を行います。

■ 関数名

I32 UcaCalcAstm3

■ 引数

● blockContext

- データ型： UcaBlockContext
- タイプ： IN
- 説明： ブロックコンテキスト

● T

- データ型： F64
- タイプ： IN
- 説明： 測定温度

● F

- データ型： F64 *
- タイプ： IN/OUT
- 説明： 測定流量

● R

- データ型： F64
- タイプ： IN
- 説明： 密度

● option

- データ型： U32
- タイプ： IN
- 説明： オプション

■ 戻り値

- SUCCEED： 正常終了
- UCAERR_TC_OVERFLOW： オーバフロー
- UCAERR_TC_ROOTMINUS： ルートの中が負
- UCAERR_TC_ZERODIV： ゼロ割り発生
- UCAERR_PARA_PTRNULL： 引数のポインタが NULL
- UCAERR_OPT_INVALID： 誤ったオプションを指定した

■ 処理内容説明

ASTM 補正（新 JIS：燃料油）を行います。

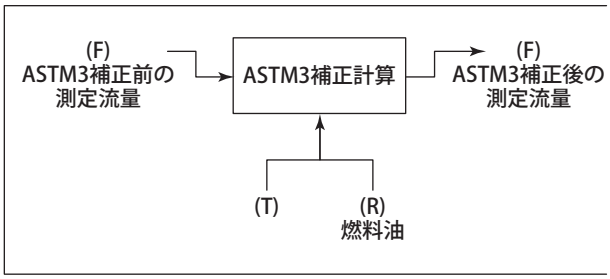
■ 引数説明

● option

NOOPTION を指定してください。

■ 詳細説明

UcaCalcAstm3 は、ASTM 補正（新 JIS）に基づき、引数 F の測定流量に対して補正流量計算を行い、結果を引数 F に格納します。燃料油を扱うときに使用します。



080207J.ai

図　UcaCalcAstm3のデータの流れ

ASTM3 補正演算式を以下に示します。

$$ASTM3（F、T、R）= \exp \times \{-\alpha \times （T - 15）+ 0.8\alpha^2 \times （T - 15）^2\} \times F$$

$$\alpha = \frac{K_0}{R^2} \times \frac{K_1}{R} \quad または \quad \alpha = A \times \frac{B}{R^2}$$

F　　：測定流量
T　　：測定温度［℃］
R　　：15 ℃密度［kg/m^3］
K0、K1、A、B ：油種ごとの定数（下表参照）

080208J.ai

表　油種ごとの定数（燃料油）

油種	15［℃］での密度Rの範囲［kg/m^3］	定数			
		K_0	K_1	A	B
燃料油	653.0 ≦ R ＜ 770.25	346.4228	0.4388	—	—
	770.25 ≦ R ＜ 787.75	—	—	− 0.00336312	2680.3206
	787.75 ≦ R ＜ 838.75	594.5418	0.0	—	—
	838.75 ≦ R ≦ 1075.0	186.9696	0.4862	—	—

8.2.4 UcaCalcAstm4

UcaCalcAstm4は、ASTM補正（新JIS：潤滑油）を行います。

■ 関数名

I32 UcaCalcAstm4

■ 引数

● blockContext

- データ型： UcaBlockContext
- タイプ： IN
- 説明： ブロックコンテキスト

● T

- データ型： F64
- タイプ： IN
- 説明： 測定温度

● F

- データ型： F64 *
- タイプ： IN/OUT
- 説明： 測定流量

● R

- データ型： F64
- タイプ： IN
- 説明： 密度

● option

- データ型： U32
- タイプ： IN
- 説明： オプション

■ 戻り値

- SUCCEED： 正常終了
- UCAERR_TC_OVERFLOW： オーバフロー
- UCAERR_TC_ROOTMINUS： ルートの中が負
- UCAERR_TC_ZERODIV： ゼロ割り発生
- UCAERR_PARA_PTRNULL： 引数のポインタが NULL
- UCAERR_OPT_INVALID： 誤ったオプションを指定した

■ 処理内容説明

ASTM 補正（新 JIS：潤滑油）を行います。

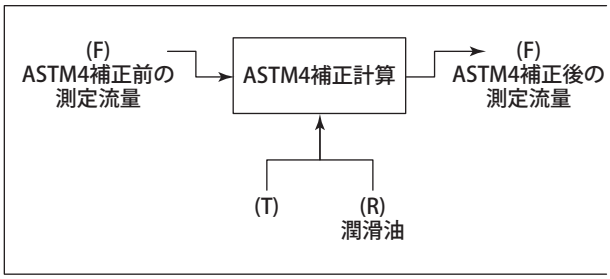
■ 引数説明

● option

NOOPTION を指定してください。

■ 詳細説明

UcaCalcAstm4 は、ASTM 補正（新 JIS）に基づき、引数 F の測定流量に対して補正流量計算を行い、結果を引数 F に格納します。潤滑油を扱うときに使用します。



080210J.ai

図　UcaCalcAstm4のデータの流れ

ASTM4 補正演算式を以下に示します。

$$ASTM4(F、T、R) = \exp \times \{-\alpha \times (T-15) + 0.8\alpha^2 \times (T-15)^2\} \times F$$

$$\alpha = \frac{K_0}{R^2} \times \frac{K_1}{R}$$

F　　：測定流量
T　　：測定温度［℃］
R　　：15 ℃密度［kg/m^3］
K_0、K_1：油種ごとの定数（下表参照）

080211J.ai

表　油種ごとの定数（潤滑油）

油種	15［℃］での密度Rの範囲［kg/m^3］	定数	
		K_0	K_1
潤滑油	800.0 ≦ R ≦ 1164.0	0.0	0.6278

Blank Page

9. その他

データ型変換、ローカルカウンタ、メッセージまたはFPU例外などの関数があります。

9.1 データ型変換

データ型変換を行う関数として、UcaDataConvertTypeが用意されています。

9.1.1 UcaDataConvertType

UcaDataConvertTypeは、データ型を変換します。

■ 関数名

I32 UcaDataConvertType

■ 引数

● blockContext

- データ型： UcaBlockContext
- タイプ： IN
- 説明： ブロックコンテキスト

● src

- データ型： void *
- タイプ： IN
- 説明： 変換前データ

● srcType

- データ型： I32
- タイプ： IN
- 説明： 変換前データ型

● dest

- データ型： void *
- タイプ： OUT
- 説明： 変換後データ

● destType

- データ型： I32
- タイプ： IN
- 説明： 変換後データ型

● option

- データ型： I32
- タイプ： IN
- 説明： オプション

■ 戻り値

- SUCCEED： 正常終了
- UCAERR_PARA_PTRNULL： 引数のポインタが NULL
- UCAERR_OPT_INVALID： 誤ったオプションを指定した

■ 処理内容説明

データ型変換を行います。

■ 引数説明

● option

NOOPTION を指定してください。

■ 詳細説明

UcaDataConvertType は、引数 src のデータを、引数 srcType で指定したデータ型から引数 destType で指定したデータ型に変換して、引数 dest に格納します。

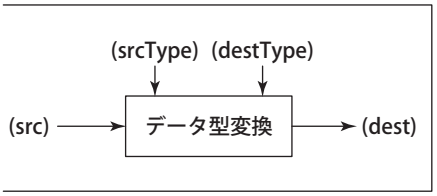


図　UcaDataConvertTypeのデータの流れ

UcaUnivType 型データから他のデータ型へ変換する場合、引数 src に UcaUnivType メンバ変数 dataValue を、引数 srcType に UcaUnivType のメンバ変数 dType を指定してください。以下に、UcaUnivType 型の変数 udata を I32S 型の変数 idata に変換する例を示します。UCA_DATATYPE_I32S は I32S 型を示すラベルです。

参照 データ型を示すラベルの一覧については、以下を参照してください。
「Appendix C.1 データ型」

```
I32 errCode;
UcaUnivType udata;
I32S idata;
...
/* uData から I32S としてデータを取り出す。*/
errCode = UcaDataConvertType(blockContext, &udata.dataValue, udata.dType,
      &idata, UCA_DATATYPE_I32S, UCAOPT_NOOPTION);
```

UcaDataConvertType は、変換前と変換後のデータ型を指定し、データ型の変換を行います。指定するデータ型により、次表のようにデータ型変換できない場合や、情報が失われる場合があります。

表　データ型と変換可否

変換後＼変換前	I16	U16	I32	U32	F32	F64	CHR	I16S	U16S	I32S	U32S	F32S	F64S	F32SR
I16	○	△ *f	△ *f	△ *f	△ *ef	△ *ef	×	△ *a	△ *af	△ *af	△ *af	△ *aef	△ *aef	△ *acef
U16	△ *f	○	△ *f	△ *f	△ *ef	△ *ef	×	△ *af	△ *a	△ *af	△ *af	△ *aef	△ *aef	△ *acef
I32	○	○	○	△ *f	△ *ef	△ *ef	×	△ *a	△ *a	△ *a	△ *af	△ *aef	△ *aef	△ *acef
U32	△ *f	○	△ *f	○	△ *ef	△ *ef	×	△ *af	△ *a	△ *af	△ *a	△ *aef	△ *aef	△ *acef
F32	○	○	○	○	○	△ *f	×	△ *a	△ *a	△ *a	△ *a	△ *a	△ *af	△ *ac
F64	○	○	○	○	○	○	×	△ *a	△ *a	△ *a	△ *a	△ *a	△ *a	△ *ac
CHR	×	×	×	×	×	×	○	×	×	×	×	×	×	×
I16S	△ *b	△ *bf	△ *bf	△ *bf	△ *bef	△ *bef	×	○	△ *f	△ *f	△ *f	△ *ef	△ *ef	△ *cef
U16S	△ *bf	△ *b	△ *bf	△ *bf	△ *bef	△ *bef	×	△ *f	○	△ *f	△ *f	△ *ef	△ *ef	△ *cef
I32S	△ *b	△ *b	△ *b	△ *bf	△ *bef	△ *bef	×	○	○	○	△ *f	△ *ef	△ *ef	△ *cef
U32S	△ *bf	△ *b	△ *bf	△ *b	△ *bef	△ *bef	×	△ *f	○	△ *f	○	△ *ef	△ *ef	△ *cef
F32S	△ *b	△ *b	△ *b	△ *b	△ *b	△ *bf	×	○	○	○	○	○	△ *f	△ *c
F64S	△ *b	△ *b	△ *b	△ *b	△ *b	△ *b	×	○	○	○	○	○	○	△ *c
F32SR	△ *bd	△ *bd	△ *bd	△ *bd	△ *bd	△ *bdf	×	△ *d	△ *d	△ *d	△ *d	△ *d	△ *df	○

○： 変換できます。
△： 変換できますが、下表の制限があります。
×： 変換できません。エラーコードを返します。

表　データ型変換の制限事項

	データ型変換の制限内容
*a	データステータスの情報を失います
*b	データステータスに 0 を格納します
*c	レンジの情報を失います
*d	レンジに 0 を格納します
*e	小数点以下を四捨五入します
*f	範囲外の場合、変換しません。エラーコードを返します

9.2 時間管理

ユーザカスタムブロックが経過時間を管理するための機能としてローカルタイマがあります。個々のユーザカスタムブロックは、ローカルタイマ機能が使用する内部変数ローカルカウンタを持っています。ユーザカスタムブロック実行管理部（システム）は、毎基本周期に内部変数ローカルカウンタの値を基本周期の秒数ずつ進めます。基本周期が4秒であれば、各基本周期に内部変数ローカルカウンタは4秒ずつ進んでいきます。

■ 時間管理処理の概要

ローカルタイマ機能は、この内部変数ローカルカウンタに操作をする以下の関数で構成されています。

表　ローカルタイマ機能の関数

関数名	機能
UcaTimeResetLocalCounter	ユーザカスタムブロックの内部変数ローカルカウンタを 0 にリセットします
UcaTimeGetLocalCounter	ユーザカスタムブロックの内部変数ローカルカウンタの現在値を取得します

UcaTimeResetLocalCounter で内部変数ローカルカウンタを 0 に初期化し、UcaTimeGetLocalCounter で内部変数ローカルカウンタの現在値を取得することにより、UcaTimeResetLocalCounter を実行してからの経過時間を得ることができます。ローカルタイマ機能の注意事項です。

- ローカルタイマの分解能は、基本周期です。つまり、UcaTimeGetLocalCounter で取得するカウンタ値（経過時間）は、基本周期の整数倍です。
- 内部変数ローカルカウンタが更新されるタイミングについて説明します。基本周期ごとに、APCS 制御機能本体は、制御ドローイングビルダで定義したブロック実行順に従い、ユーザカスタムブロックに処理タイミングを与えます。処理タイミングを与えられると、ユーザカスタムブロック実行管理部は、最初に内部変数ローカルカウンタを進めます。そして、今回の処理タイミングが、ユーザカスタムブロックの実効スキャン周期の場合には、ユーザ定義の機能ブロック定周期処理関数 UcaBlockPeriodical を呼び出します。
- ユーザカスタムブロック実行管理部は、ユーザカスタムブロックのブロックモードが O/S でなければ、内部変数ローカルカウンタを進めます。つまり、ブロックモードが O/S 以外の場合には、ローカルタイマで管理される経過時間が進んでいきます。ブロックモードが O/S の場合は、内部変数ローカルカウンタは変化しません。
- ローカルタイマの最大値は、4,294,967,295（0xffffffff）［sec］です。
  最大値に達すると 0 に戻り、カウントを続けます。

9.2.1 UcaTimeResetLocalCounter

UcaTimeResetLocalCounterは、内部変数ローカルカウンタの値をリセットします。

■ 関数名

I32 UcaTimeResetLocalCounter

■ 引数

● blockContext

- データ型： UcaBlockContext
- タイプ： IN
- 説明： ブロックコンテキスト

■ 戻り値

- SUCCEED： 成功
- UCAERR_PARA_PTRNULL： 引数のポインタが NULL

■ 処理内容説明

ローカルカウンタをリセットします。

■ 詳細説明

UcaTimeResetLocalCounter は、内部変数ローカルカウンタの値を 0 にリセットします。

9.2.2 UcaTimeGetLocalCounter

UcaTimeGetLocalCounterは、内部変数ローカルカウンタの値を取得します。

■ 関数名

I32 UcaTimeGetLocalCounter

■ 引数

● blockContext

- データ型： UcaBlockContext
- タイプ： IN
- 説明： ブロックコンテキスト

● time

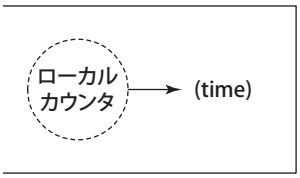
- データ型： U32 *
- タイプ： OUT
- 説明： カウンタ値

■ 戻り値

- SUCCEED： 成功
- UCAERR_PARA_PTRNULL： 引数のポインタが NULL

■ 詳細説明

UcaTimeGetLocalCounter は、内部変数ローカルカウンタの値を取得し、引数 time に格納します。



090202J.ai

図 UcaTimeGetLocalCounterのデータの流れ

システムは毎基本スキャン周期に内部変数ローカルカウンタの値を進めます。内部変数ローカルカウンタの単位は秒［sec］で、分解能は APCS の基本スキャン周期です。UcaTimeGetLocalCounter が引数 time に格納する内部変数ローカルカウンタの値は、必ず基本スキャン周期の整数倍になります。

9.3 メッセージ

メッセージ関数は、以下のメッセージを出力します。

表　メッセージ関数

メッセージ関数	出力するメッセージ	メッセージ番号	システムアラームウィンドウに表示	プリンタに印字	ヒストリカルメッセージ保存ファイルに保存
UcaMesgSendSystemAlarm	システムアラームメッセージ	0725 0726	○	○	○
UcaMesgSendPrintMessage	印字メッセージ	1301	×	○	○
UcaMesgSendHistoricalMessage	ヒストリカルメッセージ	1303	×	×	○

○： する
×： しない

参照 メッセージの詳細については、以下を参照してください。
操作監視リファレンス Vol.2（IM 33J05A11-01JA）「6. メッセージ処理」

9.3.1 UcaMesgSendSystemAlarm

UcaMesgSendSystemAlarmは、システムアラームメッセージを出力します。

■ 関数名

I32 UcaMesgSendSystemAlarm

■ 引数

● blockContext

- データ型： UcaBlockContext
- タイプ： IN
- 説明： ブロックコンテキスト

● string

- データ型： BYTE *
- タイプ： IN
- 説明： メッセージ文字列

● type

- データ型： I32
- タイプ： IN
- 説明： メッセージのタイプ

■ 戻り値

- SUCCEED： 成功
- UCAERR_MESG_INVALIDTYPE： メッセージのタイプが違う
- UCAERR_TEXT_CONVFAIL： 文字コード変換失敗
- UCAERR_PARA_PTRNULL： 引数のポインタが NULL

■ 処理内容説明

システムアラームメッセージを出力します。

■ 引数説明

● string

メッセージ文字列です。

● type

以下の中から 1 つだけ指定することができます。

- UCAOPT_MESG_SALMERR： 任意文字列システムアラーム発生
- UCAOPT_MESG_SALMNR： 任意文字列システムアラーム復帰

■ 詳細説明

UcaMesgSendSystemAlarm は、システムアラームメッセージを出力します。引数 string に指定した文字列はシステムアラームウィンドウに表示されます。プリンタに印字され、ヒストリカルメッセージ保存ファイルに格納されます。

表　UcaMesgSendSystemAlarmが発生するメッセージ

引数type	UCAOPT_MESG_SALMERR	UCAOPT_MESG_SALMNR
メッセージ番号	0725	0726
メッセージ内容	任意文字列システムアラーム発生	任意文字列システムアラーム復帰
意味	ユーザカスタムブロックなどから任意文字列システムアラームの発生メッセージを出力しました	ユーザカスタムブロックなどから任意文字列システムアラームの復帰メッセージを出力しました
備考	SEBOL の sysalarm 文も同じメッセージを出力します	SEBOL の sysalarm recover 文も同じメッセージを出力します

引数 type に UCAOPT_MESG_SALMERR を指定すると任意文字列システムアラーム発生メッセージを出力します。UCAOPT_MESG_SALMNR を指定すると、任意文字列システムアラーム復帰メッセージを出力します。
引数 string には、最大 128 バイトのメッセージ文字列を指定してください。128 バイト以上の文字列を指定した場合、UcaMesgSendSystemAlarm は 128 バイト目までしかメッセージを出力せず、129 バイト以降の文字列は無視します。
引数 string には、半角カナを使用しないでください。半角カナを使用した場合、UcaMesgSendSystemAlarm はメッセージを出力せず、エラーコード UCAERR_TEXT_CONVFAIL を返します。

9.3.2 UcaMesgSendPrintMessage

UcaMesgSendPrintMessageは、印字メッセージを出力します。

■ 関数名

I32 UcaMesgSendPrintMessage

■ 引数

● blockContext

- データ型： UcaBlockContext
- タイプ： IN
- 説明： ブロックコンテキスト

● string

- データ型： BYTE *
- タイプ： IN
- 説明： メッセージ文字列

■ 戻り値

- SUCCEED： 成功
- UCAERR_TEXT_CONVFAIL： 文字コード変換失敗
- UCAERR_PARA_PTRNULL： 引数のポインタが NULL

■ 処理内容説明

印字メッセージを出力します。

■ 引数説明

● string

メッセージ文字列です。

■ 詳細説明

UcaMesgSendPrintMessage は、印字メッセージを出力します。引数 string に指定した文字列はプリンタに印字され、ヒストリカルメッセージ保存ファイルに格納されます。システムアラームウィンドウには表示されません。

表　UcaMesgSendPrintMessageが発生するメッセージ

メッセージ番号	1303
メッセージ内容	印字メッセージ
意味	印字メッセージがユーザカスタムブロックやシーケンステーブルなどより発生しました

引数 string には、最大 128 バイトのメッセージ文字列を指定してください。128 バイト以上の文字列を指定した場合、UcaMesgSendPrintMessage は 128 バイト目までしかメッセージを出力せず、129 バイト以降の文字列を無視します。
引数 string には、半角カナを使用しないでください。半角カナを使用した場合、UcaMesgSendPrintMessage はメッセージを出力せず、エラーコード UCAERR_TEXT_CONVFAIL を返します。

9.3.3 UcaMesgSendHistoricalMessage

UcaMesgSendHistoricalMessageは、ヒストリカルメッセージを出力します。

■ 関数名

I32 UcaMesgSendHistoricalMessage

■ 引数

● blockContext

- データ型： UcaBlockContext
- タイプ： IN
- 説明： ブロックコンテキスト

● string

- データ型： BYTE *
- タイプ： IN
- 説明： メッセージ文字列

■ 戻り値

- SUCCEED： 成功
- UCAERR_TEXT_CONVFAIL： 文字コード変換失敗
- UCAERR_PARA_PTRNULL： 引数のポインタが NULL

■ 処理内容説明

ヒストリカルメッセージを出力します。

■ 引数説明

● string

メッセージ文字列です。

■ 詳細説明

UcaMesgSendHistoricalMessage は、ヒストリカルメッセージを出力します。引数 string に指定した文字列は、ヒストリカルメッセージ保存ファイルに格納されます。システムアラームウィンドウに表示されません。プリンタに印字されません。

表　UcaMesgSendHistoricalMessageが発生するメッセージ

メッセージ番号	1305
メッセージ内容	ヒストリカルメッセージ
意味	任意文字列メッセージがヒストリカルファイルに格納されました

引数 string には、最大 128 バイトのメッセージ文字列を指定してください。128 バイト以上の文字列を指定した場合、UcaMesgSendHistoricalMessage は 128 バイト目までしかメッセージを出力せず、129 バイト以降の文字列は無視します。
引数 string には、半角カナを使用しないでください。半角カナを使用した場合、UcaMesgSendHistoricalMessage はメッセージを出力させず、エラーコード UCAERR_TEXT_CONVFAIL を返します。

9.4 FPU例外

浮動小数点の演算で、オーバフローやゼロ割りを起こすと、例外が発生します。システムは、例外を検出するとシステムが保持している例外発生フラグに「例外発生」を設定します。この例外発生フラグをクリアする関数がUcaFpuExpClearです。UcaFpuExpCheckは、例外発生フラグの現在の状態を取得する関数です。UcaFpuExpClearで例外発生フラグをクリアしたあとに浮動小数点演算を行い、演算終了あとにUcaFpuExpCheckで取得した例外発生フラグを検査することにより、演算の途中で、オーバフローなどの例外が検出されたかを知ることができます。

■ FPU例外処理の概要

例外発生フラグは､32 ビットの整数です｡例外の種類ごとに 1 ビットが割り付いています。検出される例外と、例外発生フラグの値を以下に示します。

表　UcaFpcExpCheckが返す例外発生フラグ

意味	ラベル	値（10進）	値（16進）
正常（例外発生なし）	なし	0	0x00000000
オーバフロー	UCAERR_FPU_OVERFLW	4	0x00000100
ゼロ割り算	UCAERR_FPU_ZERODIV	8	0x00001000
無効演算	UCAERR_FPU_INVALID	16	0x00010000

補足 ラベルは、システム定義インクルードファイル libucadef.h で定義されています。

浮動小数点演算で例外を検出する原因と、検出された場合の演算結果を以下に示します。

表　浮動小数点演算で検出される例外の種類と演算結果

例外	説明	演算結果
オーバフロー	演算結果の絶対値が、浮動小数点の形式で表現可能な有効最大数より大きい場合にオーバフローとなります	演算結果は、正または負の無限大となります
ゼロ割り算	割り算の分母が 0（ゼロ）である場合に、ゼロ割り算となります	演算結果は、正または負の無限大となります
無効演算	「∞（無限大）÷∞」や「0 ÷ 0」のような無効な演算をすると、無効演算となります	演算結果は、非数（NaN）となります (*1)

*1： 非数（NaN）をオペランドとする演算をすると演算結果は非数になります。

ユーザカスタムアルゴリズム作成用ライブラリの関数は、例外発生によって生成されてしまった無限大や非数を、正常な浮動小数点データとは区別し、エラー処理をするようにできています。ユーザカスタムアルゴリズム作成用ライブラリの関数は、引数に指定された浮動小数点データが無限大または非数の場合、次の動作をします。

- 自ブロックのデータアイテムに設定する値が無限大または非数の場合には、データステータスを BAD にし、数値は変更しません。たとえば、UcaRWSetPv は、PV への設定値が無限大または非数であれば、PV のデータステータスを BAD とし、PV の数値は変更しません。
- 外部へのデータ出力の場合、出力値が無限大または非数であれば、指定されたデータを出力をしないでエラーで戻ります。

補足 APCS の制御機能は、「アンダーフロー」と「不正確結果」は例外として扱いません。また、UcaFpuExpCheck は、「アンダーフロー」と「不正確結果」を検出しません。

- アンダーフローは、演算結果の絶対値が、浮動小数点の形式で表現可能な最小有効数より小さい場合に発生します。APCS の制御機能は、アンダーフローを検出しても例外とはしないで、演算結果を（正の）ゼロにします。
- 不正確結果は、無限精度の演算結果が、浮動小数点数の丸め処理から得られた値と異なることを意味します。たとえばアンダーフローが起きた場合には、（非常に小さい数値をゼロにしてしまうので）不正確結果も発生します。APCS の制御機能は、不正確結果を無視し、「不正確な」演算結果をそのまま使用します。「不正確」といっても、非常に小さい誤差が出るだけですので実用上はまったく問題ありません。

補足

- 浮動小数点演算の例外は、ユーザカスタムブロック強制終了の対象ではありません。つまり、浮動数点演算でゼロ割り算やオーバフローが発生してもシステムはユーザカスタムブロックのブロックモードを O/S にしません。ユーザカスタムブロックはそのまま実行を継続します。
- 整数演算のゼロ割り算やオーバフローを UcaFpuExpClear と UcaFpuExpCheck で検出することはできません。また、整数のゼロ割り算は、ユーザカスタムブロック強制終了（レベル 4）であり、システムはユーザカスタムブロックのブロックモードを O/S にして強制終了します。

参照 詳細については、以下を参照してください。
APCS ユーザカスタムブロック（IM 33J15U20-01JA）「2.4.3 異常発生時の処理」

9.4.1 UcaFpuExpClear

UcaFpuExpClearは、例外発生フラグを0クリアします。

■ 関数名

I32 UcaFpuExpClear

■ 引数

● blockContext

- データ型： UcaBlockContext
- タイプ： IN
- 説明： ブロックコンテキスト

■ 戻り値

- SUCCEED： 成功
- UCAERR_PARA_PTRNULL： 引数のポインタが NULL

■ 処理内容説明

FPU 例外発生フラグをクリアします。

9.4.2 UcaFpuExpCheck

UcaFpuExpCheckは、例外発生フラグを取得します。

■ 関数名

I32 UcaFpuExpCheck

■ 引数

● blockContext

- データ型： UcaBlockContext
- タイプ： IN
- 説明： ブロックコンテキスト

● expFlag

- データ型： I32 *
- タイプ： OUT
- 説明： 例外フラグポインタ

■ 戻り値

- SUCCEED： 成功
- UCAERR_PARA_PTRNULL： 引数のポインタが NULL

■ 処理内容説明

FPU 例外発生フラグを検査します。

■ 引数説明

● expFlag

- 0： 例外発生なし
- UCAERR_FPU_OVERFLW： オーバフロー
- UCAERR_FPU_ZERODIV：0 割り算
- UCAERR_FPU_INVALID： 無効演算

■ 詳細説明

UcaFpuExpCheck は、浮動小数点演算により発生した例外の有無を示す例外発生フラグを取得し、引数 expFlag に格納します。UcaFpuExpCheck を使用する場合には、事前に UcaFpuExpClear で例外発生フラグをクリアしてください。例外発生の検出はつぎの順に行います。

1. UcaFpuExpClear で例外発生フラグをクリアします。
2. 浮動小数点演算を行います。
3. UcaFpuExpCheck で例外発生フラグを取得し、例外発生の有無を検査します。

UcaFpuExpCheck で取得する例外発生フラグは、例外ごとに 1 ビットが割りついた 32 ビットの整数です（下表）。

表　UcaFpcExpCheckが返す例外発生フラグ

意味	ラベル	値（10進）	値（16進）
正常（例外発生なし）	なし	0	0x00000000
オーバフロー	UCAERR_FPU_OVERFLW	4	0x00000100
ゼロ割り算	UCAERR_FPU_ZERODIV	8	0x00001000
無効演算	UCAERR_FPU_INVALID	16	0x00010000

補足 ラベルは、システム定義インクルードファイル libucadef.h で定義されています。

参照 浮動小数点演算の例外を検査するプログラム例については、以下を参照してください。
APCS ユーザカスタムブロック プログラミングガイド（IM 33J15U21-01JA）「5.2.1 入力処理と UcaFpuExpCheck による演算エラーの検出」

9.5 時間

現在時間を取得する関数と、現在時間を年月日時分秒に分解する関数が用意されています。

■ 時間を扱う関数

表　時間関数

関数名	内容
UcaTime	万国標準時の 1970 年 1 月 1 日 0 時 0 分 0 秒からの経過時間を取得します。単位は秒です
UcaLocaltime	経過時間を、現地時刻の年月日時分秒に分解します
UcaGmtime	経過時間を、万国標準時刻の年月日時分秒に分解します
UcaMktime	現地時刻の年月日時分秒を、経過時間に変換します

時間関数で使用するデータ型を以下に示します。これらのデータ型は、Windows のインクルードファイル time.h で定義されています。

表　時間に関するデータ型

データ型	内容
time_t	万国標準時の 1970 年 1 月 1 日 0 時 0 分 0 秒からの経過時間（秒）を保持するデータ型です。32 ビット符号ありの整数です
struct tm	年月日時分秒を保持する時刻構造体です

時刻構造体 struct tm を示します。

```
struct tm{
   int tm_sec;          /* 秒－［0～59］*/
   int tm_min;          /* 分－［0～59］*/
   int tm_hour;         /* 時－［0～23］*/
   int tm_mday;         /* 日－［1～31］*/
   int tm_mon;          /* 月－［0～11］*/
   int tm_year;         /* 年　1900 年からの経過年数 */
   int tm_wday;         /* 日曜日からの経過日数－［0～6］*/
   int tm_yday;         /* 1 月 1 日からの経過日数－［0～365］*/
   int tm_isdst;        /* 夏時間が無効 0、夏時間が有効 正の値、どちらかわからない 負の値 */
};
```

9.5.1 UcaTime

UcaTimeは、万国標準時（UCT）の1970年1月1日0時0分0秒からの経過時間を取得します。

■ 関数名

time_t UcaTime

■ 引数

● timerPtr

- データ型： time_t *
- タイプ： OUT
- 説明： 万国標準時（UCT）からの経過時間［秒］

■ 戻り値

経過時間を秒単位で表した数値

■ 詳細説明

UcaTime は、システムクロックに従って万国標準時（UCT）の 1970 年 1 月 1 日 0 時 0 分 0 秒からの経過時間を取得し、引数 timePtr に格納するとともに、同じ値を戻り値として返します。経過時間の単位は（秒）です。
引数 timePtr に NULL を指定した場合は、UcaTime は経過時間の戻り値のみを返します。
UcaTime を呼び出す場合、Windows のインクルードファイル time.h をインクルードしてください。

```
...
#include <time.h>
...
#include <libuca.h>
...

time_t current_time;

current_time = UcaTime(NULL);
```

9.5.2 UcaLocaltime

UcaLocaltimeは、経過時間を現地時刻に変換します。

■ 関数名

struct tm *UcaLocaltime

■ 引数

● timerPtr

- データ型： time_t *
- タイプ： IN
- 説明： 万国標準時（UCT）からの経過時間［秒］

■ 戻り値

時刻構造体へのポインタ

■ 処理内容説明

UcaLocaltime は、time_t 型の値として格納されている万国標準時（UCT）の 1970 年 1 月 1 日 0 時 0 分 0 秒からの経過時間（単位は秒）を、現地時間の年月日時分秒に変換し、時刻構造体 tm に格納します。UcaLocaltime は結果として得られた時刻構造体へのポインタを返します。

参照 tm 構造体については、以下の最初の説明を参照してください。
「9.5 時間」

UcaLocalTime と UcaTime と組み合わせて使用することにより、現地時間の年月日時分秒を取得することができます。

UcaLocaltimeを呼び出す場合には、Windowsのインクルードファイルtime.hをインクルードしてください。UcaTime と UcaLocalTime を組み合わせて使う例を示します。現在時刻をヒストリカルメッセージとして出力します。

```
...
#include <time.h>                          /* time_tと時刻構造体tm */
#include <stdio.h>                 /* sprintf( ) */
...
#include <libuca.h>
...

time_t current_time;
struct tm *time_ptr;
BYTE msgstr[128];

current_time = UcaTime(NULL);
time_ptr = UcaLocaltime(&current_time);

sprintf (msgstr, "日本時刻で%d年%d月%d日%d時%d分%d秒です",
   time_ptr->tm_year+1900,
   time_ptr->tm_mon+1,
   time_ptr->tm_mday,
   time_ptr->tm_hour,
   time_ptr->tm_min,
   time_ptr->tm_sec);
UcaMesgSendHistoricalMessage(bc, msgstr);
...
```

9.5.3 UcaGmtime

UcaGmtimeは、経過時間を万国標準時刻（UCT）に変換します。

■ 関数名

struct tm *UcaGmtime

■ 引数

● timerPtr

- データ型： time_t *
- タイプ： IN
- 説明： 万国標準時（UCT）からの経過時間［秒］

■ 戻り値

時刻構造体へのポインタ

■ 処理内容説明

UcaGmtime は、time_t 型の値として格納されている万国標準時（UCT）の 1970 年 1 月 1 日 0 時 0 分 0 秒からの経過時間（単位は秒）を、万国標準時の年月日時分秒に変換し、時刻構造体 tm に格納します。UcaGmtime は結果として得られた時刻構造体へのポインタを返します。

参照 tm 構造体については、以下の最初の説明を参照してください。
「9.5 時間」

UcaGmtime と UcaTime と組み合わせて使用することにより、万国標準時の年月日時分秒を取得することができます。

UcaGmtime を呼び出す場合には、Windows のインクルードファイル time.h をインクルードしてください。UcaTime と UcaGmtime を組み合わせて使う例を示します。万国標準時刻をヒストリカルメッセージとして出力します。

```
...
#include <time.h>                          /* time_t と時刻構造体 tm */
#include <stdio.h>                 /* sprintf( ) */
...
#include <libuca.h>
...

time_t current_time;
struct tm *time_ptr;
BYTE msgstr[128];

current_time = UcaTime(NULL);
time_ptr = UcaGmtime(&current_time);

sprintf(msgstr, "万国標準時刻で%d年%d月%d日%d時%d分%d秒です",
   time_ptr->tm_year+1900,
   time_ptr->tm_mon+1,
   time_ptr->tm_mday,
   time_ptr->tm_hour,
   time_ptr->tm_min,
   time_ptr->tm_sec);
UcaMesgSendHistoricalMessage(bc, msgstr);
...
```

9.5.4 UcaMktime

UcaMktimeは、年月日時分秒を通算時間に変換します。

■ 関数名

time_t UcaMktime

■ 引数

● timeStructPtr

- データ型： struct tm *
- タイプ： IN
- 説明： カレンダー時刻を格納した時刻構造体へのポインタ

■ 戻り値

万国標準時（UCT）からの経過時間［秒］

■ 処理内容説明

UcaMktime は、引数 timeStructPtr が指す時刻構造体（struct tm）の年月日時分秒を変換して、time_t 型万国標準時（UCT）の 1970 年 1 月 1 日 0 時 0 分 0 秒からの経過時間（単位は秒）に変換し、戻り値として返します。

参照 tm 構造体については、以下の最初の説明を参照してください。
「9.5 時間」

変換後の時刻は、UcaTime 関数によって返された値と同じ形式を持ちます。timeptr 構造体の tm_wday と tm_yday の元の値は無視されます。また、その他のメンバの元の値はそれぞれの正規範囲内にある必要はありません。
UcaMktime 関数は、変換に成功すると tm_wday と tm_yday の値を正しく設定します。また、その他のメンバも、値の正規範囲内で指定されたカレンダー時刻（年月日時分秒）を表すように設定します。tm_mday の最終的な値は、tm_mon と tm_year が決定されるまで設定されません。tm 構造体時刻を指定するとき、夏時間でないことを示すには tm_isdst のフィールドに 0 を、夏時間であることを示すには 0 より大きい値を、また夏時間かどうかわからない場合は 0 より小さい値を設定します。

UcaMktime を呼び出す場合には、Windows のインクルードファイル time.h をインクルードしてください。

```
...
#include <time.h>                          /* time_t と時刻構造体 tm */
...
#include <libuca.h>
...

time_t current_time;
time_t new_time;
struct tm *time_ptr;

current_time = UcaTime(NULL);
time_ptr = UcaLocaltime(&current_time);

new_time = UcaMktime(time_ptr); /* new_time は、current_time と同じ値に戻ります */
...
```

10. 関数呼び出し可否一覧

このライブラリに含まれる各関数が、カスタムブロックのどの処理で呼び出し可能であるかをまとめて示します。

■ CSTM-C用

表 呼び出し可否（1/4）

関数名	機能ブロック初期化処理	機能ブロック終了処理	機能ブロック定周期処理	機能ブロックワンショット処理	機能ブロックデータ設定特殊処理
UcaModeSet	○	○	○	○	○
UcaModeGet	○	○	○	○	○
UcaModeGetPrev	○	○	○	○	○
UcaBstsSetExclusive	○	○	○	○	○
UcaBstsSet	○	○	○	○	○
UcaBstsClear	○	○	○	○	○
UcaBstsGet	○	○	○	○	○
UcaAlrmSet			○	○	
UcaAlrmClear			○	○	
UcaAlrmGet			○	○	
UcaDstsIsGood	○	○	○	○	○
UcaDstsIsBad	○	○	○	○	○
UcaDstsIsQst	○	○	○	○	○
UcaDstsIsPtpf	○	○	○	○	○
UcaDstsIsCnd	○	○	○	○	○
UcaDstsIsCal	○	○	○	○	○
UcaDstsSetBad	○	○	○	○	○
UcaDstsClearBad	○	○	○	○	○
UcaDstsSetQst	○	○	○	○	○
UcaDstsClearQst	○	○	○	○	○
UcaDstsSetCnd	○	○	○	○	○
UcaDstsClearCnd	○	○	○	○	○
UcaDstsChooseBadOrQst	○	○	○	○	○
UcaRWRead			○	○	
UcaRWWrite			○	○	
UcaRWReadback			○	○	
UcaRWReadString			○	○	
UcaRWWriteString			○	○	
UcaRWWritePvToJnSub			○	○	
UcaRWReadIn			○	○	
UcaRWSetPv			○	○	
UcaRWWriteMvToOutSub			○	○	
関数名	機能ブロック初期化処理	機能ブロック終了処理	機能ブロック定周期処理	機能ブロックワンショット処理	機能ブロックデータ設定特殊処理

表　呼び出し可否（2/4）

関数名	機能ブロック初期化処理	機能ブロック終了処理	機能ブロック定周期処理	機能ブロックワンショット処理	機能ブロックデータ設定特殊処理
UcaRWWriteMv			○	○	
UcaRWReadbackMv			○	○	
UcaRWWriteSub			○	○	
UcaRWReadTrack			○	○	
UcaRWReadBin			○	○	
UcaRWReadRl			○	○	
UcaRWReadInt	○	○	○	○	○
UcaRWSeqCond			○	○	
UcaRWSeqOprt			○	○	
UcaRWIsSeqCon			○	○	
UcaDataMeasureTracking			○	○	
UcaDataSvPushback			○	○	
UcaDataCheckDv			○	○	
UcaDataConvertRange			○	○	
UcaDataConvertRange_p			○	○	
UcaDataGetReadback			○	○	
UcaDataGetTrack			○	○	
UcaDataGetMvbb			○	○	
UcaDataGetMvbl			○	○	
UcaDataGetTagName	○	○	○	○	○
UcaDataStoreErrorNumber	○	○	○	○	○
UcaDataGet***	○	○	○	○	○
UcaDataStore***	○	○	○	○	○
UcaDataGet***n	○	○	○	○	○
UcaDataStore***n	○	○	○	○	○
UcaDataGetEachSn	○	○	○	○	○
UcaDataStoreEachSn	○	○	○	○	○
UcaDataStoreF64SToMvn	○	○	○	○	○
UcaConfigNonlinearGain	○	○	○	○	○
UcaConfigCompensation	○	○	○	○	○
UcaConfigDirection	○	○	○	○	○
UcaConfigOutput	○	○	○	○	○
UcaConfigDeadband	○	○	○	○	○
UcaConfigDeadbandHys	○	○	○	○	○
UcaConfigGeneral	○	○	○	○	○
UcaCtrlHandler			○	○	
UcaCtrlInitCheck	○	○	○	○	○
UcaCtrlInitClear	○	○	○	○	○
UcaCtrlInitSet	○	○	○	○	○
関数名	機能ブロック初期化処理	機能ブロック終了処理	機能ブロック定周期処理	機能ブロックワンショット処理	機能ブロックデータ設定特殊処理

表　呼び出し可否（3/4）

関数名	機能ブロック初期化処理	機能ブロック終了処理	機能ブロック定周期処理	機能ブロックワンショット処理	機能ブロックデータ設定特殊処理
UcaCtrlPidInit			○	○	
UcaCtrlFuncInit			○	○	
UcaCtrlPid			○	○	
UcaCtrlPidTiming			○	○	
UcaCtrlPidGetTime			○	○	
UcaCtrlPidPutTime			○	○	
UcaCtrlHoldCheck			○	○	
UcaCtrlHoldClear			○	○	
UcaCtrlHoldSet			○	○	
UcaCtrlGetGapGainCoef_p			○	○	
UcaCtrlGapGain_p			○	○	
UcaCtrlDvSquareGain_p			○	○	
UcaCtrlCompensation			○	○	
UcaCtrlCompensation_p			○	○	
UcaCtrlDeadband			○	○	
UcaCtrlDeadband_p			○	○	
UcaCtrlResetLimitOprt			○	○	
UcaCtrlResetLimitOprt_p			○	○	
UcaTagReadF64S			○	○	
UcaTagReadI32S			○	○	
UcaTagReadString			○	○	
UcaTagRead			○	○	
UcaTagWriteF64S			○	○	
UcaTagWriteI32S			○	○	
UcaTagWriteString			○	○	
UcaTagWrite			○	○	
UcaTagOneshot			○	○	
UcaOtherTagReadF64S			○	○	
UcaOtherTagReadI32S			○	○	
UcaOtherTagRead			○	○	
UcaOtherTagWriteF64S			○	○	
UcaOtherTagWriteI32S			○	○	
UcaOtherTagWrite			○	○	
UcaTagReadToRvnF64S			○	○	
UcaOtherTagReadToRvnF64S			○	○	
UcaTagArray1ReadF64S			○	○	
UcaTagArray1ReadI32S			○	○	
UcaTagArray1WriteF64S			○	○	
UcaTagArray1WriteI32S			○	○	
関数名	**機能ブロック初期化処理**	**機能ブロック終了処理**	**機能ブロック定周期処理**	**機能ブロックワンショット処理**	**機能ブロックデータ設定特殊処理**

表　呼び出し可否（4/4）

関数名	機能ブロック初期化処理	機能ブロック終了処理	機能ブロック定周期処理	機能ブロックワンショット処理	機能ブロックデータ設定特殊処理
UcaOtherTagArray1ReadF64S			○	○	
UcaOtherTagArray1ReadI32S			○	○	
UcaOtherTagArray1WriteF64S			○	○	
UcaOtherTagArray1WriteI32S			○	○	
UcaCalcTc	○	○	○	○	○
UcaCalcPcp	○	○	○	○	○
UcaCalcPckp	○	○	○	○	○
UcaCalcPcmp	○	○	○	○	○
UcaCalcTpcp	○	○	○	○	○
UcaCalcTpckp	○	○	○	○	○
UcaCalcTpcmp	○	○	○	○	○
UcaCalcAstm1	○	○	○	○	○
UcaCalcAstm2	○	○	○	○	○
UcaCalcAstm3	○	○	○	○	○
UcaCalcAstm4	○	○	○	○	○
UcaDataConvertType	○	○	○	○	○
UcaTimeResetLocalCounter	○	○	○	○	○
UcaTimeGetLocalCounter	○	○	○	○	○
UcaMesgSendSystemAlarm	○	○	○	○	○
UcaMesgSendPrintMessage	○	○	○	○	○
UcaMesgSendHistoricalMessage	○	○	○	○	○
UcaFpuExpClear	○	○	○	○	○
UcaFpuExpCheck	○	○	○	○	○
UcaTime	○	○	○	○	○
UcaLocaltime	○	○	○	○	○
UcaGmtime	○	○	○	○	○
UcaMktime	○	○	○	○	○

■ CSTM-A用

表　呼び出し可否（1/3）

関数名	機能ブロック初期化処理	機能ブロック終了処理	機能ブロック定周期処理	機能ブロックワンショット処理	機能ブロックデータ設定特殊処理
UcaModeSet	○	○	○	○	○
UcaModeGet	○	○	○	○	○
UcaModeGetPrev	○	○	○	○	○
UcaBstsSetExclusive	○	○	○	○	○
UcaBstsSet	○	○	○	○	○
UcaBstsClear	○	○	○	○	○
UcaBstsGet	○	○	○	○	○
UcaAlrmSet			○	○	
UcaAlrmClear			○	○	
UcaAlrmGet			○	○	
UcaDstsIsGood	○	○	○	○	○
UcaDstsIsBad	○	○	○	○	○
UcaDstsIsQst	○	○	○	○	○
UcaDstsIsPtpf	○	○	○	○	○
UcaDstsIsCnd	○	○	○	○	○
UcaDstsIsCal	○	○	○	○	○
UcaDstsSetBad	○	○	○	○	○
UcaDstsClearBad	○	○	○	○	○
UcaDstsSetQst	○	○	○	○	○
UcaDstsClearQst	○	○	○	○	○
UcaDstsSetCnd	○	○	○	○	○
UcaDstsClearCnd	○	○	○	○	○
UcaDstsChooseBadOrQst	○	○	○	○	○
UcaRWRead			○	○	
UcaRWWrite			○	○	
UcaRWReadback			○	○	
UcaRWReadString			○	○	
UcaRWWriteString			○	○	
UcaRWReadIn			○	○	
UcaRWSetCpv			○	○	
UcaRWWriteCpvToOutSub			○	○	
UcaRWWriteCpv			○	○	
UcaRWReadbackCpv			○	○	
UcaRWCheckOutputCondition			○	○	
UcaRWWriteSub			○	○	
UcaRWSeqCond			○	○	
UcaRWSeqOprt			○	○	
関数名	機能ブロック初期化処理	機能ブロック終了処理	機能ブロック定周期処理	機能ブロックワンショット処理	機能ブロックデータ設定特殊処理

表　呼び出し可否（2/3）

関数名	機能ブロック初期化処理	機能ブロック終了処理	機能ブロック定周期処理	機能ブロックワンショット処理	機能ブロックデータ設定特殊処理
UcaRWIsSeqCon			○	○	
UcaDataConvertRange_p			○	○	
UcaDataGetReadback			○	○	
UcaDataGetTagName	○	○	○	○	○
UcaDataStoreErrorNumber	○	○	○	○	○
UcaDataGet***	○	○	○	○	○
UcaDataStore***	○	○	○	○	○
UcaDataGet***n	○	○	○	○	○
UcaDataStore***n	○	○	○	○	○
UcaDataGetEachSn	○	○	○	○	○
UcaDataStoreEachSn	○	○	○	○	○
UcaConfigGeneral	○	○	○	○	○
UcaCtrlInitCheck	○	○	○	○	○
UcaCtrlInitClear	○	○	○	○	○
UcaCtrlInitSet	○	○	○	○	○
UcaTagReadF64S			○	○	
UcaTagReadI32S			○	○	
UcaTagReadString			○	○	
UcaTagRead			○	○	
UcaTagWriteF64S			○	○	
UcaTagWriteI32S			○	○	
UcaTagWriteString			○	○	
UcaTagWrite			○	○	
UcaTagOneshot			○	○	
UcaOtherTagReadF64S			○	○	
UcaOtherTagReadI32S			○	○	
UcaOtherTagRead			○	○	
UcaOtherTagWriteF64S			○	○	
UcaOtherTagWriteI32S			○	○	
UcaOtherTagWrite			○	○	
UcaTagReadToRvnF64S			○	○	
UcaOtherTagReadToRvnF64S			○	○	
UcaTagArray1ReadF64S			○	○	
UcaTagArray1ReadI32S			○	○	
UcaTagArray1WriteF64S			○	○	
UcaTagArray1WriteI32S			○	○	
UcaOtherTagArray1ReadF64S			○	○	
UcaOtherTagArray1ReadI32S			○	○	
UcaOtherTagArray1WriteF64S			○	○	
関数名	**機能ブロック初期化処理**	**機能ブロック終了処理**	**機能ブロック定周期処理**	**機能ブロックワンショット処理**	**機能ブロックデータ設定特殊処理**

表　呼び出し可否（3/3）

関数名	機能ブロック初期化処理	機能ブロック終了処理	機能ブロック定周期処理	機能ブロックワンショット処理	機能ブロックデータ設定特殊処理
UcaOtherTagArray1WriteI32S			○	○	
UcaCalcTc	○	○	○	○	○
UcaCalcPcp	○	○	○	○	○
UcaCalcPckp	○	○	○	○	○
UcaCalcPcmp	○	○	○	○	○
UcaCalcTpcp	○	○	○	○	○
UcaCalcTpckp	○	○	○	○	○
UcaCalcTpcmp	○	○	○	○	○
UcaCalcAstm1	○	○	○	○	○
UcaCalcAstm2	○	○	○	○	○
UcaCalcAstm3	○	○	○	○	○
UcaCalcAstm4	○	○	○	○	○
UcaDataConvertType	○	○	○	○	○
UcaTimeResetLocalCounter	○	○	○	○	○
UcaTimeGetLocalCounter	○	○	○	○	○
UcaMesgSendSystemAlarm	○	○	○	○	○
UcaMesgSendPrintMessage	○	○	○	○	○
UcaMesgSendHistoricalMessage	○	○	○	○	○
UcaFpuExpClear	○	○	○	○	○
UcaFpuExpCheck	○	○	○	○	○
UcaTime	○	○	○	○	○
UcaLocaltime	○	○	○	○	○
UcaGmtime	○	○	○	○	○
UcaMktime	○	○	○	○	○

Blank Page

Appendix A. データアイテム一覧

ユーザカスタムブロックのデータアイテム一覧を以下に示します。

■ データアイテム一覧

「データアイテムの有無」欄の「○」はデータアイテムがあることを示します。「空欄」はないことを示します。たとえば、データアイテム PV は、CSTM-C にはあるので「○」、CSTM-A にはないので「空欄」となります。

表　データアイテム一覧（1/4）

シンボル	データ名	データアイテムの有無		関連する主要なユーザカスタムアルゴリズム作成用ライブラリの関数	
		CSTM-C	CSTM-A	CSTM-C	CSTM-A
MODE	ブロックモード	○	○	UcaModeSet/UcaModeGet	
ALRM	アラームステータス	○	○	UcaAlrmSet/UcaAlrmClear	
AFLS	アラームフラッシングステータス	○	○		
AF	アラーム検出指定	○	○		
AOFS	アラーム抑制指定	○	○		
PV	測定値	○		UcaRWSetPv UcaRWWritePvToJnSub	
CPV	演算出力値		○		UcaRWSetCpv UcaRWWriteCpvToOutSub
RAW	生入力データ	○	○		
PVP	変化率警報サンプリング最旧値	○		UcaRWSetPv	
SUM	積算値	○	○	UcaRWSetPv	UcaRWSetCpv
SV	設定値	○		UcaCtrlHandler UcaDataSvPushback	
CSV	カスケード設定値	○		UcaCtrlHandler UcaDataSvPushback	
RSV	リモート設定値	○		UcaCtrlHandler UcaDataSvPushback	
DV	制御偏差値	○		UcaCtrlHandler UcaDataCheckDv	
VN	入出力補償値	○		UcaCtrlCompensation	
MV	操作出力値	○		UcaRWWriteMvToOutSub	
RMV	リモート操作出力値	○		UcaCtrlHandler	
RLV1	リセット信号値 1	○		UcaCtrlResetLimitOprt	
RLV2	リセット信号値 2	○		UcaCtrlResetLimitOprt	
HH	上上限警報設定値	○		UcaRWSetPv	
LL	下下限警報設定値	○		UcaRWSetPv	
PH	上限警報設定値	○		UcaRWSetPv	
PL	下限警報設定値	○		UcaRWSetPv	
VL	変化率警報設定値	○		UcaRWSetPv	

表　データアイテム一覧（2/4）

シンボル	データ名	データアイテムの有無		関連する主要なユーザカスタムアルゴリズム作成用ライブラリの関数	
		CSTM-C	CSTM-A	CSTM-C	CSTM-A
DL	偏差警報設定値	○		UcaCtrlHandler UcaDataCheckDv	
MH	操作出力上限設定値	○		UcaRWWriteMvToOutSub	
ML	操作出力下限設定値	○		UcaRWWriteMvToOutSub	
SVH	設定上限設定値	○		UcaCtrlHandler	
SVL	設定下限設定値	○		UcaCtrlHandler	
P	比例帯	○		UcaCtrlPid	
I	積分時間	○		UcaCtrlPid	
D	微分時間	○		UcaCtrlPid	
GW	ギャップ幅	○		UcaCtrlPid	
DB	不感帯	○		UcaCtrlDeadband	
CK	補償ゲイン	○		UcaCtrlCompensation	
CB	補償バイアス	○		UcaCtrlCompensation	
PMV	プリセット操作出力値	○		UcaRWWriteMvToOutSub	
TSW	トラッキングスイッチ	○		UcaCtrlHandler	
CSW	制御スイッチ	○		UcaCtrlPidTiming	
PSW	プリセット MV スイッチ	○		UcaRWWriteMvToOutSub	
RSW	パルス幅リセットスイッチ	○		UcaRWWriteMvToOutSub	
BSW	バックアップスイッチ	○		UcaCtrlHandler	
OPHI	出力上限置針	○			
OPLO	出力下限置針	○			
OPMK	オペマーク	○	○		
SAID	システムアプリケーション ID	○	○		
UAID	ユーザアプリケーション ID	○	○		
TYPE	機能ブロック形名	○	○		
OMOD	ブロックモード（最低優先順位）	○	○		
CMOD	ブロックモード（最高優先順位）	○	○		
SH	スケール上限値	○	○	UcaRWReadIn	
				UcaCtrlHandler	
SL	スケール下限値	○	○	UcaRWReadIn	
				UcaCtrlHandler	
MSH	MV スケール上限値	○		UcaRWWriteMvToOutSub	
MSL	MV スケール下限値	○		UcaRWWriteMvToOutSub	
DR	制御動作方向	○		UcaConfigDirection	
シンボル	データ名	CSTM-C	CSTM-A	CSTM-C	CSTM-A
		データアイテムの有無		関連する主要なユーザカスタムアルゴリズム作成用ライブラリの関数	

表　データアイテム一覧（3/4）

シンボル	データ名	データアイテムの有無		関連する主要なユーザカスタムアルゴリズム作成用ライブラリの関数	
		CSTM-C	CSTM-A	CSTM-C	CSTM-A
RAWRL1	生データ（RL1）	○			
RAWRL2	生データ（RL2）	○			
RAWBIN	生データ（BIN）	○			
RAWTIN	生データ（TIN）	○			
RAWTSI	生データ（TSI）	○			
RAWINT	生データ（INT）	○			
BSTS	ブロックステータス	○	○	UcaBstsSetExclusive	
RV	入力値	○	○	UcaRWReadIn	
RV01 ～ RV32	副入力データ	○	○	UcaRWRead	
RAW01 ～ RAW32	生入力データ	○	○		
MV01 ～ MV16	副出力データ	○		UcaRWWrite	
CPV01 ～ CPV16	演算出力値		○		UcaRWWrite
P01 ～ P32	パラメータ	○	○		
S01 ～ S16	文字列パラメータ	○	○		
I01 ～ I08	整数パラメータ	○	○		
CALC	演算結果	○			
ERRL	エラー発生個所	○	○	UcaDataStoreErrorNumber	
ERRC	エラーコード	○	○		
ERRA	エラーコード（自動設定）	○	○		
TRSW	実行時間リセット	○	○		
TSC	実効スキャン周期	○	○	UcaCtrlPid UcaCtrlPidTiming UcaCtrlResetLimitOprt	
PRGN	プログラム名称	○	○		
VERS	プログラムバージョン番号	○	○		
PRGS	実行管理情報	○	○		
ICON	初期化処理起動回数	○	○		
ICER	初期化処理起動エラー回数	○	○		
ILER	初期化処理起動前回エラーコード	○	○		
ITIM	初期化処理前回実行時間	○	○		
IMIN	初期化処理最小実行時間	○	○		
IMAX	初期化処理最大実行時間	○	○		
IAVE	初期化処理平均実行時間	○	○		
ITOT	初期化処理通算実行時間	○	○		
シンボル	データ名	CSTM-C	CSTM-A	CSTM-C	CSTM-A
		データアイテムの有無		関連する主要なユーザカスタムアルゴリズム作成用ライブラリの関数	

表　データアイテム一覧（4/4）

シンボル	データ名	データアイテムの有無		関連する主要なユーザカスタムアルゴリズム作成用ライブラリの関数	
		CSTM-C	CSTM-A	CSTM-C	CSTM-A
FCON	終了処理起動回数	○	○		
FCER	終了処理起動エラー回数	○	○		
FLER	終了処理起動前回エラーコード	○	○		
FTIM	終了処理前回実行時間	○	○		
FMIN	終了処理最小実行時間	○	○		
FMAX	終了処理最大実行時間	○	○		
FAVE	終了処理平均実行時間	○	○		
FTOT	終了処理通算実行時間	○	○		
PCON	定周期起動回数	○	○		
PCER	定周期起動エラー回数	○	○		
PLER	定周期起動前回エラーコード	○	○		
PTIM	定周期起動前回実行時間	○	○		
PMIN	定周期起動最小実行時間	○	○		
PMAX	定周期起動最大実行時間	○	○		
PTOT	定周期起動通算実行時間	○	○		
PAVE	定周期起動平均実行時間	○	○		
OCON	ワンショット起動回数	○	○		
OCER	ワンショット起動エラー回数	○	○		
OLER	ワンショット起動前回エラーコード	○	○		
OTIM	ワンショット起動前回実行時間	○	○		
OMIN	ワンショット起動最小実行時間	○	○		
OMAX	ワンショット起動最大実行時間	○	○		
OTOT	ワンショット起動通算実行時間	○	○		
OAVE	ワンショット起動平均実行時間	○	○		
DCON	データ設定時特殊処理回数	○	○		
DCER	データ設定時特殊処理エラー回数	○	○		
DLER	データ設定時特殊処理前回エラーコード	○	○		
DTIM	データ設定時特殊処理前回実行時間	○	○		
DMIN	データ設定時特殊処理最小実行時間	○	○		
DMAX	データ設定時特殊処理最大実行時間	○	○		
DTOT	データ設定時特殊処理通算実行時間	○	○		
DAVE	データ設定時特殊処理平均実行時間	○	○		
シンボル	データ名	CSTM-C	CSTM-A	CSTM-C	CSTM-A
		データアイテムの有無		関連する主要なユーザカスタムアルゴリズム作成用ライブラリの関数	

Appendix B. 入出力端子一覧

ユーザカスタムブロックの入出力端子一覧を以下に示します。

■ 入出力端子一覧

「端子の有無」欄の「○」は端子があることを示します。「空欄」はないことを示します。たとえば、TIN 端子は、CSTM-C にはあるので「○」、CSTM-A にはないので「空欄」となります。

表　入出力端子一覧

入出力端子		端子の有無		関連する主要なユーザカスタムアルゴリズム作成用ライブラリの関数	
		CSTM-C	CSTM-A	CSTM-C	CSTM-A
IN	測定入力／演算入力	○	○	UcaRWReadIn	
SET	設定入力	○		UcaCtrlHandler	
OUT	操作出力	○	○	UcaRWWriteMvToOutSub	UcaRWWriteCpvToOutSub
SUB	補助出力	○	○	UcaRWWriteMvToOutSub	UcaRWWriteCpvToOutSub
RL1	リセット信号 1 入力	○		UcaCtrlHandler（UcaCtrlResetLimitOprt）(*1)	
RL2	リセット信号 2 入力	○		UcaCtrlHandler（UcaCtrlResetLimitOprt）(*2)	
BIN	補償入力	○		UcaCtrlHandler（UcaCtrlCompensation）(*3)	
TIN	トラッキング信号入力	○		UcaCtrlHandler	
TSI	トラッキング SW 入力	○		UcaCtrlHandler	
INT	インタロック SW 入力	○		UcaCtrlHandler	
Q01 ～ Q32	第 n 演算入力	○	○	UcaRWRead	
J01 ～ J16	第 n 演算出力	○	○	UcaRWWrite	

*1： UcaCtrlHandler が、RL1 端子をデータアイテム RLV1 に読み込みます。UcaCtrlResetLimitOprt は、データアイテム RVL1 にアクセスします。
*2： UcaCtrlHandler が、RL2 端子をデータアイテム RLV2 に読み込みます。UcaCtrlResetLimitOprt は、データアイテム RVL2 にアクセスします。
*3： UcaCtrlHandler が、BIN 端子をデータアイテム VN に読み込みます。UcaCtrlCompensation は、データアイテム VN にアクセスします。

Blank Page

Appendix C. ラベル一覧

ユーザカスタムアルゴリズムで使用するラベルについて説明します。
ラベルは、システム定義インクルードファイルlibucadef.hに定義されています。

Appendix C.1　データ型

データ型を示すラベルと値を示します。

■ データ型

表　データ型とラベル

データ型	ラベル	値（10進）	説明
I16	UCA_DATATYPE_I16	12	2 バイト符合付き整数
U16	UCA_DATATYPE_U16	16	2 バイト符号なし整数
I32	UCA_DATATYPE_I32	20	4 バイト符号付き整数
U32	UCA_DATATYPE_U32	24	4 バイト符号なし整数
F32	UCA_DATATYPE_F32	28	単精度浮動小数
F64	UCA_DATATYPE_F64	32	倍精度浮動小数
BYTE string[16]	UCA_DATATYPE_CHR	8	文字列
I16S	UCA_DATATYPE_I16S	13	データステータス付き 2 バイト符号付き整数
U16S	UCA_DATATYPE_U16S	17	データステータス付き 2 バイト符号なし整数
I32S	UCA_DATATYPE_I32S	21	データステータス付き 4 バイト符号付き整数
U32S	UCA_DATATYPE_U32S	25	データステータス付き 4 バイト符号なし整数
F32S	UCA_DATATYPE_F32S	29	データステータス付き単精度浮動小数
F64S	UCA_DATATYPE_F64S	33	データステータス付き倍精度浮動小数
F32SR	UCA_DATATYPE_F32SR	31	データステータス／レンジ付き単精度浮動小数
DITMN	UCA_DATATYPE_DITM	60	データアイテム名

Appendix C.2　ブロックモード

ブロックモードを示すラベルと値を示します。

■ ブロックモード

表　ブロックモードとラベル

ビット番号（10進）	シンボル	名称	ラベル	値
1	O/S	サービスオフ（Out of Service）	UCAMASK_MODE_OS	0x80000000
4	IMAN	初期化手動（Initialization MANual）	UCAMASK_MODE_IMAN	0x08000000
5	TRK	トラッキング（TRacKing）	UCAMASK_MODE_TRK	0x04000000
9	MAN	手動（MANual）	UCAMASK_MODE_MAN	0x00800000
10	AUT	自動（AUTomatic）	UCAMASK_MODE_AUT	0x00400000
11	CAS	カスケード（CAScade）	UCAMASK_MODE_CAS	0x00200000
12	PRD	プライマリダイレクト（PRimary Direct）	UCAMASK_MODE_PRD	0x00100000
25	RCAS	リモートカスケード（Remote CAScade）	UCAMASK_MODE_RCAS	0x00080000
26	ROUT	リモート出力（Remote OUTput）	UCAMASK_MODE_ROUT	0x00040000

Appendix C.3　ブロックステータス

ユーザカスタムブロックのブロックステータスは、すべてユーザが定義します。システム定義インクルードファイルには、次ページの表のように1～32ビットに対応したブロックステータスのラベルを定義してあります。しかし、このラベルをそのまま使うのではなく、ユーザ定義のブロックステータス名に合わせたラベルをユーザ定義インクルードファイルに定義して、そのラベルを使うことを推奨します。

■ ブロックステータス

参照 ユーザカスタムブロックのブロックステータスについては、以下を参照してください。
APCS ユーザカスタムブロックプログラミングガイド（IM 33J15U21-01JA）の「4.3.1 ブロックステータス」
APCS ユーザカスタムブロックプログラミングガイド（IM 33J15U21-01JA）の「4.3.3 ユーザ定義インクルードファイル usrstatus.h」

表　ブロックステータスのビット番号とラベル

ビット番号（10進）	ラベル	値（16進）
1	UCAMASK_BSTS_01	0x80000000
2	UCAMASK_BSTS_02	0x40000000
3	UCAMASK_BSTS_03	0x20000000
4	UCAMASK_BSTS_04	0x10000000
5	UCAMASK_BSTS_05	0x08000000
6	UCAMASK_BSTS_06	0x04000000
7	UCAMASK_BSTS_07	0x02000000
8	UCAMASK_BSTS_08	0x01000000
9	UCAMASK_BSTS_09	0x00800000
10	UCAMASK_BSTS_10	0x00400000
11	UCAMASK_BSTS_11	0x00200000
12	UCAMASK_BSTS_12	0x00100000
13	UCAMASK_BSTS_13	0x00080000
14	UCAMASK_BSTS_14	0x00040000
15	UCAMASK_BSTS_15	0x00020000
16	UCAMASK_BSTS_16	0x00010000
17	UCAMASK_BSTS_17	0x00008000
18	UCAMASK_BSTS_18	0x00004000
19	UCAMASK_BSTS_19	0x00002000
20	UCAMASK_BSTS_20	0x00001000
21	UCAMASK_BSTS_21	0x00000800
22	UCAMASK_BSTS_22	0x00000400
23	UCAMASK_BSTS_23	0x00000200
24	UCAMASK_BSTS_24	0x00000100
25	UCAMASK_BSTS_25	0x00000080
26	UCAMASK_BSTS_26	0x00000040
27	UCAMASK_BSTS_27	0x00000020
28	UCAMASK_BSTS_28	0x00000010
29	UCAMASK_BSTS_29	0x00000008
30	UCAMASK_BSTS_30	0x00000004
31	UCAMASK_BSTS_31	0x00000002
32	UCAMASK_BSTS_32	0x00000001

Appendix C.4　アラームステータス

デフォルトのアラームステータスのラベルと値を示します。

■ アラームステータス

参照 ユーザ定義アラームのラベルを定義する方法については、以下を参照してください。
APCS ユーザカスタムブロックプログラミングガイド（IM 33J15U21-01JA）の「4.3.2 アラームステータス」
APCS ユーザカスタムブロックプログラミングガイド（IM 33J15U21-01JA）の「4.3.3 ユーザ定義インクルードファイル usrstatus.h」

表　デフォルトのアラームステータスのラベルと値

ビット番号（10進）	シンボル	名称	ラベル	値（16進）
1 ～ 8		（使用不可）		
9	NR	正常状態	UCAMASK_ALRM_NR	0x00800000
10	OOP	出力オープン警報	UCAMASK_ALRM_OOP	0x00400000
11	IOP	上限入力オープン警報	UCAMASK_ALRM_IOP	0x00200000
12	IOP-	下限入力オープン警報	UCAMASK_ALRM_IOPM	0x00100000
13	HH	上上限警報	UCAMASK_ALRM_HH	0x00080000
14	LL	下下限警報	UCAMASK_ALRM_LL	0x00040000
15		（ユーザ定義）	UCAMASK_ALRM_15	0x00020000
16			UCAMASK_ALRM_16	0x00010000
17	HI	上限警報	UCAMASK_ALRM_HI	0x00008000
18	LO	下限警報	UCAMASK_ALRM_LO	0x00004000
19		（ユーザ定義）	UCAMASK_ALRM_19	0x00002000
20			UCAMASK_ALRM_20	0x00001000
21	DV+	正方向偏差警報	UCAMASK_ALRM_DVP	0x00000800
22	DV-	負方向偏差警報	UCAMASK_ALRM_DVM	0x00000400
23		（ユーザ定義）	UCAMASK_ALRM_23	0x00000200
24			UCAMASK_ALRM_24	0x00000100
25	VEL+	正方向変化率警報	UCAMASK_ALRM_VELP	0x00000080
26	VEL-	負方向変化率警報	UCAMASK_ALRM_VELM	0x00000040
27	MHI	出力上限警報	UCAMASK_ALRM_MHI	0x00000020
28	MLO	出力下限警報	UCAMASK_ALRM_MLO	0x00000010
29		（ユーザ定義）	UCAMASK_ALRM_29	0x00000008
30			UCAMASK_ALRM_30	0x00000004
31	ERRC	演算異常	UCAMASK_ALRM_ERRC	0x00000002
32	CNF	結合状態不良警報	UCAMASK_ALRM_CNF	0x00000001

Appendix C.5　データステータス

データステータスのラベルと値を示します。

■ データステータス

表　データステータスのラベル

ビット番号（10進）	シンボル	名称	ラベル	値（16進）
1	BAD	データ値不良（BAD value）	UCAMASK_DSTS_BAD	0x80000000
2	QST	データ値疑問（QueSTionable value）	UCAMASK_DSTS_QST	0x40000000
3	NCOM	通信不良（No COMmunication）	UCAMASK_DSTS_NCOM	0x20000000
4	NFP	非プロセス起源（Not From Process）	UCAMASK_DSTS_NFP	0x10000000
5	PTPF	出力フェイル（Path To Process Failed）	UCAMASK_DSTS_PTPF	0x08000000
6	CLP+	上限クランプ（CLamP high）	UCAMASK_DSTS_CLPP	0x04000000
7	CLP-	下限クランプ（CLamP low）	UCAMASK_DSTS_CLPM	0x02000000
8	CND	コンディショナル（ConNDitional）	UCAMASK_DSTS_CND	0x01000000
9	CAL	キャリブレーション（CALibration）	UCAMASK_DSTS_CAL	0x00800000
10	NEFV	無効データ（Not EFfectiVe）	UCAMASK_DSTS_NEFV	0x00400000
11	O/S	サービスオフ（Out of Service）	UCAMASK_DSTS_OS	0x00200000
12	MNT	メンテナンス（MaiNTenace）	UCAMASK_DSTS_MNT	0x00100000
17	IOP+	上限入力オープン（Input OPen high）	UCAMASK_DSTS_IOPP	0x00008000
18	IOP-	下限入力オープン（Input OPen low）	UCAMASK_DSTS_IOPM	0x00004000
19	OOP	出力オープン（Output OPen）	UCAMASK_DSTS_OOP	0x00002000
20	NRDY	PI/O ノットレディ（PI/O Not ReaDY）	UCAMASK_DSTS_NRDY	0x00001000
21	PFAL	PI/O 無応答（PI/O Power FAiLure）	UCAMASK_DSTS_PFAL	0x00000800
22	LPFL	PI/O 長時間無応答 （PI/O Long Power FaiLure）	UCAMASK_DSTS_LPFL	0x00000400
25	MINT	マスタ初期化（Master INiTialize）	UCAMASK_DSTS_MINT	0x00000080
26	SINT	スレーブ初期化（Slave INiTialize）	UCAMASK_DSTS_SINT	0x00000040
27	SVPB	SV プッシュバック（SV PushBack）	UCAMASK_DSTS_SVPB	0x00000020

Appendix C.6　ユーザ定義関数の呼び出し理由

機能ブロック初期化処理と機能ブロック終了処理の引数に指定される呼び出し理由のラベルと値を示します。

■ ユーザ定義関数の呼び出し理由

表　機能ブロック初期化処理の呼び出し理由

ラベル	値	説明
UCAINIT_START	1	APCS 初期化スタート
UCAINIT_NEW	2	制御ドローイングビルダより、オンラインでユーザカスタムブロックを追加
UCAINIT_PRG	3	機能ブロック詳細ビルダより、オンラインで「プログラム名称」を変更
		システムビューより、オンラインでユーザカスタムアルゴリズムをダウンロード
UCAINIT_MAINT	4	機能ブロック詳細ビルダより、オンラインで「プログラム名称」以外を変更
UCAINIT_AWAKEN	5	外部からのデータ設定によりブロックモード（MODE）が O/S から復帰

表　機能ブロック終了処理の呼び出し理由

ラベル	値	説明
UCAFINISH_DELETE	2	制御ドローイングビルダより、オンラインでユーザカスタムブロックを削除
UCAFINISH_PRG	3	機能ブロック詳細ビルダより、オンラインで「プログラム名称」を変更
		システムビューより、オンラインでユーザカスタムアルゴリズムを削除
UCAFINISH_MAINT	4	機能ブロック詳細ビルダより、オンラインで「プログラム名称」以外を変更
UCAFINISH_SLEEP	5	外部からのデータ設定によりブロックモード（MODE）が O/S に遷移

Appendix C.7　ビルダ定義項目

ビルダ定義項目の指定を取得する関数が返す値とそのラベルを示します。

■ ビルダ定義項目

表　非線形ゲイン（UcaConfigNonlinearGain関数）

指定	ラベル	値
なし	UCACONFIG_NONLINEAR_NON	0
ギャップ動作	UCACONFIG_NONLINEAR_GAPACT	1
偏差二乗動作	UCACONFIG_NONLINEAR_SQRDVACT	2

表　入出力補償（UcaConfigCompensation関数）

指定	ラベル	値
なし	UCACONFIG_COMP_NON	0
入力補償	UCACONFIG_COMP_INPUT	1
出力補償	UCACONFIG_COMP_OUTPUT	2

表　制御動作方向（UcaConfigDirection関数）

指定	ラベル	値
逆動作	UCACONFIG_DIR_REV	1
正動作	UCACONFIG_DIR_DIR	0

表　制御／演算出力動作（UcaConfigOutput関数）

指定	ラベル	値
位置形	UCACONFIG_ACT_POS	0
速度形	UCACONFIG_ACT_VEL	1

表　不感帯動作（UcaConfigDeadband関数）

指定	ラベル	値
なし	UCACONFIG_DEADBAND_FALSE	0
あり	UCACONFIG_DEADBAND_TRUE	1

Appendix C.8　制御演算

制御演算関数で使用する値とそのラベルを示します。

■ 制御演算

表　演算処理実行ブロックモード（UcaCtrlHandler関数）

ラベル	値	説明
UCACTRL_AUTCASRCAS	1	AUT/CAS/RCAS モード時のみ演算処理関数を実行
UCACTRL_ALLMODE	2	すべてのモードで演算処理関数を実行

表　不感帯動作前回運転状態（UcaConfigDeadband関数）

ラベル	値	説明
UCA_DEADBAND_SMALL	0	偏差小
UCA_DEADBAND_LARGEPLUS	1	正方向の偏差大
UCA_DEADBAND_LARGEMINUS	2	負方向の偏差大

Appendix C.9　浮動小数演算の例外発生フラグ

浮動小数演算における例外発生フラグとラベルを以下に示します。例外発生フラグは、UcaFpuExpClearでクリアし、UcaFpuExpCheckで取得します。ラベルは、システム定義インクルードファイルlibucadef.hで定義されています。

■ 浮動小数演算の例外

表　UcaFpcExpCheckが返す演算エラー

意味	ラベル	値（10進）	値（16進）
正常（例外発生なし）	なし	0	0x00000000
オーバフロー	UCAERR_FPU_OVERFLW	4	0x00000100
ゼロ割り算	UCAERR_FPU_ZERODIV	8	0x00001000
無効演算	UCAERR_FPU_INVALID	16	0x00010000
演算フラグクリア抜け (*1)	UCAERR_FPU_EXNOCLR	32	0x00100000

*1： 未使用

Blank Page

Appendix D. エラーコード一覧

ユーザカスタムブロックのエラーコードには、ユーザカスタムブロック実行管理部のエラーコードとユーザカスタムアルゴリズム作成用ライブラリのエラーコードがあります。

Appendix D.1 ユーザカスタムブロック実行管理部のエラーコード

ユーザカスタムブロック実行管理部が検出するエラーコードの一覧を以下に示します。

■ ユーザカスタムブロック実行管理部のエラーコード

表 ユーザカスタムブロック実行管理部エラーコード一覧

10進	16進	ラベル	説明
58641	0xE501	UCAERR_NOPROC	ユーザ定義関数が UCAERR_NOPROC で戻った
58675	0xE502	UCAERR_STOPME	ユーザ定義関数が UCAERR_STOPME で戻った
58676	0xE503	UCAERR_NONAME	機能ブロック詳細ビルダで「プログラム名称」を指定していない
58689	0xE504	UCAERR_NOFUNC	機能ブロック詳細ビルダで「プログラム名称」に指定したユーザカスタムアルゴリズムに、ユーザ定義関数が定義されていない
58690	0xE505	UCAERR_NOTLOADED	機能ブロック詳細ビルダで「プログラム名称」に指定したユーザカスタムアルゴリズムがロードされていない
58691	0xE506	UCAERR_EXCEPTION	ユーザ定義関数内で例外を検出
58692	0xE507	UCAERR_TIMELIMIT	ユーザ定義関数内で 1 秒以上連続して実行

これらのエラーコードは、システムにより以下のデータアイテムに設定されます。データアイテムには最後に検出したエラーコードが設定されます。

表 エラーコードが設定されるデータアイテム

データアイテム	説明
ILER	機能ブロック初期化処理の実行で検出したエラーコード
FLER	機能ブロック終了処理の実行で検出したエラーコード
PLER	機能ブロック定周期処理の実行で検出したエラーコード
OLER	機能ブロックワンショット起動処理の実行で検出したエラーコード
DLER	機能ブロックデータ設定時特殊処理の実行で検出したエラーコード

Appendix D.2 ユーザカスタムアルゴリズム作成用ライブラリのエラーコード

ユーザカスタムアルゴリズム作成用ライブラリのエラーコード一覧を示します。このエラーコードは、ユーザカスタムアルゴリズム作成用ライブラリが戻り値として返す値です。ユーザカスタムアルゴリズム作成用ライブラリが最後に検出した（つまりもっとも新しい）エラーコードは、システムによりユーザカスタムブロックのデータアイテムERRAに設定されます。

■ ライブラリのエラーコード

参照
- エラーコードの扱い方については、以下を参照してください。
  APCS ユーザカスタムブロック プログラミングガイド（IM 33J15U21-01JA）「3.8.1 ユーザカスタムアルゴリズム作成用ライブラリのエラーコード」
- 0xE5**（16 進）以外のエラーコードは、CENTUM VP のプロセス制御用プログラム言語 SEBOL の詳細エラーコードと同じです。以下を参照してください。
  SEBOL リファレンス（IM 33J05L20-01JA）「13.4 詳細エラーコード」

表 ユーザカスタムアルゴリズム作成用ライブラリエラーコード一覧（1/2）

10進	16進	ラベル	説明
58641	0xE511	UCAERR_RW_NOCONNECT	結合を定義していない入出力端子にアクセスした
58642	0xE512	UCAERR_RW_LOOPCONN	端子と端子を結合している
58643	0xE513	UCAERR_RW_MNT	入出力端子に結合してある機能ブロックがオンラインメンテナンス中
58644	0xE514	UCAERR_RW_OS	入出力端子に結合してある機能ブロックのブロックモードが O/S
58645	0xE515	UCAERR_RW_SWOPEN	IN 端子に結合してある切り換えスイッチがオープン
58646	0xE516	UCAERR_RW_IOPHI	入力端子から入力したデータの異常、または、入力データが上限方向に振り切れ（IOP アラーム検出）
58647	0xE517	UCAERR_RW_IOPLO	入力データが下限方向に振り切れ（IOP- アラーム検出）
58648	0xE518	UCAERR_RW_OOP	出力端子に結合してある出力先データの異常を検出（OOP アラーム検出）
58649	0xE519	UCAERR_RW_PTPF	出力先データのデータステータスが PTPF（出力フェイル）
58650	0xE51A	UCAERR_RW_BADVALUE	入出力端子に結合してある結合先データのデータステータスが BAD
58651	0xE51B	UCAERR_RW_INVALIDNO	端子番号の誤り（Q01 ～ Q32、J01 ～ J16 の範囲外）
57653	0xE51D	UCAERR_RW_RANGEOVER	データがレンジの範囲外
58655	0xE51F	UCAERR_RW_CONNFAIL	データアクセスエラー
58657	0xE521	UCAERR_TAG_NOSTRING	指定された機能ブロックデータは文字列型ではない
58658	0xE522	UCAERR_TAG_INVALIDDTYPE	UcaDataType 型の引数のデータ型コード dType に誤りがある
58659	0xE523	UCAERR_TAG_INVALIDPARA	UcaDataType 型の引数のメンバ indOpt または rqstCode に 0 以外を指定した
58660	0xE524	UCAERR_TAG_NOADL	BDSET-1L でステーション間結合ブロックを定義していないデータを指定した
58661	0xE525	UCAERR_TAG_INVALIDDB	データベースタイプの誤り（UcaOtherTagRead***/Write**** を使用できないデータベースタイプを使用）

表　ユーザカスタムアルゴリズム作成用ライブラリエラーコード一覧（2/2）

10進	16進	ラベル	説明
58673	0xE531	UCAERR_DATA_RANGEOVER	データがレンジの範囲外
58675	0xE533	UCAERR_DATA_INVALIDDATA	読み返し（Readback）していないデータにアクセスした
58676	0xE534	UCAERR_DATA_INVALIDNO	番号付きデータアイテム取得・保存関数に指定しているデータ番号／データ数の誤り（データアイテム P01 ～ P32 に対し P33 を取得しようとした場合など）
58689	0xE541	UCAERR_RANGE_INVALIDSCALE	引数に指定されたスケール上限値／下限値が正しくない
58705	0xE551	UCAERR_MESG_INVALIDTYPE	UcaMesgSendSystemAlarm に指定した「メッセージのタイプ」に誤りがある
58707	0xE553	UCAERR_TEXT_CONVFAIL	指定した文字列中に半角カナ（または異常な文字コード）を検出した
58721	0xE561	UCAERR_CTRL_PIDCALC	PID 演算計算中に浮動小数演算で例外発生
58722	0xE562	UCAERR_CTRL_CTRLTIME	制御周期時間が異常
58723	0xE563	UCAERR_CTRL_GAPGAIN	ギャップゲイン計算中に浮動小数演算で例外発生
58724	0xE564	UCAERR_CTRL_DVSQUAREGAIN	偏差二乗ゲイン計算中に浮動小数演算で例外発生
58725	0xE565	UCAERR_CTRL_RESETLIMITTI	リセットリミット計算で積分時間が異常
58726	0xE566	UCAERR_CTRL_PIDTI	PID 制御計算で積分時間が異常
58737	0xE571	UCAERR_TC_OVERFLOW	オーバフローを検出（温圧補正関数）
58738	0xE572	UCAERR_TC_ROOTMINUS	ルート計算の対象が負（温圧補正関数）
58739	0xE573	UCAERR_TC_ZERODIV	ゼロ割を検出（温圧補正関数）した
58753	0xE581	UCAERR_MODE_OS	モード設定禁止状態
58769	0xE591	UCAERR_FUNCID_INIT	機能ブロック初期化処理で使用できない関数を呼び出した
58770	0xE592	UCAERR_FUNCID_FINISH	機能ブロック終了処理で使用できない関数を呼び出した
58771	0xE593	UCAERR_FUNCID_PERIODICAL	機能ブロック定周期処理で使用できない関数を呼び出した
58772	0xE594	UCAERR_FUNCID_ONESHOT	機能ブロックワンショット処理で使用できない関数を呼び出した
58773	0xE595	UCAERR_FUNCID_DSET	機能ブロックデータ設定特殊処理で使用できない関数を呼び出した
58866	0xE5F2	UCAERR_MODE_INVALID	UcaModeSet に誤ったブロックモードを指定（2.1.1 UcaModeSet 参照）
58867	0xE5F3	UCAERR_ALRM_INVALID	UcaAlrmSet または UcaAlrmClear に誤ったアラームステータスを指定（2.3.1 UcaAlrmSet と 2.3.2 UcaAlrmClear 参照）した
58868	0xE5F4	UCAERR_OPT_INVALID	誤ったオプションを指定した
58869	0xE5F5	UCAERR_BLKTYPE_INVALID	実行できないライブラリ関数を呼び出した（CSTM-C で CSTM-A 専用のライブラリ関数を呼び出しなど）
58870	0xE5F6	UCAERR_NUM_INVALID	汎用指定項目の番号に 1 ～ 8 の範囲外を指定した
58871	0xE5F7	UCAERR_PARA_PTRNULL	引数のポインタが NULL である
58872	0xE5F8	UCAERR_NOTANUMBER	引数の浮動小数データが NAN（Not a Number、非数）である
58873	0xE5F9	UCAERR_INFINITY	引数の浮動小数データが無限大である
58875	0xE5FB	UCAERR_EXECMODE_INVALID	演算処理実行ブロックモードの指定に誤りがある
58879	0xE5FF	UCAERR_FATAL	引数のブロックコンテキスト（bc）が異常

Appendix E. ソース公開

以下に、ソースファイルが公開されている関数を示します。

■ ソース公開されている関数

- UcaBstsSetExclusive
- UcaDstsChooseBadOrQst
- UcaDataStoreF64SToMvn
- UcaOtherTagArray1ReadF64S
- UcaOtherTagArray1ReadI32S
- UcaOtherTagArray1WriteF64S
- UcaOtherTagArray1WriteI32S
- UcaOtherTagReadToRvnF64S
- UcaRWWriteCpvToOutSub
- UcaRWWriteMvToOutSub
- UcaRWWritePvToJnSub
- UcaTagArray1ReadF64S
- UcaTagArray1ReadI32S
- UcaTagArray1WriteF64S
- UcaTagArray1WriteI32S
- UcaTagReadToRvnF64S

Blank Page

APCS ユーザカスタムブロック ライブラリ

IM 33J15U22-01JA 2 版

索　引

テ

ニ

ヒ

フ

メ

リ

ワ

Blank Page

改訂情報

資料名称　　：APCS ユーザカスタムブロック ライブラリ
資料番号　　：IM 33J15U22-01JA

2019年8月／2版／R6.07以降
前書き　　「■ 商標」の記述変更

2018年8月／初版／R6.06
新規発行

■ お問い合わせについて
問い合わせ：http://www.yokogawa.co.jp/dcs より、お問い合わせフォームをご利用ください。
■ 著作者　　横河電機株式会社
■ 発行者　　横河電機株式会社
〒 180-8750　　東京都武蔵野市中町 2-9-32

Blank Page