KEYENCE

084699

プログラマブルコントローラ KV-8000シリーズ

ST言語プログラミングマニュアル

お使いになる前に、このマニュアルをお読みください。
いつでも使用できるように大切に保管してください。

1章 概 要
2章 ST を作成する
3章 データの取り扱い
4章 ST の文法と注意
5章 基本関数
6章 応用関数
7章 演算関数
8章 拡張関数
・コントロールメモリ、コントロールリレー一覧
・索引

対応CPUユニット
・KV-8000
・KV-8000A

本社技術スタッフ直営！
FAQ よくあるご質問
www.keyence.co.jp/seigyofaq

はじめに

本書は、《KV-8000シリーズ》のST言語プログラムを作成する上で必要となる機能の仕様や設定方法について説明しています。このマニュアルをよくお読みいただき、十分に理解した上でプログラムを作成してください。
プログラムされる際には、『KV-8000シリーズ ユーザーズマニュアル』、『KV-8000/7000/5000/3000/1000シリーズ・KV Nanoシリーズ命令語リファレンスマニュアル』『KV STUDIOユーザーズマニュアル』を、CPU内蔵機能に関する命令語や拡張ユニット専用命令語については、各ユニットのユーザーズマニュアルをご覧ください。
本書はいつでもご利用いただけますよう、大切に保管してください。
本書が最終的にご利用いただく方のお手元に届くよう、ご配慮ください。

■KV-8000シリーズ関連マニュアル

以下の全てのPDFマニュアルは、KV STUDIOのヘルプから開けます。また、最新のPDFマニュアルは弊社のホームページよりダウンロード可能です。
印刷版マニュアルのご請求は、本書裏表紙のお問い合わせ電話番号にてご用命ください。

種類	内容	製本
KV-8000シリーズ ユーザーズマニュアル	《KV-8000 シリーズ》の接続・保守方法、CPU 内蔵機能、ラダープログラミングを作成する方法を説明しています。	なし（PDFのみ）
KV-8000/7000/5000/3000/1000シリーズ、KV Nanoシリーズ 命令語リファレンスマニュアル	ラダープログラミングで使用可能な各命令語を説明しています。	なし（PDFのみ）
KV-8000シリーズST言語プログラミングマニュアル	本マニュアルです。ST言語プログラムを作成する方法と使用可能な演算子・制御文・関数を説明しています。	なし（PDFのみ）
KV-8000/7000/5000/3000/1000シリーズ、KV Nanoシリーズ スクリプトプログラミングマニュアル	スクリプトプログラムを作成する方法と使用可能な演算子・制御文・関数を説明しています。	なし（PDFのみ）
KV STUDIO ユーザーズマニュアル	《KV STUDIO》の操作方法を説明しています。	なし（PDFのみ）

安全にお使いいただくために

■記号の見方

本書では、人への危害や機器への損害を未然に防ぐため、守っていただきたい事項を下記のように表示区分しています。

危 険	ここに記載されている記載事項を遵守しない場合、結果的に死亡又は重傷を引き起こす危害が発生します。
警 告	ここに記載されている記載事項を遵守しない場合、結果的に死亡又は重傷を引き起こす危害が発生する可能性があります。
注 意	ここに記載されている記載事項を遵守しない場合、中程度の傷害又は軽傷を引き起こす危害が発生する可能性があります。
注 記	ここに記載されている記載事項を遵守しない場合、商品自体の損害(自損)のみならず、他の財物に対する損傷を引き起こす可能性があります。

▶ 重 要　かならずおこなう操作などについての注意を示しています。

ポイント　誤りやすい操作などについての注意を示しています。

参 考　本文の理解を深める事項や、知っておくと役に立つ情報を示しています。

参照すべきページや別のマニュアルの参照ページを示しています。

■一般的な注意事項

危 険	• 人体および、人体の一部を保護する目的でこの製品を使用してはいけません。 • この製品は、防爆エリアで使用することを想定していないので、防爆エリアでは決して使用しないでください。 • この商品の動作状態によって人および財物に大きな危害が及ぶ可能性のある用途（発電所、航空、鉄道、船舶、車両、医療機器、娯楽遊具等）では使用しないでください。
警 告	• 《KV STUDIO》の使用時には、《KV シリーズ》の機能および性能が正常に動作していることを確認してからご使用ください。 • 始業または操作時には、当社製品の機能および性能が正常に動作していることを確認してからご使用ください。 • プログラマブルコントローラ本体に異常が発生した場合でも、フェールセーフの点から、システム全体が安全側に働くように、プログラマブルコントローラを介さない方法で安全回路を設けてください。 • 出力回路や内部回路の故障により、正常に制御動作できなくなる場合が考えられます。火災など、重大な事故の要因となる制御については、安全回路を必ず設けてください。 • この商品を設置する際は、装置全体のリスクアセスメントを実施し、適切なリスク低減がなされていることを確認してください。異常発生時でも装置全体が安全に動作するよう、この商品を介さずに適切な保護手段を設けてください。
注 意	この取扱説明書に規定された方法以外でこの商品を使用すると、商品が備えている保護が損なわれる可能性があります。
注 記	• 仕様に示された規格以外での使用、または改造された製品については、機能および性能の保証はできかねますのでご留意ください。 • 当社製品を他の機器と組み合わせてご使用になる場合、使用条件、環境などにより、機能および性能が満足できない場合がありますので、十分ご検討のうえご使用ください。

KV-8000 CPU機能バージョンについて

KV-8000 シリーズは、CPU 機能バージョンにより使用できる機能が異なります。
CPU 機能バージョンは、KV STUDIO のメニューで【ツール】→【CPU 機能バージョン確認】→【PLC】から確認できます。（※ KV STUDIO Ver.10.0 以降が必要です）
また、CPU 機能バージョンは、KV STUDIO のメニューで【ツール】→【CPU 機能バージョン確認】→【PLC】→〔システムプログラム更新〕から更新できます。（※KV STUDIO Ver.10.0 以降が必要です。）

■CPU機能バージョンによるCPU機能差分

公開		2021/09	2021/07	2021/04	2020/04	2019/04
CPU機能バージョン		2.3	2.2	2.1	2.0	1.0
KV STUDIOの対応バージョン		Ver.11.3以降	Ver.11.2以降	Ver.11.1以降	Ver.11.0以降	Ver.10.0以降
CPU機能	デバイスコメント	最大128文字	最大128文字	最大128文字	最大32文字	最大32文字
	変数	○	○	○	○	―
	配列	ファンクションブロック型に 対応	STRING型に対応	―	―	―
	構造体	○	○	○	○	―
	ファンクションブロック	データ型の自動変換に対応	○	○	○（EN/ENOに対応、インスタンスに対応）	○
	ST言語	○	○	○	―	―
命令語スクリプト	ファンクションブロック関数	○	○	○	2関数追加（FBCALL, FBSTRT）	追加なし
	データサイズ取得関数	○	○	○	2関数追加（ACNT,WSIZE）	―

用語について

このマニュアルの説明では一部をのぞき、次のような用語を使っています。

用　語	説　明
プログラマブルコントローラ	プログラムを変更することで自由に機器を制御できる電子装置で、PLC（Programmable Logic Controller）とも呼ばれています。
《KV-8000》	当社製プログラマブルコントローラKV-8000/8000Aです。
KV-8000シリーズ	CPUユニット《KV-8000》と接続可能な拡張ユニットの総称です。
CPUユニット	《KV-8000》です。
KV-8000シリーズ 拡張ユニット	KV-8000シリーズで使用できるCPUユニット以外の拡張特殊ユニットです。
KV-8000/7000シリーズ 拡張ユニット	KV-8000/7000シリーズで使用できるCPUユニット以外の拡張入出力ユニット、拡張特殊ユニットです。
KV-5000/3000/1000シリーズ 拡張ユニット	KV-5000/3000/1000シリーズで使用できるCPU ユニット以外の拡張入出力ユニット、拡張特殊ユニットです。
《KV STUDIO》	KVシリーズのプログラム作成をサポートするソフトウェアです。
ラダープログラム	《KV STUDIO》で作成されたプログラムです。
命令語リファレンスマニュアル	KV-8000/7000/5000/3000/1000･KV Nanoシリーズ命令語リファレンスマニュアルです。

記号について

このマニュアルの説明では、メニューやボタンを次のような記号であらわしています。

用　語	説　明
16#	「16#」を先頭に表記している値は、16進数を表わしています。
[DM0･DM1]	32ビットの数値を扱うときは、デバイスを2点使用します。プログラム上では、下位16ビットを格納しているデバイス番号（左の例ではDM0）だけを記述します。特に気を付けていただきたいときには、左の例のように記述して、32ビットのデータを扱っていることを表わしています。
↑	「信号の立ち上がり（ONになったとき）」を表わしています。
⎍	「信号がON状態になっている間」を表わしています。
↓	「信号の立ち上がり（OFFになったとき）」を表わしています。

目次

はじめに
安全にお使いいただくために 1
用語について 3
記号について 3
目次 4

1章　概要

ST の種類 1-2
ボックス型ST 1-2
エリア型ST 1-3
ST の機能 1-4
代入文について 1-4
演算子について 1-4
制御文について 1-5
サフィックスについて 1-7
ST関数について 1-8

2章　STを作成する

ST を作成する 2-2
ST記述 2-2
ボックス型 ST 2-4
ボックス型STの特長 2-4
ボックス型STを作成する 2-4
ボックス型STを使用するときの注意 2-9
エリア型 ST 2-11
エリア型STの特長 2-11
エリア型STを作成する 2-11
エリア型STを作成する 2-12
変数プログラミングと配列・構造体 2-15
変数プログラミング 2-15
グローバル変数とローカル変数 2-15
変数とデータ型一覧 2-15
配列 2-16
構造体 2-17
変数の設定 2-20
変数使用時の注意事項 2-22
ST を記述するときの注意事項 2-23
エリア型STの並列接続について 2-23
STで演算式を記述するときの注意事項 2-23
ST 変換時のメッセージ 2-24
警告メッセージ 2-24
エラーメッセージ 2-24
ファンクションブロック 2-27
ファンクションブロックの呼び出し 2-27
ST のデバッグ機能 2-28
モニタ機能 2-28
入力補助機能・その他 2-30
入力補助機能について 2-30
コメント機能について 2-34
編集画面の注意事項とカスタマイズ 2-34
ST/スクリプト用ワークメモリ(ワークデバイス)について 2-35
XYM表記 2-35

3章　データの取り扱い

デバイス **3-2**
デバイス一覧 3-2
デバイスの表記方法 3-5
ビットデバイス 3-7
ワードデバイス 3-13
ビットデバイスのワード単位扱い 3-19
ワードデバイスのビットデバイス扱い 3-20
ダイレクト入出力リレー 3-21
変数 3-21
定数 3-22
インデックス修飾と間接指定 **3-25**
インデックス修飾 3-25
間接指定 3-27
インデックス修飾と間接指定を併用する 3-29
データの型（サフィックス） **3-31**
サフィックス 3-31
演算の型について 3-36

4章　STの文法と注意

代入文 **4-2**
代入文の使用方法 4-2
演算子 **4-4**
演算子一覧 4-4
制御文 **4-6**
制御文一覧 4-6
IF ～ 文(条件分岐 ①) 4-8
IF ～ ELSE文(条件分岐 ②) 4-8
IF ～ ELSIF ～ 文(条件分岐 ③) 4-9
CASE ～ 文(多分岐制御) 4-11
FOR ～ END_FOR文(指定回数の繰り返し制御) 4-12
WHILE文(前判定繰返し制御) 4-14
REPEAT文(後判定繰り返し制御) 4-14
EXIT処理 4-15
RETURN処理 4-16
制御文を使う時の注意 4-16
ST を記述するときの注意 **4-18**

5 章　基本関数

関数の読み方 **5-2**
BMOV 5-2
接点関数 **5-3**
LDP 5-3
LDF 5-3
LDPB 5-4
LDFB 5-5
BLD 5-5
ビット接点関数 **5-7**
BLDB 5-7
出力関数 **5-8**
SET 5-8
RES 5-8

ビット出力関数 5-10
BOUT 5-10
BOUB 5-10
BSET 5-11
BRES 5-12
タイマ・カウンタ関数 5-13
TMR 5-13
TMH 5-14
TMS 5-15
TMU 5-16
CNT 5-16
CURVAL 5-17
ダイレクト入出力関数 5-18
RFSX 5-18
RFSY 5-18

6 章　応用関数

フロー関数 6-2
CALL 6-2
ECALL 6-2
ファンクションブロック関数 6-4
FBSTRT 6-4
FEND 6-5
MDSTRT 6-5
MDSTOP 6-6
インデックスレジスタ関数 6-7
ZPUSH 6-7
ZPOP 6-8
間接指定関数 6-9
ADRSET 6-9
ADRINC 6-9
ADRDEC 6-10
ADRADD 6-11
ADRSUB 6-12
ファイルレジスタ関数 6-13
FRSET 6-13
FRSTM 6-13
FRLDM 6-14
ユニットプログラム関数 6-15
UPSTRT 6-15
UPSTOP 6-16
UFSUS 6-17
UFRSM 6-18
UMALLOC 6-19
バッファメモリ関数 6-21
UREAD 6-21
UWRIT 6-21
UFILL 6-22
ユニット種別判定関数 6-23
IS_ □□□ 6-23
配列要素数取得関数 6-24
ACNT 6-24
ワードサイズ取得関数 6-25
WSIZE 6-25

7 章　演算関数

データ転送関数 7-2

BMOV 7-2
FMOV 7-2
PMOV 7-3
BYBMOV 7-4
BYLMOV 7-5

算術関数 7-6

INC 7-6
DEC 7-7
ROOT 7-7
SQRT 7-8

論理演算関数 7-9

ANDA 7-9
ORA 7-9
EORA 7-10
ENRA 7-11
COM 7-12
NEG 7-12
ABS 7-13

データシフト関数 7-14

SHR 7-14
SHL 7-14
ASRA 7-15
ASLA 7-16
RRA 7-16
RLA 7-17
ROR 7-17
ROL 7-18
WSR 7-19
WSL 7-19
BSR 7-20
BSL 7-21

データ制御関数 7-22

LIMIT 7-22
BANDC 7-22
ZONE 7-23
APR 7-24
RAMP 7-25
TPOUT 7-26
LLFLT 7-28

データ変換関数 7-29

TBCD 7-29
TBIN 7-29
MPX 7-30
DMX 7-30
GRY 7-31
RGRY 7-32
DISN 7-32
UNIN 7-33
DISB 7-34
UNIB 7-34
SWAP 7-35
BSWAP 7-35
DECO 7-36
ENCO 7-37
CPMSET 7-37
CPMGET 7-38
*_TO_** 7-39
TO_** 7-40
*_TRUNC_** 7-40
TRUNC_** 7-41
*_TO_BCD_** 7-42
TO_BCD_** 7-42
*_BCD_TO_** 7-43
BCD_TO_** 7-43

浮動小数点関数 7-44

FLOAT 7-44

DFLOAT 7-44
INTG 7-45
INT 7-46
DISF 7-46
UNIF 7-47
EXP 7-48
EXPT 7-48
LN 7-49
LOG 7-50
RAD 7-50
DEG 7-51
SIN 7-52
COS 7-53
TAN 7-54
ASIN 7-55
ACOS 7-55
ATAN 7-56
ATAN2 7-57

文字処理関数 7-59

ASC 7-59
RASC 7-59
DASC 7-60
STR 7-61
RDASC 7-62
HASC 7-63
RHASC 7-64
FASC 7-65
RFASC 7-66
VAL 7-67
LEN 7-67
SRGHT 7-68
RIGHT 7-69
SLEFT 7-69
LEFT 7-70
SMID 7-71
MID 7-72
SRPLC 7-72
REPLACE 7-73
SINS 7-74
INSERT 7-75
SDEL 7-75
DELETE 7-76
TRIM 7-77
LTRIM 7-78
RTRIM 7-78
STRIM 7-79
SPLIT 7-80
SFIND 7-81
SFINDN 7-82
INSTR 7-82
FIND 7-83
DISS 7-84
UNIS 7-85
CHR 7-85
CPSASC 7-86
RCPSASC 7-86
CONCAT 7-87

8 章　拡張関数

データ処理関数 8-2

SEG 8-2
BCNT 8-3
DCNT 8-3
SER 8-4
DSER 8-5
MAX 8-5
MAX_RANGE 8-6
MIN 8-7

MIN_RANGE 8-7
SEL 8-8
MUX 8-8
AVG 8-9
WSUM 8-10
BSUM 8-10
CRC 8-11
ZRES 8-12
BCMP 8-13
BCMPI 8-14
RND 8-14
SORT 8-15
SORTN 8-16
テーブル処理関数 8-17
FIFOW 8-17
FIFOR 8-17
LIFOW 8-18
LIFOR 8-19
FWRIT 8-19
FINS 8-20
FDEL 8-21
時計処理関数 8-22
WTIME 8-22
SEC 8-23
RSEC 8-23
AJST 8-24
高速処理関数 8-25
DI 8-25
EI 8-25
DIC 8-26
IEDGE 8-26
PID 関数 8-28
PIDAT 8-28
ログ関数 8-29
LOGE 8-29
LOGD 8-29
TRGD 8-30
TRGR 8-30
ストレージ関数 8-32
MWRIT 8-32
MREAD 8-33
MFREE 8-34
MFREEK 8-35
MMKDIR 8-36
MRMDIR 8-37
MDEL 8-38
MPRINT 8-39
MREADL 8-40
MCOPY 8-41
MMOV 8-42
MREN 8-43
MSTAT 8-43
アクセスウィンドウ関数 8-45
AWSHOW 8-45
AWHIDE 8-45
フリーランカウンタ関数 8-47
RFSFRC 8-47
センサ設定関数 8-48
SPRD 8-48
SPWR 8-49
SSVC 8-50
サイクリック通信リフレッシュ関数 8-52
RFSCI 8-52

RFSCO 8-52

付章　付 録

CR/CM 一覧 ..... 付 -2
コントロールリレー CR 付-2
コントロールメモリ CM 付-4
ASCII コード表 ..... 付 -9
使用できない文字一覧 ..... 付 -10
使用できない関数一覧 ..... 付 -12
標準関数として対応している関数一覧 ..... 付 -13
索引 ..... 付 -14
関数索引 ..... 付 -17

1

概要

STの種類 ····· 1-2
STの機能 ····· 1-4

STの種類

ST言語には、記述スタイルの異なる2種類のSTがあります。

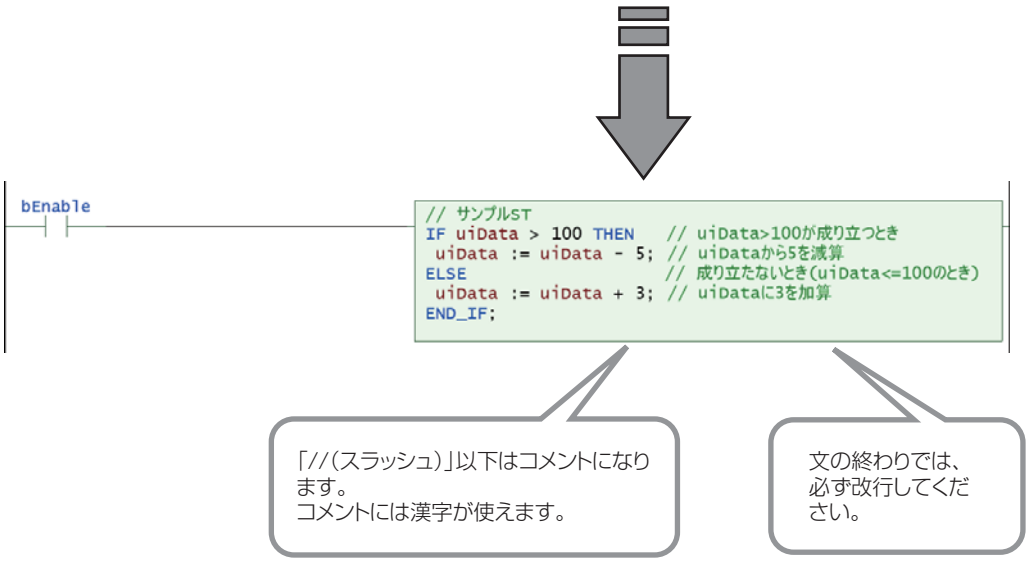
ボックス型ST

ボックス型STは、実行条件をラダーで記述できるSTです。
編集画面の記述範囲は、1行の第5セルから第10セルまでの6セル内になります。

ST記述

```
// サンプルST
IF uiData > 100 THEN          // uiData>100が成り立つとき
    uiData := uiData - 5;     // uiDataから5を減算
ELSE                          // 成り立たないとき(uiData<=100のとき)
    uiData := uiData + 3;     // uiDataに3を加算
END_IF;
```

ボックス型ST



ポイント　**実行条件が満たされないとき、ボックス型ST内は実行されず、スキップされます(微分型命令など、一部の関数において注意が必要です)。**

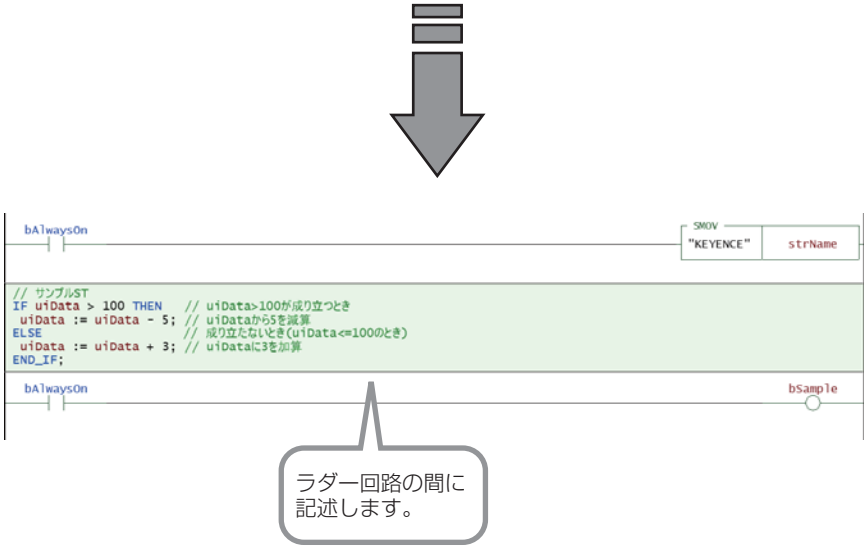
エリア型ST

エリア型STは、ラダーの実行条件を持たない、毎スキャン実行型STです。
編集画面上の記述範囲は、1行の全てを占有します。

ST記述

```
// サンプルST
IF uiData > 100 THEN          // uiData>100が成り立つとき
    uiData := uiData - 5;     // uiDataから5を減算
ELSE                          // 成り立たないとき(uiData<=100のとき)
    uiData := uiData + 3;     // uiDataに3を加算
END_IF;
```

エリア型ST



STの機能

STでは、デバイスや変数に値や演算結果を格納する処理を、代入文を使って記述します。

代入文について

STでは、デバイスや変数に値や演算結果を格納する処理を、代入文を使って記述します。

■代入文

uiValue1 := uiValue2 + 10;

":="の左辺に、右辺の結果(値)を格納します。

●数値代入

定数1000をuiValueに格納します。

ST記述

```
uiValue := 1000;
```

●文字列代入

文字列'KEYENCE'をstrNameに格納します（文字列の終わりを示すNUL(00H)は自動的に加えられます）。

ST記述

```
strName := 'KEYENCE';
```

●BOOL代入

BOOL変数bValue1の状態を、bValue2に反映します。

ST記述

```
bValue2 := bValue1;
```

演算子について

STでは、算術演算や論理演算などをラダープログラムのような専用の命令語に置きかえる必要がなく、演算子をそのまま記述できるので、より直感的なプログラミングが可能です。

●演算例

uiValue1の値を10倍し、uiValue2を加えた結果を2倍します。演算結果をuiValue3に代入（格納）します。

ST記述

```
uiValue3 := 2 * (uiValue1 * 10 + uiValue2);
```

●演算子一覧

記　号		処理内容	記述例
算術演算子	**	累乗を求めます	uiValue1 := uiValue2 ** 2;
	*	2つの値の積を求めます（乗算）	uiValue1 := uiValue2 * 3;
	/	2つの値の商を求めます（除算）	uiValue1 := uiValue2 / 4;
	MOD	2つの値の除算を行い、その剰余を返します	uiValue1 := uiValue2 MOD 5;
	+	2つの値の和を求めます（加算）	uiValue1 := uiValue2 + 6;
	-	2つの値の差を求めます（減算）	uiValue1 := uiValue2 - 7;
比較演算子	<	より小さい	bFlag := uiValue < 10;
	<=	小さいか等しい（以下）	bFlag := uiValue <= 20;
	>	より大きい	bFlag := uiValue > 30;
	>=	大きいか等しい（以上）	bFlag := uiValue >= 40;
	=	等しい（等価）	bFlag := uiValue = 50;
	<>	等しくない（不等価）	bFlag := uiValue <> 60;
論理演算子	NOT	値の論理否定を求めます	bFlag1 := NOT bFlag2;
	AND, &	2つの値（ビットデータ同士、ワードデータ同士）の論理積を求めます（かつ）	bFlag1 := bFlag2 AND bFlag3;
	OR	2つの値（ビットデータ同士、ワードデータ同士）の論理和を求めます（または）	bFlag1 := bFlag2 OR bFlag3;
	XOR	2つの値（ビットデータ同士、ワードデータ同士）の排他的論理和を求めます	bFlag1 := bFlag2 XOR bFlag3;

演算子の詳細については、「演算子」4-4ページを参照してください。

制御文について

STでは、さまざまな制御文を記述できます。制御文を利用することにより、簡単にプログラムの流れを変える（制御）ことができます。制御文には、条件により処理を分岐させる条件分岐文と、繰り返し処理をおこなわせる繰り返し制御文があります。

■条件分岐制御文

条件によって実行する処理を選択します。

●条件分岐文の種類

制御文	基本書式	説明
IF文	IF␣<条件式>␣THEN 　<処理文> END_IF;	<条件式>の結果により、処理を実行する/しないを決める分岐制御文
	IF␣<条件式>␣THEN 　<処理文1>; ELSE␣ 　<処理文2>; END_IF;	<条件式>の結果に従って別々の処理を実行する2分岐制御文
	IF␣<条件式1>␣THEN 　<処理文1> ELSIF␣<条件式2>␣THEN 　<処理文2> 　・ 　・ ELSE 　<処理文n> END_IF;	複数の条件式（<条件式1>、<条件式2>・・<条件式n>）の結果に従って別々の処理を実行する多分岐制御文
CASE文	CASE␣<ワードデータ>␣OF 　<比較値1>、<比較値2>:<処理文1> 　<比較値3>:<処理文2> ELSE 　<処理文3> END_CASE;	<ワードデータ>の値に従い、対応する処理を選択して実行する多分岐制御文

※　␣は半角スペースの挿入を表します。

■繰り返し制御文

指定された条件で処理を繰り返します。

●繰り返し制御文の種類

制御文	基本書式	説明
FOR文	FOR␣<ワードデータ>:=<初期値> 　　␣TO␣<最終値>␣STEP␣<増分値> ␣DO 　　<処理文> END_FOR;	<ワードデータ>の値が<最終値>に到達するまで処理を繰り返す制御文
WHILE文	WHILE␣<条件式> ␣DO 　　<処理文> END␣WHILE;	条件式が成立している（真である）間、(不定回数)処理を繰り返す制御文
REPEAT文	REPEAT 　　<処理文> UNTIL␣<条件式> END_REPEAT;	条件式が成立しない（偽である）間、(不定回数)処理を繰り返す制御文
EXIT	<繰り返し条件文始まり> 　　EXIT; <繰り返し条件文終わり>;	繰り返し制御文の中で、条件とは関係なく繰り返し制御から抜ける (FOR文、WHILE文、REPEAT文の処理文の中でのみ使用)

※　␣は半角スペースの挿入を表します。

※制御文の詳細については，「制御文」4-6ページを参照してください。

■プログラム実行終了文

記述されているプログラムの実行を終了します。

●繰り返し制御文の種類

制御文	基本書式	説明
RETURN文	<処理文>; RETURN; <処理文>;	記述されているプログラムの実行を終了します。

サフィックスについて

ラダー言語では各命令語にサフィックスを付けることで、命令の動作の型が決まるのに対して、ST言語ではデバイスにサフィックスを付けることで、関数の動作の型や、デバイスに格納されている値の意味が決まります。データの型（サフィックス）の詳細については、「データの型（サフィックス）」（3-31ページ）を参照してください。

●サフィックスの省略

デバイスのサフィックスを省略して記述した場合、デバイスの種別により扱われる型が決まります。

デバイスの種類	サフィックスを省略した場合の型（デフォルトの型）
R(DR),B,MR,LR,T,C,CTC,CR,UR	.B（ビット型）
DM,W,EM,ZF,FM,TM,CM,V,P,UM,UV, 間接指定(*)するデバイス	.U（16ビット符号無しデータ型）
Z	.L（32ビット符号付きデータ型）

●変数のサフィックス

グローバル変数、ローカル変数を記述した場合、変数登録時に選択する「データ型」により扱われる型が決まります。

変数のデータ型		対応するデータ型
BOOL	ビット	.B
INT	1ワード符号あり整数	.S
DINT	2ワード符号あり整数	.L
UINT	1ワード符号なし整数	.U
UDINT	2ワード符号なし整数	.D
REAL	単精度浮動小数点型実数	.F
LREAL	倍精度浮動小数点型実数	.DF
STRING	文字列	.T
TIMER	タイマ	.B
COUNTER	カウンタ	.B
ARRAY		変数登録時に指定するデータ型のデフォルトの型
構造体型		構造体定義で設定した各メンバのデータ型のデフォルトの型
ファンクションブロック型		引数設定で設定した各引数のデータ型のデフォルトの型

ST関数について

ST関数は、KV-8000シリーズのラダー命令語をSTから利用できるようにしたものです。
ラダー命令語と区別するため、STで使用する命令語は「ST関数」と呼びます。

●ST関数の種類

ST関数は、大きく分けて以下の種類に分けられます。

種　類	説　明
演算処理関数	ワードデータやビットデータの演算をおこなう関数群です。論理演算をおこなう ANDA･ORAや、データ転送をおこなうBMOV･FMOV、浮動小数点型演算をおこなう SIN･COSなどがあります。
文字列処理関数	文字列処理をおこなうための関数群です。文字列長を検出するLENや、文字列取り出しをおこなうSMID･SLEFTなどがあります。
データ処理関数	データ処理をおこなうための関数群です。データの最大値を求めるMAXや、最小値を求めるMIN、指定ワードデータ内のONしているビット数を数えるBCNTなどがあります。
その他の関数	演算、文字列、データ処理以外の処理をおこなう関数群です。タイマを操作するTMRや、間接指定をおこなうADRSET、メモリカード制御をおこなうMWRITやMREADなどの関数があります。

■演算フラグについて（CR2009～CR2012）

各STはラダープログラムに変換され、関数を実行したとき演算フラグも変化します。そのため演算フラグの状態は、STプログラムの実行の前後で変化しています。
また、STを変換して生成されるラダープログラムは、関数の組み合わせなどにより色々な記述ができるため、STプログラムによって変化した演算フラグを利用することは基本的にできません。

■間接指定やインデックス修飾時の注意事項について

間接指定やインデックス修飾などを使用する場合、範囲外のデバイスやワーク領域を指定しないように注意してください。
もしも各関数の引数に対して、間接参照やインデックス修飾を使用して範囲外のデバイスやワーク領域にアクセスしようとした場合、それぞれ次のような動きになります。

●不正なアドレスを指定したときの各関数の動作一覧

関数の動作	関数名
戻り値に0を格納する	LDP, LDF, BLD, BLDB, LDPB, LDFB, UNIF, RDASC, RHASC, RFASC, VAL, LEN, SFIND, INSTR, FIND, SEG, MAX_RANGE, MIN_RANGE, AVG, WSUM, BSUM, BCMP, BCMPI, SEC
戻り値にNUL（00H）を格納する	DASC, STR, HASC, FASC, SRGHT, RIGHT, SLEFT, LEFT, SMID, MID, SRPLC, REPLACE, SINS, INSERT, SDEL, DELETE
戻り値に第1引数の値を格納する※1	ROOT, ANDA, ORA, EORA, ENRA, COM, NEG, SHR、SHL, RRA, RLA, ROR, ROL, TBCD, TBIN, MPX, DMX, GRY, RGRY, SWAP, DECO, ENCO, TO,USDLF, FLOAT, INTG, INT, EXP, LOG, RAD, DEG, SIN, COS, TAN, ASIN, ACOS, ATAN, ASC, RASC, DASC, CHR, BCNT, DCNT, SER, CRC, WTIME, AWNUM, AWMSG, WTIME, SQRT, ABS, RND
関数を実行しない	SET, RES, BOUT, BOUB, BSET, BRES, ZPUSH, ZPOP, ADRSET, ADRINC, ADRDEC, ADRADD, ADRSUB, BMOV, FMOV, PMOV, INC, DEC, WSR, WSL, BSR, BSL, DISF, DISS, UNIS, ZRES, FIFOW, FIFOR, LIFOW, LIFOR, FWRIT, FINS, FDEL, RSEC, DIC, LOGE, LOGD, MWRIT, MREAD, MFREE, MMKDIR, RFSX, RFSY, ECALL, MDSTRT, MDSTOP, FRSET, FRSTM, FRLDM, UREAD, UWRIT, UFILL, LIMIT, BANDC, ZONE, APR, DISN, DISB, UNIN, UNIB, DSER, RFSCTH, TRGD, MRMDIR, MDEL, RFSFRC、ACNT

※1　ただし、第1引数に不正な間接参照をした場合、戻り値の値は保証されません。

2

STを作成する

STを作成する ····· 2-2
ボックス型ST ····· 2-4
エリア型ST ····· 2-11
変数プログラミングと配列･構造体 ····· 2-15
STを記述するときの注意事項 ····· 2-23
ST変換時のメッセージ ····· 2-24
ファンクションブロック ····· 2-27
STのデバッグ機能 ····· 2-28
入力補助機能･その他 ····· 2-30

STを作成する

ボックス型ST/エリア型STの作成方法、メニューや編集画面について説明します。

ST記述

STの記述方法

ボックス型ST/エリア型STはラダー編集ウィンドウ上でラダープログラムに混在させた形で記述できます。
STは、ラダー編集ウィンドウ上でST記述エリアを確保してから、記述します。

①メニューから呼び出す

【ST/スクリプト(S)】→【エリア型ST挿入(R)】または【ボックス型ST挿入(T)】を選択する。

②右クリックメニューから呼び出す

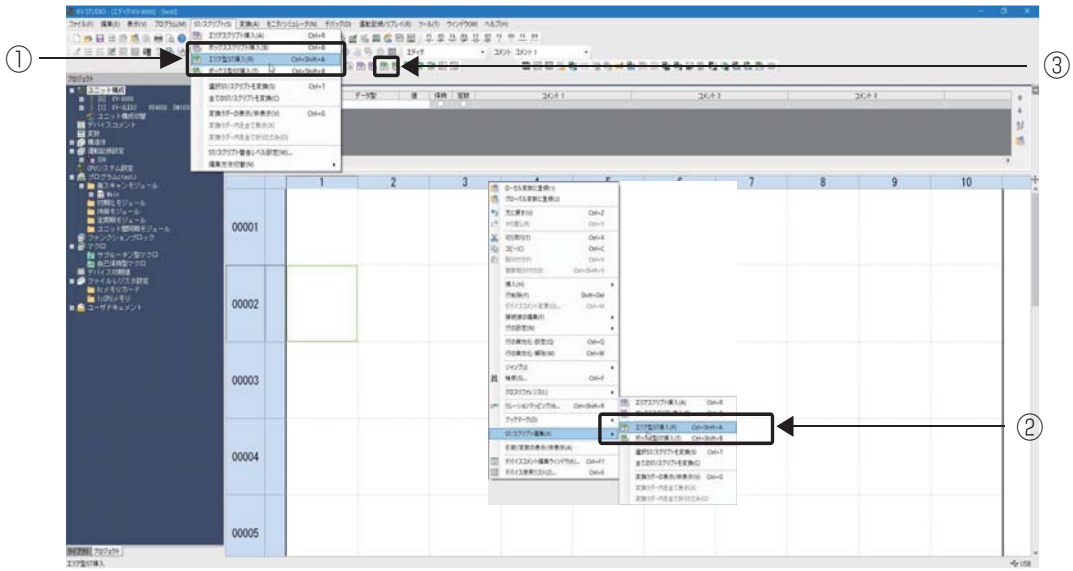
カーソルがラダー編集ウィンドウの上にあるとき、右クリックメニューから【ST/スクリプト編集(X)】を選択し、【エリア型ST挿入(R)】または【ボックス型ST挿入(T)】を選択する。

③アイコンから呼び出す

ツールバーのアイコンをクリックする。
・エリア型ST：
・ボックス型ST：

④ショートカットキーで呼び出す

・エリア型ST：Ctrl キー＋Shift キー＋A キー
・ボックス型ST：Ctrl キー Shift キー＋B キー

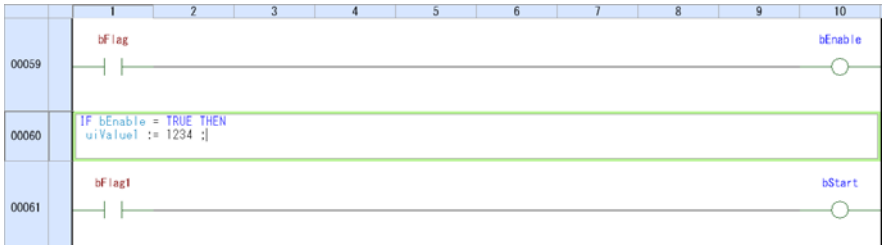


STの入力方法

STには2種類の入力方法があります。入力方法の違いによる機能の違いはありませんので、お好みにより選択してください。

■直接入力(デフォルト)

ラダープログラムの命令語と同じように、ラダーエディタ上に直接記述します。

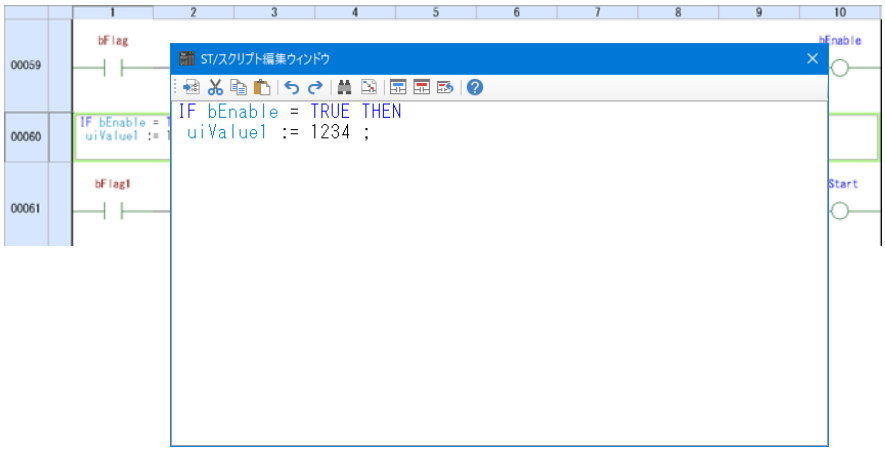


1 **メニューから【ST/スクリプト(S)】→【編集方法切替(N)】→【直接入力(D)】を選択します。**

別手順　メニューから【ツール(T)】→【オプション(O)】、「ST/スクリプト設定」タブの[入力形式]から「直接入力」を選択。

■ウィンドウ入力

ST記述時に表示されるテキストエディタウィンドウに入力します。



1 **メニューから【ST/スクリプト(S)】→【編集方法切替(N)】→【ウィンドウ(W)】を選択します。**

別手順　メニューから【ツール(T)】→【オプション(O)】、「ST/スクリプト設定」タブの[入力形式]から「直接入力」を選択。

ボックス型ST

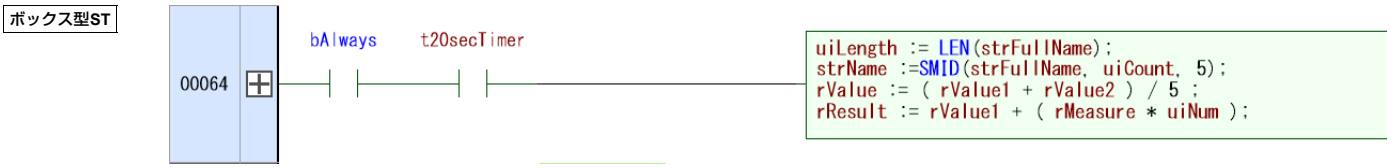
ここではボックス型STについての作成/編集方法、使用時の注意点について説明します。

ボックス型STの特長

ボックス型STは、STで記述されたプログラムをどのタイミングで実行するかをラダーで指定できる、ラダー実行条件付きSTです。

ボックス型STはこんなときに使う

ボックス型STは数行程度の演算式や複雑な制御文を含まないプログラムの記述に適しています。
また、実行条件をラダーで記述することで、ボックス型ST内のプログラムは実行処理文だけの記述が可能ですので、制御文を使う必要がなく、簡単に利用できます。

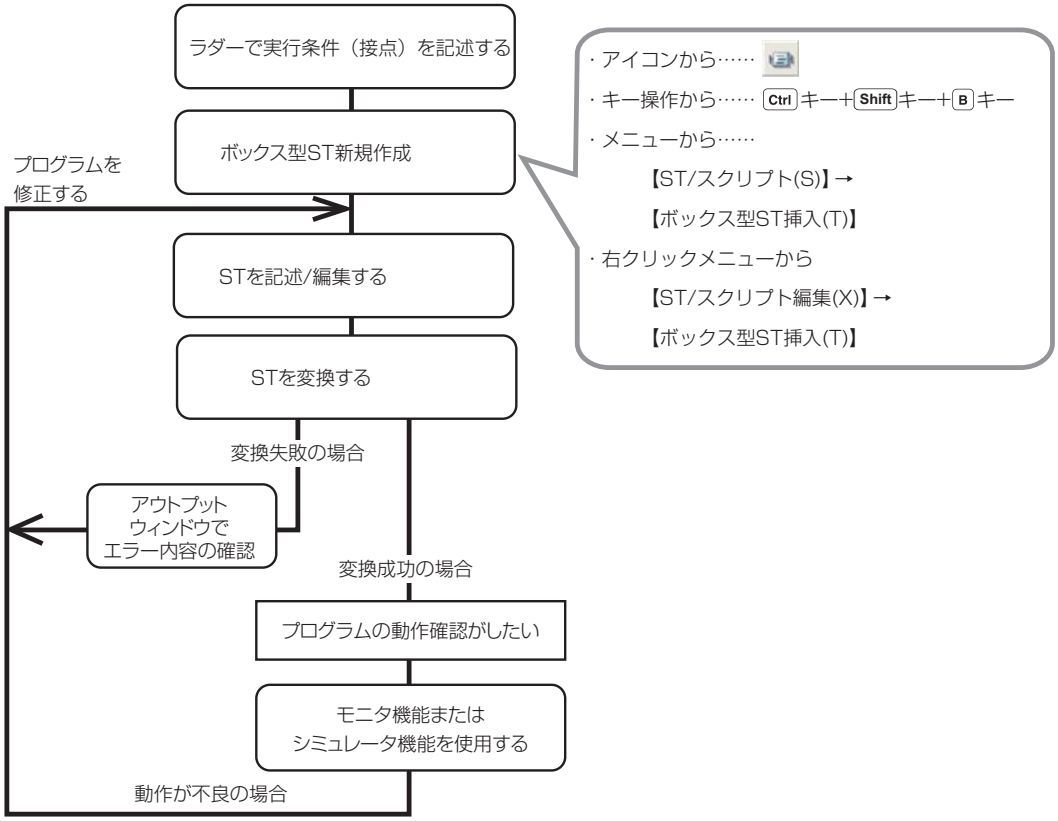


ボックス型STを作成する

ボックス型STの作成からデバッグまでの流れを説明します。

操作の流れ

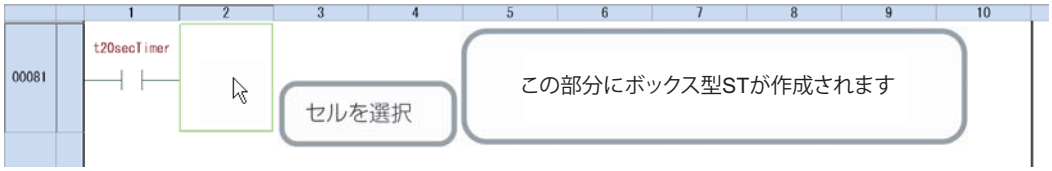
ボックス型STを作成/編集するときの、操作の流れについて説明します。



ボックス型STを新規作成する

ボックス型STを新規に作成する時の手順を説明します。

1 **ラダーエディタで実行条件を記述し、ボックス型STを作成したいセルをクリックします。**

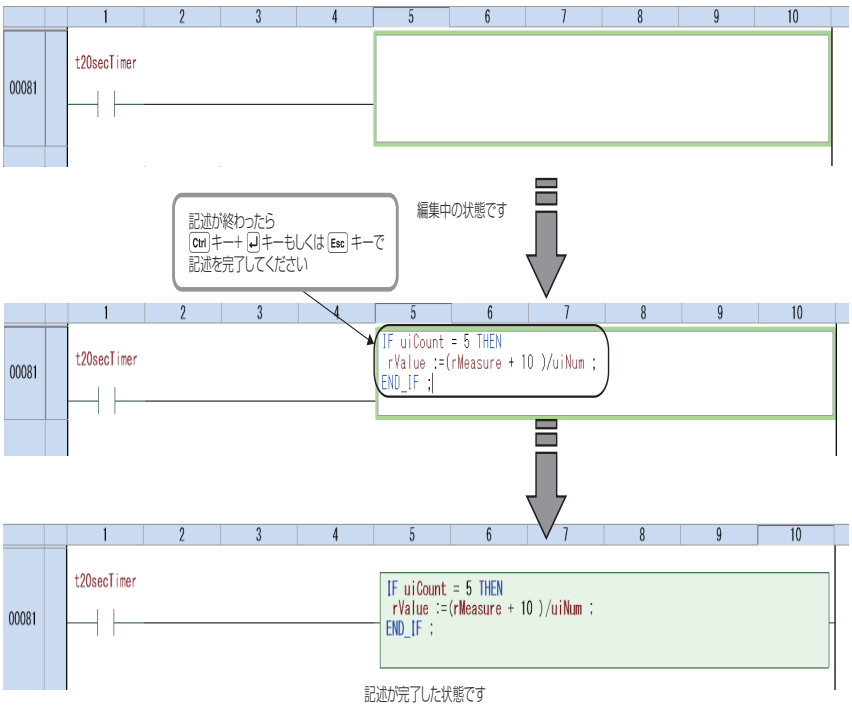


参 考
- ボックス型STは6セル(5～10)を占有します。実行条件は1～4セル目までに記述してください。
- 選択したセルがどの位置でも5～10セル部分にボックス型STを作成します。

2 **メニューから【ST/スクリプト(S)】→【ボックス型ST挿入(T)】を選択します。**

別手順
- Ctrl キー+Shift キー+B キー
- 右クリックメニューから【ST/スクリプト編集(X)】→【ボックス型ST挿入(T)】

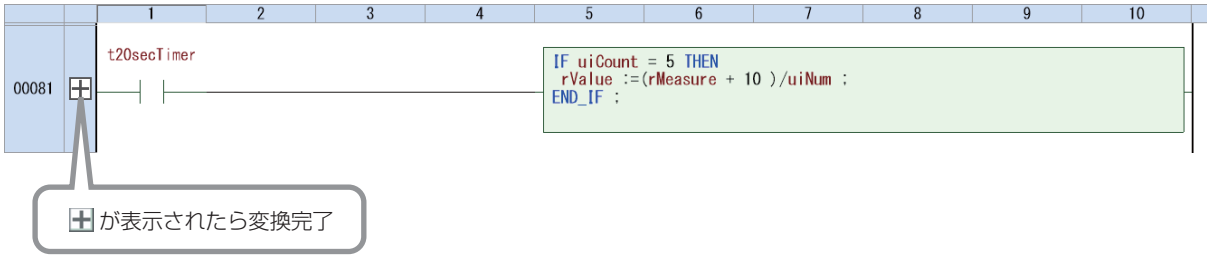
3 **ボックス型STが挿入されますので、プログラムを記述してください。**
記述が終わったら、Ctrl キー +↵ キーもしくは Esc キーを押すか、編集中のボックス型ST以外のラダー編集ウィンドウをクリックし、記述を完了してください。



4 記述が完了したらボックス型STを変換します。

別手順　右クリックメニューから【ST/スクリプト編集(X)】→【全てのST/スクリプトを変換(C)】

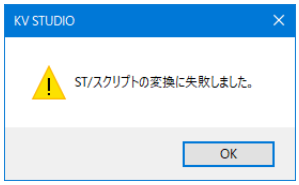
●正常に変換された場合



●変換できなかった場合

「ST変換エラー」が表示されます。
また、アウトプットウィンドウにエラーメッセージが表示されます。

「変換エラーが発生したときは」(2-24ページ)



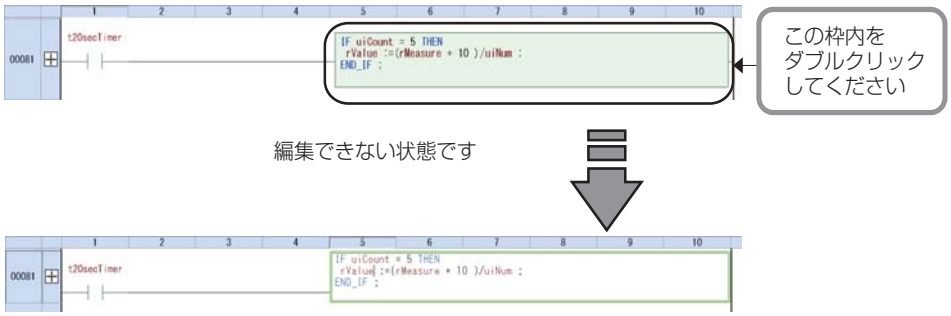
ボックス型STを再編集する

ボックス型STを再編集する場合の手順について説明します。

■ボックス型STを編集するには

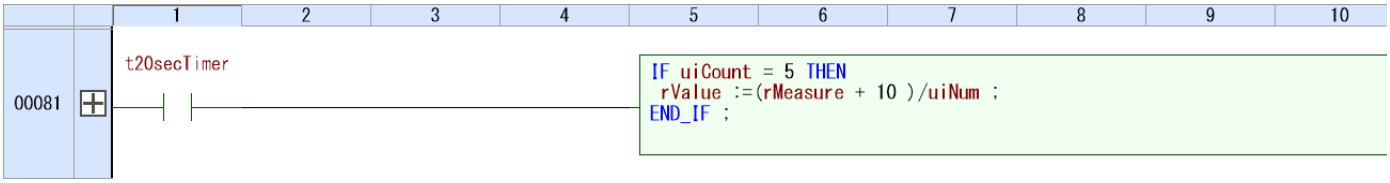
1 **すでに記述しているボックス型STを再編集する場合は、編集したいボックス型STをダブルクリックしてください。**

別手順　編集したいボックス型STを選択してから
- Esc キー
- F2 キー
- ↵ キー

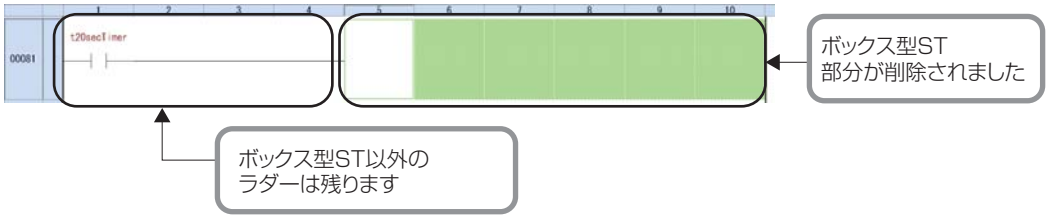


■ボックス型STを削除するには

1 **削除したいボックス型STを選択します。**

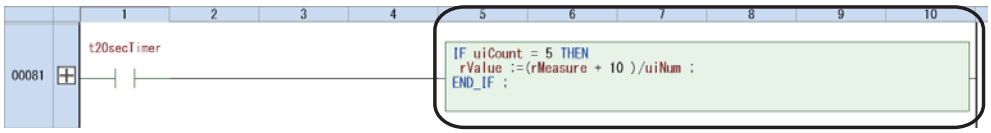


2 **キーボードの Delete キーを入力します。**



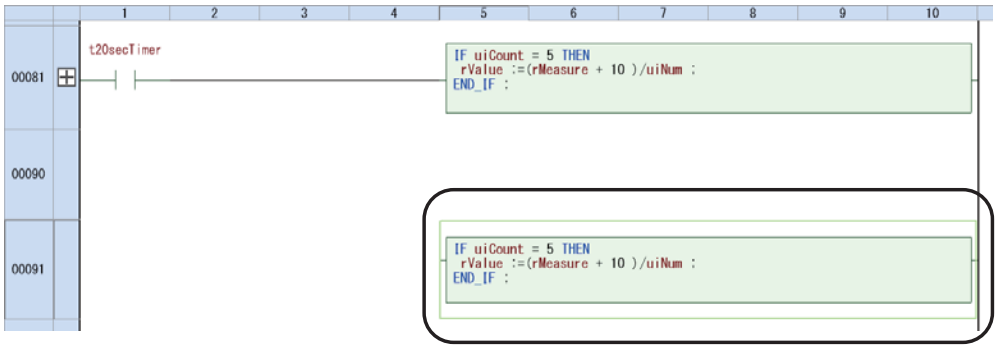
■ボックス型STをコピーするには

1 コピーしたいボックス型STを選択して、メニューから【編集(E)】→【コピー (C)】を選択します。



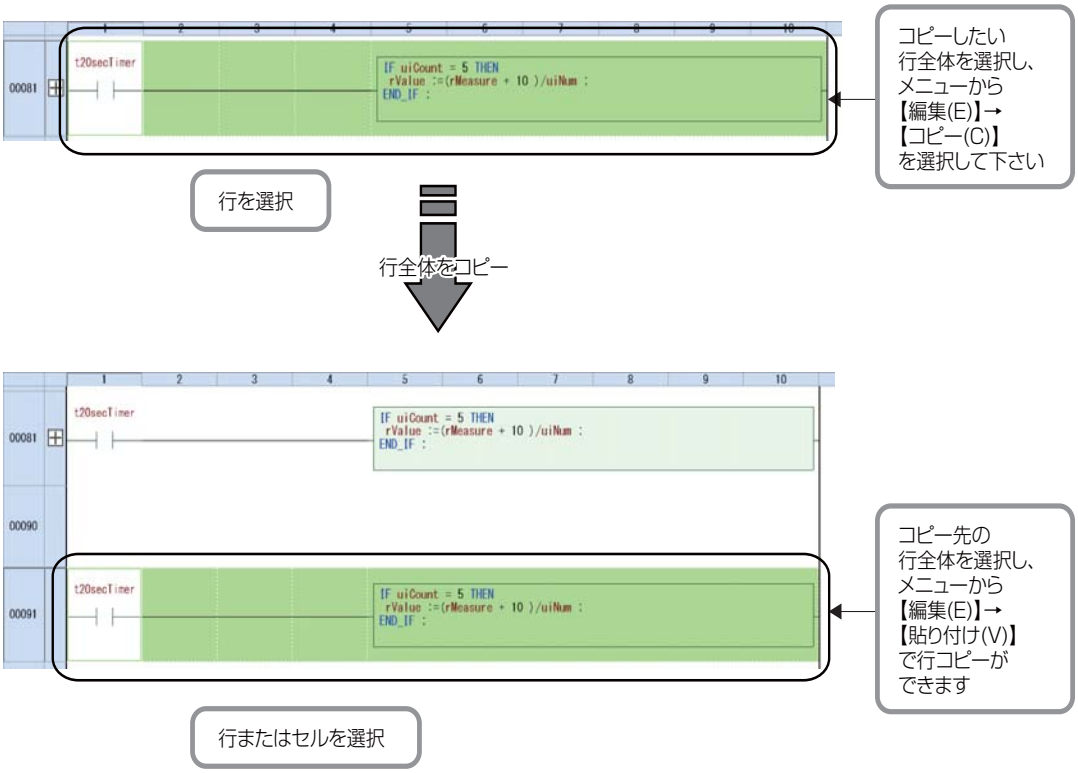
別手順
- Ctrl キー+ C キー
- 右クリックメニューから【コピー(C)】

2 手順1でコピーしたボックス型STを貼り付けるセル(行)を選択します。
メニューから【編集(E)】→【貼り付け(P)】を選択します。



別手順
- Ctrl キー+ V キー
- 右クリックメニューから【貼り付け(P)】

参 考　ボックス型STを含む行全体をコピーすることもできます。



ボックス型STを使用するときの注意

ボックス型STは実行条件がOFFのとき、STプログラムを実行しません。このため、以下のような場合に注意が必要です。

●微分実行型関数を使用する場合

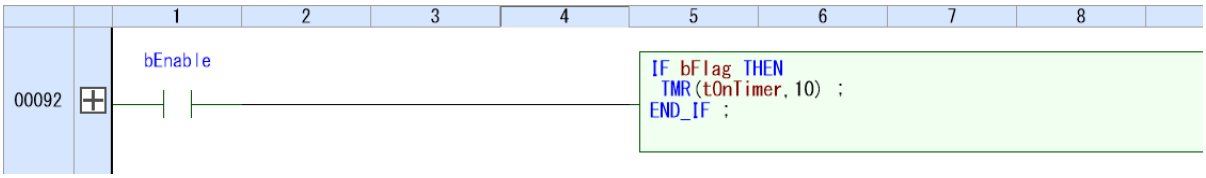
微分実行型関数の実行条件の前状態は前回ボックス型ST(サブルーチンプログラム)が実行されたときの状態になります。
n回目のボックス型ST実行と、n+1回目のボックス型ST実行が連続したスキャンで実行されない場合や、1スキャン内に2回実行されるとき注意が必要です。

詳細は、『命令語リファレンスマニュアル』「微分実行型命令を使用する時の注意」を参照してください。

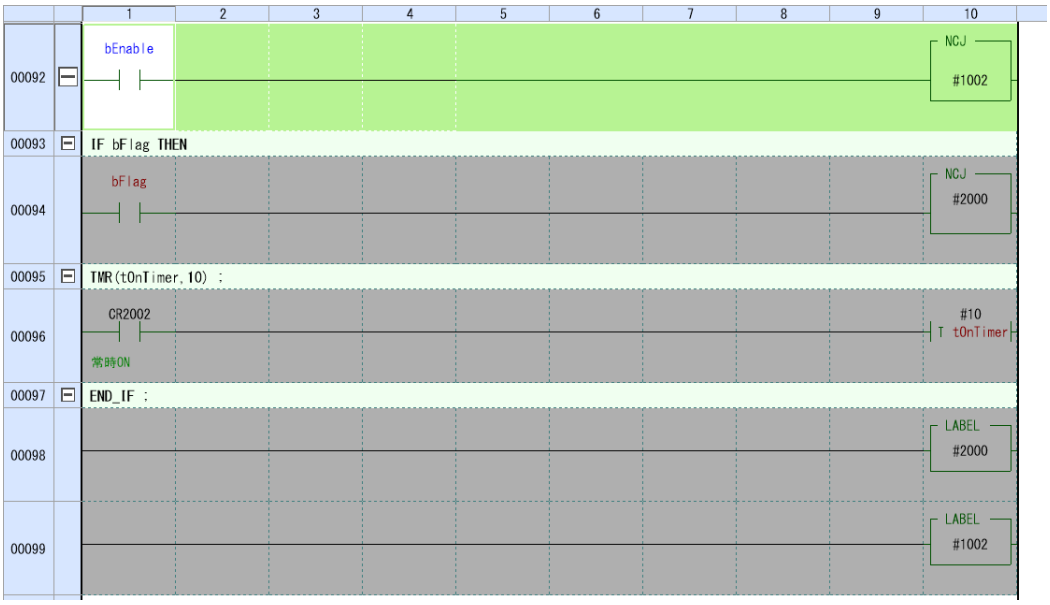
●タイマ関数を使用する場合

ボックス型STでタイマ関数を使用した場合、展開したラダープログラムでは、NCJ～LABEL間にTMR命令が記述されるため、タイマ接点が正しくON/OFFしないことがあるので注意してください。また、実行条件を省略したTMR関数も注意が必要です。
例えば、次のようなプログラムを記述した場合、

ボックス型ST



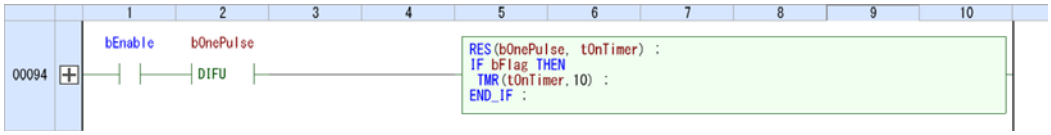
ラダー展開



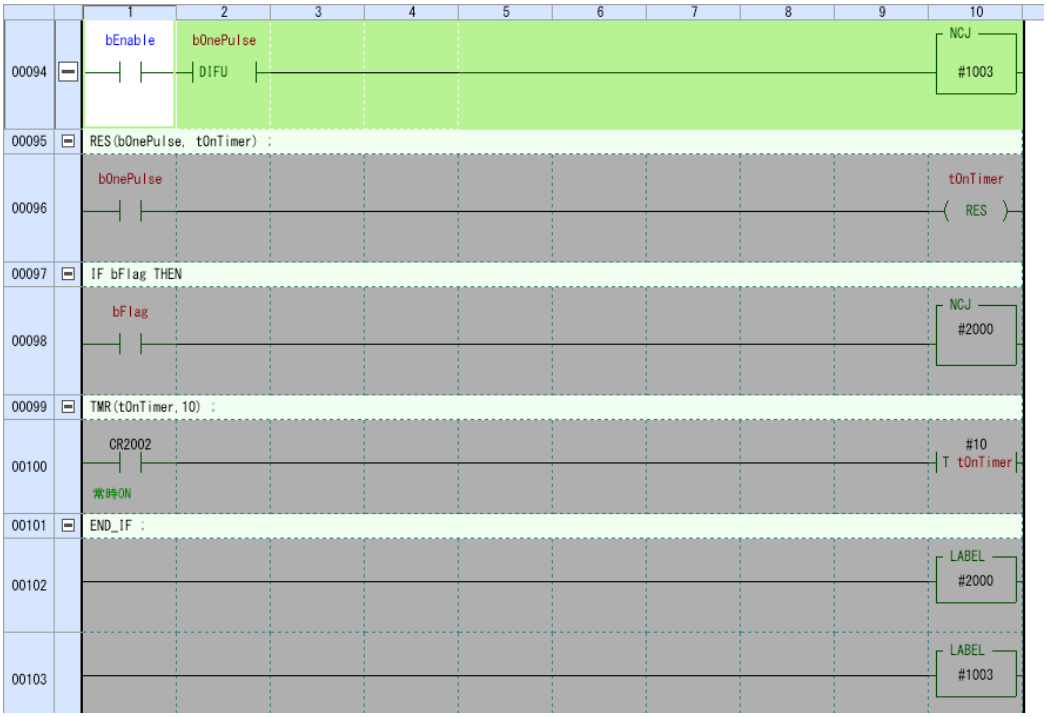
展開したラダープログラムを見ると、タイマ命令のリセット条件がCR2002(常時ON)になっているため、一度実行するとタイマをリセットできません。
この問題を解決するためには、「実行条件としてDIFU命令をAND接続する方法」があります。

実行条件としてDIFU命令をAND接続する方法

ボックス型ST



ラダー展開



ボックス型STの実行条件(MR1000)の立ち上がり時、DIFU命令によりMR1100が1スキャンだけONします。このとき、ボックス型STの1行目に記述されたRES関数がタイマをリセットするので、タイマの前状態にかかわらず、常にボックス型STの実行開始直後はタイマがクリアされた状態になるので、タイマ関数(接点)は正しく動作します。

エリア型ST

ここではエリア型STについての作成/編集方法、使用時の注意点について説明します。

エリア型STの特長

エリア型STは、行のすべてをSTで記述します。

エリア型STはこんなときに使う

エリア型STは毎スキャン実行型なので記述内容は、条件分岐や繰り返し制御を含んだ記述に適しています。
複雑な条件分岐や多数分岐を記述する場合に、プログラムをコンパクトにまとめることができます。

エリア型ST

※　//(スラッシュ)より右側の記述はコメントとして処理されます。

参 考　(* と *)で囲まれた領域はコメントとして処理されます。間接参照(*EM0など)を使用する場合、(と * の間を1文字空けて使用してください。

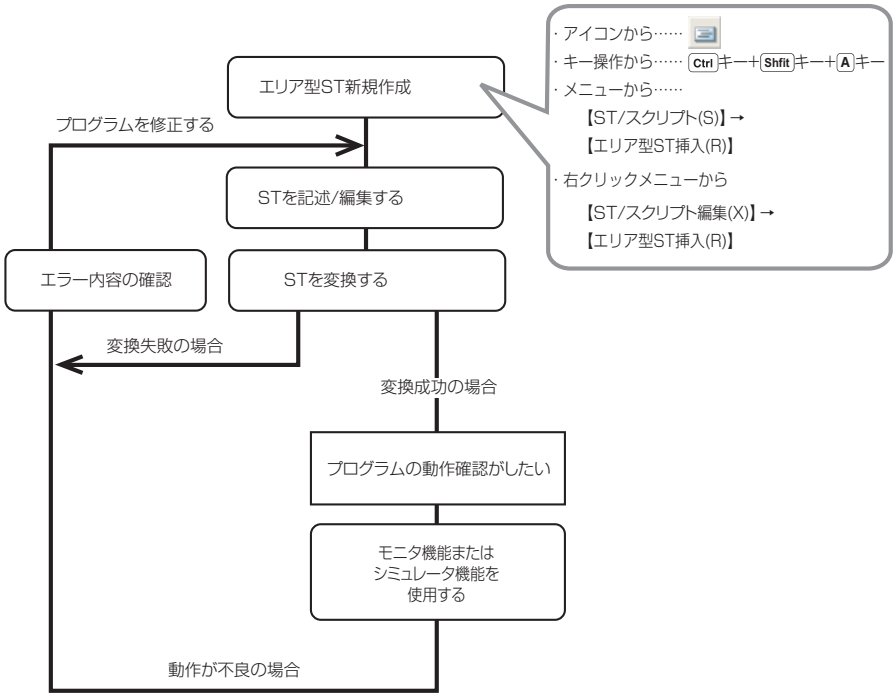
条件分岐が複数ある場合や、ラダーで記述すると長くなる処理などは、エリア型STを使用すると作業効率が上がります。

エリア型STを作成する

エリア型STの作成からデバッグまでの流れを説明します。

エリア型STはこんなときに使う

エリア型STを作成/編集するときの、操作の流れについて説明します。

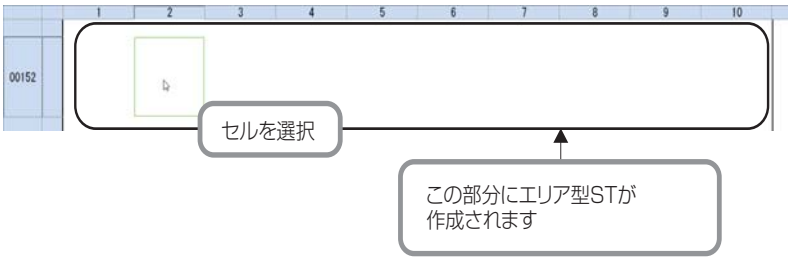


エリア型STを新規作成する

エリア型STを作成する

エリア型STを新規に作成するときの手順を説明します。

1 **エリア型STを作成したいセル(行)をクリックします。**



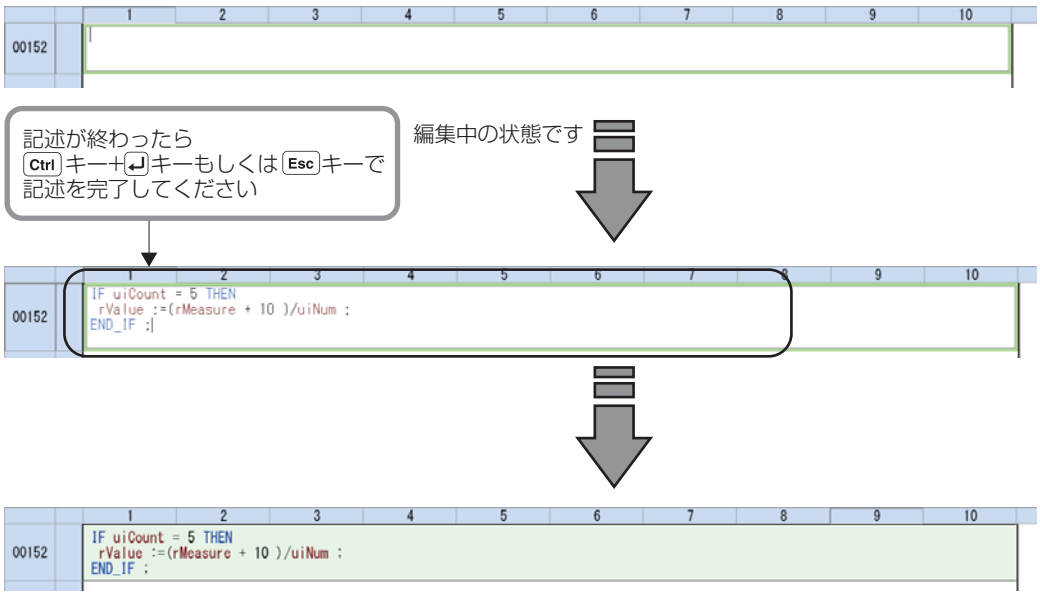
参考　エリア型STは1行(10セル)全てを占有します。

2 **メニューから【ST/スクリプト(S)】→【エリア型ST挿入(R)】を選択します。**

※エリア型STはメニューバーとツールバーとキーボード上から挿入できます。

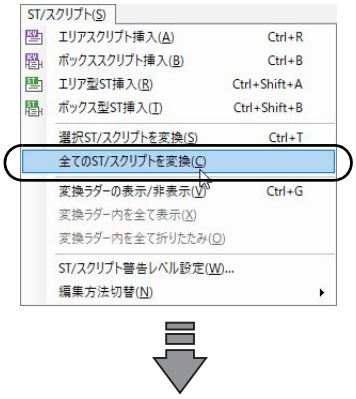
別手順
- Ctrl + Shift + A を押します。
- 右クリックメニューから【ST/スクリプト編集(X)】→【エリア型ST挿入(R)】

3 **エリア型STが挿入されますので、プログラムを記述してください。**
記述が終わったら、Ctrl キー + ↵ キーもしくは Esc キーを押すか、編集中のエリア型ST以外のラダー編集ウィンドウをクリックしてください。

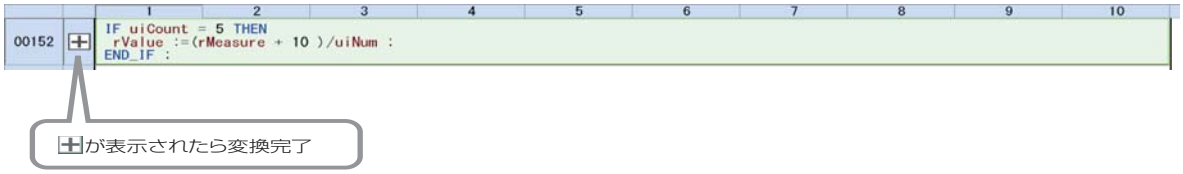


記述が完了した状態です

4 記述を確定したらエリア型STを変換します。



●正常に変換された場合



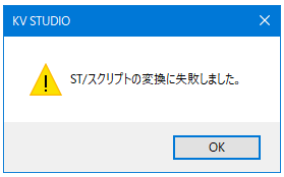
アウトプットウィンドウに「警告」メッセージが表示される場合もあります。

「警告メッセージ」(2-24ページ)

●変換できなかった場合

「ST変換エラー」が表示されます。
また、アウトプットウィンドウにエラーメッセージが表示されます。

「変換エラーが発生したときは」(2-24ページ)



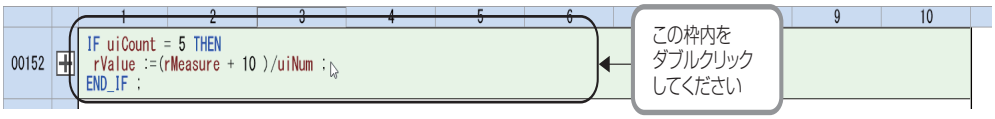
エリア型STを再編集する

エリア型STを再編集する場合の手順について説明します。

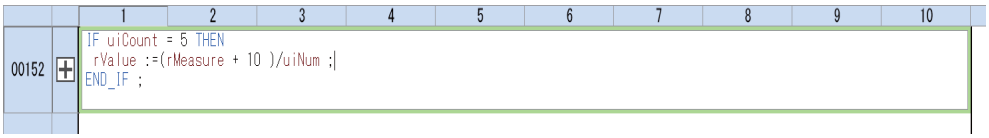
■エリア型STを編集するには

1 すでに記述しているエリア型STを再編集する場合は、編集をおこないたいエリア型STをダブルクリックしてください。

別手順　編集したいエリア型STを選択してから
- Esc キー
- F2 キー



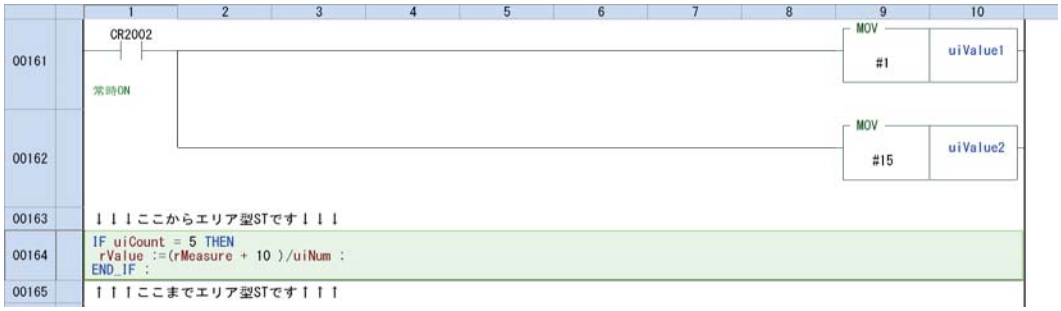
編集できない状態です



編集可能になりました

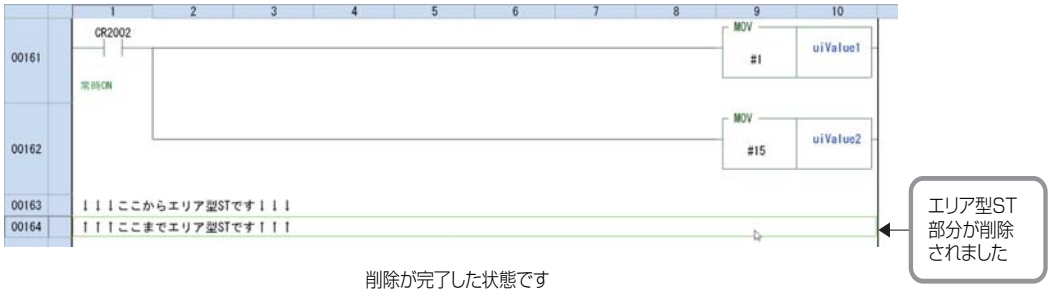
■エリア型STを削除するには

1 削除したいエリア型STを選択します。



エリア型STが選択されている状態です

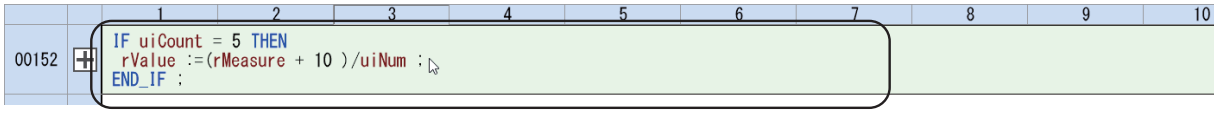
2 キーボードの Delete キーを入力します



削除が完了した状態です

■エリア型STをコピーするには

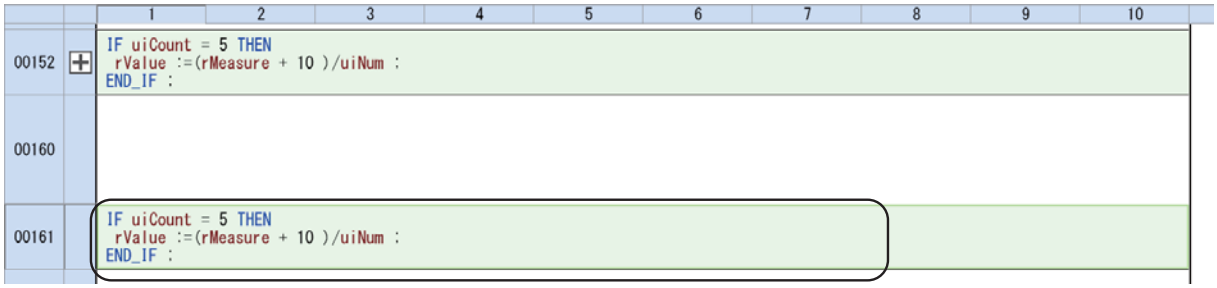
1 コピーしたいエリア型STを選択して、メニューから【編集(E)】→【コピー (C)】を選択します。



エリア型STが選択されている状態です

別手順
- Esc キー+ C キー
- 右クリックメニューから【コピー(C)】

2 手順1でコピーしたエリア型STを選択したセル(行)に貼り付けます。
メニューから【編集(E)】→【貼り付け(P)】を選択します。



コピーが完了した状態です

別手順
- Esc キー+ V キー
- 右クリックメニューから【貼り付け(P)】

変数プログラミングと配列・構造体

変数を使用したプログラミングの方法や注意事項と、配列･構造体の使用方法を説明します。

変数プログラミング

変数とは、データ型と名前が定義された、プログラミングで使用できる要素のことを言います。KV STUDIOでは、デバイスの他に変数を用いてプログラミングできます。変数を用いたプログラミングには以下のような特長があります。

- あらかじめ定義した名前を用いてプログラミングできるため、プログラムの可読性が向上します。
- 変数は、定義した変数に合わせて必要なサイズがCPUユニットのワーク領域に自動的に割り付けられます。そのため、割り付けを意識する必要がありません。
- あらかじめデバイスマップを作成しておく必要がなくなる、デバッグ効率が向上するなど、プログラムの作成工数が短くなります。
- 異なるプロジェクト間でプログラムをコピーした場合も、変数の割り付けは自動で行われます。デバイスのように、デバイスマップを変更してコピーする必要がなく、プログラムの流用性が向上します。
- タッチパネルディスプレイVT5シリーズも、KV-8000シリーズの変数に対応しているため、デバイスのようにPLCのデバイス番号を変更した際にタッチパネル側のデバイス番号もあわせて変更する、といった作業が不要になります。

グローバル変数とローカル変数

変数にはグローバル変数とローカル変数の2種類があります。それぞれの特徴を以下に示します。

内　容	グローバル変数	ローカル変数
実体	プロジェクト全体で同じ実体を共通して扱います。異なるプログラムで同じグローバル変数名を記述した場合は同じ実体を参照します。	プログラム内だけで使用できる変数です。異なるプログラムで同じローカル変数名を記述した場合でも異なる実体を参照します。
割付	デバイスに割り付ける手動割付と、ワークメモリにCPUユニットが自動で割り付ける自動割付があります。	ワークメモリにCPUユニットが自動で割り付けます。デバイスには割り付けられません。
用途	タッチパネルなど外部機器との通信領域に使用できます。また、拡張ユニットのデバイスを割り付けておくと、動作許可などの拡張ユニットのデバイスを変数でプログラミングできます。	同じ変数名でもプログラムごとに競合しないため、演算のための領域として使用できます。

変数とデータ型一覧

変数を使用する場合、あらかじめデータ型を定義する必要があります。CPUユニットは定義されたデータ型に応じて必要なデータサイズを自動的にワーク領域に割り付けます※1。
≪KV-8000≫で使用できるワーク領域のサイズと、変数の仕様は以下のとおりです。

項　目		内　容
ワーク領域のサイズ	ビット	64000点
	ワード	589824ワード
	タイマ	4000点
	カウンタ	4000点
変数の仕様	定義可能数	100000点※2
	変数名の最大長	128文字※3

※1　グローバル変数として定義した場合は、手動でCPUユニットのデバイスに割り付けることも可能です。
※2　定義可能数とは変数編集ウィンドウで登録できる変数名を表します。例えば10000個の要素をもつ配列型変数を1つ定義した場合は1点としてカウントします。
※3　変数名の文字コードはUTF-16です。

ポイント　**BOOL型変数は自動割付時にCH先頭に割り付かない場合が有ります。CHの先頭を指定する必要があるオペランドにBOOL型変数を使用する場合は以下のいずれかの方法でCH先頭に割り付けた状態で指定してください。**
・ビットオフセットを0にしたBOOL型メンバを定義した構造体型のローカル変数/グローバル変数を作成する。
・割り付けデバイスをグローバルデバイスのCH先頭に指定して、グローバル変数を作成する。

≪KV-8000≫で使用できる変数のデータ型と、データサイズは以下のとおりです。

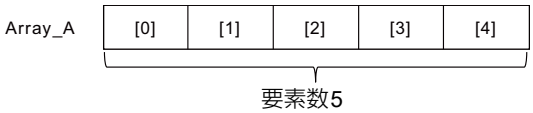
データ型		データサイズ (ビット単位)
型　名	説　明	
BOOL	ビット	1
UINT	1ワード符号なし整数	16
INT	1ワード符号あり整数	16
UDINT	2ワード符号なし整数	32
DINT	2ワード符号あり整数	32
REAL	単精度浮動小数点型	32
LREAL	倍精度浮動小数点型	64
STRING※1	文字列	(文字数※2+1)×8 を16の整数倍に切り上げ
TIMER	タイマ	—
COUNTER	カウンタ	—
ARRAY	配列	データ型のデータサイズ × 配列サイズ
構造体	—	定義による※3
ファンクションブロック	ファンクションブロック	定義による※3

※1　STRING 型で扱う文字コードはプロジェクト言語設定で設定した言語に従います。プロジェクト言語設定の詳細は『KV STUDIO ユーザーズマニュアル』を参照してください。
※2　文字数とはNULLまでのバイト数のことです。Shift-JISなどの2バイト文字の場合は1文字で文字数を2として計算します。
※3　構造体およびファンクションブロックは複数のデータ型を設定して使用するデータ型です。そのため、設定内容によりデータサイズが決まります。

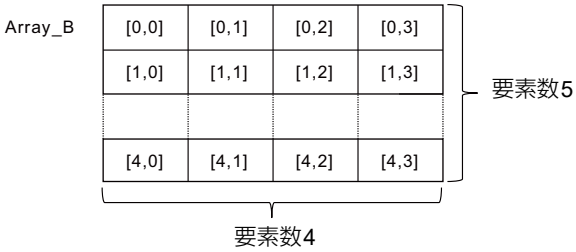
配列

配列とは、同じデータ型のデータの集合体を1つの名前で定義するデータ型です。配列を構成する1つ1つのデータを要素といいます。配列では、変数名の他に、データ型と要素数と次元数を設定します。次元数がnの配列を、n次元配列と呼びます。以下に、1次元配列～3次元配列の例とイメージ図を示します。

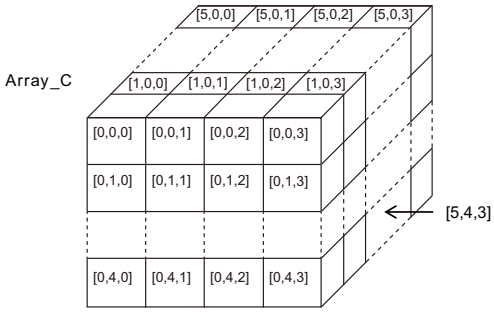
•1次元配列　変数名:Array_A　要素数:5の場合



•2次元配列　変数名:Array_B　要素数:5×4の場合



•3次元配列　変数名:Array_C　要素数:6×5×4の場合



■配列型変数の記載

●配列型変数の要素の指定

プログラム上で配列の要素を指定して使用する場合は、配列名と[　]の添え字を組み合わせて使用します。2次元以上の配列の場合は、要素番号を「カンマ(,)」で区切って指定します。例えば、3次元配列Array_Cの[5,0,3]の要素を使用する場合、プログラム上では以下のとおり「Array_C[5,0,3]」と記載します。

```
Array_C[5,0,3] := 0 ;
```

●配列型変数の全体の指定

配列を構成する全要素のことを「配列全体」といいます。ST上で配列全体を指定する場合は、変数名だけを記載します。例えば、Array_Cという変数の配列全体を指定する場合は「Array_C」と記載します。

```
uiCount := ACNT(Array_C, 2);
```

配列の仕様

項　目	内　容
1次元あたりの最大要素数	589823
最大次元数	8
1配列あたりで占有可能な最大データサイズ	589823ワード
指定可能なデータ型	BOOL / UINT / INT / UDINT / DINT / REAL / LREAL / STRING/ ファンクションブロック / 構造体
指定可能な添え字※1	デバイス/定数/変数/演算式※2
添え字に使用できるデータ型	UINT / INT / UDINT / DINT

※1　添え字には0または正の整数を指定してください。その他の数を指定した場合、変換エラーまたは演算エラーが発生します。
※2　添え字に演算式を使用できるのはST/スクリプトのみです。ラダープログラム上では添え字に演算式を使用できません。

ポイント

- **要素数「10」で設定したArray_Aという配列に対してArray_A[10](11番目の要素)と指定するなど、指定可能な範囲を超えて要素を指定した場合、意図しない領域の値を参照する可能性があります。特に、添え字に変数や演算式を使用する場合は、配列を参照する前にACNT関数やWSIZE関数を実行して配列の要素数や配列全体のデータサイズを取得し、指定可能な範囲を超えた要素を参照しないようにしてください。**
  「ACNT」(6-24ページ)
  「WSIZE」(6-25ページ)
- **配列の添え字にデバイスを指定した場合は16ビット符号無し整数として扱われます。**
- **添え字にビットデバイス(RやMRなど)、ワードデバイスのビット指定、BOOL型の変数は指定できません。**
- **KV STUDIOのバージョンによって、配列のデータ型の表記が以下のとおり異なります。**
  例：要素数が3のUINT型配列の場合
  Ver.11.20以降　ARRAY[0..2] OF UINT
  Ver.11.10以前　UINT[3]

構造体

構造体とは、複数のデータ型のデータの集合体を1つの名前で定義するデータ型です。構造体を構成する1つ1つのデータをメンバといいます。構造体では、定義名の他に、構造体を構成するメンバ名とメンバのデータ型を設定します。構造体には以下のような特長があります。

- 事前に構造体のメンバとデータ型を登録しておくことで、同じ構造を持った変数を作りやすくなる
- 登録モニタウィンドウなどでモニタする際に構造体変数を登録すると、すべてのメンバを一括でモニタできる

モータ軸の以下の情報をプログラムで使用する場合を例に説明します。

現在座標(CurCoordinate)
現在速度(CurSpeed)
軸制御中(Control)
軸エラー(Error)
軸エラーコード(ErrorCode)

構造体を使用しない場合、軸を複数使用しようとすると以下のように軸番号が異なるだけの同一内容の変数を登録する必要があります。構造体を使用する場合は、軸情報のデータ構造を構造体として事前に定義しておくことで、複数の軸を使用する場合でもデータ型に同じ構造体を設定するだけで済みます。

<構造体を使用しない場合>

変数名	データ型
Ax1_CurCoordinate	DINT
Ax1_CurSpeed	DINT
Ax1_Control	BOOL
Ax1_Error	BOOL
Ax1_ErrorCode	UINT
Ax2_CurCoordinate	DINT
Ax2_CurSpeed	DINT
Ax2_Control	BOOL
Ax2_Error	BOOL
Ax2_ErrorCode	UINT

<構造体を使用する場合>

変数名	データ型
Ax1	AxisInfo
Ax2	AxisInfo

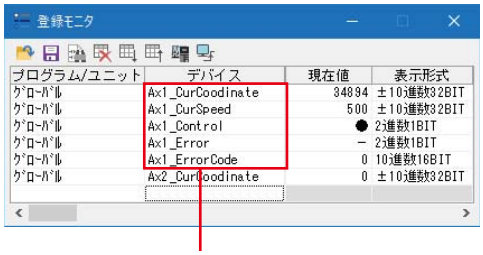
「AxisInfo」型データの定義

メンバ名	データ型
CurCoordinate	DINT
CurSpeed	DINT
Control	BOOL
Error	BOOL
ErrorCode	UINT

事前に設定した構造体を流用

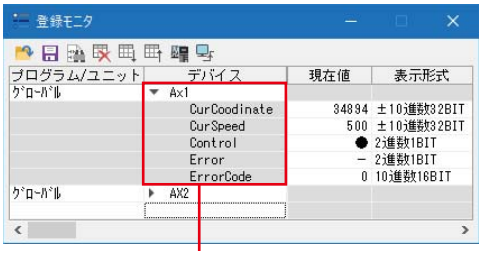
また、KV STUDIO上でモニタする際も、構造体を使用しない場合はすべての変数を手動で入力する必要がありますが、構造体を使用する場合は変数名を入力するだけで構造体内のメンバも一括でモニタできます。

<構造体を使用しない場合>



モニタしたい変数はすべて入力する必要があります。

<構造体を使用する場合>



構造体の場合、変数名を入力するとメンバを一括でモニタできます。

■構造体型変数の記載

プログラム上で構造体型変数を使用する際の記載方法について説明します。

●構造体型変数のメンバの指定

プログラムで構造体型変数のメンバを使用する場合は、「構造体型変数名+ピリオド(.)+メンバ名」と記載します。例えば、Ax1というAxisInfo型変数のCurSpeedというメンバを参照する場合は、「Ax1.CurSpeed」と記載します。

```
diSpeed := Ax1.CurSpeed ;
```

また、構造体型変数のメンバのデータ型に構造体を指定できます。以下の例では、「ProductInfo」という構造体型の変数Product_Aに「ProductionDate」という構造体型変数がネストされています。

変数名	データ型
Product_A	ProductInfo

「ProductInfo」型データの定義

メンバ名	データ型
Name	STRING[20]
SerialCode	STRING[20]
ProductionDate	DateInfo

「DateInfo」型データの定義

メンバ名	データ型
Year	UINT
Month	UINT
Day	UINT

Product_AのDayをプログラムで指定する場合は、「Product_A.ProductionDate.Day」と記載します。

```
Product_A.ProductionDate.Day :=13 ;
```

ポイント　**構造体のネストは最大8段までです。**

●構造体型変数の全体の指定

構造体型変数を構成する全メンバを「構造体全体」といいます。プログラムで構造体全体を指定する場合は、変数名だけを記載します。例えば、Product_Aという構造体型変数の構造体全体を指定する場合は「Product_A」と記載します。

```
WSIZE(Product_A) ;
```

ポイント　**FMOV関数などを使用して構造体全体に値を書き込む場合、転送データ数に構造体全体のデータ数より大きな値を指定すると、意図しない領域の値を参照する可能性があります。FMOV関数などで構造体全体を参照する場合は、関数実行前にWSIZE関数を実行して構造体全体のデータサイズを取得してください。**
「WSIZE関数」(6-62ページ)

構造体の仕様

項　目	内　容
構造体の最大定義数	800
構造体1つあたりの最大メンバ数	500
構造体のネスト最大段数	8※1
メンバとして指定可能なデータ型	BOOL / UINT / INT / UDINT / DINT / REAL / LREAL / STRING/ 配列 / 構造体

※1　stAAA.memberと記載した場合、ネスト段数は1段です。stAAA.stBBB.memberと記載した場合、ネスト段数は2段です。

変数の設定

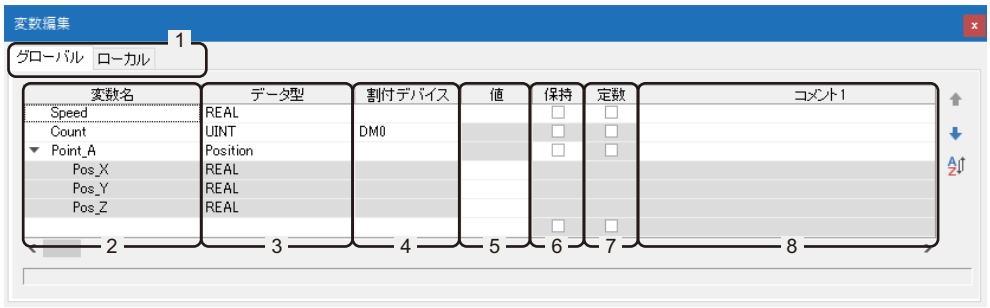
変数を定義してラダープログラムで使用するまでの手順を説明します。

■変数編集ウィンドウから設定する方法

この方法ではグローバル変数、およびローカル変数を設定できます。

1 **【表示(V)】→【変数編集ウィンドウ(L)】を選択し、【編集設定】ウィンドウを開きます。**

別手順　ワークスペースの「変数」をダブルクリック



番　号	項　目	内　容
1	グローバル/ローカルタブ	グローバル変数編集タブとローカル変数編集タブを切り換えます。 「グローバル変数とローカル変数」(2-15ページ)
2	変数名	変数名を入力します(半角128文字/全角128文字)。
3	データ型	変数のデータ型を以下から選択します。 BOOL/UINT/INT/UDINT/DINT/REAL/LREAL/STRING/ TIMER/COUNTER/ARRAY/構造体/ファンクションブロック
4	割付 デバイス	変数をCPUユニットのデバイスに手動で割り付ける場合に設定します。割付は必須ではありません。ローカル変数編集タブでは存在しない項目です。
5	値	「定数」の項目にチェックを入れた場合: 変数の初期値を指定します。定数のチェックが入っている場合は値が固定され、プログラムから値を変更できなくなります。 「定数」の項目にチェックを入れない場合: 変数の初期値を設定します。プログラム内で値を変更できます。割付デバイスを設定している場合は、変数編集ウィンドウでは初期値を設定できません。デバイス初期値設定で、割り付けたデバイスの初期値を設定してください。詳細はCPUユニットのユーザーズマニュアルを参照してください。
6	保持	チェックを入れた変数は、電源断時も値が保持されます。保持できる変数はビット、ワード、カウンタの合計で最大30000個です。 割付デバイスを設定している場合は、上記の30000個に含まれないため設定したデバイスに対して停電保持設定を行ってください。 保持にチェックを入れていない変数は、PROGRAMモード→RUNモード切り換え時に「値」で設定した値に初期化されます。設定していない場合は0に初期化されます。 詳細はCPUユニットのユーザーズマニュアルを参照してください。
7	定数	チェックを入れた変数は、定数として扱われます。定数に設定した変数は、最大定義数100000個の中に含まれますが、ワーク領域を消費しません。
8	コメント	変数のコメントを設定します(半角128文字、全角128文字)※1。設定したコメントはKV STUDIOのラダー編集ウィンドウに表示されます。1つの変数に対してコメント1～コメント8までを設定できます。

※1　CPU機能バージョン2.1未満の場合、半角32文字、全角32文字です。

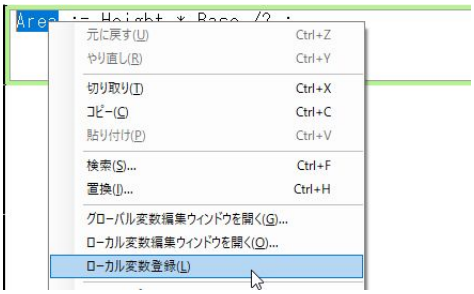
■ST内の未登録変数を取り込んで設定する方法

この方法ではローカル変数のみ設定できます。

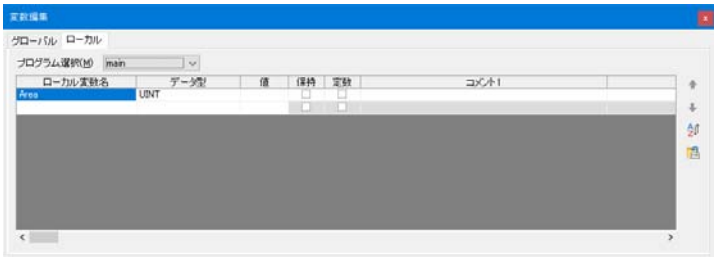
1 変数を使用して、STを記述します。

```
Area := Height * Base /2 ;
```

2 ST内のローカル変数として登録したい文字列を選択した状態で、右クリックメニューの【ローカル変数登録(L)】を選択します。

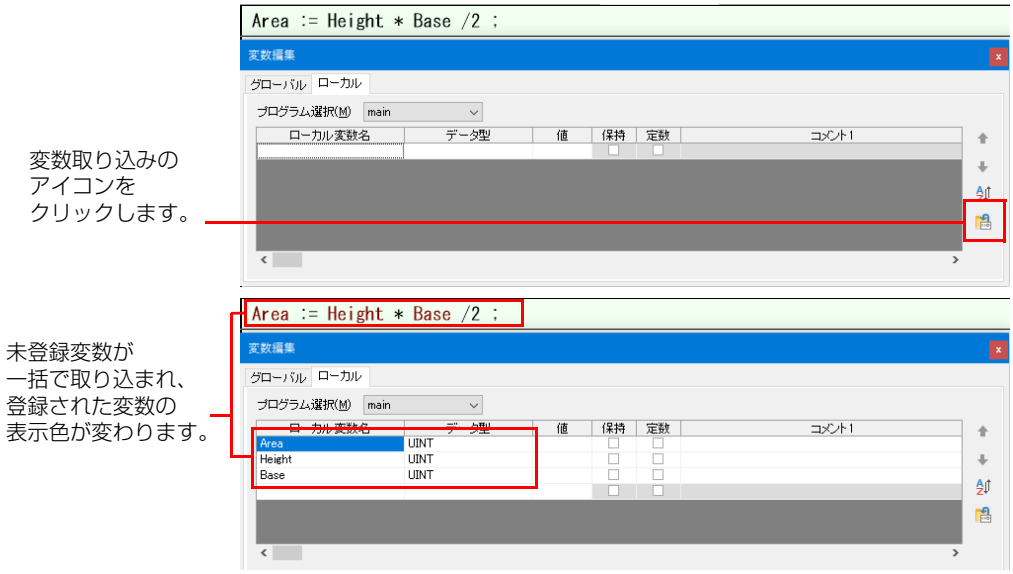


3 変数名が取り込まれた状態で変数編集ウィンドウが表示されるので、データ型を設定します。



●ローカル変数の一括登録

モジュール内の未登録変数をローカル変数として一括で取り込む方法を説明します。未登録変数が存在する状態で、変数編集ウィンドウの をクリックします。対象モジュール内に存在するすべての未登録変数がローカル変数として取り込まれるので、データ型を設定します。



ポイント

一括登録を実行すると、モジュール内のすべての変数が登録されます。そのため、誤記部も変数として登録されるので、本来変換時に発生するエラーが、正常変換される可能性があります。

例　DN1000:=10; .....一括登録前：ST変換時にエラーになる

.....一括登録後：ST変換時にエラーは発生しない

一括登録後に、変数編集ウィンドウにて登録内容を確認することをおすすめします。

変数使用時の注意事項

変数を使用する際の制限や注意事項について説明します。

● 変数の範囲チェック

KV-8000では関数を実行した結果、変数の範囲を超えた領域を参照しないように範囲チェックを行っています。範囲チェックの内容は変数の設定により異なります。
要素数を10に設定したArray_Aという変数を使用する場合を例に説明します。

グローバル変数（自動割付）/ローカル変数の場合
Array_A[20]のように、要素数を超える定数を指定した場合は、変換時にエラーとなります。
Array_A[idx]のように、要素数に定数以外を指定した場合は、変換時にはエラーにはなりません。プログラム実行時に、idxの値が0～9の範囲を超えている場合は命令実行時に演算エラーが発生します。

グローバル変数（手動割付）の場合
Array_Aの先頭をDM0に割り付けている場合を例に説明します。
Array_A[20]のように、要素数を超える定数を指定した場合は、変換時にエラーとなります。
Array_A[idx]のように、要素数に定数以外を指定した場合は、変換時に警告が出ますがエラーにはなりません。プログラム実行時にidxの値が0～9の範囲を超えている場合は、対象となるデバイスが扱われ、演算エラーは発生しません。例えば、idx=19の場合、Array_A[idx]はDM19として扱われます。

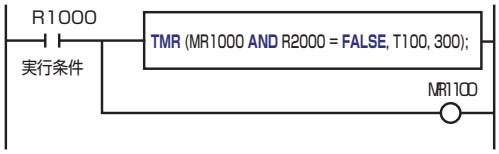
● その他の注意事項

- グローバル変数とローカル変数で同じ変数名を設定できますが、使用した場合はローカル変数として動作します。
- 変数名には、予約語など使用できない文字列があります。
  「使用できない文字一覧」付-10ページ
- 自動割付したグローバル変数およびローカル変数はインデックス修飾できません。
- 配列型、構造体型の変数はインデックス修飾や間接指定はできません。
- ［変数編集］ウィンドウで「定数」にチェックを入れた場合、インデックス修飾や間接指定はできません。

STを記述するときの注意事項

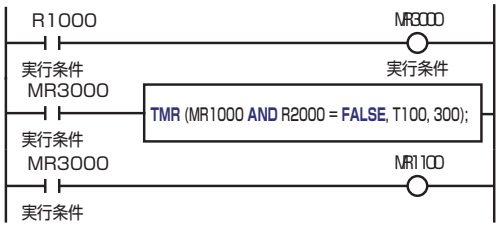
エリア型STの並列接続について

エリア型STは、ラダー命令をOR接続できません。
右のような記述は、変換時にエラーが発生します。



● OR接続と等価なラダープログラム

いったん別のデバイス(右の場合は、MR3000)を経由して、論理的にOR接続しているように記述してください。



STで演算式を記述するときの注意事項

精度落ちが発生した場合「警告」を表示する設定も可能です。「警告メッセージ」(2-24ページ)

ST変換時のメッセージ

STプログラムは、いったんラダープログラムに変換されてからCPUユニットに転送されます。
STプログラムをラダープログラムに変換するときに表示される、警告メッセージとエラーメッセージについて説明します。

警告メッセージ

STプログラムからラダープログラムに変換はできますが、実行すると期待する演算結果が得られない場合などに表示されるメッセージです。できるかぎり警告の発生原因は取り除くことをお勧めします。
代入文で右辺の結果の型と左辺の型が異なっていたり、演算式に.Dが含まれたり、現在の機種では実行できない関数であったりする場合などに発生します。

■型変換によるデータの欠落

以下のSTではデータが欠落し、必要なデータを得られない可能性があります。

ST記述　uiValue := iValue * 10000;

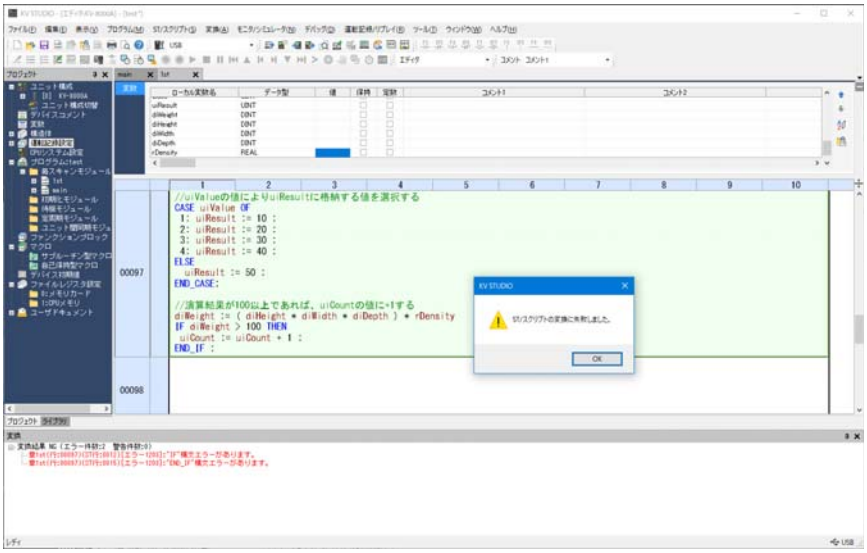
- 例えばiValueの値が－1の場合、uiValueの型はUINTなので正しい演算結果を格納できません。また、iValueの値が32767の場合、演算結果がUINTの範囲(0~65535)を超えるので、期待する(正しい)演算結果を代入できません。

エラーメッセージ

STプログラム内に、ラダープログラムに変換できない記述があった場合、エラーが発生しエラー要因をアウトプットウィンドウに表示します。
エラー要因を取り除き変換に成功しないと、プログラムはCPUユニットに転送できません。アウトプットウィンドウに表示されているエラーメッセージを元に、STを修正してください。

変換エラーが発生したときは

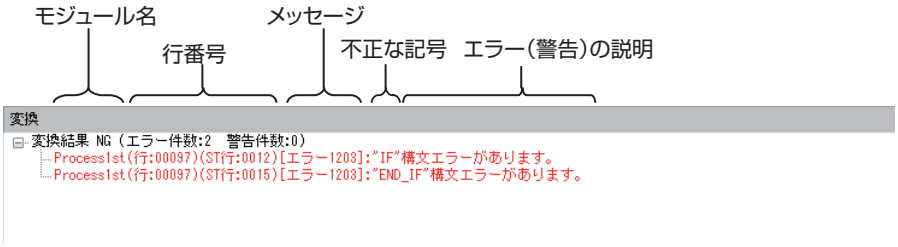
関数名のタイプミスや、不正な型(サフィックス)指定、制御文の使い方を間違えるなど、STプログラムの記述が不正な場合、STを変換すると「ST変換エラー」が発生し、ダイアログボックスが表示されます。



変換エラーが発生すると、アウトプットウィンドウにエラー発生した場所とエラーの内容が表示されます。

■アウトプットウィンドウのエラー表示

変換エラーが発生すると、次のようにエラー内容が表示されます。



メッセージが[エラー]の場合　:ラダープログラムに変換不可能な状態です。この状態ではプログラムをPLCに転送できません。

メッセージが[警告]の場合　:ラダープログラムに変換されます。プログラムを PLC に転送して実行したり、シミュレータで実行可能な状態です。
ただし、プログラムを実行すると、期待する演算結果が得られない恐れがあります。

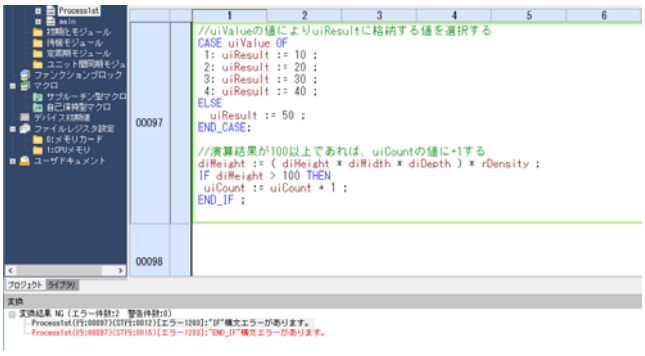
■アウトプットウィンドウの使いかた

変換エラーが発生すると、アウトプットウィンドウにエラー(警告)内容が表示されます。
表示された行をダブルクリックすると、ラダー編集ウィンドウのエラー行にカーソルが移動します。
エラー内容を元に修正してください。

1 **アウトプットウィンドウのエラー内容(①)をダブルクリックします。**

対応するエラー記述のある行(②)にカーソルが移動(フォーカス移動)し、STが編集可能状態になります。

2 **エラー記述を修正します。**



3 **修正後、修正したSTだけを変換します。**

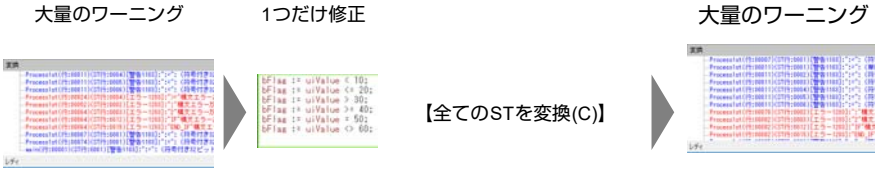
メニューから【ST/スクリプト(S)】→【選択STを変換(S)】を選択します。

別手順
- Ctrl キー＋ T キー
- 右クリックメニューから【ST/スクリプト編集(X)】→【選択STを変換(S)】

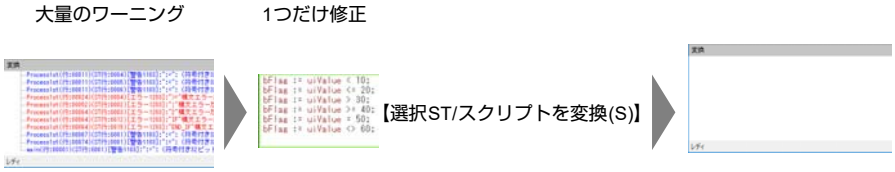
アウトプットウィンドウに、エラーが表示されなければ修正完了です。

参 考

・ エラーやワーニングが大量に発生している状態で、エラーを1つだけ修正してから【全てのST/スクリプトを変換(C)】を選択すると、他の修正しなかったエラーが再度大量に表示されます。



・ このようなときは、メニューから【選択ST/スクリプトを変換(S)】を選択します。



・ 修正したSTだけが変換されますので、そのSTに残っているエラーだけが表示され、正しく修正できたか確認しやすくなります。
・ 変換ラダーの表示/非表示(Ctrl+G)でSTを編集したまま、変換ラダーを表示して確認したり、展開ラダーを閉じたりできます。

ファンクションブロック

ファンクションブロックはプロジェクト内で何度も繰り返して使用するラダー回路に、任意の名前を付けて登録し、共通化したものです。登録されたファンクションブロックはSTプログラムから呼び出すこともできますし、ファンクションブロック内でSTで使用することも可能です。

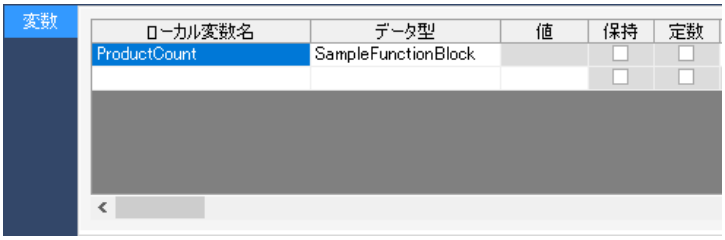
ファンクションブロックの呼び出し

■ファンクションブロックの呼び出し

STプログラムでは、以下のようにファンクションブロックインスタンスを関数として使用できます。

FBInstanceName (arg1, arg2, arg3);

ファンクションブロックを呼び出す際は、まずファンクションブロックインスタンスを作成します。
ファンクションブロックインスタンスは[変数編集]ウィンドウで変数を作成し、変数のデータ型をファンクションブロック型に設定することで、作成した変数をファンクションブロックインスタンスとして使用できます。




```
ProductCount(uiValue1, uiValue2, uiValue3);
```

ポイント

- 自身のファンクションブロックインスタンスを使用した ST プログラムが含まれるファンクションブロックを、呼び出すことはできません。
- 別箇所から同一のファンクションブロックインスタンスを呼び出した場合、共通のリソースが使用されます。実体を分けて使用する場合は、別名のファンクションブロックインスタンスを用意して使用してください。

■ファンクションの呼び出し

STプログラムでは、以下のようにファンクション名を関数として使用できます。

FunctionName (arg1, arg2, arg3);

STプログラム内でファンクション名と引数を設定するだけでファンクションを呼び出すことができます。

```
SampleFunction(uiValue1, uiValue2, uiValue3);
```

STのデバッグ機能

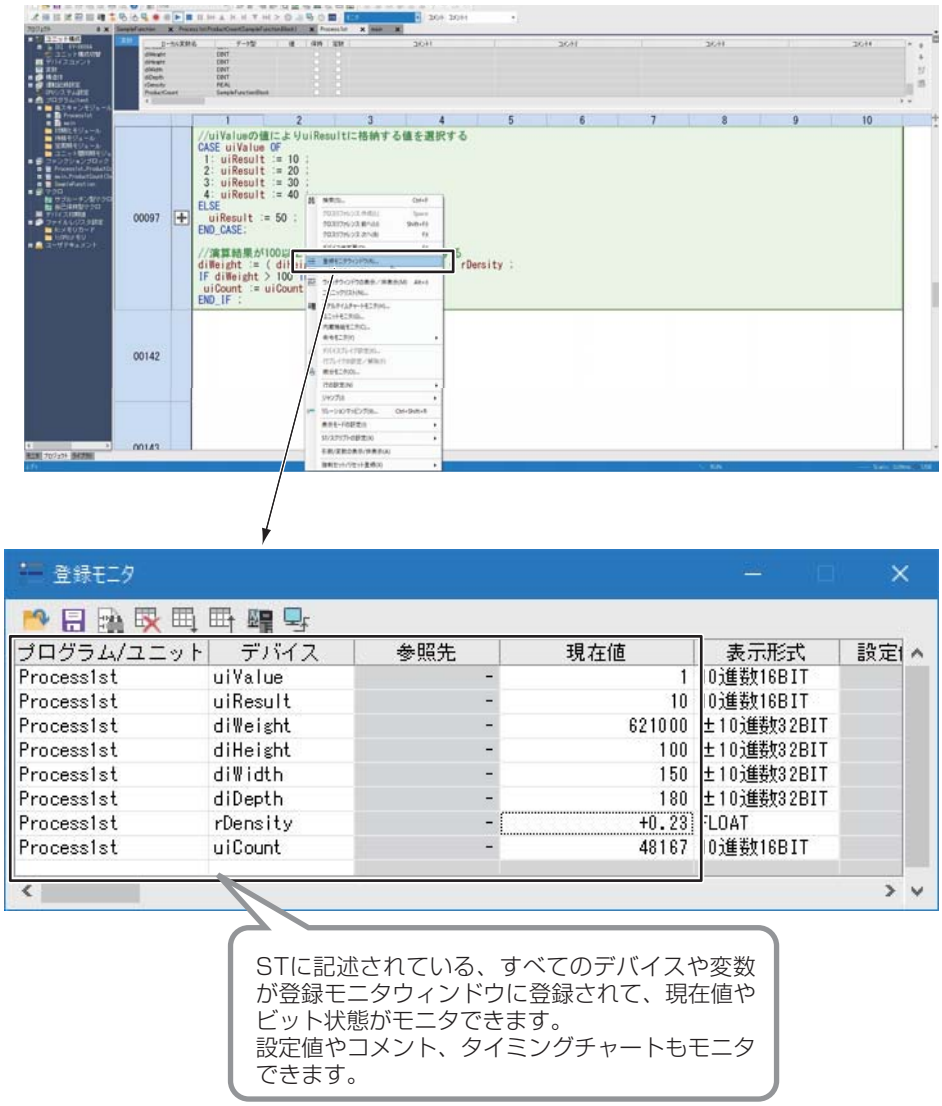
モニタ機能

オンライン中もしくは、シミュレータ起動中にモニタ機能を利用して、STプログラムのデバッグをおこないます。

■登録モニタ

STに記述されている各デバイスや変数の現在値がモニタできます。

1 KV STUDIOがモニタ状態、もしくはシミュレータ状態で、モニタしたいSTをクリックし、右クリックメニューから［登録モニタ］を選択します。

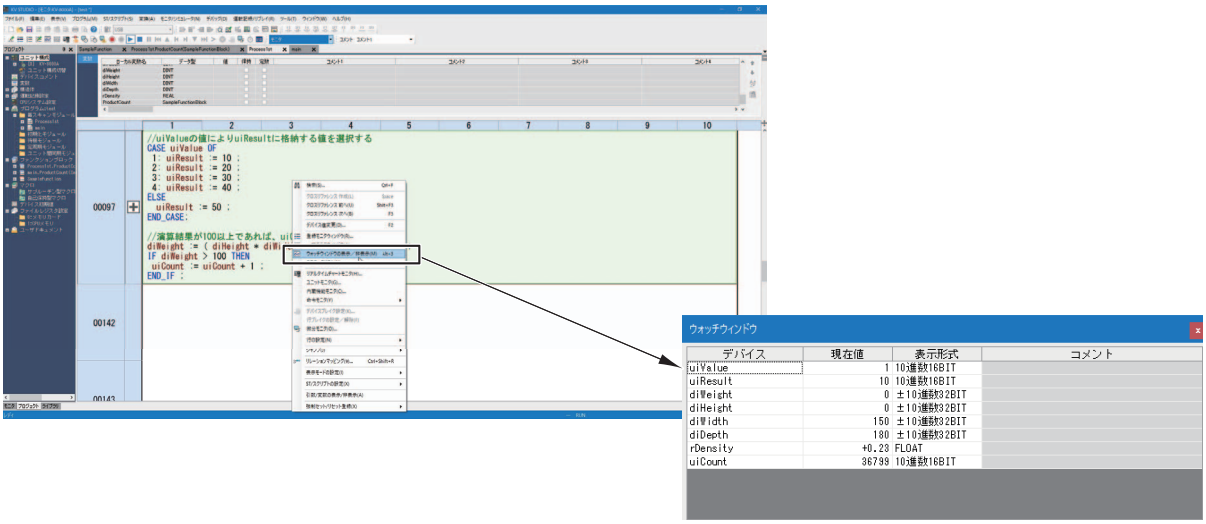


■ウォッチウィンドウ

ウォッチウィンドウは、STを選択するだけで、ST内のすべてのデバイスや変数の現在値をモニタできます。

1 KV STUDIOがモニタモード、もしくはシミュレータモードで、モニタしたいSTをクリックし、右クリックメニューから［ウォッチウィンドウ］を選択します。

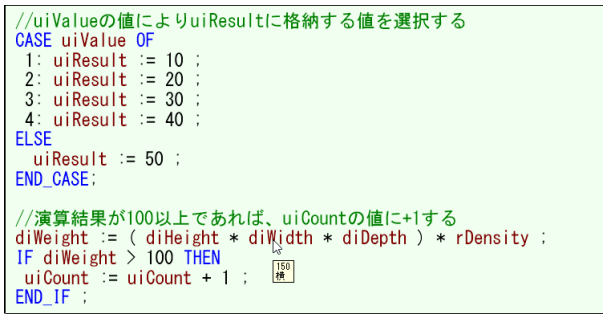
別手順



別のST記述を選択すると、ウォッチウィンドウに表示されている変数やデバイスが自動的に切り換わります。

■ツールチップによるモニタ

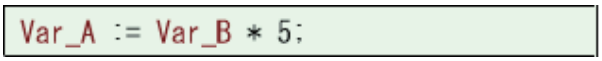
KV STUDIOがモニタモード、もしくはシミュレータモードで、モニタしたいデバイスにマウスカーソルをあわせるとツールチップで現在値とデバイスコメント（変数の場合は変数コメント）を表示します。



■STモニタによるモニタ

KV STUDIOがモニタモード、もしくはシミュレータモードの場合、変数の隣に変数の値が表示されるようになります。

エディタモード



モニタモード/シミュレータモード

入力補助機能・その他

入力補助機能について

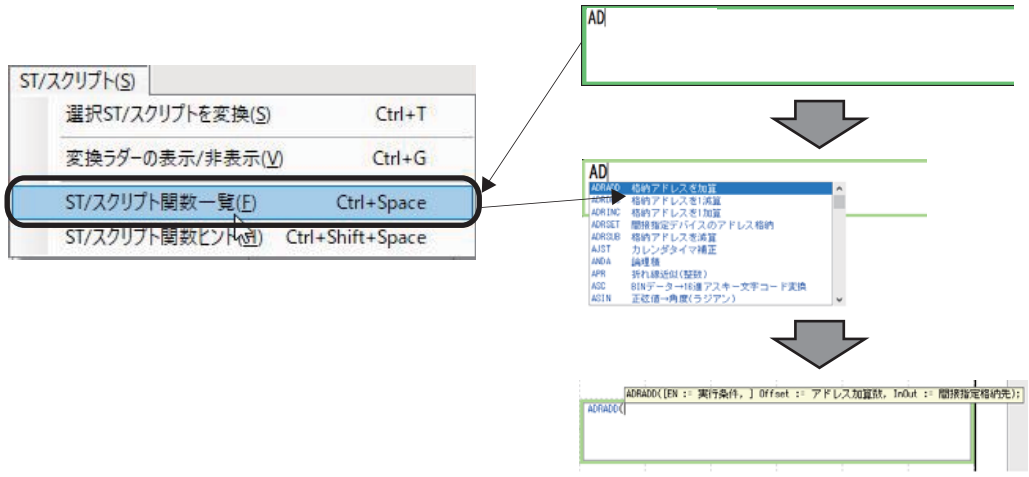
KV STUDIOには、STの記述をサポートするために、様々な便利機能や入力補助機能が存在します。

ST関数一覧

関数名そのものがわからなくなった場合、関数を途中まで入力した後に、次のいずれかの方法で関数候補を呼び出すことができます。

●関数一覧の呼び出し

メニューから【ST/スクリプト(S)】→【ST/スクリプト関数一覧(F)】を選択します。



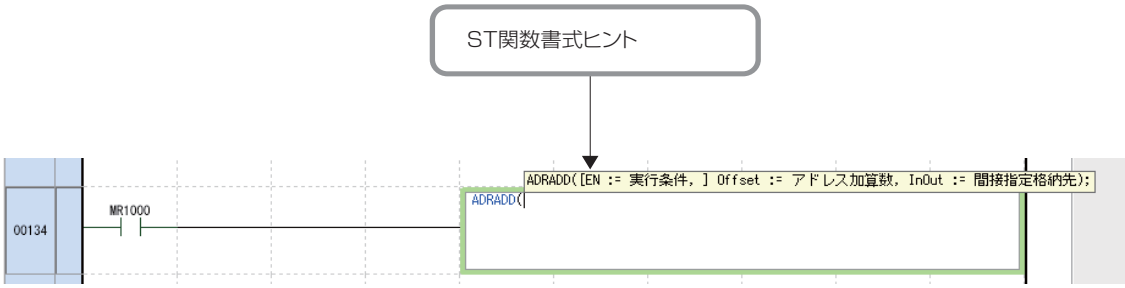
使用する関数を選択し↵キーを押すか、ダブルクリックすると、STエディタのカーソルがある場所に選択した関数が挿入され、ST関数書式ヒントが表示されます。

別手順
- Ctrl キー＋ Space キー
- 右クリックメニューから【ST/スクリプト関数一覧(F)】

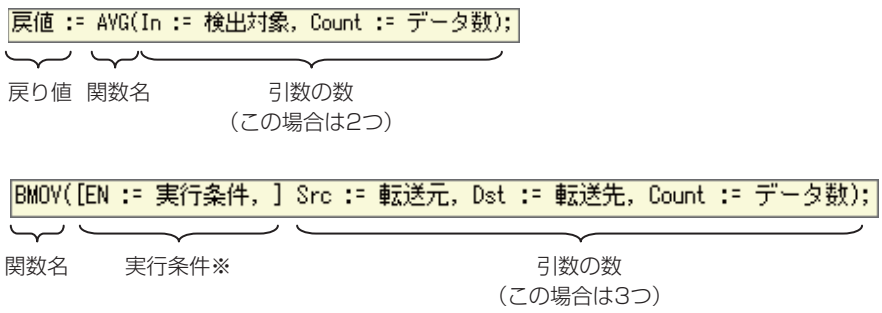
参考
- 何も入力していない状態で、ST関数候補を呼び出すこともできます。その場合アルファベット順にすべての関数候補が表示されます。
- 「ST関数候補」はST記述中のみ呼び出すことができます。

ST関数書式ヒント

関数を入力する時、引数や戻り値の有り無しなどの情報を、簡易的に「ST関数書式ヒント」が教えてくれます。



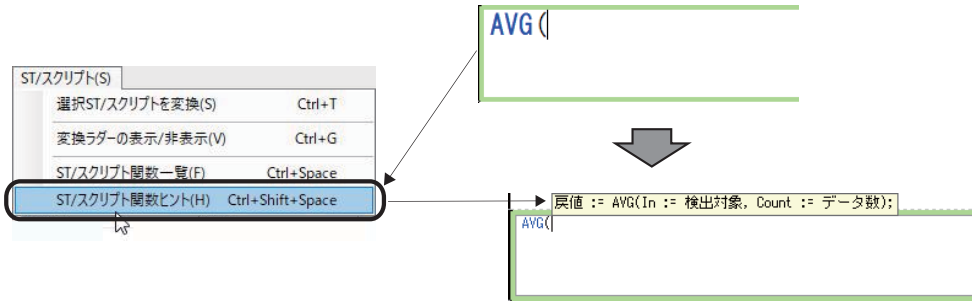
「関数名(」まで入力すると、自動的に「ST関数書式ヒント」が表示されます。



※[　]内に書かれている引数は省略可能です。

●ST関数ヒントの呼び出し

カーソル「Ｉ」を「(」の右側に置き、メニューから【ST(S)】→【ST関数ヒント(H)】を選択します。



別手順　Ctrl キー+ Shift キー+ Space キー

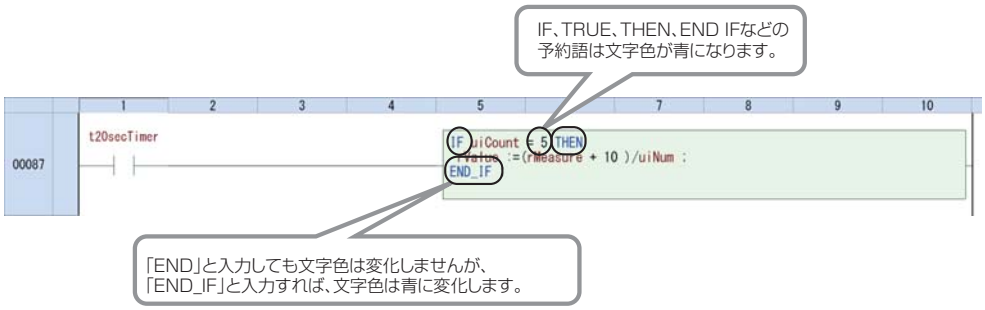
参考　「ST関数ヒント」はST記述中のみ呼び出すことができます。

ポイント　**「ST関数ヒント」を呼び出すときは、必ずカーソルを「(」の右側か、関数の先頭に置いてください。他の位置にカーソルがあるとST関数ヒントは呼び出されません。**

予約語

制御文や関数など、STに登録されている単語(予約語)については、記述時に文字色が青になります。文字色が変化しなかった場合、入力が間違っている可能性があります。

参考　予約語の文字色は、KV STUDIOの［オプション設定］ダイアログで変更可能です。



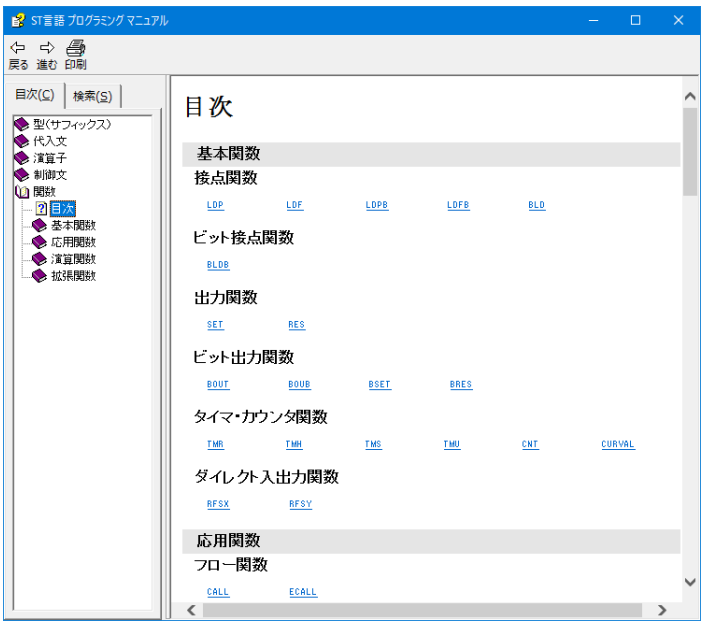
HELP機能

関数の書式や使い方、制御文の説明や演算子の詳細は「ST関数ヘルプ」で調べることができます。

■ST関数ヘルプを起動する

1 **メニューから【ヘルプ(H)】→【ST関数ヘルプ(M)】を選択します。**

「ST関数ヘルプ」の目次が表示されます。

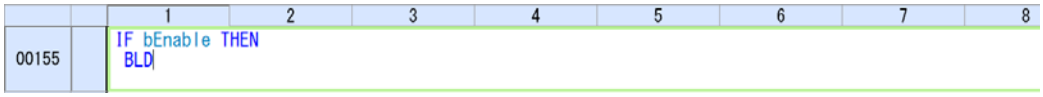


2 **調べたい関数を選択します。**

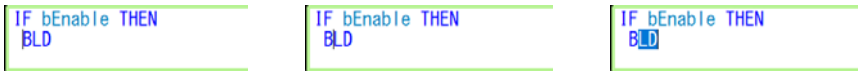
関数の詳細な説明が表示されます。

■ST編集画面上で指定した関数のヘルプを表示する

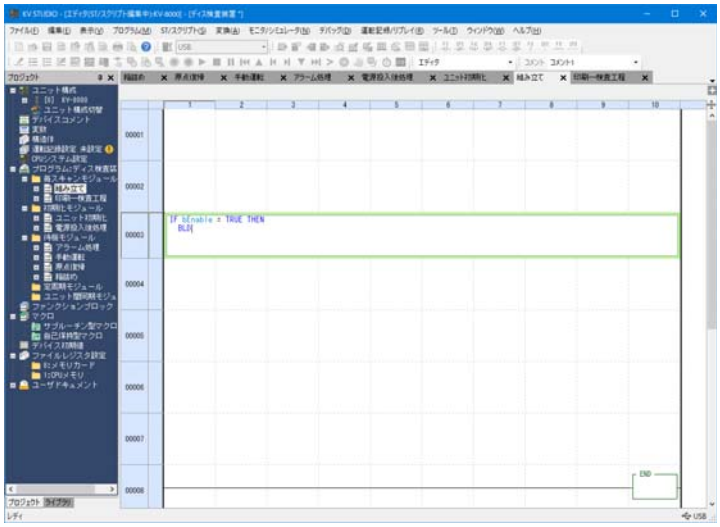
1 調べたい関数にカーソルをあわせます。



カーソルの位置は、下図のように関数の前後や中、または関数の一部でも範囲選択した状態にしてください。



2 F1 キーを押す。



選択した関数のヘルプ画面が表示されます。

コメント機能について

STでは、「//」による1行コメントと、「(*...*)」「/*...*/」による範囲コメントに対応しています。コメントの記述は、効率の良いデバッグやプログラムの改造をおこなう上で効果的です。STとあわせて積極的に記述されることをお勧めします。

参 考

- コメント行が増えてもプログラムのサイズやスキャンタイムには影響しません。
- 一行に入力できる文字数はST、コメントを含み1024文字(半角)までです。
- 以下の方法で選択している複数の行をまとめてコメント行にできます(KV STUDIO Ver.11.30以降)。
  - 右クリックメニューの【行の無効化-設定(Q)】
  - メニューの【ST/スクリプト(S)】→【行の無効化-設定(Q)】
  - Ctrl + Q
- 以下の方法で選択している複数の行のコメントを解除できます(KV STUDIO Ver.11.30以降)。
  - 右クリックメニューの【行の無効化-解除(W)】
  - メニューの【ST/スクリプト(S)】→【行の無効化-解除(W)】
  - Ctrl + W

ポイント

引数に間接参照(先頭に*を付ける)を使用する場合、「(*」と続けて記載するとコメントとして認識されます。「(」と「*」の間に半角スペースを追加して、コメントにならないようにしてください。

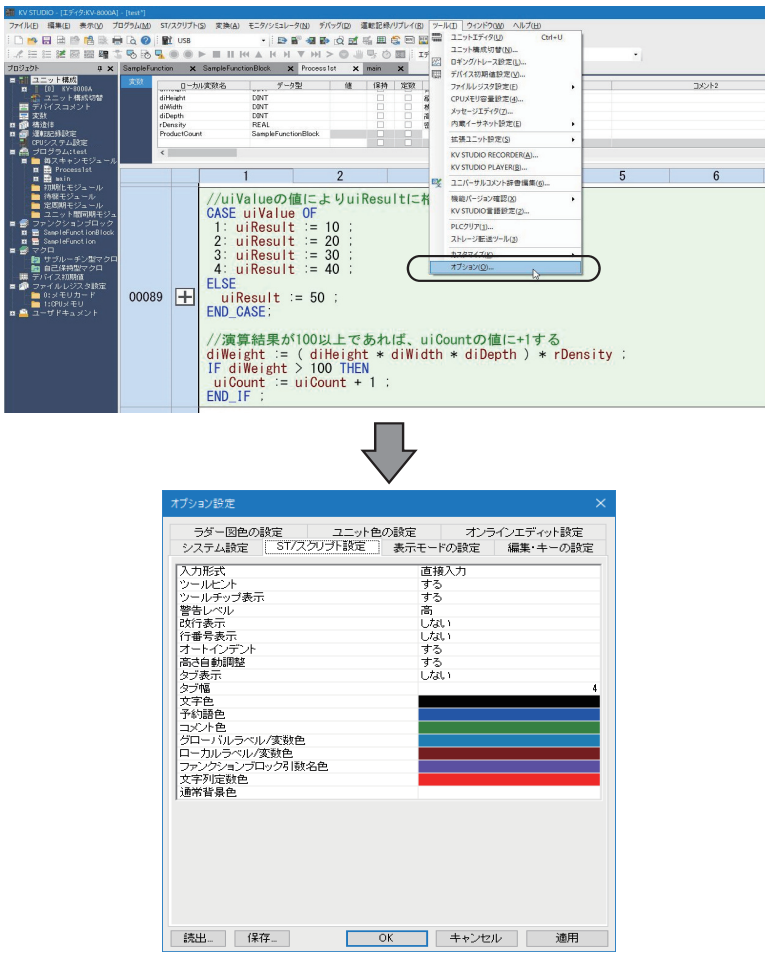
編集画面の注意事項とカスタマイズ

■ST記述時の注意事項

- 1行の文字数はST、コメントを含めて1024文字(半角)までです。
- STを展開したラダーを直接は編集できません。

■STウィンドウのカスタマイズ

メニューから、【ツール(T)】→【オプション(O)】→【ST/スクリプト設定】で、編集ウィンドウのカスタマイズがおこなえます。
詳細は、『KV STUDIOユーザーズマニュアル』を参照してください。



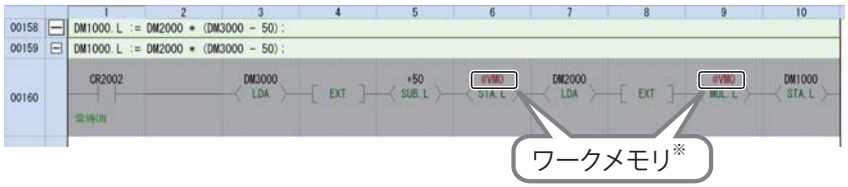
ST/スクリプト用ワークメモリ(ワークデバイス)について

ワークメモリ(ワークデバイス)

ワークメモリ(ワークデバイス)とは、複雑な演算式などで部分的な演算結果を一時的に保存しておくシステム領域のデバイスのことです。STプログラムでは、ST変換時にKV STUDIOがあらかじめモジュール/マクロ用に確保している領域から、自動的にワークメモリを割り当ててくれるので、デバイスの重複使用などを意識せずに使用できます。

ST記述　DM1000.L := DM2000 * (DM3000 - 50);

ラダー変換

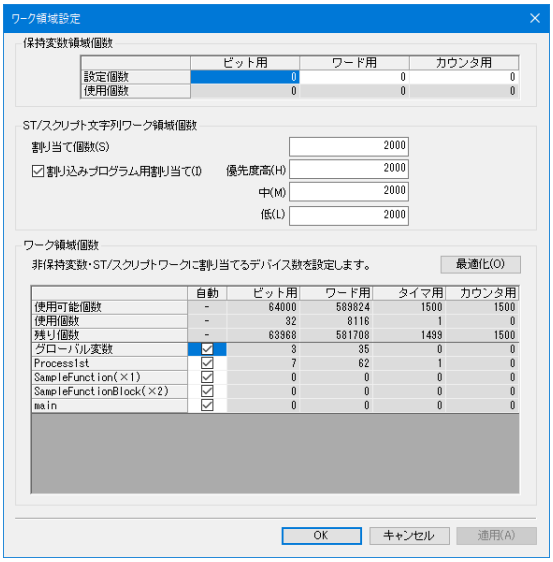


動作説明　最初に(DM3000-50)の演算をおこなってワークメモリ(@VM0)に一時的に格納し、ワークメモリ(@VM0)に格納されている値とDM2000を乗算した結果を、左辺のDM1000.Lに格納します。

■ワークメモリの使用領域

ST/スクリプト用ワークメモリは、[ワーク領域設定]ダイアログで使用状況を確認します。
『KV STUDIOユーザーズマニュアル』

〔ワーク領域設定(W)〕をクリックすると、割り込みプログラム用にワーク領域を割り当てられます。



XYM表記

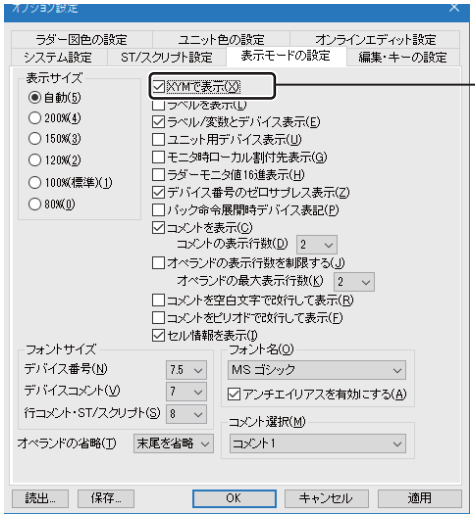
XYM表記でSTプログラムが作成できます。

■XYM表記でプログラムを作成する

1 **メニューから【ツール(T)】→【オプション(O)】を選択します。**

[オプション設定]ダイアログボックスが表示されます。

2 ［表示モードの設定］タブをクリックします。



［XYMで表示(X)］のチェックボックスをチェックして、OKをクリックします。

3 XYM表記でプログラミングします。

```
IF D1000 > 20 THEN
 D0 := 50;
 SET (Y20);
END_IF;
```

参 考　「XYMで表示」を選択後、通常デバイス番号での記述(混在可)もできます。

XYM表記でSTを編集したあとXYM表記を解除すると、X、Yを使用しているSTを再変換するとエラーが発生します。M、Lについては、デバイス番号が10進数の扱いからRのようなCH表記に変わるため、変換結果のデバイス番号が変わることがあるので注意してください。

例) M100 := TRUE;

このSTのM100のラダーへの変換結果は、

・XYM表記時は、MR604に変換される。

・XYM表記でないときは、MR100に変換される。

3

データの取り扱い

デバイス ····· 3-2
インデックス修飾と間接指定 ····· 3-25
データの型(サフィックス) ····· 3-31

デバイス

デバイスとは、CPUユニット内にあるリレーやデータメモリなど、プログラムで用いられる要素（変数）のことをいいます。演算や関数で使用するデータを与えたり、関数の実行結果を格納するために使用します。

デバイス一覧

■ビットデバイス

デバイス名	範囲 ※1	点数	主な機能	運転開始時（電源 ON、PRG → RUN）	ローカルデバイス（@）※1	インデックス修飾（:Z）	間接指定（＊）※8 参照	間接指定（＊）※8 格納
入力リレー※10	R000 ～ R199915※2	合計 32000 点	外部機器から ON/OFF の情報を取り込むためのデバイス	クリア / 保持 ※3※12	○	○	○	×
出力リレー※10			外部機器へ ON/OFF 情報を送り出すためのデバイス					
内部補助レー※10			CPU 内部と拡張ユニットで使用できるデバイス					
リンクリレー	B00 ～ B7FFF※9	32768 点	CPU 内部でのみ使用できるデバイス	クリア / 保持※3	×	○	○	×
内部補助リレー	MR000 ～ MR399915	64000 点		クリア	○	○	○	×
ラッチリレー	LR000 ～ LR99915	16000 点		保持	○	○	○	×
タイマ（接点）	T0 ～ T3999	4000 点	10μs、1ms、10ms、100ms の減算タイマ・10ms のアップダウンタイマの接点デバイス	クリア	○	○	○	×
カウンタ（接点）	C0 ～ C3999	4000 点	加算カウンタ・アップダウンカウンタの接点デバイス	クリア / 保持※7	○	○	○	×
コントロールリレー	CR000 ～ CR7915※4	1280 点	PLC の機能を制御したり、状態を取得するためのデバイス	保持	×	×	×	×

■ワードデバイス

デバイス名	範囲 ※1	点数	主な機能	運転開始時（電源 ON、PRG → RUN）	ローカルデバイス（@）※1	インデックス修飾（:Z）	間接指定（＊）※8 参照	間接指定（＊）※8 格納
データメモリ	DM0 ～ DM65534	65535 点	数値データを格納できるデバイス（16 ビット）	クリア / 保持 ※3	○	○	○	○
拡張データメモリ	EM0 ～ EM65534	65535 点		クリア / 保持 ※3	○	○	○	○
ファイルレジスタ	連番方式：ZF0 ～ ZF524287	524288 点		クリア / 保持 ※3	×	○	○	○
	カレントバンク：FM0 ～ FM32767	（32768×16 バンク）点			○※11			
リンクレジスタ	W0 ～ W7FFF※9	32768 点		クリア / 保持 ※3	×	○	○	○
テンポラリメモリ	TM0 ～ TM511	512 点	数値データを格納できるデバイス（16 ビット）	クリア	○	○	○	○
タイマ（現在値 / 設定値）	T0 ～ T3999	4000 点	10μs、1ms、10ms、100ms の減算タイマ・10ms のアップダウンタイマの現在値 / 設定値（32 ビット）	現在値：クリア ※5 現在値：保持	○	○	○	×
カウンタ（現在値 / 設定値）	C0 ～ C3999	4000 点	加算カウンタ・アッ プダウンカウンタの現在値 / 設定値（32 ビット）	クリア / 保持 ※7	○	○	○	×
インデックスレジスタ	Z1 ～ Z23※6	23 点	インデックス修飾で使用するデバイス（32 ビット）	クリア	×	×	×	×
コントロールメモリ	CM0 ～ CM7599※4	7600 点	PLC の機能を制御したり、状態を取得するためのデバイス（16 ビット）	保持	×	×	×	×

※1　ローカルデバイスとして使用するために、ローカルデバイスに予約した個数分、グローバルデバイスの使用可能な範囲が制限されます。「ローカルデバイス一覧」(3-4)
※2　R000～R199915は、《ユニットエディタ》で拡張入出力リレーに設定できます。
※3　CPUシステム設定の[停電保持設定]でクリア/ 保持の設定が可能です。
※4　「CR/CM一覧」(付-2ページ)
※5　アップダウンタイマ(UDT) は、現在値を保持します。
※6　Z11～Z23 は、システム予約されているため使用できません。
※7　CPUシステム設定の[停電保持設定]でクリア/ 保持の設定が可能です。(接点、現在値のみ)
※8　「格納」は、「＊(デバイス)」と記述できるデバイスです。「参照」は、間接指定できるデバイスです。
『KV-8000シリーズ ユーザーズマニュアル』「間接指定」
※9　デバイス番号は、16進数で指定します。
※10　ダイレクト処理する場合は、「DR」で表記します。
『KV-8000シリーズ ユーザーズマニュアル』「ダイレクト処理について」
※11　ローカルデバイスは、バンク切り替えできません。
※12　入力リレーは、クリア/保持の設定はできません。

■ファンクションブロック引数

ファンクションブロック内のみで使用される引数は以下のとおりです。

引数タイプ		引数名 / デバイス		説明	間接指定（＊）	ローカルデバイス（@）	インデックス修飾（：）
IN		任意のラベル		デバイスの値または定数を引数として受け取ります。	×	×	×
OUT		任意のラベル		演算結果を指定したデバイスに書き込みます。	×	×	×
IN-OUT		任意のラベル		デバイスを引数として受け取ります。	×	×	○
ユニット	先頭 R	UR	UR0 ～ UR7	ユニットに割り付けられた先頭リレー(R)を引数として受け取ります。	×	×	○
	先頭 DM	UM	UM0 ～ UM7	ユニットに割り付けられた先頭データメモリ(DM)を引数として受け取ります。	×	×	○
	番号	UV	UV0 ～ UV7	ユニットの番号を引数として受け取ります。	×	×	×

■定数

定数	範囲	主な機能
10 進数	0～65535(16 ビット符号無し) -32768～+32767(16 ビット符号有り) 0～4294967295(32 ビット符号無し) -2147483648～+2147483647(32 ビット符号有り)	10 進定数を指定する時に使用します。
単精度浮動小数点型実数	-3.4E+38 ≦ N ≦ -1.4E-45 N ＝ 0 +1.4E-45 ≦ N ≦ +3.4E+38(有効桁数:約7 桁)	単精度浮動小数点型定数を指定する時に使用します。
倍精度浮動小数点型実数	-1.79E+308 ≦ N ≦ -2.23E-308 N ＝ 0 +2.23E-308 ≦ N ≦ +1.79E+308(有効桁数:約16 桁)	倍精度浮動小数点型定数を指定する時に使用します。
16 進数 (16#)	16#0000～16#FFFF(16ビット) 16#00000000～16#FFFFFFFF(32ビット)	16 進定数を指定する時に使用します。
固定文字列 (' ')	(例) '型式ABC'、'月/08:30:15'	固定文字列を指定する時に使用します。

■内部レジスタ

デバイス名	範囲	点数	主な機能	運転開始時（電源 ON、PROG → RUN）
内部レジスタ	－	1 点	データを一時的に格納するデバイス (16ビット/32ビット/64ビット)	クリア

ローカルデバイス一覧

プロジェクト作成時に確保されるローカルデバイス/グローバルデバイスの個数･範囲の初期値を下記に示します。ローカルデバイスの予約個数は、[ワークスペース]上の[CPUシステム設定]→[ローカル全体割付設定]で変更できます。

デバイス名 ※1	ローカルデバイス	グローバルデバイス	合計個数
	プロジェクト全体での予約個数 ※2	個数・範囲	
リレー(R)※3	なし	32000点(2000ch) R000～R199915	32000点(2000ch)
内部補助リレー(MR)※3	32000点(2000ch)	32000 点(2000ch) MR000 ～ MR199915	64000点(4000ch)
ラッチリレー(LR)※3	8000点(500ch)	8000点(500ch) LR000～LR49915	16000点(1000ch)
タイマ(T)※4	1500点	2500点T0～T2499	4000点
カウンタ(C)※4	1500点	2500点C0～C2499	4000点
データメモリ(DM)	25000点	40535点DM0～DM40534	65535点
拡張データメモリ(EM)	45000点	20535点EM0～EM20534	65535点
テンポラリデータメモリ(TM)	なし	512点TM000～TM511	512点

※1　リンクリレー(B)、リンクレジスタ(W)、ファイルレジスタ(ZF)、高速カウンタ(CTH)、高速カウンタコンパレータ(CTC)、コントロールリレー(CR)、コントロールメモリ(CM)、インデックスレジスタ(Z)は、ローカルデバイスとして使用できません。
@FMxxxを使用した場合、ワーク領域をローカルデバイスとして消費します。

※2　プロジェクト新規作成時に、ローカルデバイスとして使用可能なデバイスの領域を一括で確保します。(モジュールを10個使用して、マクロを50回呼び出すと想定して計算しています。)
各デバイスの予約個数は任意に変更できます。

※3　チャンネル(ch)単位で使用個数を設定します。

※4　タイマ･カウンタ命令のオペランドとしてデバイス番号を指定する場合のみ、
「@＋(デバイス番号)」と記述します(ニモニック入力例:　TMR @0 #100)。

ポイント

- **リレー(R)とテンポラリデータメモリ(TM)をローカルデバイスとして使用する場合は、使用個数を設定する必要があります。**
- **以下のデバイスは、グローバルデバイスとして扱う必要があります。**
  - **拡張ユニットに割り当てられているデバイス**
  - **タッチパネルなどの外部接続機器で参照するデバイス**
  - **プロジェクト全体で共通して使用するデバイス**
  - **モジュール間でデータの受け渡しが必要なデバイス**
- **ローカルデバイスは、0番を先頭にできるだけ詰めて使用してください。**
- **R や DM は拡張ユニットに割り当てられるため、グローバルデバイスとして使用する領域が必要です。ローカルデバイスは、MRやEMを優先して使用してください。**
- **使用可能範囲外でローカルデバイス/グローバルデバイスを使用すると、変換エラーになります。**
- **インデックスデバイスや、間接参照を使用して、ローカルデバイス / グローバルデバイスの使用可能範囲外を指定した場合演算エラーが発生します。**
- **オペランドに先頭デバイスを指定する各命令語、インデックス修飾、間接指定を使用する場合は、ローカルデバイス / グローバルデバイスの使用可能範囲を超えないようにしてください。**

デバイスの表記方法

●インデックス修飾(インデックスレジスタによる修飾、インデックス定数による修飾)の記述

•インデックスレジスタによる修飾

デバイス(R(DR) / MR / LR / B / T / C / DM / EM / FM / ZF / W / TM / ＊DM / ＊EM / ＊FM / ＊ZF / ＊ W / ＊TM)の後に : Z□(□ : 1～10の数値) を続けて入力します。

例

DM0:Z01	**(Z1の値が100なら、DM100を意味します)**
R1000:Z01	**(Z1の値が100なら、R1604を意味します)**

•インデックス定数による修飾

デバイス(R(DR) / MR / LR / B / T / C / DM / EM / FM / ZF / W / TM / ＊DM / ＊EM / ＊FM / ＊ZF / ＊ W / ＊TM)の後に :□(□ : －524287～524287) を続けて入力します。

例

DM0:1	**(DM1を意味します)**
R1000:1	**(R1001を意味します)**

•演算式によるインデックス修飾

デバイス(R(DR) / MR / LR / B / T / C / DM / W / TM / CM / ＊DM / ＊ W / ＊TM)の後に演算式を続けて入力します。

例

DM0:(DM100＋20)	**(DM100の値が50なら、DM70を意味します)**
R1000:(DM0＊4)	**(DM0の値が2なら、R1008を意味します)**

ポイント

- **演算式によるインデックス修飾はST/スクリプト内でのみ使用できます。**

●間接指定(＊)の記述

デバイス(DM/EM/FM/ZF/W/TM)の前に ＊ を入力します。間接指定命令とあわせて使用します。

例

＊DM0　＊TM100

「間接指定」3-27ページ

●ローカルデバイスの記述

デバイス(R/MR/LR/T/C/DM/EM/FM/TM)の前に @ を入力します。
モジュール･マクロ内のみで有効なデバイスです。間接指定やインデックス修飾も可能です。

例

@DM0　@R10000

●XYM表記について

デバイス(R(DR)/MR/LR/DM/EM/FM)については、下表のXYM表記によるデバイスの指定が可能です。

デバイス名	キーエンス表記	XYM表記	XYM表記時のデバイス番号
リレー	R(DR)	X(DX)	0000～999F
		Y(DY)	0000～999F
内部補助リレー	MR	M	00000～63999
ラッチリレー	LR	L	0000～15999
データメモリ	DM	D	00000～65534
拡張データメモリ	EM	E	00000～65534
ファイルレジスタ	FM	F	00000～32767

XYM表記にするには、《KV STUDIO》のメニューから【ツール】→【オプション】を選択して、[表示モードの設定]で「XYMで表示」にチェックを付けます。

●ワードデバイスのビット指定の記述

デバイス(R(DR) / MR / LR / B / T / C / DM / EM / FM / ZF / W / TM / CM)の後に.□(□:0~15の数値)を続けて入力します。

例

DM0.0　(DM0の1ビット目を意味します)
EM10.4　(EM10の5ビット目を意味します)

「ワードデバイスのビットデバイス扱い」(3-20)

●サフィックスの記述

デバイス(R(DR)/ MR / LR / B / T / C / DM / EM / FM / ZF / W / TM / CM / ＊DM / ＊EM / ＊FM / ＊ZF / ＊W / ＊TM)の後に.□(□:U,S,D,L,F,DF,B,T)を続けて入力します。

例

DM0.F　(DM0～DM1の値を浮動小数点型データとして演算します)
R1000.U　(R1000～R1015までの16ビットデータとして演算します)

ポイント　**サフィックスの記述はST/スクリプト内でのみ使用できます。**

●定数値の型指定

STでは、10 進定数にサフィックスを付けることで、定数がもてる範囲を設定することができます。

型	サフィックス	データの範囲	記述例
16ビット符号無し	UINT#	0～65535	UINT#910, UINT#505
16ビット符号付き	INT#	－32768～＋32767	INT#23, INT#-600
32ビット符号無し	UDINT#	0～4294967295	UDINT#1234567, UDINT#5
32ビット符号付き	DINT#	－2147483648～＋2147483647	DINT#123, DINT#-98335963
32ビット単精度浮動小数点型	REAL#	－3.4E＋38≦N≦－1.4E－45 N＝0 ＋1.4E－45≦N≦＋3.4E＋38	REAL#100, REAL#-3.4E38, REAL#0.12
64ビット倍精度浮動小数点型	LREAL#	－1.79E＋308　≦N≦－2.23E－308 N＝0 ＋2.23E－308　≦N≦＋1.79E＋308	LREAL#100, LREAL#-1.7E123, LREAL#0.0256

ビットデバイス

■ビットの状態

ビットデバイスの状態を示すのに、以下の表記を用いることができます。

ビット状態	表　記	記述例
ON（真）	1、TRUE	MR1001 := 1;、MR1002 := TRUE;
OFF（偽）	0、FALSE	MR2001 := 0;、MR2002 := FALSE;

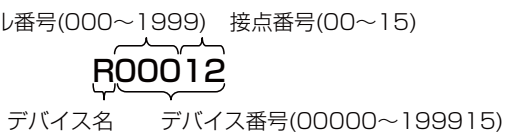
参 考　これら2状態を「ブール値」といいます。

入力/出力リレー　R

概要　入力リレーは、外部機器からPLCにON/OFF情報を取り込むためのデバイスです。
出力リレーは、PLCからPLCの外部機器へON/OFF指令や演算結果を送り出すためのデバイスです。

表記　デバイス番号は、次のような構成になっています。

例）



参 考
- ゼロサプレスして記述できます。
  例）R12
- サフィックスを付けて記述できます。
  例）R12.B
- サフィックスを省略して記述したとき、型宣言がなければ.B（ビット型）として扱います。
- ワードデバイスとして扱えます。
  例）R2000.U（R2000を先頭に16ビットを占有します）
- デバイス名に「D」を付けるとダイレクト入出力リレーとして動作します。
  例）DR500（R500はダイレクト出力リレーとして動作します）

「ダイレクト入出力リレー」（3-21）

範囲
- R000～R199915

用途　入力リレー
- 接続する外部機器として、ボタンやスイッチ（切り換えスイッチ、リミットスイッチ、光電スイッチ、デジタルスイッチ）などがあります。

出力リレー
- 接続する外部機器として、ソレノイドや電磁開閉器、表示器（信号灯、デジタル表示器）などがあります。

詳細
- 入力端子1点につき入力リレーが1ビット割り当てられます。
- 出力端子1点につき出力リレーが1ビット割り当てられます。
- a接点（ノーマルオープン）、b接点（ノーマルクローズ）のどちらも指定できます。
- 同じデバイス番号の入力リレーを、1プロジェクト内で何度でも記述できます。

初期状態　出力リレーは、デフォルトで、電源ON時、PROG→RUN時に状態がクリアされてOFFになります。

参 考

・リレー(R)のユニット用デバイス表記について
《KV STUDIO》では、通常の表記以外に、ユニット番号を基準としたリレー表記ができます。ユニット用デバイス表記を使用するには、メニューから【表示(V)】→【ユニット用デバイス表示(Y)】を選択します。

ユニット番号　：CPUを先頭に00、01、02、03、04…と、ユニットの接続順に割り付けられます。

ユニットリレー番号：00000から占有するリレー点数分がリレー番号として割り付けられます。

ユニット番号	00	01		02	03	04	05
KV-PU1	CPU ユニット	KV-SIR32XT 入力32点 出力32点	KV-7000C	KV-B16XC 入力16点	KV-C32TD 出力32点	KV-C32XTD 入力32点 出力32点	KV-C64TD 出力64点
リレー番号		0000～0315		0000～0015	0000～0015	0000～0315	0000～0315

リンクリレー　B

概要　EtherNet/IPやFL-net接続時などで、リンクリレーとして使用します。他のリレーと異なり、16進表記でデバイス番号を指定します。

表記　デバイス番号は、次のような構成になっています。

例）

参 考

・ゼロサプレスして記述できます。
例）B7A
・サフィックスを付けて記述できます。
例）B7A.B
・サフィックスを省略して記述したとき、型宣言がなければ.B（ビット型）として扱います。
・ワードデバイスとして扱えます。
例）B80.U（B80を先頭に16ビットを占有します）

ポイント

- **16進数（0～F）で指定してください。**
- **ローカルデバイス（@）は、使用できません。**

範囲　B0000～B7FFF

用途　EtherNet/IPやFL-net接続時などでは、リンクリレーとして使用できます。

詳細
・拡張ユニットのユニットデバイスに割り付けできません。
・同じデバイス番号のリンクリレーBを、1プロジェクト内で何度でも記述できます。

初期状態　リンクリレーは、デフォルトで、電源ON時、PROG→RUN時に状態がクリアされてOFFになります。CPUシステム設定の［停電保持設定］により状態を保持する設定が可能です。

内部補助リレー　R・B・MR・LR

■R

概要　実入出力リレーと拡張ユニットリレーに割り当てていないRを内部補助リレーとして使用できます。

表記　デバイス番号は、次のような構成になっています。

例）

チャンネル番号(000～1999)　接点番号(00～15)

R00012

デバイス名　デバイス番号(00000～199915)

参考
- ゼロサプレスして記述できます。
  例）R12
- サフィックスを付けて記述できます。
  例）R12.B
- サフィックスを省略して記述したとき、型宣言がなければ.B（ビット型）として扱います。
- ワードデバイスとして扱えます。
  例）R2000.U（R2000を先頭に16ビットを占有します）

範囲
- R000～R199915（拡張入力ユニットの番号として割り付け可能）

■B

概要　EtherNet/IPやFL-netに接続しない場合や、リンクリレーに割り当てていないBを内部補助リレーとして使用できます。

表記　デバイス番号は、次のような構成になっています。

例）

チャンネル番号(000～7FF)　接点番号(0～F)

B1EE7

デバイス名　デバイス番号(0000～7FFF)

参考
- ゼロサプレスして記述できます。
  例）B7A
- サフィックスを付けて記述できます。
  例）B7A.B
- サフィックスを省略して記述したとき、型宣言がなければ.B（ビット型）として扱います。
- ワードデバイスとして扱えます。
  例）B80.U（B80を先頭に16ビットを占有します）

ポイント
- **16進数（0～F）で指定してください。**
- **ローカルデバイス（@）は、使用できません。**

範囲　B0000～B7FFF

詳細
- 拡張ユニットのユニットデバイスに割り付けできません。
- 同じデバイス番号のリンクリレーBを、1プロジェクト内で何度でも記述できます。

初期状態　デフォルトでは、電源ON時、PROG→RUN時に状態がクリアされてOFFになります。CPUシステム設定の［停電保持設定］により状態を保持する設定が可能です。

■MR

概要　CPU内部のみで使用できるデバイスです。

表記　デバイス番号は、次のような構成になっています。

例）　チャンネル番号(000～3999) 接点番号(00～15)

MR02809

デバイス名　デバイス番号(00000～399915)

参考
- ゼロサプレスして記述できます。
  例）MR2809
- サフィックスを付けて記述できます。
  例）MR2809.B
- サフィックスを省略したとき、型宣言がなければ.B（ビット型）として扱います。
- ワードデバイスとして扱えます。
  例）MR2900.U（MR2900を先頭に16ビットを占有します）

範囲　MR00000～MR399915

ポイント　**MR200000～MR399915、ローカルデバイスに割り当てられているため使用できません。（CPUシステム設定の［ローカル全体割付設定］で予約領域を変更できます）**

詳細
- 拡張ユニットのユニットデバイスに割り付けできません。
- 同じデバイス番号の内部補助リレーMRを、1プロジェクト内で何度でも記述できます。

初期状態　電源ON時、PROG←→RUN時に状態がクリアされてOFFになります。

■LR（ラッチリレー）

概要　CPU内部のみで使用でき、状態を保持するデバイスです。

表記　デバイス番号は、次のような構成になっています。

例）　チャンネル番号(000～999)　接点番号(00～15)

LR02308

デバイス名　デバイス番号(00000～99915)

参考
- ゼロサプレスして記述できます。
  例）LR2308
- サフィックスを付けて記述できます。
  例）LR2308.B
- サフィックスを省略したとき、型宣言がなければ.B（ビット型）として扱います。
- ワードデバイスとして扱えます。
  例）LR2400.U（LR2400を先頭に16ビットを占有します）

範囲　LR00000～LR99915

詳細
- 拡張ユニットのユニットデバイスに割り付けできません。
- 同じデバイス番号のラッチリレーLRを、1プロジェクト内で何度でも記述できます。

初期状態　前回電源をOFFしたときの直前の状態が保持されます。

タイマ（接点） T

概要　タイマは、接点（1ビット）、現在値（32ビット）を持ちます。減算式タイマの現在値が0と一致（タイムアップ）すると、接点がONするデバイスです。

表記　デバイス番号は、次のような構成になっています。

例）

デバイス番号（0000～3999）

T0501.B

デバイス名　サフィックス

参考
- ゼロサプレスして記述できます。
  例）T501.B
- サフィックスを付けて記述できます。タイマ接点として扱うときは.Bを、タイマの現在値を扱うときは.Dを付けてください。
- サフィックスを省略したとき、型宣言がなければ.B（ビット型）として扱います。

範囲　T0000.B～T3999.B

詳細　タイマ関数には10μsタイマ･1msタイマ･10msタイマ･100msタイマの4種類があります。

初期状態　接点は電源ON時、PROG→RUN時に状態がクリアされてOFFになります。
現在値は電源ON時、PROG→RUN時に設定値と同じ値になります。
設定値は保持されます。

■STでタイマ関数を動作させる時の記述方法

タイマ関数の各引数に、使用するタイマの実行条件（タイマのリセット用リレー）、タイマ番号、設定値（タイムアップするまでの時間）を指定することで、タイマの動作を記述します。

例）

タイマデバイス（T0000～T3999）　設定値（0～4294967295, デバイス番号）

TMR　(R1000, T501, 1000)

タイマ関数名（100ms減算タイマ）　タイマ実行条件

【タイマの種類による動作の違い】

タイマの種類（関数）	デバイス番号	設定値	ONするまでの時間
100ms減算タイマ（TMR）	T0～T3999	0～4294967295	n×100ms
10ms減算タイマ（TMH）			n×10ms
1ms減算タイマ（TMS）			n× 1ms
10μs減算タイマ（TMU）			n×10μs

参考　設定値にはデバイス（32ビット符号無しデータ）や式を指定できます。
「TMR関数」（5-13ページ）

■タイマの誤差

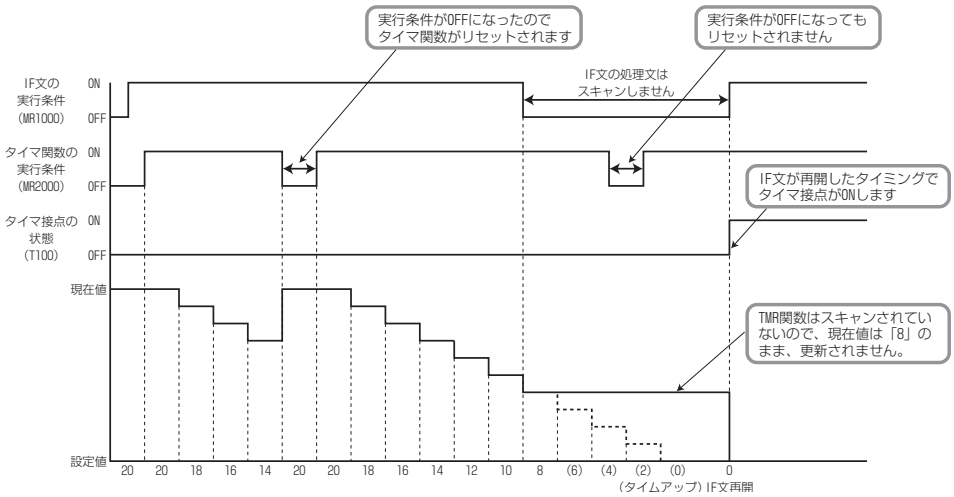
タイマの誤差は、次のとおりです。

TMR（100msタイマ）　：±（100ms+1スキャンタイム）以内
TMH（10msタイマ）　：±（10ms+1スキャンタイム）以内
TMS（1msタイマ）　：±（1ms+1スキャンタイム）以内
TMU（10μsタイマ）　：±（10μs+1スキャンタイム）以内

■タイマ関数使用時の注意

IF文、ボックス型STでタイマ関数を使用する場合、タイマ関数実行中にIF文、ボックス型STを実行しなくすると、タイマの現在値は更新されません。
時間計測は引き続きされており、各命令間の実行が再開した時に経過した時間がタイマの現在値に反映されます。設定時間が経過していた場合には、通常は実行を再開した直後にタイマ接点がONします。

ST記述
```
IF MR1000 = TRUE THEN
    TMR(MR2000 = TRUE, T100, 200);
END_IF;
```



■タイマの設定値・現在値を扱う方法

ラダー中のタイマの設定値･現在値をSTで読み出したり、変更する方法を以下の表に示します。

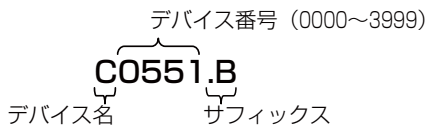
		読み出し方法	変更方法
現在値		格納データ:=CURVAL(タイマデータ); 例)udiValue1 := CURVAL(timer);	現在値は変更できません。 TMR関数などで初期化してください。
設定値	デバイス指定	格納データメモリ＝設定値格納デバイス	設定値格納デバイス＝変更する値(定数、デバイス)
	定数	設定値は読み出せません。	設定値は変更できません。

ポイント　**変更時に現在値＞設定値となる場合、現在値は設定値と同じ値に変更されます。**

カウンタ（接点）　C

概要　カウンタは、接点(1ビット)、現在値(32ビット)を持ちます。
カウンタの現在値が設定値と同じになる(カウントアップする)と、接点がONするデバイスです。

表記　デバイス番号は、次のような構成になっています。
例）



参考
- ゼロサプレスして記述できます。
  例）C551.B
- サフィックスを付けて記述できます。タイマ接点として扱うときは.Bを、タイマの現在値を扱うときは.Dを付けてください。
- サフィックスを省略したとき、型宣言がなければ.B(ビット型)として扱います。

範囲　C0000.B～C3999.B

■STでカウンタ関数を動作させる時の記述方法

カウンタ関数の引数に使用するカウンタの実行条件（カウンタリセット）、カウンタ番号、設定値、カウント入力リレーを指定することでカウンタの動作を記述します。

カウンタデバイス番号　設定値

CNT（R1000，C600，50，MR1000）

カウンタ関数　カウンタ実行条件　カウント入力リレー

カウンタ関数は、カウント入力リレーがOFF→ONに変化（微分型）すると現在値を1ずつ加算し、現在値が設定値に達したらカウンタ接点をONします。

参考　設定値にはデバイス（32ビット符号無しデータ）や式を指定できます。
「CNT関数」（5-16ページ）

■カウンタの設定値・現在値を扱う方法

ラダー中のカウンタの設定値･現在値をSTで読み出したり、変更する方法を以下の表に示します。

		読み出し方法	変更方法
現在値		格納データ:=CURVAL（カウンタデータ）; 例)udiValue1 := CURVAL(counter);	現在値は変更できません。 CNT関数で初期化してください。
設定値	デバイス指定	格納データメモリ＝設定値格納デバイス	設定値格納デバイス＝変更する値（定数、デバイス）
	定数	設定値は読み出せません。	設定値は変更できません。

ポイント　**変更時に現在値＞設定値となる場合、現在値は設定値と同じ値に変更されます。**

コントロールリレー（接点）　CR

概要　PLCの機能を制御したり、PLCの状態を知るために使用する特殊な内部補助リレーです。

範囲　CR0000～CR7915

参考
- サフィックスを付けて記述できます。
  例）CR2002.B
- サフィックスを省略したとき、型宣言がなければ.B（ビット型）として扱います。

詳細　「CR/CM一覧」（付-2ページ）

ワードデバイス

データメモリ/拡張データメモリ　DM/EM/FM

概要　データメモリと拡張データメモリは、関数が演算する際に参照するデータを保存したり、演算の結果を格納するための1点当たり16ビットのデバイスです。

表記　デバイス番号は、次のような構成になっています。

例）

デバイス名　デバイス番号

DM06143

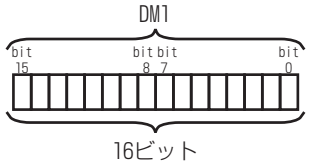
参 考
- ゼロサプレスして記述できます。
  例）DM6143
- サフィックスを付けて記述できます。
  例）DM1000.S（16ビット符号付き）、DM2000.D（32ビット符号なし）
- サフィックスを省略して記述したとき、型宣言がなければ.U（16ビット符号なし）として扱います。
- ビットデバイスとして扱えます。
  例）DM1100.B DM1100の最下位ビットを扱います
- ビット位置を指定したビットデバイスとして扱えます。
  例）DM1200.12DM1200の13ビット目を扱います

範囲
- DM00000～DM65534
- EM00000～EM65534
- FM00000～FM32766

詳細
- データメモリ（DM）のみ拡張ユニットのユニットデバイスに割り付けできます。

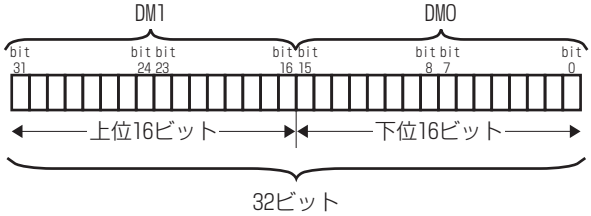
●16ビットデータを扱うとき

- デバイス1点のビット幅は16ビットで、符号無し10進数として扱う時は0～65535、符号付き10進数として扱う時は–32768～+32767、16進数として扱う時は16#0～16#FFFFの値が扱えます。
- 16ビットデータを扱う時は、デバイス番号に.U（省略可能）あるいは.Sのサフィックスを付けます。



●32ビットデータを扱うとき

- デバイス2点を使用して32ビットのデータを扱うこともできます。32ビットの場合、指定したデバイス番号のデータメモリに下位16ビット、その次のデバイス番号のデータメモリに上位16ビットを格納します。符号無し10進数として扱う時は0～4294967295、符号付き10進数として扱う時は–2147483648～+2147483647、16進数として扱う時は16#0～16#FFFFFFFFの値が扱えます。
- 32ビットデータを扱う時は、下位16ビットを示しているデバイス番号に.Dあるいは.Lのサフィックスを付けます。

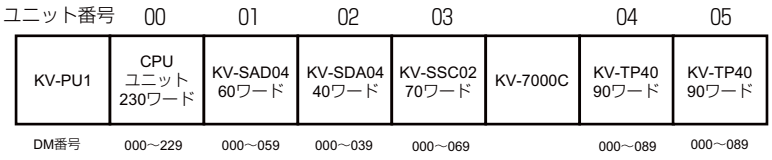


初期状態 デフォルトでは、値が保持されます。
- CPUシステム設定の［停電保持設定］により状態をクリアする（保持しない）設定が可能です。

参 考
- データメモリ（DM）のユニット用デバイス表記について
  《KV STUDIO》では、通常の表記以外に、ユニット番号を基準とした表記ができます。
  ユニット用デバイス表記を使用するには、メニューから【表示(V)】→【ユニット用デバイス表示(Y)】を選択します。

ユニット番号：	CPUを先頭に00、01、02、03、04･･･と、ユニットの接続順に割り付けられます。
ユニットDM番号：	0000から占有するデータメモリ分が割り付けられます。
サフィックス：	サフィックスを付けて記述できます。省略した場合は16ビット符号無し(.U)として扱われます。

例）



KV-SAD04の5番目 ..... DM01_0005
KV-TP40（ユニット番号04）の10番目 ..... DM04_0009
KV-TP40（ユニット番号05）の2番目 ..... DM05_0001

ファイルレジスタ　ZF/FM

概要　ファイルレジスタは、すべての領域を連番方式で扱うZFと、領域を複数（16または4）のバンクに分割してバンク切り換え方式で扱うFMがあります。
詳細は使用しているCPUユニットのユーザーズマニュアルの「ファイルレジスタ」を参照してください。

表記　デバイス番号は、次のような構成になっています。

例）

デバイス名　デバイス番号

ZF0372

参考
- ゼロサプレスして記述できます。
  例）ZF372
- サフィックスを付けて記述できます。
  例）ZF1000.S（16ビット符号付き）、FM12300.D（32ビット符号なし）
- サフィックスを省略して記述したとき、型宣言がなければ.U（16ビット符号なし）として扱います。
- ビットデバイスとして扱えます。
  例）FM1100.BFM1100の最下位ビットを扱います
- ビット位置を指定したビットデバイスとして扱えます。
  例）ZF1200.12ZF1200の13ビット目を扱います

ポイント
- **ZFのローカルデバイス（@ZFxxx）は、使用できません。**
- **FMのローカルデバイス（@FM0～@FM32767）は、バンク切り換えできません。**
- **FMのローカルデバイス（@FM0～@FM32767）は、ワーク領域を使用します。**

範囲
- ZF000000～ZF524287
- FM00000～FM32767

「データメモリ/拡張データメモリ　DM/EM/FM」（3-13）

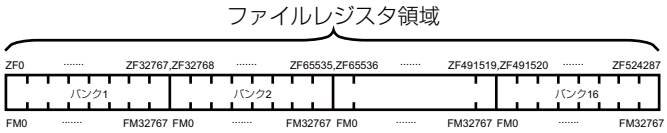
詳細
- 拡張ユニットのユニットデバイスに割り付けできません。
- 連番方式とバンク切り換え方式では、デバイスの種別と範囲が異なります。

●ZF：連番方式でファイルレジスタを扱うとき

ファイルレジスタの全領域を、1つのデバイスとして扱います。

●FM：バンク切り換え方式でファイルレジスタを扱うとき

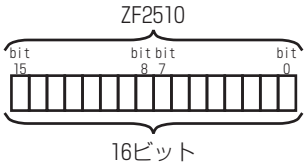
ファイルレジスタの領域を16分割し、バンクを切り換えて扱います。



バンクを切り換えることで、1つのデバイスを16の異なるデバイスとして使用できます。
ただし、ZFデバイスと重複して使用できるため、データの取り扱いには注意が必要です。
バンクの切り換え方法については、使用しているCPUユニットのユーザーズマニュアルの「ファイルレジスタ」を参照してください。

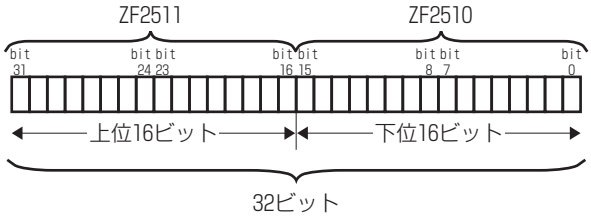
●16ビットデータを扱うとき

- デバイス1点のビット幅は16ビットで、符号無し10進数として扱う時は0～65535、符号付き10進数として扱う時は –32768～+32767、16進数として扱う時は16#0～16#FFFFの値が扱えます。
- 16ビットデータを扱う時は、デバイス番号に.U（省略可能）あるいは.Sのサフィックスを付けます。



●32ビットデータを扱うとき

- デバイス2点を使用して32ビットのデータを扱うこともできます。32ビットの場合、指定したデバイス番号のデータメモリに下位16ビット、その次のデバイス番号のデータメモリに上位16ビットを格納します。符号無し10進数として扱う時は0～4294967295、符号付き10進数として扱う時は–2147483648～+2147483647、16進数として扱う時は16#0～16#FFFFFFFFの値が扱えます。
- 32ビットデータを扱う時は、下位16ビットを示しているデバイス番号に.Dあるいは.Lのサフィックスを付けます。



初期状態　デフォルトでは、値が保持されます。
・ CPUシステム設定の[停電保持設定]により状態をクリアする（保持しない）設定が可能です。

リンクレジスタ　W

概要　EtherNet/IPやFL-net接続時などで、リンクレジスタとして使用します。他のワードデバイスと異なり、16進表記でデバイス番号を指定します。

表記　デバイス番号は、次のような構成になっています。

例）

デバイス名　デバイス番号（0～7FFF）

W3A6F

参考
・ ゼロサプレスして記述できます。
例）W7A
・ サフィックスを付けて記述できます。
例）W7A.S（16ビット符号付き）、W3E76.D（32ビット符号無し）
・ サフィックスを省略して記述したとき、型宣言がなければ.U（16ビット符号無し）として扱います。
・ ビットデバイスとして扱えます。ビット位置の指定も可能です。
例）W1200.12W1200の13ビット目を扱います

ポイント
- **16進数(0～F)で指定してください。**
- **ローカルデバイス(@)は、使用できません。**

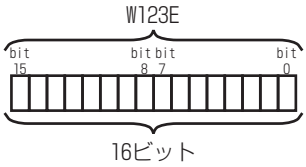
範囲　W0000～W7FFF

用途　EtherNet/IPやFL-net接続時などでは、リンクレジスタとして使用できます。

詳細　拡張ユニットのユニットデバイスに割り付けできません。

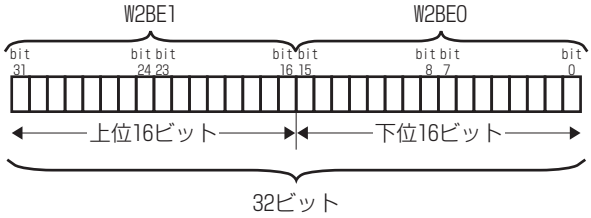
●16ビットデータを扱うとき

- デバイス1点のビット幅は16ビットで、符号無し10進数として扱う時は0～65535、符号付き10進数として扱う時は –32768～+32767、16進数として扱う時は16#0～16#FFFFの値が扱えます。
- 16ビットデータを扱う時は、デバイス番号に.U(省略可能)あるいは.Sのサフィックスを付けます。



●32ビットデータを扱うとき

- デバイス2点を使用して32ビットのデータを扱うこともできます。32ビットの場合、指定したデバイス番号のデータメモリに下位16ビット、その次のデバイス番号のデータメモリに上位16ビットを格納します。符号無し10進数として扱う時は0～4294967295、符号付き10進数として扱う時は–2147483648～+2147483647、16進数として扱う時は16#0～16#FFFFFFFFの値が扱えます。
- 32ビットデータを扱う時は、下位16ビットを示しているデバイス番号に.Dあるいは.Lのサフィックスを付けます。



初期状態　デフォルトでは、値が保持されます。
CPUシステム設定の[停電保持設定]により状態をクリアする(保持しない)設定が可能です。

テンポラリデータメモリ　TM

概要　テンポラリデータメモリは、データメモリと同じように扱うことができる1点当たり16ビットのワードデバイスです。一部のデバイス領域は、演算のためのデータや演算の結果データを一時格納するためにも使用します。

表記　デバイス番号は、次のような構成になっています。

例)　デバイス名　デバイス番号

TM010

参考
- TM000～TM003はラダー中のDIV、MULなどの演算命令で、また、TM000、TM001は32ビットを超えるデータを扱うときに使用するため、それ以外の用途で使用しないことをおすすめします。
- ゼロサプレスして記述できます。
  例) TM10
- サフィックスを付けて記述できます。
  例) TM10.D(32ビット符号無し)
- サフィックスを省略したとき、型宣言がなければ.U(16ビット符号無し)として扱います。

範囲　TM000～TM511

TM番号	用　途
TM000･TM001	64ビットデータ用
TM002･TM003	除算の余り格納用
TM004～TM511	汎用※

※ STで簡易間接指定は使用できません。

初期状態　電源ON時、PROG→RUN時に値がクリアされて0になります。

タイマ（現在値）　T

概要　タイマデバイスにサフィックス.Dを付けることで、現在値を扱います。

範囲　T0000.D～T3999.D

参考　タイマデバイスを現在値として扱うときは、サフィックス.Dを省略できません。サフィックスを省略した場合、型宣言がなければ.B（タイマ接点）として扱います。

詳細　タイマデバイスの現在値は、0～4294967295の範囲で変更できます。

ポイント　**現在値変更時に現在値＞設定値となる場合、現在値は設定値と同じ値に変更されます。**

「タイマ（接点）　T」（3-11）

カウンタ（現在値）　C

概要　カウンタデバイスにサフィックス.Dを付けることで、現在値を扱います。

範囲　C0000.D～C3999.D

参考　カウンタデバイスを現在値として扱うときは、サフィックス.Dを省略できません。サフィックスを省略した場合、型宣言がなければ.B（カウンタ接点）として扱います。

詳細　カウンタデバイスの現在値は、0～4294967295の範囲で変更できます。

ポイント　**現在値変更時に現在値＞設定値となる場合、現在値は設定値と同じ値に変更されます。**

「カウンタ（接点）　C」（3-12）

インデックスレジスタ　Z

概要　インデックス修飾する際に、対象デバイス番号に加える数を指定するための1点当たり32ビットのワードデバイスです。データメモリと同じように、データを保存したり、演算の結果を格納できます。

範囲　CPU機能バージョン2.0以降のKV-8000：Z1～Z23
上記以外の機種：Z1～Z12

参考
- サフィックスを付けて記述できます。
  例）Z01.L（32ビット符号付き）
  Z02.D（32ビット符号なし）
- サフィックスを省略して記述したとき、型宣言がなければ.L（32ビット符号付き）として扱います。

ポイント
- **Z11以降は、システムで予約されているため使用できません。**
- **サフィックスに.F、.DF、.T、.Bは指定できません。**
- **ワードデバイスのビット指定はできません。**
- **ローカルデバイスはありません。ZPUSH、ZPOP関数を使用してください。**
  「ZPUSH」(6-7)
  「ZPOP」(6-8)

初期状態 電源ON時、PROG→RUN時に値がクリアされて0になります。
「インデックス修飾」(3-25)

コントロールメモリ　CM

概要 PLCの機能を制御したり、PLCの状態を知るために使用する特殊なワードデバイスです。

範囲 CM0000～CM7599

参考
- サフィックスを付けて記述できます。
  例) CM1720.U
- サフィックスを省略したとき、型宣言がなければ.U(16ビット符号無し)として扱います。

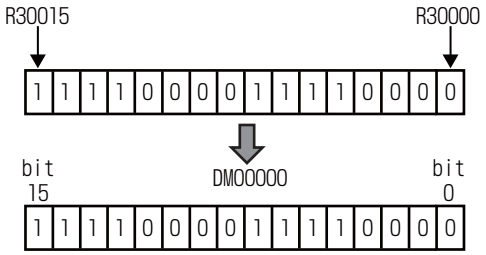
詳細 「CR/CM一覧」(付-2ページ)

ビットデバイスのワード単位扱い

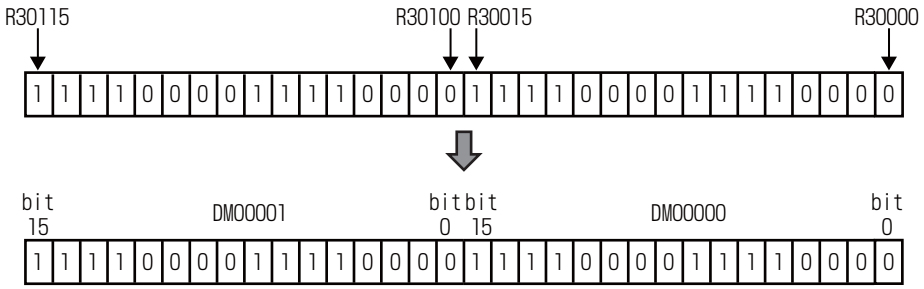
ワードデバイスを使用する関数や代入文などに、.B以外のサフィックスを付加したビットデバイスを指定すると、指定したビットデバイスを先頭としたワード単位に扱われます。

例)

DM0000.U := R3000.U;

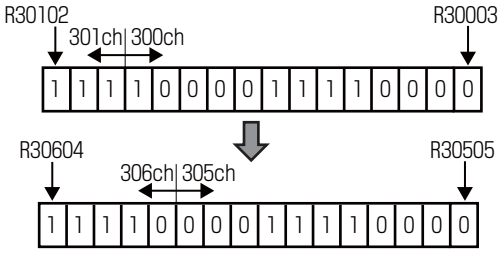


DM0000.D := R3000.D;



サフィックスを付加したビットデバイスがチャンネルの先頭以外の時は、次のチャンネルにまたがったワード単位として扱われます。

R30505.U := R30003.U;



ワードデバイスのビットデバイス扱い

通常、ワードデバイスは、16ビット/32ビットのデバイスとして処理されますが、ビット型のサフィックスを付加することで、ビットデバイスとして扱えます。

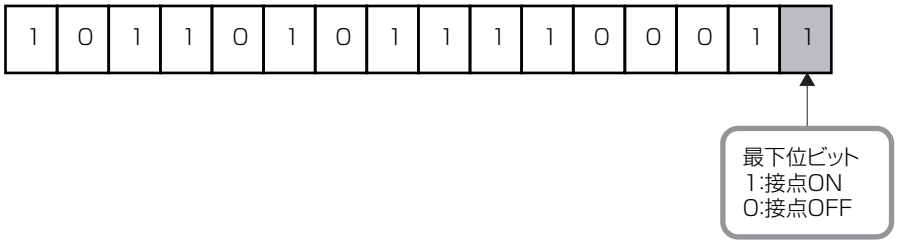
ポイント　**インデックスレジスタ(Z)は、ビットデバイス扱いできません。**

●最下位ビットでビットデバイスとして扱う

ワードデバイスに.Bサフィックスを付加したり、ビットデバイスを処理する接点･出力関数の引数や代入文などにワードデバイスを指定すると、指定したワードデバイスの最下位ビットの状態でビットデバイスとして扱います。

DM1000の値が46563(16#B5E3)の場合

DM1000 := 46563(16#B5E3);



DM1000の最下位ビットが1なので、DM1000.Bは「ON」として扱います。

●任意のビット位置でビットデバイスとして扱う

ワードデバイスに「.0～.15」のビット位置を指定すると、ワードデバイスの指定したビット位置の状態でビットデバイスとして扱います。

表記　例）

DM1234.12

任意のワードデバイス　ビット位置（.0～.15）

参考
- DM123.BとDM123.0は、同じビットを扱います。
- デバイスコメントは、ワードデバイスに対して設定されたコメントが表示されます。

ポイント
- **ビット位置は、10進数(0～15)で指定してください。**
- **ローカルデバイス(@)は、使用できません。**
- **ビット指定可能なデバイスは、「DM、W、EM、FM、ZF、TM、CM」のみです。**
- **ビット指定可能な変数は、データ型が「INT、DINT、UINT、UDINT」の変数、およびこれらのデータ型に設定した配列型変数の要素、構造体型変数のメンバのみです。**
- **ビット位置指定されたデバイスは、間接参照やインデックス修飾できません。**

DM123の値が59(16#3B)の場合

bit 15	bit 14	bit 13	bit 12	bit 11	bit 10	bit 9	bit 8	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
0	0	0	0	0	0	0	0	0	0	1	1	1	0	1	1

DM123をビット位置指定でビットデバイスとして扱った場合、各ビットの状態は以下の通りです。

デバイス	ビットの状態
DM123.0	ON
DM123.1	ON
DM123.2	OFF
DM123.3	ON
DM123.4	ON
DM123.5	ON
DM123.6～DM123.15	OFF

ダイレクト入出力リレー

入出力リレー(R)のデバイス番号の先頭に「D」を付けることで、ダイレクト入出力リレーとして動作します。

表記　例)

ダイレクト入出力リレーの表記

DR001

入出力リレー

ポイント　**.B以外のサフィックスは付加できません。**

範囲　入出力リレー

変数

変数とは、データ型と名前が定義された、プログラミングで使用できる要素のことを言います。KV STUDIOでは、プログラミングにデバイスもしくは変数を使用できます。
変数は、デバイスと異なりKV STUDIOで定義した名前でプログラミングでき、定義した変数は自動的にCPUユニットのワーク領域に割り付きます。そのため、各機能が割り当たっているデバイス番号を管理するためだけにデバイスマップを作成するなどの作業が不要になります。また、VT5シリーズおよびSoft-VTも変数に対応しているため、PLC側のデバイス番号に合わせてタッチパネル側のデバイス番号を変更する作業も不要になります。

ポイント
- **変数を使用できるのは、KV STUDIO Ver11.0以降で、対応機種がCPU機能バージョン2.0以降のKV-8000の場合のみです。**
- **タッチパネルから変数を読み書きできるのは、VT　STUDIO　Ver.8.0以降で、対応機種がVT5シリーズで、PLC機種がKV-8000(イーサネット)の場合のみです。**

変数の設定方法については、「変数プログラミングと配列･構造体」(2-15)を参照してください。

変数とデータ型一覧

変数を使用する場合、あらかじめデータ型を定義する必要があります。CPUユニットは定義されたデータ型に応じて必要なデータサイズを自動的にワーク領域に割り付けます※1。
≪KV-8000≫で使用できるワーク領域のサイズと、変数の仕様は以下のとおりです。

項　目		内　容
ワーク領域のサイズ	ビット	64000点
	ワード	589824ワード
	タイマ	4000点
	カウンタ	4000点
変数の仕様	定義可能数	100000
	変数名の最大長	128文字

※ 1 ワーク領域ではなく、CPUユニットのデバイスに手動で割り付けることも可能です。詳細は『KV-8000シリーズ　ユーザーズマニュアル』を参照してください。

≪KV-8000≫で使用できる変数のデータ型と、データサイズは以下のとおりです。

データ型		データサイズ(ビット単位)
型　名	説　明	
BOOL	ビット	1
UINT	1ワード符号なし整数	16
INT	1ワード符号あり整数	16
UDINT	2ワード符号なし整数	32
DINT	2ワード符号あり整数	32
REAL	単精度浮動小数点型	32
LREAL	倍精度浮動小数点型	64
STRING	文字列	(文字数+1)×8 を16の整数倍に切り上げ
タイマ	タイマ	―
カウンタ	カウンタ	―
配列	―	データ型 × 配列サイズ
構造体	―	定義による※1
ファンクションブロック	ファンクションブロック	定義による※1

※1　構造体およびファンクションブロックは複数のデータ型を設定して使用するデータ型です。そのため、設定内容によりデータサイズが決まります。詳細は『KV-8000シリーズ　ユーザーズマニュアル』を参照してください。

定数

使用する関数やデバイスに付属するサフィックス、変数のデータ型によって、表記可能な定数やその範囲が変わります。

10進定数

概要　16ビットと32ビットのビット幅で表される数値を10進表記で示します。

表記　10進表記の数値をそのまま表記します。(符号付き表記も可能です)

例)

数値
06143

参 考
- ゼロサプレスして記述できます。
- 符号付きの場合、「+」のみ符号の省略が可能です。

範囲
- 16ビット符号無しの場合：0～65535
- 32ビット符号無しの場合：0～4294967295
- 16ビット符号付きの場合：–32768～+32767
- 32ビット符号付きの場合：–2147483648～+2147483647

参 考 定数にサフィックスを付けて記述することで、定数の範囲が拡張します。（通常記述では定数にサフィックスを付ける必要はありません）

UINT#300　:16ビット符号無し（0～65535）
INT#500　:16ビット符号付き（–32768～+32767）
UDINT#3000　:32ビット符号無し（0～4294967295）
DINT#5000　:32ビット符号付き（–2147483648～+2147483647）
REAL#123　:単精度浮動小数点型（–3.4E+38≦N≦1.4E–45
N＝0
1.4E–45≦N≦3.4E+38）
（有効桁数：約7桁）
LREAL#1.2345　:倍精度浮動小数点型（–1.79E+308≦N≦–2.23E–308
N＝0
2.23E–308≦N≦1.79E+308）
（有効桁数：約16桁）

16進定数

概要　16ビットと32ビットの2種類のビット幅で表される数値を16進表記で示します。

表記　先頭に「16#」、続いて16進表記の数値を記述します。

例）

16進数を表す記号　数値

16#1AF7

範囲
- 16ビットの場合：16#0~16#FFFF
- 32ビットの場合：16#0~16#FFFFFFFF

ポイント
- **「16#」をつけずにA～Fを含む16進数を記述した場合はエラーになり、A～Fを含まない場合は10進数と認識されます。**
- **16進定数は符号を付けて記述できません。**

単精度浮動小数点型定数

概要　単精度浮動小数点型実数として記述できます。

表記
- 符号を付けて10進表記の数値をそのまま表記します。

例）

数値

1234.56

- 8桁以上で指数表記になります。
- 10進定数にサフィックスを付けて表記します。

例）

型指定　数値

REAL#1234

参 考　サフィックスを付けずに表現することも可能です。

例）　**1234.0**

単精度浮動小数点型を示す

範囲　−3.4E+38≦N≦−1.4E−45
N=0
1.4E−45≦N≦+3.4E+38（有効桁数：約7桁）

倍精度浮動小数点型定数

概要　倍精度浮動小数点型実数として記述できます。

表記
- 符号を付けて10進表記の数値をそのまま表記します。

例）　数値

12345.6789

- 17桁以上で指数表記になります。
- 10進定数にサフィックスを付けて表記します。

例）　型指定　数値

LREAL#1234

範囲　−1.79E+308≦N≦−2.23E−308
N=0
+2.23E−308≦N≦+1.79E+308（有効桁数：約16桁）

固定文字列（文字列型定数）

概要　任意の文字列を「'」で囲むことで記述できます。

表記　「' '」の間に文字列を表記します。

例）　文字列

'演算結果'

参 考
- 「'」を文字として扱うには、「$'」と記述します。
- 「'」を他の文字に付けて扱う場合は、文字を「+」で結合して使用します。

strValue := '$'' + '演算結果' +'$'';　strValueには「'演算結果'」が格納されます。

範囲　扱える文字列の最大は1999文字です。

インデックス修飾と間接指定

インデックス修飾と間接参照で使用できるデバイスは以下のとおりです。

デバイス			KV-8000シリーズ	
種類	記号	名称	インデックス修飾	間接指定
ビットデバイス	R	リレー	○	○※
	CR	コントロールリレー	―	―
	T	タイマ(接点)	○	○※
	C	カウンタ(接点)	○	○※
	MR	内部補助リレー	○	○※
	LR	ラッチリレー	○	○※
	B	リンクリレー	○	○※
ワードデバイス	DM	データメモリ	○	○
	EM	拡張データメモリ	○	○
	FM	ファイルレジスタ(バンク方式)	○	○
	ZF	ファイルレジスタ(連番方式)	○	○
	W	リンクレジスタ	○	○
	TM	テンポラリメモリ	○	○
	TS	タイマ(設定値)	○	○
	TC	タイマ(現在値)	○	○
	CS	カウンタ(設定値)	○	○
	CC	カウンタ(現在値)	○	○
	CM	コントロールメモリ	―	―

※間接指定はできますが、「*(デバイス)」と記述してアドレスの格納はできません。

ポイント
- **CR、Z、変数(変数配列)、およびワードデバイスのビット指定デバイスはインデックス修飾できません。**
- **CR、Zおよびワードデバイスのビット指定デバイスは間接指定できません。**

インデックス修飾

インデックス修飾とは、各デバイスに、インデックスレジスタ(Z)の値や定数を加算した番号のデバイスを指定する方法です。
加算できる値の範囲は、符号付き32ビット(-2147483648～+2147483647)です。
使用できるデバイスはユニットごとに異なります。
間接指定(*)、ローカルデバイス(@)と併用できます。

ポイント
- **ラベル(ラベル配列)およびワードデバイスのビット指定デバイスは、インデックス修飾できません。**
- **変数はインデックス修飾できません。ただし、グローバルデバイスに手動で割り付けた変数はインデックス修飾できます。**
- **インデックス修飾して使用可能範囲外のデバイスを引数に指定した場合は、コントロールリレーCR2012 が ON し、その関数は実行されません。運転は継続します。**

■インデックスレジスタによる修飾

デバイス番号を運転中に変更する場合に使用します。
インデックスレジスタによるデバイス修飾の記述方法は

(デバイス番号):Z(01～10)

となります。

例)
DM1000:Z01 ―― Z01の値が-10の場合、DM0990を参照します。
R0000:Z05 ―― Z05の値が+16の場合、R0100を参照します。

ポイント **Z11以降は、システム予約されているため使用できません。**

例

DM1000に格納される現在値を、R0001の立ち上がりをトリガ信号として保存します。
運転開始時にインデックスレジスタを初期化し、R0001の立上り(OFF→ONの変化)で、DM0000から順番にデータを保存していきます。

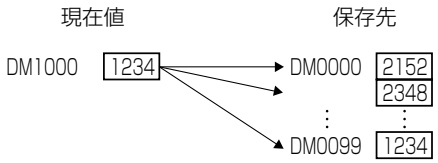
ST記述

```
IF LDP(CR2008) THEN
   Z01 := 0;
END_IF;
IF LDP(R0001) THEN
   DM0000:Z01 := DM1000;
   INC(Z01);
END_IF;
```

運転開始時、インデックスレジスタZ01を0にします。

R0001の立ち上がりで、インデックス修飾されている対象デバイスにDM1000の値を書き込みます。

インデックスレジスタZ01をインクリメント(+1)します。



ポイント **インデックス修飾するデバイス(値)は、32ビット符号付きデータとして扱うため、型宣言をおこなっていないインデックスレジスタは自動的に「.L(32ビット符号付き)」サフィックスとして実行されます。**

参考 インデックスレジスタ (Z) をプロジェクト内で 10 個以上使用する必要がある場合は、インデックスレジスタ関数 (ZPUSH/ZPOP)を使用して、モジュールやマクロごとにインデックスレジスタを一括で退避･復帰させます。
「ZPUSH関数」(6-7ページ)
「ZPOP関数」(6-8ページ)

■インデックス定数による修飾

定数を使用してインデックス修飾する場合の記述方法は

(デバイス番号):(10進数表記の定数)

となります。

例)
DM1000:10 ―― DM1000+10ワードでDM1010になります。
R0000:16 ―― R0000+16ビット(1チャンネル分)でR0100になります。

参考 ユニットデバイスを使用したラダープログラムを作成する場合に使用すると、割り付けられた先頭デバイスから何番目かがわかりやすくなります。また、マクロ作成時に、引数デバイスとしてUR/UMを使用する場合にも有効です。

例)
UR0:2
UM1:32

■デバイスや式による修飾

STでは、インデックスレジスタ以外でもインデックス修飾ができます。
この場合、ラダー変換時に自動的にシステムが予約しているインデックスレジスタ(Z11、Z12)を使用します。

ST記述　DM1000:(DM10+2) := 100;

※ 式でインデックス修飾するときは、必ず式を(　)でくくってください。

■サフィックスの付いたデバイスの修飾

インデックス修飾するデバイスのサフィックスは以下のように記述してください。

DM1000.D:Z01.S
DM1100.F:(DM10+10)

■不正なインデックス修飾

次のような場合、インデックス修飾はできません。

- **(DM1000**:**Z01)**:**Z02**インデックス修飾したデバイスの修飾
- **(DM1000+3)**:**Z02**演算結果に対する修飾
- **DM1000**:**Z01+5**インデックス修飾する式に**(　)**がない
- **Z01**:**Z02**インデックス修飾できないデバイスの修飾

間接指定

■間接指定(*)とは

間接指定(*)は、デバイスを直接指定するのではなく、デバイスが指し示す値(アドレス)が格納されたデバイスを記述することで、間接的に指定することです。

●間接指定の仕組み

各デバイスはCPUユニット内のメモリ上で管理され、それぞれに個別のアドレスを持っています。

- 間接指定**(*)**は、そのアドレスを専用関数で操作して参照するデバイスを変更します。
- インデックス修飾が「基準デバイス**+**相対オフセット値**(**インデックス定数**)**」で表記するのに対して、間接指定は「**"*"+**参照先アドレス格納デバイス」で表記します。
- インデックス修飾**(**:**Z/:10**進定数**)**、ローカルデバイス**(@)**と併用できます。
  アドレスは**32**ビットで管理されるため、アドレス格納用にデバイスが**2**ワード必要になります。

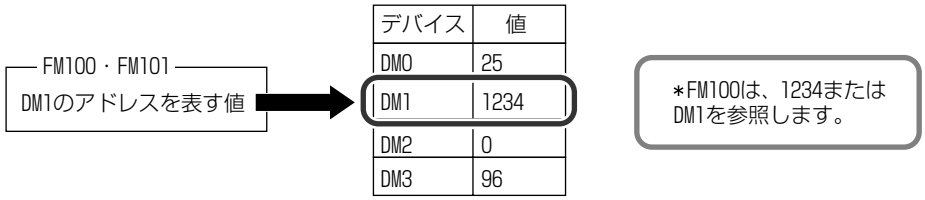
ポイント
- **間接指定して使用可能範囲外のデバイスを引数に指定した場合、その関数の動作は保証されません。**
- **演算式の項に使用可能範囲外のデバイスを指定した場合、演算結果は保証されません。**

●間接指定の記述方法

「*(デバイス)」と記述すると、そのデバイスに格納されているアドレスに該当するデバイスまたは、デバイス値を参照します。

例)

＜DM1のアドレスがFM100にセットされていて、*FM100と記述した場合＞



使用できるデバイスはユニットごとに異なります。「インデックス修飾と間接参照で使用できるデバイス」を参照してください。

ポイント
- 変数も間接指定できます。ただし、以下の場合は間接指定できません。
- 配列型変数の添え字
- 配列型変数の要素
- 構造体型変数のメンバ
- ワードデータのビット位置指定は間接指定できません。

参考
- 間接指定するデバイスがタイマ(T)、カウンタ(C)の場合に、接点、現在値、設定値は使用する関数の引数や付加するサフィックスによって決まります。
- ローカルデバイス使用時の記述は、「*@(デバイス)」になります。
- 間接指定したデバイスは.U(16ビット符号無し)として扱います。
- 間接指定に対しても型宣言できます。

●間接指定の使用方法

次の手順で間接指定します。

1 **アドレスセット関数(ADRSET)を使用して、参照するデバイスのアドレスを書き込みます。**

例)

```
ADRSET(R1000,DM0,FM100.D);
```

実行条件(R1000)がONのとき
DM0を示すアドレスをFM100、FM101に格納します。

ポイント
アドレスを格納できるデバイスはユニットごとに異なります。「インデックス修飾と間接参照で使用できるデバイス」を参照してください。

「ADRSET関数」(6-9ページ)

2 **関数の各引数(デバイス)を間接指定するには、アドレスを格納しているデバイスの先頭に「*」を付けて使用します。**

例)

```
EM1000.U := AVG( *FM100.U, 10);
```

*FM100(DM0)から10ワード分のデータの平均値を
EM1000.Uに格納します。

ポイント
アドレスを格納するデバイスに「*」を付けて使用する前に、必ずADRSET関数で間接指定するデバイスのアドレスを格納してください。

3 **参照するデバイスを変更する場合は、間接指定専用関数(ADRINC/ADRDEC/ADRADD/ADRSUB)を使用して、格納されているアドレスを変更します。**

例)

```
ADRSET(EM300, FM100.D);
ADRINC(R1001, FM100.D);
*FM100 := DM1000;
```

EM300のアドレスをFM100にセットします。

実行条件(R1001)がONのとき、FM100、FM101に格納されているDM0のアドレスをインクリメント(+1)します。

* FM100(EM300)にDM1000の値を格納します。

ポイント
参照するアドレスを変更する場合に、通常の演算子(+,－,*,/など)を使用しないでください。ラダープログラムが正しく動作しません。

●間接指定デバイスの型

間接指定デバイス(*[デバイス])は、その他のデバイスと同様に扱えるデータの範囲がサフィックスにより変化します。

- 型は、間接指定デバイスにサフィックスを付加して表します。

ST記述　*DM1000.F := 123.567;　//DM1000で間接指定されるデバイスは、.F(単精度浮
//動小数点型)として扱います。

- 型宣言をおこなわなかった間接指定デバイスのサフィックスを省略して記述すると、.Uとして扱われます。

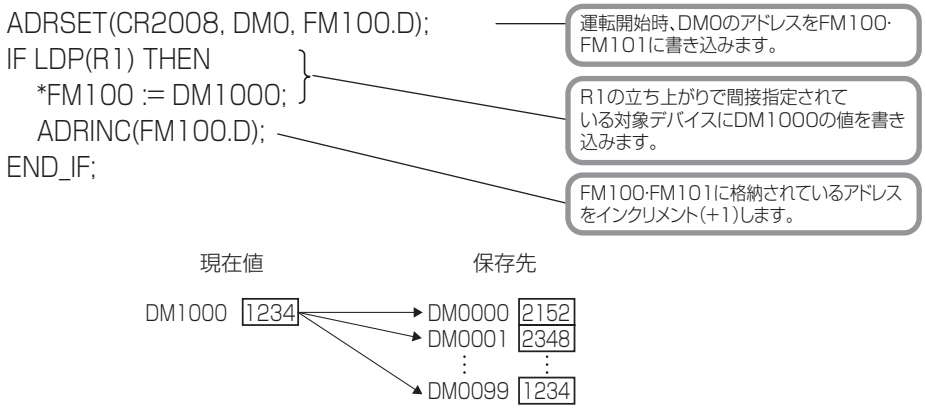
ST記述　*DM1000 := INT#65535;　//型宣言していない*DM1000で間接指定されるデバ
//イスは、.U(16ビット符号無し)として扱います。

このとき間接指定されるデバイスの型が.U以外でも、間接指定デバイスのサフィックスを省略すると.Uとして扱われます。

ST記述
```
ADRSET (MR2000, DM1000.D);   //DM1000.DにMR2000のアドレスを格納します。
*DM1000 := 123;              //MR2000はビットデバイスですが、サフィックスを
                             //省略すると、*DM1000(=MR2000)は、.U(ワードデバイス)として扱われます。
```

●間接指定の使用例

運転開始の最初のスキャンで、DM0を示すアドレスをFM100、FM101に格納します。
R0001の立ち上がりをトリガにして、DM1000の現在値をDM0から順番に格納していきます。



インデックス修飾と間接指定を併用する

■記述方法について

間接指定デバイスにインデックス修飾をおこなう場合、以下のように記述します。

例)　(*DM1000.U):Z01 ―― *DM1000をZ01でインデックス修飾します。
　　*(DM1000.U:Z01) ―― Z01でインデックス修飾されたDM1000を間接参照します。

(　)を省略して記述すると、以下の意味になります。

例)　*DM1000.U:Z01 ―― *DM1000をZ01でインデックス修飾します。

●インデックス修飾と間接指定の併用例

R1の立ち上がりをトリガ信号として、DM1000に格納される現在値を保存します。
R0の立ち上がり時にR2がONの場合はDM0から、OFFの場合はEM0から順番にデータを保存していきます。

```
IF LDP(R0000) THEN
   Z01 := 0; . . . . . . . . . . . . . . . . . . . . . . . . . . . . ①
   ADRSET(R2, DM0, FM100.D);. . . . . . . . . . . . . ②
   ADRSET(R2 = FALSE, EM0, FM100.D);. . . . . . ③
END_IF;
IF LDP(R1) THEN
   *FM100:Z01 := DM1000; . . . . . . . . . . . . . . . . . . ④
   Z01 := Z01 +1;. . . . . . . . . . . . . . . . . . . . . . . . . ⑤
END_IF;
```

［測定開始時］

①インデックスレジスタZ01に0を書き込みます。(インデックスレジスタの初期化)

②R2がONの場合は、FM100･FM101にDM0のアドレスを書き込みます。

③R2がOFFの場合は、FM100･FM101にEM0のアドレスを書き込みます。

［トリガ発生時］

④FM100･FM101に格納されているアドレスのデバイスを、Z01でインデックス修飾(Z01の値を加算)したデバイスにDM1000の値を書き込みます。

例) R2がOFFの時、

＊FM100→EM0　　→　＊FM100:Z01→EM23

Z01 := 23;

⑤Z01をインクリメント(＋1)します。

■簡易間接指定

STでは簡易間接指定(#TM)は使用できません。簡易間接指定を利用する際はラダープログラムで記述してください。

データの型(サフィックス)

STではデバイスにサフィックスを指定することにより、デバイスや定数の型を表現します。
末尾に「.U .S .D .L .F .DF .B .T」の記号"サフィックス"がつくデバイスは、サフィックスの違いにより扱うデータの種類が異なります。

サフィックス

サフィックスとはデバイスに接尾語としてつくもので以下の8種類があります。

□.U　□.S　□.D　□.L　□.F　□.DF　□.B　□.T

※ □ にはサフィックス対応のデバイスが入ります。

各デバイスはサフィックスに応じて、データメモリなどの値を以下のように扱います。

□.U 16ビット符号無しデータとして扱います。(0～65535)
□.S 16ビット符号付きデータとして扱います。(–32768～32767)
□.D 32ビット符号無しデータとして扱います。(0～4294967295)
□.L 32ビット符号付きデータとして扱います。(–2147483648～2147483647)
□.F 単精度浮動小数点型実数データとして扱います。

－3.4E+38 ≦ N ≦ －1.4E－45
N = 0
1.4E－45 ≦ N ≦ 3.4E+38
(有効桁数:約7桁)

□.DF 倍精度浮動小数点型実数データとして扱います。

－1.79E+308 ≦ N ≦ －2.23E－308
N = 0
2.23E－308 ≦ N ≦ 1.79E+308
(有効桁数:約16桁)

□.B ビット型データとして扱います。(ON:真, OFF:偽)
□.T 文字列型データとして扱います。

参考　ラダープログラムで指定するサフィックスは、命令語にのみ付き、デバイスには付きません(STではデバイスのみに付き、関数には付きません)。

例

```
DM1000 (.U)
DM1000.S
DM1000.D
DM1000.L
DM1000.F
DM1000.DF
DM1000.B
DM1000.T
```

ポイント　**ST言語内では、変数にサフィックスを付与して使用することはできません。**

●サフィックスの省略

デバイスのサフィックスを省略して記述した場合、デバイスの種別により扱われる型が決まります。

デバイスの種類	サフィックスを省略した場合の型(デフォルトの型)
R(DR),B,MR,LR,T,C,CR,UR	.B(ビット型)
DM,W,EM,ZF,FM,TM,CM,間接指定するデバイス	.U(16ビット符号無しデータ型)
Z※	.L(32ビット符号付きデータ型)

※　使用するCPUユニットにより、デフォルトの型が異なります。

ポイント　**変数でサフィックスを省略した場合は、変数編集ウィンドウで設定したデータ型で扱われます。**

ST記述

```
DM1000 := 2000;                  //省略した場合16ビット符号無し(.U)扱い
DM1100.D := 75535;               //DM1100は32ビット符号無し(.D)扱い
IF DM1000 = 3000 THEN            //
   DM3000.L := DM1100 + 1000;    //右辺は32ビット符号付きで演算します。
END_IF;                          //
                                 //
DM1000.L := -21474800;           //DM1000は32ビット符号付き(.L)扱い
IF DM1000 > 0 THEN               //
   DM3000.D := DM1000.S * -1;    //右辺は.Lに拡張して演算し、解を.Dに変換し
END_IF;                          //て代入します。
```

■.Uと.Sについて

□.UはBINデータを16ビット符号無しデータとして扱います。
□.SはBINデータを16ビット符号付きデータとして扱います。
したがって、BINデータとサフィックスに応じた10進表記の関係は、次のようになります。

デバイス(DM1000)	定数の種類	1111111111111111というビットデータの場合
DM1000.U	符号無し10進	65535
DM1000.S	符号付き10進	–1

※ DM1000に格納されている16ビットBINデータ(1111111111111111)を10進定数100と比較する場合、サフィックスが.Uのとき、DM1000の値は65535と解釈されます。

ST記述

```
IF DM1000.U > 100 THEN
   INC(DM2000);
END_IF;
```

したがって、DM1000 > 100 が成り立つので、DM2000に1が加算されます。

- サフィックスが.Sのとき、DM1000の値は–1と解釈されます。

ST記述

```
IF DM1000.S > 100 THEN
   DEC(DM2000);
END_IF;
```

したがって、DM1000 < 100 となるので、DM2000は減算されません。
制御文の詳細については、「制御文」(4-6)を参照してください。

■.Dと.Lについて

□.D はBINデータを32ビット符号無しデータとして扱います。
□.L はBINデータを32ビット符号付きデータとして扱います。

したがって、BINデータとサフィックスに応じた10進表記の関係は、次のようになります。

デバイス(DM1000)	定数の種類	11111111111111111111111111111111というビットデータの場合
DM1000.D	符号無し10進	4294967295
DM1000.L	符号付き10進	–1

- DM1000･DM1001 に格納されている 32 ビット BIN データ(11111111111111111111111111111111)を 10 進定数 100 と比較する場合サフィックスが.Dのとき、[DM1000･DM1001]の値は4294967295と解釈されます。

ST記述
```
IF DM1000.D > 100 THEN
    DM20000 := DM2000 + 1;
END_IF;
```

したがって、DM1001（上位）･DM1000（下位）（4294967295）> 100となり、接点はONされ、DM2000に1が加算されます。

•サフィックスが.Lのとき、[DM1000･DM1001]の値は–1として扱われます。

ST記述
```
IF DM1000.L > 100 THEN
    DM2000 := DM2000 -1;
END_IF;
```

したがって、DM1001（上位）･DM1000（下位）（–1）< 100 となり、接点はOFFされ、DM2000は減算されません。

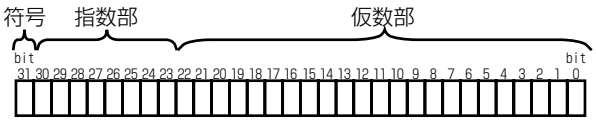
ポイント **32ビット型のデータ（.D、.L、.F）を扱うときは、偶数デバイス番号を使用してください。**

■.Fについて

□.FはBINデータを単精度浮動小数点型実数データとして扱います。データは32ビットです。
単精度浮動小数点型実数データは、特殊なビット構成で値を表現（内部表現）しています。
したがって、内部表現を解析して直接ビットを操作することは、できるだけ避けるようにしてください。
単精度浮動小数点型実数値は、次のように表記される数値です。

$$-1.234\times2^{-3}$$

符号　仮数部　指数部



符号（1ビット）
0＝正
1＝負

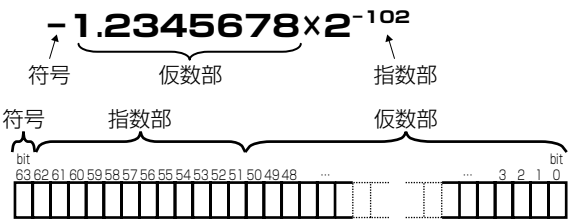
仮数部（23ビット）
0～8388607

指数部（8ビット）
00000000（16#00）＝未使用
00000001（16#01）＝-126
00000010（16#02）＝-125
00000011（16#03）＝-124
：
01111110（16#7E）＝-1
01111111（16#7F）＝0
10000000（16#80）＝1
10000001（16#81）＝2
11111100（16#FC）＝125
11111101（16#FD）＝126
11111110（16#FE）＝127
11111111（16#FF）＝未使用

ポイント **DM/W/EM/FM/TM/ZFなどのワードデバイスを使用して32ビットデータを扱う場合は、デバイスにサフィックス.D、.L、.Fを付加します。**
このときデバイスは番号が連続する2ワードを使用します。
指定するデバイスは下位16ビットにサフィックスを付けたものになります。
例）DM1000.D→[DM1001,DM1000]を使用します。

■.DFについて

□.DFはBINデータを倍精度浮動小数点型実数データとして扱います。データは64ビットです。
倍精度浮動小数点型実数データは、特殊なビット構成で値を表現（内部表現）しています。
したがって、内部表現を解析して直接ビットを操作することは、できるだけ避けるようにしてください。
倍精度浮動小数点型実数値は、次のように表記される数値です。



符号（1ビット）
0＝正
1＝負

仮数部（51ビット）
0〜2251799813685248

指数部（12ビット）
000000000000（16#000）＝未使用
000000000001（16#001）＝－2046
000000000010（16#002）＝－2045
000000000011（16#003）＝－2044
:
011111111110（16#000）＝－1
011111111111（16#7FF）＝0
100000000000（16#800）＝1
100000000001（16#801）＝2
:
000000000000（16#000）＝2045
000000000000（16#000）＝2046
000000000000（16#000）＝2047
111111111111（16#FFF）＝未使用

ポイント

DM/W/EM/FM/TM/ZFなどのワードデバイスを使用して倍精度浮動小数点型実数を扱う場合は、デバイスにサフィックス.DFを付加します。
このときデバイスは番号が連続する4ワードを使用します。.DFを扱うときは、偶数番号のデバイスを用いてください。
指定するデバイスは最下位16ビットにサフィックスを付けたものになります。
例）DM1000.DF→[DM1003・DM1002・DM1001・DM1000]を使用します。

■.Bについて

□.BはBINデータをビット型データとして扱えます。

例）ビットデバイスMR1000をON(1, TRUE)にします。

ST記述　SET(MR1000.B);

参考　ワードデバイスにサフィックスに.Bを付加すると、最下位ビットがビット型判定の対象になります。上記例の場合、SET関数の引数にDM10.Bを指定すると、DM10の最下位ビットが1になります。ワードデバイスの真偽判定は、以下のようになります。

```
例）IF DM10.B = TRUE THEN
        処理文;
    END_IF;
```

例）ビットデバイスMR1000をOFF(0, FALSE)にします。

ST記述　RES(MR1000.B);

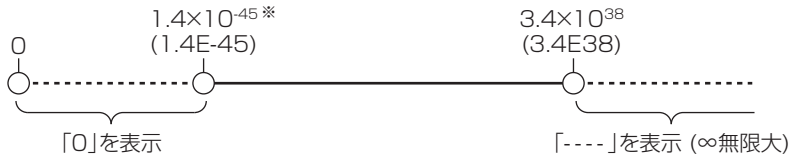
■.Tについて

□.TはBINデータを文字列型データとして扱います。

例）DM100を先頭とする連続したデバイスに 'KEYENCE' の文字列を格納します。

ST記述　DM100.T := 'KEYENCE' + CHR(16#0D) + CHR(16#0A);

上記例の場合、データ格納状態は以下のようになります。



参 考　文字列の終端には自動的にNUL(00H)が格納されます。

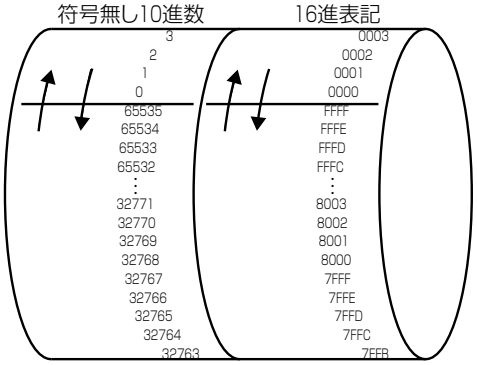
ポイント　**扱える文字列の最大は1999文字です。**

■最大値・最小値を超える時の数値の変化

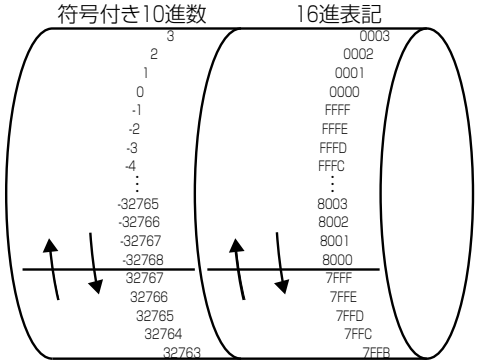
算術演算結果が最大値または最小値を超えた場合、データ型式によりそれぞれ次のように数値が変化します。

● 16ビットデータの場合

□.Uの65535を超える時の値の変化

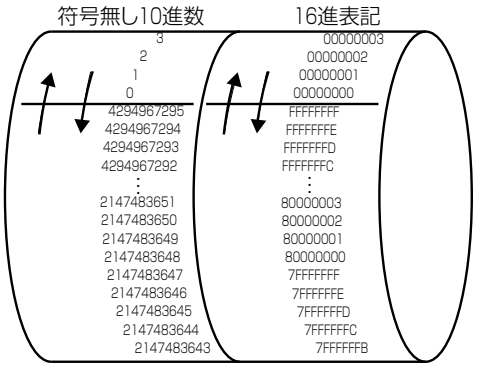


□.Sの最大値、最小値を超える時の値の変化

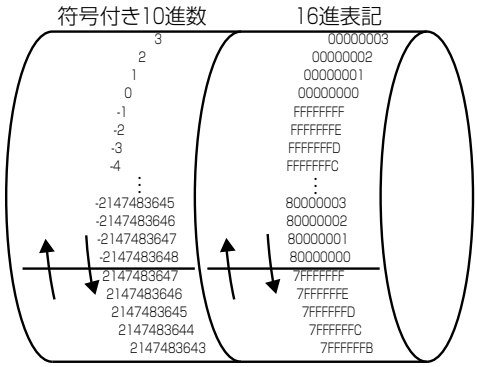


● 32ビットデータの場合

□.Dの4294967295を超える時の値の変化



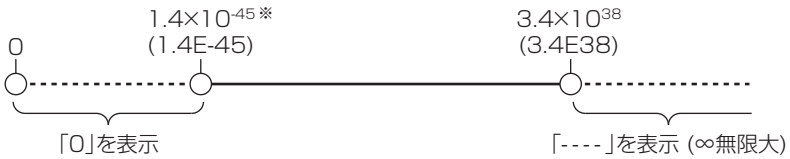
□.Lの最大値、最小値を超える時の値の変化



参 考　・単精度浮動小数点型データの場合

オーバーフロー：絶対値が3.4×10^{38}より大きくなる場合、「∞(無限大)」になります。

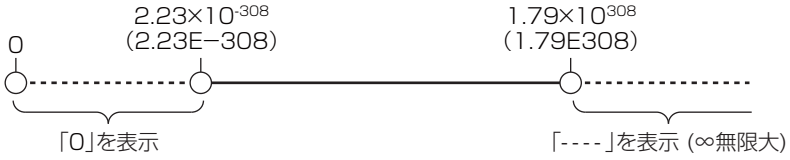
アンダーフロー：絶対値が1.4×10^{-45}※より小さくなる場合、「0」になります。



・倍精度浮動小数点型データの場合

オーバーフロー：絶対値が1.79×10^{308}より大きくなる場合、「∞(無限大)」になります。

アンダーフロー：絶対値が2.23×10^{-308}より小さくなる場合、「0」になります。



演算の型について

STは、数値演算をおこなう場合、ラダー言語のように専用の命令語を使うことなく、演算子(+, − , ＊ ,/)を使って記述できます。

■演算式で扱えるデータの範囲

演算式の各項は、すべて.L(32ビット符号付き)、.F(単精度浮動小数点型)もしくは.DF(倍精度浮動小数点型)として扱われます。このため式のすべてのデバイスは、順番に実行する演算式に応じて自動的に.L、.Fもしくは.DFに符号拡張されてから演算されます。また、演算結果は格納先のデータ型に応じて変換されて格納されます。したがって、演算の順番や結果の値により精度落ちなどが発生することがありますのでご注意ください。

4

STの文法と注意

代入文 ····· 4-2
演算子 ····· 4-4
制御文 ····· 4-6
STを記述するときの注意 ····· 4-18

代入文

代入文は、左辺と右辺が等号(=)によって結ばれた式のみで構成された文で、右辺の結果あるいは値を左辺に代入(格納)します。

代入文の使用方法

代入文の右辺として記述できるもの(デバイスや変数、式)には、次に示すように数値、文字列、ビットがあり、それぞれ定数か演算式、または(戻り値をもつ)関数で記述します。

●数値代入文

右辺の数値定数または数値演算の結果を、左辺に代入します。

ST記述

```
uAnalogData := 123;                          //数値定数の代入
lSettingParam := uTemp + 123;                //数値演算結果の代入
uCount := DCNT(uSettingPrm,auMeasure[0],20); //関数が処理した結果の代入
```

●文字列代入文

右辺の文字列

定数または文字列演算の結果を、左辺に代入します。

ST記述

```
stProductName := 'ABCD';                     //文字列定数の代入
DM200.T := '現在' + DM300.T + 'です';        //文字列連結結果の代入
```

●ビット代入文

- 右辺のビット定数またはビット演算(条件式の評価)の結果を、左辺に代入します。

ST記述

```
R1000 := TRUE;                               //ビット定数(ブール値)の代入
R1000 := R1100 OR (R1200 AND R1300);         //ビット演算結果の代入
R1000 := DM100.S > 100;                      //条件式評価の代入
R1000 := R2000 = R3000;                      //比較結果の代入
```

- ビット状態は、1、0やTRUE、FALSEで表せます。

ST記述　R1000 := INT#1;

動作説明　R1000をONにします。

- ビットデータはブール値のみ代入できます。

```
R1000 := TRUE;     R1000 を ON
R1000 := FALSE;    R1000 を OFF
```

- ビットデータには1、0以外の定数の代入や数値演算式の結果の代入はできません。

×:　R1000 := DM1000 + DM2000; . . . . . "+"は不可

- ビットデータに論値演算式の結果や比較演算の結果(どちらも結果はブール値)の代入は可能です。

○:　R1000 := R2000 AND DM2100.B; . . . 論理演算子の使用は可能
　　R1000 := R2000 = R2100;. . . . . 等号、不等号(演算結果がブール値しかもたない式)は可能

■代入文を使用するときの注意

●右辺と左辺で型が異なる場合

- 代入文は、右辺と左辺で型が異なる(型変換ができない)場合成立しません。

ST記述　stCurData := uTemp;　//エラーになります

動作説明　右辺のuTemp(定数：16ビット符号無し)を、stCurData(文字列型)に代入しようとしています(変換エラー)

- ただし型が異なる場合でも.U.S.D.L.F間での型の違いは、変換できます(精度落ちが発生する場合があります)。

ST記述　DM1100.U := DM2100.D;

動作説明　右辺のDM2100.D(32ビット符号無し)を、DM1100.U(16ビット符号無し)に代入します。
DM2100.Dの下位16ビットだけの代入になります。

●左辺に代入できない場合

- 下記のSTは、左辺の「3」が数値であり、結果や値を代入できないため、代入文として正しくありません。

ST記述　3 := DM1000.U;　//エラーになります(定数には代入できません)

動作説明　右辺のDM1000.U(16ビット符号無し)を、3(定数)に代入しようとしています。(変換エラー)

- 左辺を式や関数で記述しているときも、代入文として正しくありませんので、エラーになります。

ST記述　DM1100.U + 30 := DM1000.U;　//エラーになります(式には代入できません)

動作説明　右辺のDM1000.U(16ビット符号無し)を、左辺の演算式に代入しようとしています。(変換エラー)

演算子

演算子とはデータに処理を加える記号のことで、式の中で使われる足し算(＋)や、掛け算(*)などのことをいいます。

演算子一覧

■演算子の種類

演算子には、下記に示す「算術演算子」、「比較演算子」、「文字列連結演算子」、および「論理演算子」の4種類があります。

●算術演算子

2つの値の算術計算の結果を返します。

記　号	処　理　内　容	記　述　例
**	累乗を求めます	uiValue1 := uiValue2 ** 2;
*	2つの値の積を求めます(乗算)	uiValue1 := uiValue2 * 3;
/	2つの値の商を求めます(除算)	uiValue1 := uiValue2 / 4;
MOD※1	2つの値の除算を行い、その剰余を返します	uiValue1 := uiValue2 MOD 5;
+	2つの値の和を求めます(加算)	uiValue1 := uiValue2 + 6;
－	2つの値の差を求めます(減算)	uiValue1 := uiValue2 -7;

※1　記号の前後に半角スペースが必要です。

●文字列連結演算子

2つの文字列の連結結果を返します。

記　号	処　理　内　容	記　述　例
+※2	2つの文字列を連結します	strName := 'KEY' + 'ENCE';

※2　「+」は記述内容に応じて算術演算と文字列連結に自動的に判断されます。

●比較演算子

2つの値の大小関係を評価した結果をブール値(TRUEまたはFALSE)で返します。

記　号	処　理　内　容	記　述　例
<	より小さい	bFlag := uiValue < 10;
<=	小さいか等しい(以下)	bFlag := uiValue <= 20;
>	より大きい	bFlag := uiValue > 30;
>=	大きいか等しい(以上)	bFlag := uiValue >= 40;
=	等しい(等価)	bFlag := uiValue = 50;
<>	等しくない(不等価)	bFlag := uiValue <> 60;

●論理演算子

2つの値の論理演算の結果を、ビット同士の演算の場合はブール値(TRUEまたはFALSE)で、ワード同士の演算の場合はワード値で返します。

記　号	処　理　内　容	記　述　例
NOT※3	値の論理否定値を求めます	bFlag1 := NOT bFlag2;
AND※3	2つのデータ(ビットデータ同士、ワードデータ同士)の論理積を求めます	bFlag1 := bFlag2 AND bFlag3;
OR※3	2つのデータ(ビットデータ同士、ワードデータ同士)の論理和を求めます	bFlag1 := bFlag2 OR bFlag3;
XOR※3	2つのデータ(ビットデータ同士、ワードデータ同士)の排他的論理和を求めます	bFlag1 := bFlag2 XOR bFlag3;

※3　記号の前後に半角スペースが必要です。

ポイント
- 演算するデータ(演算要素)は、ビットデータ同士またはワードデータ同士を指定してください。ビットデータとワードデータなどの論理演算はできません。
- 演算するデータ(演算要素)の型の大きさが異なる場合、大きい方の型に合わせて演算をおこないます。
  例)　DM1000.U AND DM2000.D ・・・・32ビット型で演算します

演算子の優先順位

1つの式の中に複数の演算子が使われている場合、式の各演算部分は決められた順序にしたがって評価(計算)されます。この順序を演算子の優先順位といいます。

■優先順位について

異なる種類の演算子が1つの式の中で使われている場合、以下の表の、優先順位の高い順で評価されます。

マイナス符号、論理否定	-, NOT
累乗	**
乗算、除算、剰余	*, /, MOD
加算、減算	+, -
より小さい、より大きい、以下、以上	<, >, <=, >=
等しい、等しくない	=, <>
論理積	AND, &
排他的論理和	XOR
論理和	OR

制御文

制御文とはプログラムの流れを変えるもので、条件により処理の流れを分岐させたり、繰り返しをおこない、実行の選別や回数を変えていきます。

制御文一覧

制御文には、次に示す「条件分岐文」と「繰り返し制御文」と「プログラム実行終了文」の3種類があります。

■条件分岐文

条件分岐文は、指定した条件で処理の流れを分岐させます。

制御文	制御内容	
IF文	① 条件式に従って処理を実行する分岐制御文	
	IF ␣条件式A ␣THEN 　処理文B; END_IF;	条件式A が成り立つとき、 処理文B を実行します。 条件式A が成り立たないとき、何も処理しません。
	② 条件式に従って別々の処理を実行する2分岐制御	
	IF ␣条件式A ␣THEN 　処理文B; ELSE 　処理文C; END_IF;	条件式A が成り立つとき、 処理文B を実行します。 条件式A が成り立たないとき、 処理文C を実行します。
	③ 条件値に従って別々の処理をする多分岐制御文	
	IF ␣条件式A ␣THEN 　処理文B; ELSIF ␣条件式C ␣THEN 　処理文D; 　. 　. 　. END_IF;	条件式A が成り立つとき、 処理文B を実行します。 条件式A が成り立たないで条件式C が成り立つとき、 処理文D を実行します。 条件式A,C･･･のいずれも成り立たないのとき、何も処理しません。
CASE文	条件値に従って対応する処理を選択して実行する多分岐制御文	
	CASE ␣比較元 ␣OF 条件値A : 　処理文B; 条件値C : 　処理文D; 　: ELSE 　処理文E; END_CASE;	 比較元と条件値Aが等しいとき、 処理文Bを実行します。 比較元と条件値Cが等しいとき、 処理文Dを実行します。 比較元が条件値A,C･･･以外のとき、 処理文Eを実行します。

■繰り返し制御文

繰り返し制御文は、指定した条件で繰り返して処理を実行させます。

制御文	制御内容	
FOR文	指定した回数の条件式が成立するまで処理を繰り返すループ制御文	
	FOR␣初期値␣TO␣最終値␣BY␣増分値␣DO 　処理文A; END_FOR;	初期値から最終値まで、増分値を加算しながら、 処理文を繰り返し実行します。
WHILE文	条件式が成立している（真である）間、処理を繰り返す不定回数ループ制御文	
	WHILE␣条件式␣DO 　処理文; END_WHILE;	条件式が成立している間、 処理文を繰り返し実行します。
REPEAT文	条件式が成立する（真になる）までの間、処理を繰り返す不定回数ループ制御文	
	REPEAT 　処理文; 　UNTIL␣条件式 END_REPEAT;	条件式が成り立たない間、 処理文を繰り返し実行します。
EXIT文	繰り返し制御文の中で、繰り返し条件とは関係なくループ制御から抜ける	
	<繰り返し処理文> 　・ 　・ EXIT; 　・ 　・ <繰り返し処理文>	EXITが実行されると、 条件に関係なくEXITを含む繰り返し制御を終了します

■プログラム実行終了文

記述されているプログラムの実行を終了します。

制御文	制御内容	
RETURN文	記述されているプログラムの実行を終了する制御文	
	処理文; RETURN; 処理文;	記述されているプログラムの実行を終了します。

IF ～ 文(条件分岐 ①)

1つの条件式で処理をおこなうかどうかを判断する、最も簡単な条件分岐文です。

IF ～ THEN ～ END_IF	
基本書式	書式説明
IF ␣ <条件式> ␣ THEN 　<処理文1>; END_IF;	もし条件式が成り立つ時、 処理文1を実行します。 IF文はここまで

※ ␣は半角スペース挿入を表します。
↵

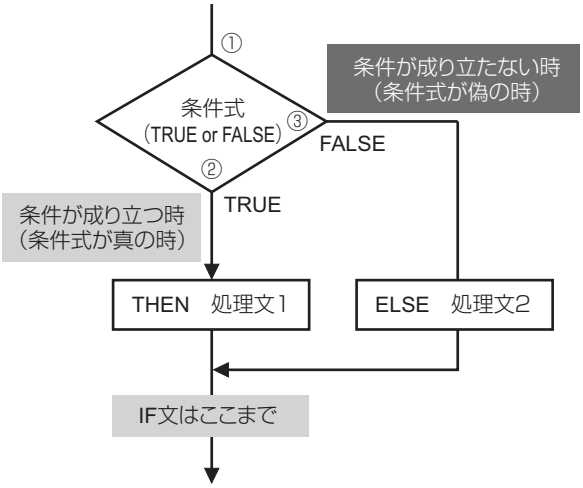
書式例	解説

IF ␣ bFlag = TRUE ␣ THEN
　uiValue := uiValue + 1;
END_IF;
	bFlagがONのとき
	uiValueに定数1を加算します。

ワンポイント

- 条件式には、結果がビット型になる(ブール値を返す)式であれば、どのような式でも使用できます。
- 処理文は、THEN～END_IFの間に記述する一連の処理で、必要なだけ複数行に渡って記述できます。
- IF文はネスト(入れ子)も可能ですので、処理文の中にさらにIF文を記述することにより、AND(かつ)条件で絞り込みながらの多重分岐処理がおこなえます。

処理の流れ



①条件式で「TRUE(真)」か「FALSE(偽)」を判断します。

②条件式が「TRUE(真)」の場合、THEN以下の処理文を実行し、END IFを実行します。

③条件式が「FALSE(偽)」の場合、END_IFまでジャンプし、何も実行せずにEND_IF以降の処理に移ります。

IF ～ ELSE文(条件分岐 ②)

1つの条件式で処理方法を2通りに分けることのできる、簡単な条件分岐文です。

IF ～ THEN ～ ELSE ～ END_IF	
基本書式	書式説明

IF ␣ ＜条件式＞ ␣ THEN
　＜処理文1＞;
ELSE
　＜処理文2＞;
END_IF;
	もし条件式が成り立つ時、
	処理文1を実行します。
	条件式が成り立たない時、
	処理文2を実行します。
	IF文はここまで

※ ␣は半角スペース挿入を表します。
↵

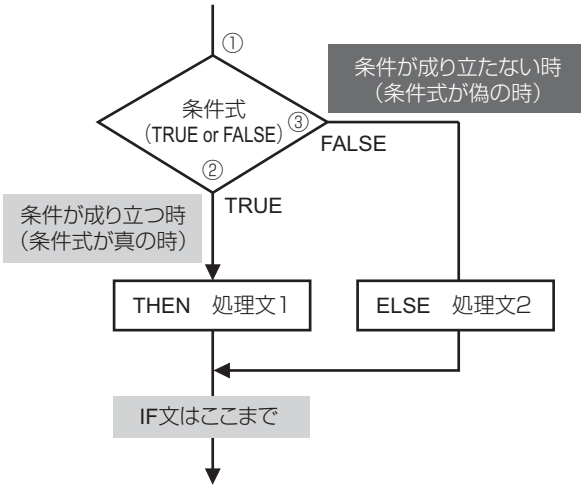
書式例	解説

IF␣bFlag = TRUE␣THEN
　uiValue := 50;
ELSE
　uiValue := 10;
END_IF;
	bFlagがONのとき
	uValueに定数50を格納します。
	bFlagがOFFのとき
	uValueに定数10を格納します。

ワンポイント

- 条件式には、結果がビット型になる(ブール値を返す)式であれば、どのような式でも使用できます。
- 処理文は、THEN～END_IFの間に記述する一連の処理で、必要なだけ複数行に渡って記述できます。
- IF文はネスト(入れ子)も可能ですので、処理文の中にIF文を記述することで、AND(かつ)条件で絞り込みながらの多重分岐処理がおこなえます。

処理の流れ



① 条件式で「TRUE(真)」か「FALSE(偽)」を判断します。

② 条件式が「TRUE(真)」の場合、THEN以下の処理文1を実行し、END_IFまでジャンプします(途中の式は実行しません)。

③ 条件式が「FALSE(偽)」の場合、THEN～ELSE間は実行されず、ELSE以下の処理文2を実行します。

IF ～ ELSIF ～ 文(条件分岐 ③)

最初の条件式を満たさない場合、次の条件式にて再度判定をおこないます。条件をいくつも持てる条件分岐文です。

IF ～ THEN ～ ELSIF ～ END_IF	
基本書式	書式説明
IF␣＜条件式1＞␣THEN 　＜処理文1＞; ELSIF␣＜条件式2＞␣THEN 　＜処理文2＞; ELSIF␣＜条件式3＞␣THEN 　… ELSE 　＜処理文n＞; END_IF;	条件式1が成り立つ時、 処理文1を実行します。 条件式1は成り立たないが、条件式2が成り立つ時 処理文2を実行します。 条件式1,2は成り立たないが、条件式3が成り立つ時 ※2 … 全ての条件式が成り立たない時、 処理文nを実行します。 IF文はここで終わりです。

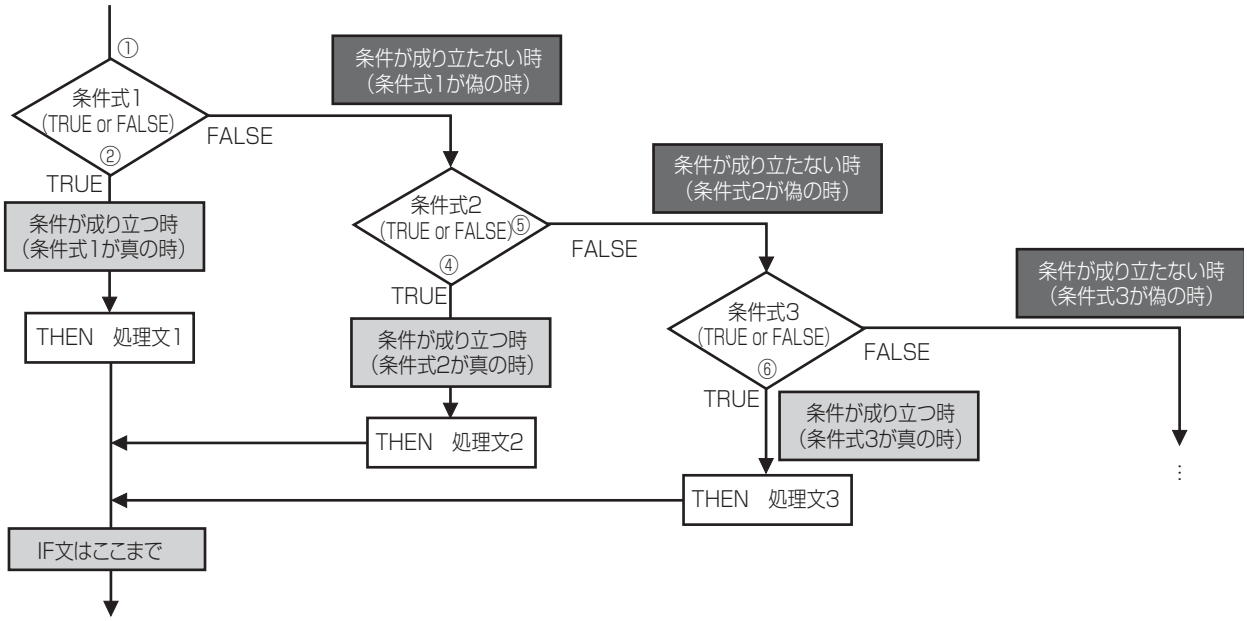
※ ␣は半角スペース挿入を表します。
※1　省略可能です。
※2　任意の数の条件式と処理文を設定できます。上限は有りません。

書式例	解説
IF ␣ bFlag ␣ THEN	bFlagがONのとき(bFlag =ONの省略記法)
uiValue := 1000;	uValueに1000を格納します。
ELSIF ␣ bFlag1 ␣ THEN	bFlagがONでなく、bFlag1がONの時
uiValue := 1001;	uValueに1001を格納します。
ELSIF ␣ bFlag2 ␣ THEN	bFlag,bFlag1がONでなく、bFlag2がONの時
uiValue := 1002;	uValueに1002を格納します。
ELSIF ␣ bFlag3 ␣ THEN	bFlag,bFlag1,bFlag2がONでなく、bFlag3がONの時
uiValue := 1003;	uValueに1003を格納します。
ELSIF ␣ bFlag4 ␣ THEN	bFlag,bFlag1,bFlag2,bFlag3がONでなく、bFlag4がONの時
uiValue := 1004;	uValueに1004を格納します。
ELSE	bFlagの全てがONでない時
uiValue := 1005;	uValueに1005を格納します。
END_IF;	

ワンポイント

- 条件式には、結果がビット型になる(ブール値を返す)式であれば、どのような式でも使用できます。
- 処理文は、制御文の間に記述する一連の処理で、必要なだけ複数行に渡って記述できます。
- ELSIF文はIF文と異なり、複数行記述してもネスト構造にはなりません。

処理の流れ



① 条件式1で「TRUE(真)」か「FALSE(偽)」を判断します。
② 条件式1が「真」の場合、THEN以下の処理文1を実行し、END_IFまでジャンプします(途中の式は実行しません)。
③ 条件式が「偽」の場合、条件式2で「真」か「偽」を判断します。
④ 条件式2が「真」の場合、処理文2を実行し、END_IFまでジャンプします(途中の式は実行しません)。
⑤ 条件式2が「偽」の場合、条件式3で「真」か「偽」を判断します。
⑥ 条件式3が「真」の場合、処理文3を実行し、END_IFまでジャンプします(途中の式は実行しません)。
⑦ 条件式3が「偽」の場合、条件式4で「真」か「偽)」を‥‥以下同じ

CASE ～ 文(多分岐制御)

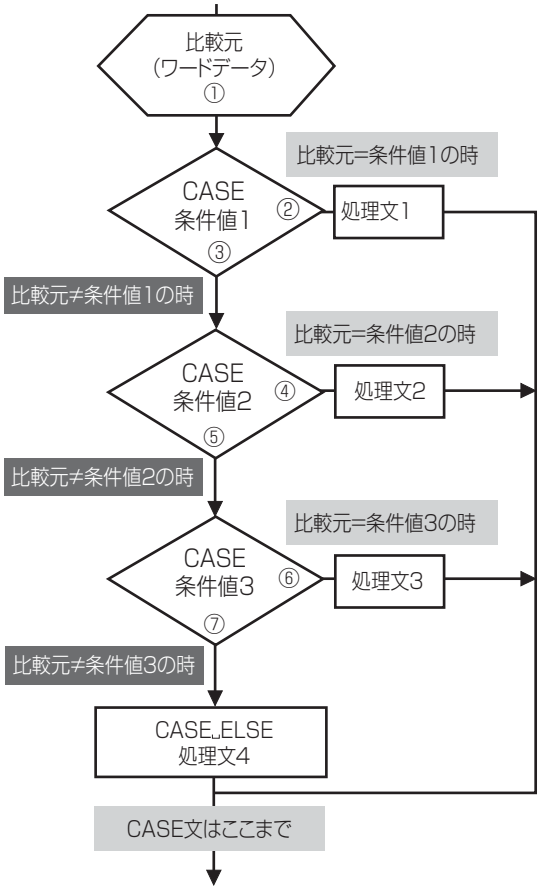
基準となるワードデータに格納されている値により、処理を選択できる条件分岐文です。

CASE	
基本書式	書式説明
CASE ␣ <ワードデータ> ␣ OF 　<定数1> : 処理文1; 　<定数2> : 処理文2; 　<定数3>,<定数4> : 処理文3; 　<定数5>..<定数6> : 処理文4; ELSE 　<処理文n>; END_CASE;	ワードデータの値を基準に 条件分岐定数1とワードデータの値が等しい時処理文1を実行します。 定数2とワードデータの値が等しい時処理文2を実行します。 定数3とワードデータの値が等しい時 … 定数とワードデータの値が何れも等しくない場合、 処理文nを実行します。 セレクト文は、ここで終わります。

※ ␣は半角スペース挿入を表します。
※1　省略可能です。
※2　《CASE定数(ELSE)》は最大200個まで記述可能です。
※3　定数1に、文字列や文字列を扱うデバイスおよび変数、ビット型デバイスやBOOL型の変数は指定できません。

書式例	解説
CASE ␣ uiCount ␣ OF	uiCountに格納されている値を基準とします。
1 : uiValue := 1000;	uiCount = 1のとき、uiValueに1000を格納します。
2..10 : uiValue := 2000;	uiCount = 2～10のとき、uiValueに2000を格納します。
11,15,60 : uiValue := 3000;	uiCount = 11 or 15 or 60のとき、uiValueに3000を格納します。
101..65535 : uiValue := 4000;	uiCountが100より大きい時、uiValueに4000を格納します。
ELSE	
uiValue := 5000;	uiCountの値が上記以外のとき、uiValueに5000を格納します。
END_CASE;	

処理の流れ



① 以降の条件値との判断基準となる値を宣言します。

② 条件値1と比較元を比較して、「条件値1=比較元」が成り立つ時、処理文1を実行します。処理文1を実行した後、END ␣ CASEにジャンプします。

③ 「条件値1=比較元」が成り立たなかった時、次の条件値比較に移ります。

④ 条件値2と比較元を比較して、「条件値2=比較元」が成り立つ時、処理文2を実行します。処理文2を実行した後、END ␣ CASEにジャンプします。

⑤ 「条件値2=比較元」が成り立たなかった時、次の条件値比較に移ります。

⑥ 条件値3と比較元を比較して、「条件値3=比較元」が成り立つ時、処理文3を実行します。処理文3を実行した後、END ␣ CASEにジャンプします。

⑦ 「条件値3=比較元」が成り立たなかった時、処理文4を実行します。

処理の流れ

ワンポイント

- 選択される定数は、単一のものでも、複数指定･範囲指定を含んだものでも可能です。

  例）　1 : 単一指定（ 1のみ ）
  3, 9, 12, 50 : 複数指定（ 3 or 9 or 12 or 50 ）
  8..20 :範囲指定（8以上、20以下の範囲）

- ELSE　<処理文>の記述は省略可能です。
  ただし、この『もし条件が全て成り立たなかったとき』というELSEは、想定外の事象が発生した場合に問題解決の一助になります。

FOR ～ END_FOR文（指定回数の繰り返し制御）

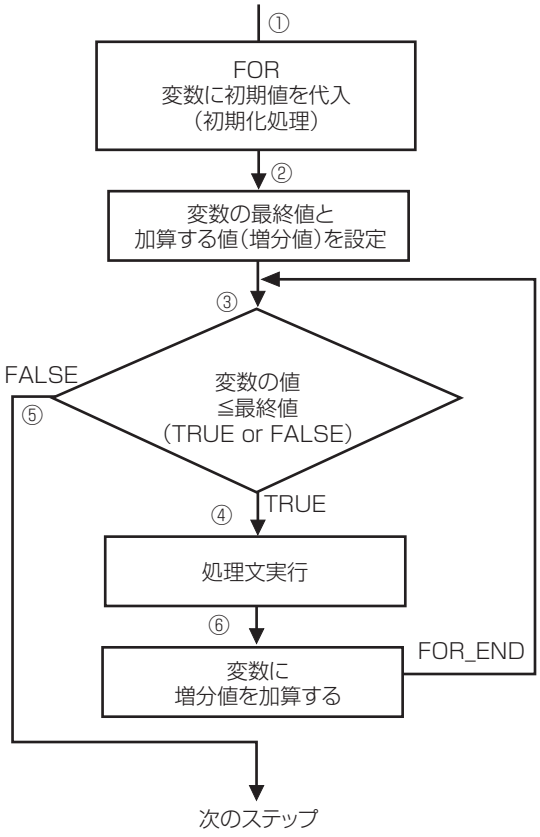
繰返し条件により、処理文の実行回数を決めることのできる制御文です。

FOR～END_FOR	
基本書式	
1	FOR ␣<変数> := <初期値>␣TO ␣<最終値>␣BY ␣<増分値>␣DO
2	<処理文>;
3	END_FOR;
書式説明	
1	繰り返し制御の回数を制限する変数の初期化と、終了値を決めます。BYで処理ごとに変数の値をいくつ増加させるかを決めます。
2	変数が最終値を超えるまで、毎回処理文を実行します。
3	変数に増分値を加算します。

※1　[　]内の記述は省略可能です。省略した場合BY ␣1として実行します。
␣は半角スペース挿入を表します。

書式例	解説
FOR ␣uiIdx := 0 ␣TO ␣100 ␣BY ␣5 ␣DO	uiIdxの値が0から100を超えるまで、繰り返します。
uiValue := uiValue + 5;	uiValueに5を加えた結果をuiValueに格納します。
END_FOR;	uiIdxの現在値に増分値5を加算して、FOR~に戻ります。

処理の流れ



①変数に初期値を代入します。

②繰返し処理の終わりを表す最終値を設定します。また、変数の値に加算する増分値も設定します。

③変数の値が最終値よりも小さいかどうかを判定します。

④③の条件が「TRUE（真）」のとき、処理文を実行します。

⑤「FALSE（偽）」のときは繰返し処理文を終了します。

⑥変数の値に増分値を加算して、③にもどります。

WHILE文(前判定繰返し制御)

繰返し条件が「TRUE(真)」の間、繰返し処理文を実行する制御文です。

WHILE ~ END_WHILE;	
基本書式	書式説明
WHILE ␣ <条件式> ␣ DO 　<処理文>; END_WHILE;	条件式が成り立つ時、※1 処理文を実行します。 文頭(WHILE)に戻ります。;

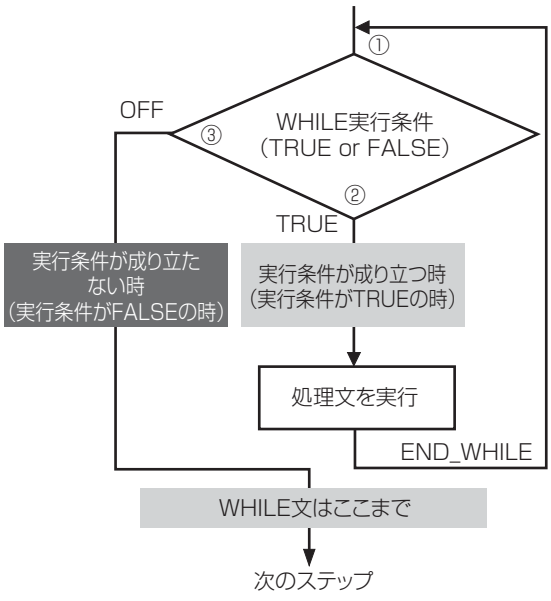
※ ␣は半角スペース挿入を表します。
※1　条件式が成り立たない時、繰返し処理を終了します。

書式例	解説

WHILE ␣ uiValue < 100 ␣ DO
　uiValue := uiValue + 5;
END_WHILE;
	uiValueの値が100未満の場合、
	uiValueに5を加算します。
	文頭(WHILE)に戻る。

※2　uiValueの値が100以上だったら、処理文を実行せずにEND_WHILEの次の行から実行します。

処理の流れ



①WHILE実行条件で「TRUE(真)」か「FALSE(偽)」を判定し、WHILE～END_WHILE間の処理方法を決めます。

②実行条件が「TRUE」の場合、処理文を実行し、①(WHILE文の文頭)に戻ります。

③条件式が「FALSE」の場合、END_WHILE文の次のステップに進みます。

REPEAT文(後判定繰り返し制御)

UNTILの条件が「FALSE」の間繰り返し処理文を実行する制御文です。

REPEAT~ END_REPEAT	
基本書式	書式説明
REPEAT 　<処理文> 　UNTIL ␣ <条件式> END_REPEAT;	繰り返し制御を行います。 処理文を実行します。 条件式が成り立たないとき、文頭(REPEAT)に戻ります。※1

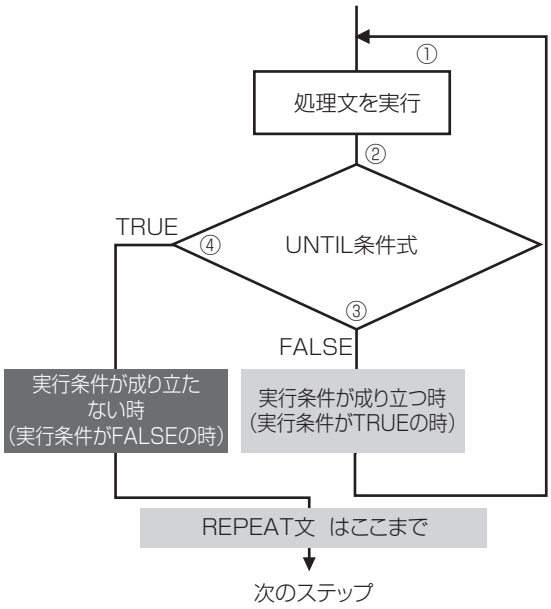
※1　条件式が成り立つとき、繰り返し処理を終了します。

書式例	解説

REPEAT
　uiValue := uiValue + 5;
　UNTIL␣uiValue >= 100
END_REPEAT;
	次ステップからの処理を行います。
	uiValueに5を加算します。
	uiValueの値が100以上でない場合、文頭(REPEAT)に戻ります。※2

※2　uiValueの値が100以上だったら、処理文を実行せずにEND_REPEATの次の行から実行します。

処理の流れ



①処理文を実行します。

②UNTIL条件式で「TRUE」か「FALSE」かを判定し、繰り返し処理を継続するかを決めます。

③条件式が「FALSE」条件式、①に戻ります。

④条件式が「TRUE」の場合、次のステップに進みます。

EXIT処理

繰り返し処理を強制的に終了(繰り返しから抜ける)する際に使用します。

EXIT	
基本書式	書式説明
<繰り返し制御文の始まり> 　EXIT; <繰り返し制御文の終わり>	繰り返し処理の中に記述します。※1 この繰り返し処理を、強制的に終了します。 繰り返し制御文の終わり。

※1　EXITは必ず、繰り返し制御文の中に記述してください。

書式例	解説

FOR␣uiValue1 := 0␣TO␣50␣BY␣2 DO
　uiValue2 := uiValue2 + 5;
　IF␣uiValue2 > 70␣THEN
　　EXIT;
　END_IF;
END_FOR;
	uiValue1が0から50になるまで、繰り返します。
	uiValue2に5を加算します。
	もし、uiValue2の値が70より大きくなったら、
	この繰り返し制御(FOR~END_FOR)を強制終了します。
	IF文の終わりです。
	uiValue1に2を加算して、FOR文の文頭に戻ります。

※ 繰り返し処理文は、FOR以外にも、WHILE文、REPEAT文の文中にも使用できます。

ポイント　**EXITは必ず繰り返し処理の内側に書かなければいけません。**
繰り返し処理の外に書くと、変換エラーになります。

RETURN処理

RETURNが記述されているプログラム（モジュール/マクロ/ファンクションブロック/ファンクション）の実行を強制的に終了する際に使用します。

RETURN	
基本書式	書式説明
<処理文> RETURN; <処理文>	記述されているプログラムの実行を終了します。

書式例	解説
FOR␣uiValue := 0␣TO␣50␣BY␣2␣DO 　uiValue2 := uiValue2 + 5; 　IF␣uiValue2 > 70␣THEN 　　RETURN; 　END_IF; END_FOR;	uiValue1が0から50になるまで、繰り返します。 uiValue2に5を加算します。 もし、uiValue2が70より大きくなったら、 このプログラムのEND命令にジャンプして、プログラムの実行を終了します。 IF文の終わりです。 uiValue1に2を加算して、FOR文の文頭に戻ります。

制御文を使う時の注意

繰返し制御文と処理時間

STでは、繰返し制御文(FOR, WHILE, REPEAT)を使うことにより、繰り返し処理を簡単に記述することができますが、PLCの許容スキャンタイムを越えない範囲で繰返し処理をおこなう必要があります。

■処理時間について

繰り返し回数が多いプログラムを記述した場合のスキャンタイムを検討します。
例) 1+2+3+‥‥200,000を計算して1～200,000の総和を求めます。

ST記述

```
udiIdx := 0;
udiSum := 0;
WHILE␣udiIdx < 200000␣DO                //①～②の処理を200,000回
                                        //繰り返します
    udiIdx := udiIdx + 1;               //①
    udiSum := udiSum + udiIdx;          //②
END_WHILE;
```

使用できるデバイスはユニットごとに異なります。「インデックス修飾と間接指定」3-25ページを参照してください。
詳細は、「WHILE文(前判定繰返し制御)」（4-14ページ）を参照してください。
上記サンプルプログラムの処理時間は以下のようになります。

1 回の繰り返し時間	→	約1.65μs
200,000 回の繰り返し時間	→	約1.65μs×200,000=約330,000μs（330ms）

※上記処理時間はあくまで参考例です。実際の処理時には、接続している拡張ユニットの台数や使用環境により異なります。

この場合、繰り返し処理が1回だけだと問題ありませんが、200,000回の繰り返し処理をおこなうようになっているため、総処理時間が許容スキャンタイム（300ms）を超えてしまいます。

STでプログラムを使用すると、少ない記述で複雑なプログラムや繰り返し処理が簡単に記述できますが、スキャンタイムに注意する必要があります。

参 考　ラダープログラムが許容スキャンタイム(300ms)を越えた場合スキャンタイムオーバー(エラー番号:30)が発生します。

繰り返し制御文内の処理時間が長い(複雑な演算処理を実行する)場合は、少ない繰り返し回数でも、スキャンタイムオーバーが発生する場合があります。
スキャンタイムオーバーを回避するためには、数スキャンに渡って(あるいはスキャン毎に)処理を実行するように、記述を変更する必要があります。

例）1+2+3+‥‥200,000を計算し、1～200,000の総和を求めます。
(スキャンタイムオーバーを回避する記述)

ST記述

```
IF ␣bFlag = FALSE ␣THEN
    bFlag := TRUE;                    //演算中のフラグ
    udiIdx := 0;                      //udiIdxの初期化
    udiSum := 0;                      //udiSumの初期化
END_IF;
IF ␣udiIdx < 200000 ␣THEN
    udiIdx := udiIdx + 1;             //カウンタ
    udiSum := udiSum + udiIdx;        //値1～200,000の総和の演算
END_IF;
```

動作説明　bFlagがFALSEの時、udiIdxとudiSumに0を格納(初期化)し、bFlagをTRUEにします。
bFlagがTRUEで、udiIdxが200000未満の時、udiIdxの現在値に1を加算し、1~200000の値を順に加算しudiSumに格納します(演算はスキャン毎に実行されます)。

詳細は、「IF ～ 文(条件分岐 ①)」(4-8ページ)を参照してください。

■ 繰返し制御文中のタイマ動作について

繰返し制御文の強制終了にタイマを使用することはできません。
繰り返し制御中はタイマ･カウンタ接点が変化しないため、下記のSTは無限ループになります。

【実行させてはいけないST記述】

```
TMH(timer, 1);                        //10msタイマ
WHILE ␣bFlag = FALSE ␣DO              //bFlagがTRUEになれば繰り返し処理は終了
    udiValue := udiValue + 1;
    udiSum := udiSum + udiValue;
    IF ␣timer ␣THEN                   //EXITはタイマ接点timerがTRUEになれば実行
        EXIT;                         //タイマの接点が変化しないので処理文は実行されません
    END_IF;
END_WHILE;
```

「TMR関数」(5-13ページ)

STを記述するときの注意

ST言語を記述するときの注意

タイマ/カウンタは現在値、接点状態を同じ変数で扱います。
接点状態は変数そのもので扱えます。
現在値を取得する場合はCURVAL関数の引数に対象の変数を指定してください。

●タイマ/カウンタ現在値の読出し

udiValue1 := CURVAL(timer);
udiValue2 := CURVAL(counter);
※ STから変数の現在値を直接書き換えることはできません。タイマ･カウンタ関数を使用してください。

●タイマ/カウンタ接点

SET(timer, bFlag1);　　//タイマ接点timerがTRUEのときbFlag1をON(保持)します。
SET(counter = FALSE, bFlag2);　//カウンタ接点counterがFALSEのときbFlag2をON(保持)します。

■タイマ関数を使用するときの注意

タイマ関数は、書式に実行条件を持ちますが他の関数と異なり実行条件はタイマのリセット命令を兼ねています。タイマ関数の実行条件は、省略して記述することができますが、その場合リセットすることができず、常時実行型関数として動作し続けます。
TMR(timer, 500);　//実行条件を省略した場合

動作説明　タイマ変数timerはプログラム起動後5秒経過したら、接点がONであり続けます。
RES(timer)を実行するか、PLCの電源OFFもしくは「RUN⇒PROG」切り替え時にしか、接点はクリアできません。

■ボックス型STやIF文中にタイマ関数を使用するときの注意

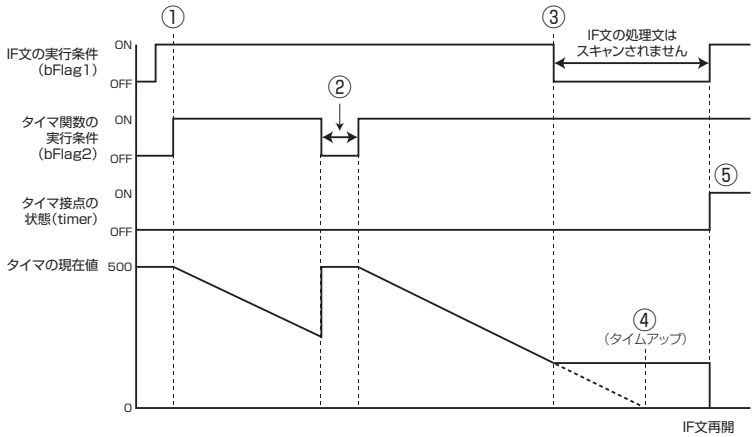
●エリア型STのIF文の中でタイマ関数を使用する

エリア型ST使用時、タイマ関数をIF文の中で使用する場合は、以下の注意が必要です。

ST記述

```
IF ␣bFlag1 = TRUE ␣THEN                    //IF文の中に記述
    TMR(bFlag2 = TRUE, timer, 500);
END_IF;
```

動作説明



① IF文の実行条件(bFlag1)がONして、かつタイマの実行条件(bFlag2)がONしている間、タイマtimerが動作します。動作開始後5秒経過したら、タイマ接点timerをONする設定です。
② IF文の実行条件(bFlag1)がON、かつタイマの実行条件(bFlag2)がOFFになると、タイマはリセットされます。
③ いったん、タイマが動作を開始しタイムアップする前に、タイマの実行条件ONにもかかわらず、IF文の実行条件がOFFになったとき、IF文の処理文(タイマ関数)はスキャンされません。このときこのときタイマ関数は見かけ上一時停止しているように見えますが、実際は動作を続けます(リセット入力が無いため)。
④ ただしタイムアップしても処理文はスキャンされないのでタイマ接点は出力せず、待機状態のようになります。
⑤ 次のIF文の実行条件が成立したスキャンで、タイマ接点はタイマ動作なしにONします。

回避方法　IF文の実行条件がOFFの場合、処理文に含まれるタイマ関数をリセットすることで、この問題は回避できます。
STの記述を以下のように変更します。

ST記述

```
IF ␣bFlag1 = TRUE ␣THEN          //IF文中に記述
   TMR(bFlag2 = TRUE, timer, 500);
ELSE                             //タイマをリセット
   RES(timer);
END_IF;
```

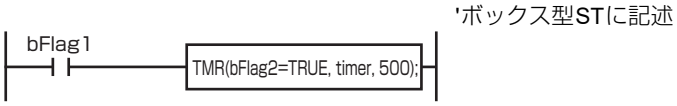
動作説明　IF文の実行条件(bFlag1)がONして、かつタイマの実行条件(bFlag2)がONしている間、タイマtimerが動作します。
動作開始後5秒経過したら、タイマ接点timerをONします。

IF文の実行条件(bFlag1)がOFFになると、タイマ(timer)をリセットします。

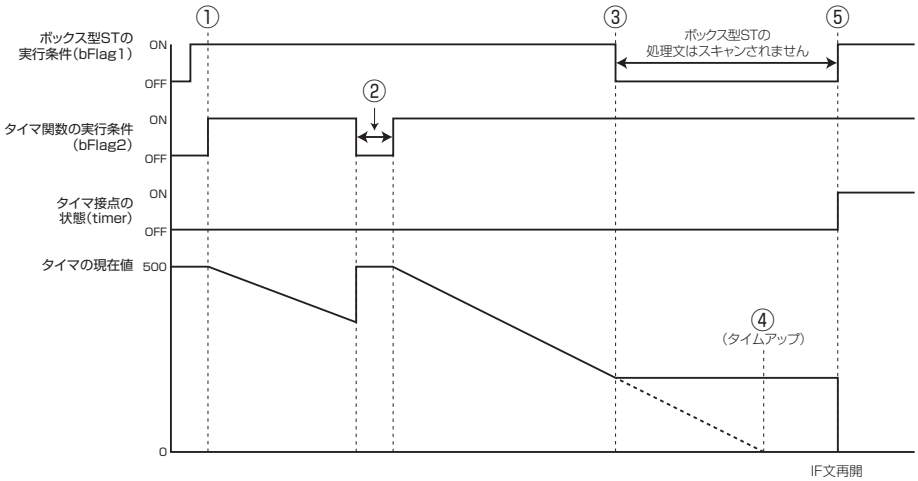
●ボックス型STでタイマ関数を指定する

ボックス型STで使用するとき、タイマ関数はIF文中で使用する場合と同様の注意が必要です。

ST記述



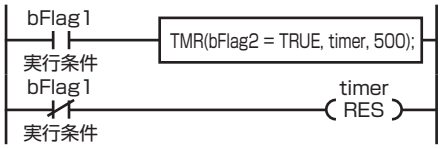
動作説明



① ボックス型STの実行条件(bFlag1)がONして、かつタイマの実行条件(bFlag2)がONしている間、タイマtimerが動作します。動作開始後5秒経過したら、タイマ接点timerをONする設定です。
② ボックス型STの実行条件(bFlag1)がON、かつタイマの実行条件(bFlag2)がOFFになるとタイマはリセットされます。
③ いったん、タイマが動作を開始しタイムアップする前に、タイマの実行条件ONにもかかわらず、ボックス型STの実行条件がOFFになったとき、ボックス型STの処理文(タイマ関数)はスキャンされません。このときタイマ関数は見かけ上一時停止しているように見えますが、実際は動作を続けます(リセット入力が無いため)。
④ ただし設定時間に到達しても処理文はスキャンされないのでタイマ接点は出力せず、待機状態のようになります。
⑤ 次のボックス型STの実行条件が成立したスキャンで、タイマ接点はタイマ動作無しにONします。

回避方法　ボックス型STの実行条件がOFFの場合、ボックス型STに含まれるタイマ関数をリセットすることで、この問題は回避できます。
STの記述を以下のように変更します。

ST記述



動作説明　ボックス型STの実行条件のB接点でタイマをリセットします。

5

基本関数

関数の読み方.....5-2
接点関数.....5-3
出力関数.....5-8
ビット出力関数.....5-10
タイマ･カウンタ関数.....5-13
ダイレクト入出力関数.....5-18

関数の読み方

BMOV　ワードブロックの一括データ転送

[]: 省略可能な引数

ブロックムーブ
BMOV ([EN], Src, Dst, n);

引数のデータ型に対し、実際認識する型を表記

引数/戻り値	説明	認識する型 .U	.S	.D	.L	.F	.DF	.B	.T	定数
EN	実行条件。省略可能。省略時、常時(毎スキャン)実行。	–	–	–	–	–	–	.B	–	○
Src※1	転送元データブロックの先頭を指定。	.U	.U	.D	.D	.D	.DF※2	–	–	○
Dst※1	転送先データブロックの先頭を指定。	型はSrcと同じ。						–	–	–
n	転送データ数、または転送するデータ数の格納先を指定。	.U	.U	.U	.U	–	–	–	–	○
R	戻り値はなし。	–	–	–	–	–	–	–	–	–

nに対し.D(UDINT)のデータ型がの引数が指定された場合、.U(UINT)のデータとして認識して関数を実行することを意味しています。

デバイス / 変数以外に定数の直接入力の可否

ポイント　**表記中のSrc、DstなどはSrc:=DM0などと記載することで、関数内の引数の順を入れ違えて使用できます。**

●変数のデータ型

ST関数で指定する変数のデータ型は先頭のプレフィックスで確認してください。
割り付けるプレフィックスは以下の通りです。

データ型	プレフィックス	データ型	プレフィックス
BOOL	b	BOOL[n]	ab
INT	i	INT[n]	ai
DINT	di	DINT[n]	adi
UINT	ui	UINT[n]	aui
UDINT	udi	UDINT[n]	audi
REAL	r	REAL[n]	ar
LREAL	l	LREAL[n]	al
STRING	str		
TIMER	t		
COUNTER	c		

例) iAnalogData…INT型の変数
　audiSendData…配列型のUDINTの変数

■表記方法の説明

Dst+(1)などの記載については、Dstで指定した変数･デバイスをオフセットしたアドレスを表します。

- デバイスの場合Dstが**DM0**とすると**Dst＋(1)**は**DM1**を表します。
- 変数の場合　**Dst auiValue**(**1**ワードの配列)とすると、**auiValue[1]**を表します。
  Dst adiData(**2**ワードの配列)とすると、**adiData[0]**の上位ワードを表します。
  Dst Sample(構造体型変数)とすると、**Sample**から**1**ワードオフセットしたメンバを表します。

接点関数

LDP　指定ビットの立ち上がり微分

ロードパルス
[R:=] LDP (In);

引数/戻り値	説明	認識する型								定数
		.U	.S	.D	.L	.F	.DF	.B※2	.T	
In※1	立ち上がりを検出する対象を指定。	—	—	—	—	—	—	.B	—	—
R	戻り値。省略可能。動作させる対象を指定。省略時は、演算結果は内部レジスタに格納。	—	—	—	—	—	—	.B	—	—

※1　Inにワードデータを指定した場合、最下位ビットの状態を検出。
※2　ダイレクト入出力リレー(DR)が指定可能。
ワードデータのビット位置を指定可能。

●動作説明

Inの立ち上がり(OFFからONの変化)を検出すると、Rを1スキャンだけONします。

「演算フラグについて（CR2009～CR2012）関数」（1-8ページ）
詳細については、『命令語リファレンスマニュアル』「接点命令」を参照してください。

●書式例

bEdge := **LDP** (bEnable);

ポイント
ボックス型ST内やIF文内で使用するときは注意が必要です。
「ボックス型STを使用するときの注意」（2-9ページ）

参考
・演算式の結果を実行条件に設定することも可能です。
・他関数の実行条件に組み込んで使用することも可能です。

LDF　指定ビットの立ち下がり微分

ロードパルフ
[R:=] LDF (In);

引数/戻り値	説明	認識する型								定数
		.U	.S	.D	.L	.F	.DF	.B※2	.T	
In※1	立ち下がりを検出する対象を指定。	—	—	—	—	—	—	.B	—	—
R	戻り値。省略可能。動作させる対象を指定。省略時は、演算結果は内部レジスタに格納。	—	—	—	—	—	—	.B	—	—

※1　Inにワードデータを指定した場合、最下位ビットの状態を検出。
※2　ダイレクト入出力リレー(DR)が指定可能。
ワードデータのビット位置を指定可能。

●動作説明

Inの立ち下がり(ONからOFFの変化)を検出すると、**R**を1スキャンだけONします。

「演算フラグについて（CR2009～CR2012）」（1-8ページ）
詳細については、『命令語リファレンスマニュアル』「接点命令」を参照してください。

●書式例

bEdge := **LDF** (bEnable);

ポイント
ボックス型ST内やIF文内で使用するときは注意が必要です。
「ボックス型STを使用するときの注意」（2-9ページ）

参考
・演算式の結果を実行条件に設定することも可能です。
・他関数の実行条件に組み込んで使用することも可能です。

LDPB　指定ビットの立ち上がり微分の反転

[R:=] LDPB (In);
（ロードパルスバー）

引数/戻り値	説明	認識する型								定数
		.U	.S	.D	.L	.F	.DF	.B※2	.T	
In※1	立ち上がりを検出する対象を指定。		–	–	–	–	–	.B	–	–
R	戻り値。省略可能。動作させる対象を指定。省略時は、演算結果は内部レジスタに格納。	–	–	–	–	–	–	.B	–	–

※1　**In**にワードデータを指定した場合、最下位ビットの状態を検出。
※2　ダイレクト入出力リレー(DR)が指定可能。
ワードデータのビット位置を指定可能。

●動作説明

Inの立ち上がり(OFFからONの変化)を検出すると、**R**を1スキャンだけOFFします。

「演算フラグについて（CR2009～CR2012）」（1-8ページ）
詳細については、『命令語リファレンスマニュアル』「接点命令」を参照してください。

●書式例

bEdge := **LDPB** (bEnable);

ポイント
ボックス型ST内やIF文内で使用するときは注意が必要です。
「ボックス型STを使用するときの注意」（2-9ページ）

LDFB　指定ビットの立ち下がり微分の反転

ロードパルフバー
[R:=] LDFB (In);

引数/戻り値	説明	認識する型								定数
		.U	.S	.D	.L	.F	.DF	.B※2	.T	
In※1	立ち下がりを検出する対象を指定。	—	—	—	—	—	—	.B	—	—
R	戻り値。省略可能。動作させる対象を指定。省略時は、演算結果は内部レジスタに格納。	—	—	—	—	—	—	.B	—	—

※1　Inにワードデータを指定した場合、最下位ビットの状態を検出。
※2　ダイレクト入出力リレー(DR)が指定可能。
　　ワードデータのビット位置を指定可能。

●動作説明

Inの立ち下がり(ONからOFFの変化)を検出すると、Rを1スキャンだけOFFします。

「演算フラグについて（CR2009～CR2012）」（1-8ページ）
詳細については、『命令語リファレンスマニュアル』「接点命令」を参照してください。

●書式例

bEdge := **LDFB** (bEnable);

ポイント　**ボックス型ST内やIF文内で使用するときは注意が必要です。**
「ボックス型STを使用するときの注意」（2-9ページ）

ビット接点関数

BLD　ワードデータの指定ビット情報

[R:=] BLD (In, Pos);
（ビットロード）

引数/戻り値	説明	対応サフィックス								定数
		.U	.S	.D	.L	.F	.DF	.B※3	.T	
In※1	ビットの状態を読み出す対象を指定。	.U	.U	.D	.D	—	—	—	—	—
Pos※2	対象のON/OFFを判定するビット位置を指定。（16ビットバイナリデータ）	.U	.U	.U	.U	—	—	—	—	○
R	戻り値。省略可能。動作させる対象を指定。省略時は、演算結果は内部レジスタに格納。	—	—	—	—	—	—	.B	—	—

※1　Inにビットデータを指定時は、指定ビット＋Posの位置のビットが指定。チャンネルをまたぐ場合も正常に動作。
※2　ビット位置の値は、Posのサフィックスが.U (UINT) のとき0～15、.D (UDINT) のときは0～31を指定。
※3　ダイレクト入出力リレー(DR)が指定可能。
　　ワードデータのビット位置を指定可能。

●動作説明

InのPosで指定したビット位置のON/OFF状態を、Rに返します。

「演算フラグについて（CR2009～CR2012）」（1-8ページ）
詳細については、『命令語リファレンスマニュアル』「ビット接点命令」を参照してください。

●書式例

bFlag := **BLD** (uiValue, INT#10);

ポイント　**Posに定数以外を記述するとき、演算結果がInのサフィックスに従った範囲を(.Uのときは0～15、.Dのときは0～31)超えた場合、次のようになります。**
Inのサフィックスが.U：Posの下位4ビットの値が使用されます。
Inのサフィックスが.D：Posの下位5ビットの値が使用されます。

BLDB　ワードデータの指定ビット反転情報

ビットロードバー
[R:=] BLDB (In, Pos);

引数/戻り値	説明	対応サフィックス								定数
		.U	.S	.D	.L	.F	.DF	.B※3	.T	
In※1	ビットの状態を読み出す対象を指定。	.U	.U	.D	.D	–	–	–	–	–
Pos※2	対象のON/OFFを判定するビット位置を指定。(16ビットバイナリデータ)	.U	.U	.U	.U	–	–	–	–	○
R	戻り値。省略可能。動作させる対象を指定。省略時は、演算結果は内部レジスタに格納。	–	–	–	–	–	–	.B	–	–

※1　Inにビットデータを指定時は、指定ビット＋Posの位置のビットが指定。チャンネルをまたぐ場合も正常に動作。
※2　ビット位置の値は、Posのサフィックスが.U (UINT) のとき0～15、.D (UDINT) のときは0～31を指定。
※3　ダイレクト入出力リレー(DR)が指定可能。
　　ワードデータのビット位置を指定可能。

●動作説明

InのPosで指定したビット位置のON/OFF状態を、反転してRに返します。

「演算フラグについて(CR2009～CR2012)」(1-8ページ)
詳細については、『命令語リファレンスマニュアル』「ビット接点命令」を参照してください。

●書式例

bFlag := **BLDB** (uiValue, INT#10);

ポイント　**Posに定数以外を記述するとき、演算結果がInのサフィックスに従った範囲を(.Uのときは0～15、.Dのときは0～31)超えた場合、次のようになります。**
Inのサフィックスが.U:Posの下位4ビットの値が使用されます。
Inのサフィックスが.D:Posの下位5ビットの値が使用されます。

出力関数

SET　指定ビットの出力ON

セット
SET ([EN], InOut);

引数/戻り値	説明	認識する型								定数
		.U	.S	.D	.L	.F	.DF	.B※1	.T	
EN	実行条件。省略可能。省略時、常時(毎スキャン)実行。	—	—	—	—	—	—	.B	—	—
InOut	ONして保持する対象を指定。	—	—	—	—	—	—	.B	—	—
R	戻り値なし。	—	—	—	—	—	—	—	—	—

※1　ダイレクト入出力リレー(DR)が指定可能。
ワードデータのビット位置を指定可能。

●動作説明

実行条件が成り立つ(TRUE)とき、**InOut** をONします。一度ONすると、実行条件が不成立になってもONのまま保持されます。(ONした対象はRES関数でOFFします)

「演算フラグについて(CR2009～CR2012)」(1-8ページ)
詳細については、『命令語リファレンスマニュアル』「出力命令」を参照してください。
「RES関数」(5-8ページ)

●書式例

SET (bEnable, bStart);

bEnableがONするとbStartをONします。

RES　指定ビットを出力OFF

リセット
RES ([EN], InOut);

引数/戻り値	説明	認識する型								定数
		.U	.S	.D	.L	.F	.DF	.B※1	.T	
EN	実行条件。省略可能。省略時、常時(毎スキャン)実行。	—	—	—	—	—	—	.B	—	—
InOut	ONして保持する対象を指定。	—	—	—	—	—	—	.B	—	—
R	戻り値なし。	—	—	—	—	—	—		—	—

※1　ダイレクト入出力リレー(DR)が指定可能。
ワードデータのビット位置を指定可能。

●動作説明

実行条件が成り立つ(TRUE)とき、**InOut**をOFFします。(SET関数でONした対象はRES関数でOFFします)

「演算フラグについて(CR2009～CR2012)」(1-8ページ)
詳細については、『命令語リファレンスマニュアル』「出力命令」を参照してください。

「SET関数」(5-8ページ)

●書式例

RES (bComp, bStart);

bCompがONするとbStartをOFFします。

ビット出力関数

BOUT　ワードデータの指定ビット出力

ビットアウト
BOUT ([EN], InOut, Pos);

引数/戻り値	説明	認識する型								定数
		.U	.S	.D	.L	.F	.DF	.B	.T	
EN	実行条件。省略可能。省略時、常時(毎スキャン)実行。	–	–	–	–	–	–	.B	–	–
InOut※1	入力状態を出力する対象を指定。	.U	.U	.D	.D	–	–	–	–	–
Pos※2	出力するビット位置を指定。	.U	.U	.U	.U	–	–	–	–	○
R	戻り値なし。	–	–	–	–	–	–	–	–	–

※1　InOutにビットデータを指定した場合、対象＋Posの値で計算される対象のリレーに出力。チャンネルをまたぐ動作も可能。
※2　Posの値は、対象のサフィックスが.U (UINT) のとき0～15、.D (UDINT) のときは0～31を指定。

●動作説明

実行条件が成り立つ(TRUE)とき、InOutの指定されたPosのビットをONし、実行条件が不成立(FALSE)時、OFFにします。

「演算フラグについて(CR2009～CR2012)」(1-8ページ)
詳細については、『命令語リファレンスマニュアル』「ビット出力命令」を参照してください。

●書式例

- **InOut:対象が16ビットの場合**
  BOUT (bEnable, uiTarget, INT#12);
- **InOut:対象が32ビットの場合**
  BOUT (iAnalogData > INT#1000, diTarget, INT#28);

ポイント
Posに定数以外を記述するとき、演算結果がInOutのサフィックスに従った範囲を(.Uのときは0～15、.Dのときは0～31)超えた場合、次のようになります。
InOutのサフィックスが.U:Posの下位4ビットの値が使用されます。
InOutのサフィックスが.D:Posの下位5ビットの値が使用されます。

BOUB　ワードデータの指定ビット反転出力

ビットアウトバー
BOUB ([EN], InOut, Pos);

引数/戻り値	説明	認識する型								定数
		.U	.S	.D	.L	.F	.DF	.B	.T	
EN	実行条件。省略可能。省略時、常時(毎スキャン)実行。	–	–	–	–	–	–	.B	–	–
InOut※1	入力状態を出力する対象を指定。	.U	.U	.D	.D	–	–	–	–	–
Pos※2	ONして保持する対象を指定。	.U	.U	.U	.U	–	–	–	–	–
R	戻り値なし。	–	–	–	–	–	–	–	–	–

※1　InOutにビットデータを指定した場合、指定ビット＋Posの位置のビットに出力。チャンネルをまたぐ動作も可能。

※2　Posの値は、対象のサフィックスが.U (UINT) のとき0～15、.D (UDINT) のときは0～31を指定。

●動作説明

実行条件が成り立つ(TRUE)とき、InOutの指定されたPosのビットをOFFにし、実行条件が不成立(FALSE)時、ONにします。

「演算フラグについて（CR2009～CR2012）」（1-8ページ）
詳細については、『命令語リファレンスマニュアル』「ビット出力命令」を参照してください。

●書式例

- **InOut：対象が16ビットの場合**
  BOUB (bEnable, uiTarget, INT#12);
- **InOut：対象が32ビットの場合**
  BOUB (iAnalogData > INT#1000, diTarget, INT#28);

ポイント　**Posに定数以外を記述するとき、演算結果がInOutのサフィックスに従った範囲を(.Uのときは0～15、.Dのときは0～31)超えた場合、次のようになります。**
InOutのサフィックスが.U：Posの下位4ビットの値が使用されます。
InOutのサフィックスが.D：Posの下位5ビットの値が使用されます。

BSET　ワードデータの指定ビット出力ON

ビットセット
BSET ([EN], InOut, Pos);

引数/戻り値	説明	認識する型								定数
		.U	.S	.D	.L	.F	.DF	.B	.T	
EN	実行条件。省略可能。省略時、常時（毎スキャン）実行。	—	—	—	—	—	—	.B	—	—
InOut※1	ONして保持する対象を指定。	.U	.U	.D	.D	—	—	—	—	—
Pos※2	ONして保持するビット位置を指定。	.U	.U	.U	.U	—	—	—	—	—
R	戻り値なし。	—	—	—	—	—	—	—	—	—

※1　InOutにビットデータを指定した場合、指定ビット＋ Pos の位置のビットに出力。チャンネルをまたぐ動作も可能。
※2　Posの値は、対象のサフィックスが.U (UINT) のとき0～15、.D (UDINT) のときは0～31を指定。

●動作説明

実行条件が成り立つ(TRUE)とき、InOutの指定されたPosのビットをONします。一度ONすると、実行条件が不成立になってもONのまま保持されます。
(ONした対象はBRES関数でOFFできます)

「演算フラグについて（CR2009～CR2012）」（1-8ページ）
詳細については、『命令語リファレンスマニュアル』「ビット出力命令」を参照してください。
「BRES関数」（5-12ページ）

●書式例

- **InOut：対象が16ビットの場合**
  BSET (bEnable, uiTarget, INT#10);

- **InOut**:対象が32ビットの場合
  BSET (iAnalogData > INT#1000, diTarget, INT#20);

ポイント　**Pos**に定数以外を記述するとき、演算結果が**InOut**のサフィックスに従った範囲を(.U (UINT) のとき0～15、.D (UDINT) のときは0～31) 超えた場合、次のようになります。
InOutのサフィックスが.U:**Pos**の下位4ビットの値が使用されます。
InOutのサフィックスが.D:**Pos**の下位5ビットの値が使用されます。

BRES　ワードデータの指定ビット出力OFF

ビットリセット
BRES ([EN], InOut, Pos);

引数/戻り値	説明	認識する型								定数
		.U	.S	.D	.L	.F	.DF	.B	.T	
EN	実行条件。省略可能。省略時、常時(毎スキャン)実行。	—	—	—	—	—	—	.B	—	—
InOut※1	OFFする対象を指定。	.U	.U	.D	.D	—	—	—	—	—
Pos※2	OFFするビット位置を指定。	.U	.U	.U	.U	—	—	—	—	—
R	戻り値なし。	—	—	—	—	—	—	—	—	—

※1　**InOut**にビットデータを指定した場合、指定ビット＋**Pos**の位置のビットに出力。チャンネルをまたぐ動作も可能。
※2　**Pos**の値は、対象のサフィックスが.U (UINT) のとき0～15、.D (UDINT) のときは0～31を指定。

●動作説明

実行条件が成り立つ(TRUE)とき、**InOut**の指定された**Pos**のビットをOFFします。

「演算フラグについて(CR2009～CR2012)」(1-8ページ)
詳細については、『命令語リファレンスマニュアル』「ビット出力命令」を参照してください。
「BSET関数」(5-11ページ)

●書式例

- **InOut**:対象が16ビットの場合
  BRES (bEnable, uiTarget, INT#5);
- **InOut**:対象が32ビットの場合
  BRES (iAnalogData > INT#1000, diTarget, INT#30);

ポイント　**Pos**に定数以外を記述するとき、演算結果が**InOut**のサフィックスに従った範囲を(.U (UINT) のとき0～15、.D (UDINT) のときは0～31)超えた場合、次のようになります。
InOutのサフィックスが.U:**Pos**の下位4ビットの値が使用されます。
InOutのサフィックスが.D:**Pos**の下位5ビットの値が使用されます。

タイマ・カウンタ関数

TMR　100ms32ビット減算式タイマ

タイマ
TMR ([EN], Timer, In100ms);

引数/戻り値	説明	認識する型								定数
		.U	.S	.D	.L	.F	.DF	.B	.T	
EN	リセットを兼ねる実行条件(OFF時はTMR関数をリセット)。省略可能。省略時は別途RES関数によりタイマをリセット。	—	—	—	—	—	—	.B	—	—
Timer	タイマを指定。(T0～T3999)	—	—	—	—	—	—	—	—	—
In100ms※1	タイマの設定値を100ms単位で指定。(0～4294967295)	.D	.D	.D	.D	—	—	—	—	○
R	戻り値なし。	—	—	—	—	—	—	—	—	—

※1　In100msにビットデータを指定時は連続する32ビット、ワードデータを指定時は連続する2ワードを占有。

●動作説明

実行条件が成り立つ(TRUE)とき、Timerで指定したタイマを起動します。In100msで指定した値が現在値となり、100msおきに減算(-1)されます。現在値が0になると指定したタイマ(接点)がONします。実行条件が不成立となったとき、指定したタイマをOFFして現在値をリセットします。

「演算フラグについて(CR2009～CR2012)」(1-8ページ)
詳細については、『命令語リファレンスマニュアル』「タイマ･カウンタ命令」を参照してください。

●書式例

TMR (bEnable, tTimer, INT#100);

参考　TMRの誤差は、「±(100ms+1スキャンタイム)以内」となります。

ポイント
- **他のTMR、TMH、TMS、TMU関数(命令)、UDT命令と同じデバイス番号は使用できません。**
- **設定値には演算式を指定できます。演算式が扱える範囲は－2147483648～+2147483647の範囲になりますので以下の場合に注意が必要です。**
- **演算式の結果の値が負の場合、演算式の結果に4294967295を加算した値が設定値としてセットされます。**
- **ボックス型STで使用する場合や、サブルーチン内に記述する場合は注意が必要です。**
  「ボックス型STやIF文中にタイマ関数を使用するときの注意」(4-18ページ)

TMH　10ms32ビット減算式タイマ

高速タイマ
TMH ([EN], Timer, In10ms);

引数/戻り値	説明	認識する型								定数
		.U	.S	.D	.L	.F	.DF	.B	.T	
EN	リセットを兼ねる実行条件(OFF時はTMH関数をリセット)。省略可能。省略時は別途RES関数によりタイマをリセット。	—	—	—	—	—	—	.B	—	—
Timer	タイマを指定。(T0～T3999)	—	—	—	—	—	—	—	—	—
In10ms※1	タイマの設定値を10ms単位で指定。(0～4294967295)	.D	.D	.D	.D	—	—	—	—	○
R	戻り値なし。	—	—	—	—	—	—	—	—	—

※1　In10msにビットデータを指定時は連続する32ビット、ワードデータを指定時は連続する2ワードを占有。

●動作説明

実行条件が成り立つ(TRUE)とき、Timerで指定したタイマを起動します。In10msで指定した値が現在値となり、10msおきに減算(-1)されます。現在値が0になると指定したタイマ(接点)がONします。実行条件が不成立となったとき、指定したタイマをOFFして現在値をリセットします。

「演算フラグについて(CR2009～CR2012)」(1-8ページ)
詳細については、『命令語リファレンスマニュアル』「タイマ･カウンタ命令」を参照してください。

●書式例

TMH (bEnable, tTimer, INT#100);

参考　TMHの誤差は、「±(10ms+1スキャンタイム)以内」となります。

ポイント
- **他のTMR、TMH、TMS、TMU関数(命令)、UDT命令と同じデバイス番号は使用できません。**
- **設定値には演算式を指定できます。演算式が扱える範囲は－2147483648～+2147483647の範囲になりますので以下の場合に注意が必要です。**
- **演算式の結果の値が負の場合、演算式の結果に4294967295を加算した値が設定値としてセットされます。**
  ボックス型STで使用する場合や、サブルーチン内に記述する場合は注意が必要です。
  「ボックス型STやIF文中にタイマ関数を使用するときの注意」(4-18ページ)

「TMR関数」(5-13ページ)

TMS　1ms32ビット減算式タイマ

高速1msタイマ
TMS ([EN], Timer, In1ms);

引数/戻り値	説明	認識する型								定数
		.U	.S	.D	.L	.F	.DF	.B	.T	
EN	リセットを兼ねる実行条件(OFF時はTMS関数をリセット)。省略可能。省略時は別途RES関数によりタイマをリセット。	—	—	—	—	—	—	.B	—	—
Timer	タイマを指定。(T0～T3999)	—	—	—	—	—	—	—	—	—
In1ms※1	タイマの設定値を1ms単位で指定。(0～4294967295)	.D	.D	.D	.D	—	—	—	—	○
R	戻り値なし。	—	—	—	—	—	—	—	—	—

※1　In1msにビットデータを指定時は連続する32ビット、ワードデータを指定時は連続する2ワードを占有。

●動作説明

実行条件が成り立つ(TRUE)とき、Timerで指定したタイマを起動します。In1msで指定した値が現在値となり、1msおきに減算(-1)されます。現在値が0になると指定したタイマ(接点)がONします。実行条件が不成立となったとき、指定したタイマをOFFして現在値をリセットします。

「演算フラグについて(CR2009～CR2012)」(1-8ページ)
詳細については、『命令語リファレンスマニュアル』「タイマ･カウンタ命令」を参照してください。

●書式例

TMS (bEnable, tTimer, INT#100);

参考　TMSの誤差は、「±(1ms+1スキャンタイム)以内」となります。

ポイント
- **他のTMR、TMH、TMS、TMU関数(命令)、UDT命令と同じデバイス番号は使用できません。**
- **設定値には演算式を指定できます。演算式が扱える範囲は－2147483648～+2147483647の範囲になりますので以下の場合に注意が必要です。**
- **演算式の結果の値が負の場合、演算式の結果に4294967295を加算した値が設定値としてセットされます。**
  ボックス型STで使用する場合や、サブルーチン内に記述する場合は注意が必要です。
  「ボックス型STやIF文中にタイマ関数を使用するときの注意」(4-18ページ)

「TMR関数」(5-13ページ)

TMU　10μs32ビット減算式タイマ

高速10μsタイマ
TMU ([EN], Timer, In10μs);

引数/戻り値	説明	認識する型								定数
		.U	.S	.D	.L	.F	.DF	.B	.T	
EN	リセットを兼ねる実行条件(OFF時はTMU関数をリセット)。省略可能。省略時は別途RES関数によりタイマをリセット。	—	—	—	—	—	—	.B	—	—
Timer	タイマを指定。(T0～T3999)	—	—	—	—	—	—	—	—	—
In10μs※1	タイマの設定値を10μs単位で指定。(0～4294967295)	.D	.D	.D	.D	—	—	—	—	○
R	戻り値なし。	—	—	—	—	—	—	—	—	—

※1　In10μsにビットデータを指定時は連続する32ビット、ワードデータを指定時は連続する2ワードを占有。

●動作説明

実行条件が成り立つ(TRUE)とき、Timerで指定したタイマを起動します。In10μsで指定した値が現在値となり、10μsおきに減算(-1)されます。現在値が0になると指定したタイマ(接点)がONします。実行条件が不成立となったとき、指定したタイマをOFFして現在値をリセットします。

「演算フラグについて(CR2009～CR2012)」(1-8ページ)
詳細については、『命令語リファレンスマニュアル』「タイマ･カウンタ命令」を参照してください。

●書式例

TMU (bEnable, tTimer, INT#100);

参考　TMUの誤差は、「±(10μs+1スキャンタイム)以内」となります。

ポイント
- **他のTMR、TMH、TMS、TMU関数(命令)、UDT命令と同じデバイス番号は使用できません。**
- **設定値には演算式を指定できます。演算式が扱える範囲は－2147483648～+2147483647の範囲になりますので以下の場合に注意が必要です。**
- **演算式の結果の値が負の場合、演算式の結果に4294967295を加算した値が設定値としてセットされます。ボックス型STで使用する場合や、サブルーチン内に記述する場合は注意が必要です。**
  「ボックス型STやIF文中にタイマ関数を使用するときの注意」(4-18ページ)

「TMR関数」(5-13ページ)

CNT　32ビット加算式カウンタ

カウンタ
CNT ([EN], Counter, PV, CD);

引数/戻り値	説明	認識する型								定数
		.U	.S	.D	.L	.F	.DF	.B	.T	
EN	リセットを兼ねる実行条件(OFF時はCNT関数をリセット)。省略可能。省略時は別途RES関数によりカウンタをリセット。	—	—	—	—	—	—	.B	—	—
Counter	カウンタを指定。(C0～C3999)	—	—	—	—	—	—	—	—	—
PV※1	カウンタの設定値を指定。(0～4294967295)	.D	.D	.D	.D	—	—	—	—	○
CD	カウント入力に使用する対象を指定。	—	—	—	—	—	—	.B	—	—
R	戻り値なし。	—	—	—	—	—	—	—	—	—

※1　PVにビットデータを指定時は連続する32ビット、ワードデータを指定時は連続する2ワードを占有。

●動作説明

カウンタ実行条件が成立したとき、CDで指定したカウント入力の立ち上がりで、指定したカウンタの現在値が加算(+1)されます。現在値がPVで指定した設定値に達すると、CounterがONします。実行条件が不成立となったとき、指定したカウンタをOFFして現在値をリセットします。

「演算フラグについて（CR2009～CR2012）」（1-8ページ）
詳細については、『命令語リファレンスマニュアル』「タイマ･カウンタ命令」を参照してください。

●書式例

CNT (bEnable, cCounter, INT#100, bDetect);

ポイント
- **OUTC、UDC命令と同じデバイス番号は使用できません。**
- **設定値には演算式を指定できます。演算式が扱える範囲は－2147483648～+2147483647の範囲になりますので以下の場合に注意が必要です。**
- **演算式の結果の値が負の場合、演算式の結果に4294967295を加算した値が設定値としてセットされます。**

CURVAL　タイマ/カウンタ現在値取得

タイマ/カウンタ現在値取得
[R:=] CURVAL (In);

引数/戻り値	説明	認識する型								定数
		.U	.S	.D	.L	.F	.DF	.B	.T	
In	現在値を読み出す対象データを指定。	—	—	—	—	—	—	—	—	—
R※1	タイマ/カウンタの現在値が格納される戻り値。省略可能。省略時、内部レジスタに格納。	.U	.S	.D	.L	.F	.DF	—	—	—

※1　Rに演算対象と異なる型を指定した場合、変換時に自動的に型変換を実行。

●動作説明

Inに格納されたタイマ/カウンタの現在値を読み出し、結果をRに格納します。

●書式例

udiValue1 := **CURVAL**(timer);

udiValue2 := **CURVAL**(counter);

※STから変数の現在値を直接書き換えることはできません。タイマ･カウンタ関数を使用してください。

「演算フラグについて（CR2009～CR2012）」（1-8ページ）

ダイレクト入出力関数

RFSX　入力リフレッシュ

リフレッシュX
RFSX ([EN], Start, Count);

引数/戻り値	説明	認識する型								定数
		.U	.S	.D	.L	.F	.DF	.B	.T	
EN	実行条件。省略可能。省略時、常時(毎スキャン)実行。	—	—	—	—	—	—	.B	—	—
Start	入力状態をリフレッシュする対象の先頭を指定	—	—	—	—	—	—	.B	—	—
Count※1	入力状態をリフレッシュする対象のリレー点数を指定。(1～256)※1	.U	.U	.U	.U	—	—	—	—	○
R	戻り値なし。	—	—	—	—	—	—	—	—	—

※1　ビットデータを指定時、連続した16ビットで演算。
チャンネルの先頭以外(R002, R1012など)を指定時、次のチャンネルにまたがって16ビットで演算。
リレー点数の指定可能範囲は1～256。

●動作説明

通常、ラダープログラムのEND命令実行時後に一括しておこなうリフレッシュ処理を、1スキャン内の任意のタイミングで入力リフレッシュする場合に使用します。
実行条件が成り立つ(TRUE)とき、Startを先頭にCountで指定した点数分のビットデータの状態を入力リフレッシュします。

「演算フラグについて(CR2009～CR2012)」(1-8ページ)
詳細については、『命令語リファレンスマニュアル』「ダイレクト入出力命令」を参照してください。

●書式例

RFSX (bFlag1, R30000, INT#128);

RFSY　出力リフレッシュ

リフレッシュY
RFSY ([EN], Start, Count);

引数/戻り値	説明	認識する型								定数
		.U	.S	.D	.L	.F	.DF	.B	.T	
EN	実行条件。省略可能。省略時、常時(毎スキャン)実行。	—	—	—	—	—	—	.B	—	—
Start	出力状態をリフレッシュする対象の先頭を指定。	—	—	—	—	—	—	.B	—	—
Count※1	出力状態をリフレッシュする対象のリレー点数を指定。(1～256)?1	.U	.U	.U	.U	—	—	—	—	○
R	戻り値なし。	—	—	—	—	—	—	—	—	—

※1　ビットデータを指定時、連続した16ビットで演算。
チャンネルの先頭以外(R002, R1012など)を指定時、次のチャンネルにまたがって16ビットで演算。
リレー点数の指定可能範囲は1～256。

●動作説明

通常、ラダープログラムのEND命令実行時後に一括しておこなうリフレッシュ処理を、1スキャン内の任意のタイミングで出力リフレッシュする場合に使用します。
実行条件が成り立つ(TRUE)とき、**Start**を先頭に**Count**で指定した点数分のビットデータの状態を出力リフレッシュします。

「演算フラグについて（CR2009～CR2012）」（1-8ページ）
詳細については、『命令語リファレンスマニュアル』「ダイレクト入出力命令」を参照してください。

●書式例

RFSY (bFlag1, R30200, INT#32);

6

応用関数

フロー関数 6-2
ファンクションブロック関数 6-4
インデックスレジスタ関数 6-7
間接指定関数 6-9
ファイルレジスタ関数 6-13
ユニットプログラム関数 6-15
バッファメモリ関数 6-21
ユニット種別判定関数 6-23
配列要素数取得関数 6-24
ワードサイズ取得関数 6-25

フロー関数

CALL　指定サブルーチンの起動

コール
CALL ([EN], In);

引数/戻り値	説明	認識する型								定数
		.U	.S	.D	.L	.F	.DF	.B	.T	
EN	実行条件。省略可能。省略時、常時(毎スキャン)実行。	—	—	—	—	—	—	.B	—	—
In	起動するサブルーチンNo.を指定。(0～99)※1	—	—	—	—	—	—	—	—	○
R	戻り値なし。	—	—	—	—	—	—	—	—	—

※1　Inは0～99の範囲で指定。16# (16進数指定) では指定不可。

●動作説明

実行条件が成り立つ(TRUE)とき、指定されたInのサブルーチンを起動します。
詳細については、『命令語リファレンスマニュアル』「フロー命令」を参照してください。

●書式例

CALL (bFlag, INT#10);

参 考　サブルーチンは16重までネスティング(入れ子構造に)できます。

ポイント
- **サブルーチン番号は定数のみ指定可能で、対象や対象を含む式は使用できません。**
- **サブルーチン番号の定数は、16# (16進数指定) での指定はできません。**
- **SBN命令やRET命令はST言語関数で用意されていないので、ラダーで記述してください。**

ECALL　モジュール間サブルーチンコール

Eコール
ECALL ([EN], Name, No);

引数/戻り値	説明	認識する型								定数
		.U	.S	.D	.L	.F	.DF	.B	.T	
EN	実行条件。省略可能。省略時、常時(毎スキャン)実行。	—	—	—	—	—	—	.B	—	—
Name	モジュール名を文字列で指定。	—	—	—	—	—	—	—	—	○※1
No	呼び出すサブルーチンNo.を指定。(0～99)※2	—	—	—	—	—	—	—	—	○
R	戻り値なし。	—	—	—	—	—	—	—	—	—

※1　《KV STUDIO》で設定したモジュール名を' 'で囲って入力。
※2　Noは0～99の範囲で指定。16# (16進数指定) では指定不可。

●動作説明

実行条件が成り立つ(TRUE)とき、Nameで指定したモジュールのNoで指定したサブルーチンプログラムを起動します。
すべての種類のモジュール(毎スキャン、初期化、待機、定周期)を指定できますが、ファンクション、ファンクションブロック、マクロは指定できません。

「演算フラグについて（CR2009～CR2012）」（1-8ページ）
詳細については、『命令語リファレンスマニュアル』「フロー命令」を参照してください。

●書式例

ECALL (bFlag, 'NewModule', INT#1);

ポイント
- **サブルーチン番号の定数は、16# (16進数指定) での指定はできません。**
- **SBN命令やRET命令はST言語関数で用意されていないので、ラダーで記述してください。**

ファンクションブロック関数

FBSTRT　ファンクションブロックスタート

ファンクションブロックスタート
FBSTRT ([EN], 'FuncBlock', Instance, Argument);

引数/戻り値	説明	認識する型								定数
		.U	.S	.D	.L	.F	.DF	.B	.T	
EN	実行条件。省略可能。省略時、常時(毎スキャン)実行。	–	–	–	–	–	–	.B	–	–
FuncBlock	ファンクションブロック名を文字列で指定。	–	–	–	–	–	–	–	–	○※2
Instance	インスタンスを文字列で指定。	–	–	–	–	–	–	–	–	○
Argument※1	引数を指定。	–	–	–	–	–	–	–	–	○
R	戻り値なし。	–	–	–	–	–	–	–	–	–

※1　引数の個数や属性は、実行するファンクションブロックに設定されている引数設定で設定。
『命令語リファレンスマニュアル』「ファンクションブロック命令」
※2　《KV STUDIO》で設定したファンクションブロック名を' 'で囲って入力。

●動作説明

実行条件が成り立つ(TRUE)とき、**FuncBlock**で指定したファンクションブロックの**Instance**で指定したインスタンスを実行します。実行条件をOFFしてもインスタンスの動作は継続します。ファンクションブロック内でFEND命令が実行されると動作を停止します。**Argument**で指定した引数をファンクションブロックに引き渡します。ファンクションブロックは引数設定のデータ型に従い、引数を扱います。

詳細については、『命令語リファレンスマニュアル』「ファンクションブロック命令」を参照してください。

●書式例

FBSTRT (bEnable, 'NewFunctionBlock', FunctionA, uiValue);

参考　名前付き引数で記述すると、ファンクションブロックの引数設定で引数の順番が変わった場合でも引数の指定順を変更する必要がなくなります。名前付き引数は以下のとおり記述します。
FBSTRT(R3000, 'ファンクションブロックA', インスタンスB, Input1:=1, OutPut1=>DM10, ENO=>MR000)
IN引数、IN-OUT引数は:=、OUT引数は=>で引数を指定します。ENOは使用しない場合、省略可能です。
名前付き引数では、ユニット引数は記述できません。ユニット引数を設定しているファンクションブロックは、ラダープログラムで記述してください。

ポイント
- **ファンクションブロック名には' 'で囲んだファンクションブロック名以外の記述はできません。**
- **インスタンスは' 'で囲む必要はありません。**
- **ファンクションブロックインスタンスを使用してファンクションブロックを呼び出すことも可能です。ファンクションブロックの呼び出し方法については、「ファンクションブロックの呼び出し」(2-25ページ)を参照してください。**
  参考例)FunctionA(uiValue);
- **ファンクションを呼び出す場合は、ファンクション名を指定して直接呼び出してください。ファンクションの呼び出し方法については、「ファンクションブロックの呼び出し」(2-25ページ)を参照してください。**
  参考例)NewFunction(uiValue);
- **詳細は『KV-8000シリーズユーザーズマニュアル』「ファンクションブロック」を参照してください。**

FEND　ファンクションブロックの終了

ファンクションブロックエンド
FEND ([EN]);

引数/戻り値	説明	認識する型								定数
		.U	.S	.D	.L	.F	.DF	.B	.T	
EN	実行条件。省略可能。省略時、常時(毎スキャン)実行。	—	—	—	—	—	—	.B	—	—
R	戻り値なし。	—	—	—	—	—	—	—	—	—

●動作説明

実行条件が成り立つ(TRUE)とき、FEND関数が記述されているファンクションブロックを停止します。

詳細については、『命令語リファレンスマニュアル』「ファンクションブロック命令」を参照してください。

●書式例

FEND (bFlag);

参考　「FBSTRT関数」(6-4ページ)

ポイント　**FEND関数は、ファンクションブロック内でのみ使用可能です。**

MDSTRT　モジュール実行開始

モジュールスタート
MDSTRT ([EN], Name);

引数/戻り値	説明	認識する型								定数
		.U	.S	.D	.L	.F	.DF	.B	.T	
EN	実行条件。省略可能。省略時、常時(毎スキャン)実行。	—	—	—	—	—	—	.B	—	—
Name	モジュール名を文字列で指定。	—	—	—	—	—	—	—	—	○※1
R	戻り値はなし。	—	—	—	—	—	—	—	—	—

※1　《KV STUDIO》で設定したモジュール名を' 'で囲って入力。

●動作説明

実行条件が成り立つ(TRUE)とき、**Name**で指定したモジュールを起動します。
起動したモジュールは、次のスキャンから動作を開始します。
すでにモジュールが起動している場合、何もしません。

「演算フラグについて(CR2009～CR2012)」(1-8ページ)
詳細については、『命令語リファレンスマニュアル』「モジュール命令」を参照してください。

●書式例

MDSTRT (bEnable, 'NewModule');

ポイント
- モジュール実行開始時における微分実行型命令の初期状態は、立ち上がり検出の場合が ON、立ち下がり検出の場合がOFFになります。実行開始時に微分を直ちに検出することはありません。
- 同じモジュールに対して同じスキャン内にMDSTRT関数(命令)、MDSTOP関数(命令)を複数回実行した場合は最後に実行したモジュール命令に従います。
- 初期化モジュールは実行開始・停止できません。

MDSTOP モジュール実行停止

モジュールストップ
MDSTOP ([EN], Name);

引数/戻り値	説明	認識する型								定数
		.U	.S	.D	.L	.F	.DF	.B	.T	
EN	実行条件。省略可能。省略時、常時(毎スキャン)実行。	–	–	–	–	–	–	.B	–	–
Name	モジュール名を文字列で指定。	–	–	–	–	–	–	–	–	○※1
R	戻り値はなし。	–	–	–	–	–	–	–	–	–

※1　《KV STUDIO》で設定したモジュール名を' 'で囲って入力。

●動作説明

実行条件が成り立つ(TRUE)とき、Nameで指定したモジュールを停止します。
すでに停止しているモジュールに対しては、何もしません。
自分自身のモジュールを停止することも可能です。

「演算フラグについて(CR2009～CR2012)」(1-8ページ)
詳細については、『命令語リファレンスマニュアル』「モジュール命令」を参照してください。

●書式例

MDSTOP (bDisable, 'NewModule');

ポイント
- 同じモジュールに対して同じスキャン内にMDSTRT関数(命令)、MDSTOP関数(命令)を複数回実行した場合は最後に実行したモジュール命令に従います。
- 全てのモジュールを停止しないでください。全モジュールを停止した場合、以下のいずれかの方法で復帰してください。(一旦プログラムを停止しなければ復帰できません。)
  - CPUユニットの電源を一度切り、再度、電源を入れてください。
  - CPUユニットの「RUN-PROG切換スイッチ」を一度、'PROGモード' に切り換えて再度、RUNモードにしてください。
  - 《KV STUDIO》でプログラムのRUNを一度OFFし、再度ONしてください。
- 初期化モジュールは実行開始・停止できません。

インデックスレジスタ関数

ZPUSH　インデックスレジスタの一括退避

Zプッシュ
ZPUSH ([EN], Dst);

引数/戻り値	説明	認識する型								定数
		.U	.S	.D	.L	.F	.DF	.B	.T	
EN	実行条件。省略可能。省略時、常時(毎スキャン)実行。	—	—	—	—	—	—	.B	—	—
Dst※1	退避先データブロックの先頭を指定。	S※2	S※2	.L※3	.L※3	—	—	—	—	—
R	戻り値はなし。	—	—	—	—	—	—	—	—	—

※1　T、C、CR、Zは使用不可。
※2　退避先の型を.Sで指定した場合:
- 連続する13ワードを占有。ネスティングをする場合、(退避する回数×12ワード＋1) ワードを占有。
- 下位16ビットのみ占有。

※3　退避先の型を.Lで指定した場合:
連続する25ワードを占有。ネスティングをする場合、(退避する回数×24ワード＋1) ワードを占有。

●動作説明

実行条件が成り立つ(TRUE)とき、**Dst**で指定したデータブロックにZ1～Z10のインデックスレジスタの値を格納(退避)します。関数実行後**Dst**+(0)の値に+1します。

ポイント
- **最初の「ZPUSH関数」の実行前に、Dst+(0)に0をセットしてください。2回目以降の「ZPUSH関数」の場合は必要ありません。**
- **「ZPUSH関数」と「ZPOP関数」は対で使用します。**
- **「ZPUSH関数」で占有するの範囲には、他の関数、命令等で書き込まないでください。**
- **データブロックの範囲がデバイスの範囲や変数として設定した範囲を超えた場合、関数は実行されません。**

「演算フラグについて(CR2009～CR2012)」(1-8ページ)

参考　「ZPUSH関数」と「ZPOP関数」はネスティング構造にできます。ネスティングをおこなう場合、同じ先頭を指定します。

ポイント
Z(インデックスレジスタ)を.L(32ビット符号付き)として扱うため、Dstに16ビット型サフィックス(.S、.U)を指定した場合、全てのZの下位16ビットのみを退避先に格納し、上位16ビットを切り捨てます。
この場合、ZPOP関数(命令)を使用しても退避時の値は復帰できません。
Zに.Sの範囲(－32768～+32767)を超える値を格納するプログラムを使用している場合は、Dstに.Sを指定しないでください。

●書式例

ZPUSH (bEnable, auiValue[0]);

詳細については、『命令語リファレンスマニュアル』「インデックスレジスタ命令」を参照してください。

ZPOP　インデックスレジスタの一括復帰

Zポップ
ZPOP ([EN], Src);

引数/戻り値	説明	認識する型								定数
		.U	.S	.D	.L	.F	.DF	.B	.T	
EN	実行条件。省略可能。省略時、常時(毎スキャン)実行。	–	–	–	–	–	–	.B	–	–
Src※1	退避先データブロックの先頭を指定。	S※2	S※2	.L※3	.L※3	–	–	–	–	–
R	戻り値はなし。	–	–	–	–	–	–	–	–	–

※1　T、C、CR、Zは使用不可。
※2　退避先の型を.Sで指定した場合：
- 連続する13ワードを占有。ネスティングをする場合、(退避する回数×12ワード＋1) ワードを占有。
- Zの下位16ビットのみ占有。

※3　退避先の型を.Lで指定した場合：
- 連続する25ワードを占有。ネスティングをする場合、(退避する回数×24ワード＋1)ワードを占有。

●動作説明

実行条件が成り立つ(TRUE)とき、Srcで指定したデータブロック(「ZPUSH関数」で格納したインデックスレジスタ)の値を、Z1～Z10のインデックスレジスタに読み出します。関数実行後Src+(0) の値を-1します。

ポイント
- **「ZPUSH関数」と「ZPOP関数」は対で使用します。**
- **次のいずれかの条件が成立した場合、関数は実行されません。**
  - **[Src+0]が0の場合**
  - **データブロックの範囲がデバイスの範囲や変数として設定した範囲を超えた場合**

「演算フラグについて（CR2009～CR2012）」（1-8ページ）

ポイント
(インデックスレジスタ)を.L(32ビット符号付き)として扱うため、Srcに16ビット型サフィックス(.S、.U)を指定した場合、全てのZの下位16ビットのみを退避先に格納し、上位16ビットを切り捨てます。
この場合、ZPOP関数(命令)を使用しても退避時の値は復帰できません。
Zに.Sの範囲(-32768～+32767)を超える値を格納するプログラムを使用している場合は、Src退避先に.Sを指定しないでください。

●書式例

ZPOP (bEnable, auiValue[0]);

詳細については、『命令語リファレンスマニュアル』「インデックスレジスタ命令」を参照してください。

間接指定関数

ADRSET　間接指定対象のアドレス格納

アドレスセット
ADRSET ([EN], Src, Dst);

引数/戻り値	説明	認識する型								定数
		.U	.S	.D	.L	.F	.DF	.B	.T	
EN	実行条件。省略可能。省略時、常時(毎スキャン)実行。	—	—	—	—	—	—	.B	—	—
Src	間接指定される対象を指定。	.U	.S	.D	.D	.F	.DF	.B	.T	—
Dst※1※2	アドレス格納先を指定。	—	—	.D	.D	—	—	—	—	—
R	戻り値はなし。	—	—	—	—	—	—	—	—	—

※1　連続する2ワードを使用。
※2　CMは指定不可。

●動作説明

実行条件が成り立つ(TRUE)とき、Srcのアドレスを、Dstを先頭とする連続した2ワードに格納します。格納後、間接指定(*格納先)で記述すると、間接指定を直接指定するのと同じ結果になります。

「演算フラグについて(CR2009～CR2012)」(1-8ページ)
詳細については、『命令語リファレンスマニュアル』「間接指定命令」を参照してください。

●書式例

ADRSET (bEnable, R0, udiAddress);

ポイント
- **アドレス格納先は、対象番号を先頭とする連続した2ワードを必ず占有します。**
- **間接指定(*格納先)を使用する前には、必ずADRSET関数で間接指定アドレスを格納先にセットしてください。**
- **アドレスの生成、および操作は、必ずADRSET, ADRINC, ADRDEC, ADRADD, ADRSUB関数を使用しておこなってください。**

ADRINC　格納アドレスに1を加算

アドレスインクリメント
ADRINC ([EN], InOut);

引数/戻り値	説明	認識する型								定数
		.U	.S	.D	.L	.F	.DF	.B	.T	
EN	実行条件。省略可能。省略時、常時(毎スキャン)実行。	—	—	—	—	—	—	.B	—	—
InOut※1※2	間接指定のアドレスの格納先を指定。	—	—	.D	.D	—	—	—	—	—
R	戻り値はなし。	—	—	—	—	—	—	—	—	—

※1　連続する2ワードを使用。
※2　CMは指定不可。

●動作説明

実行条件が成り立つ(TRUE)とき、InOutに格納されている間接指定アドレスを1つ次のアドレスにします。ワードデータアドレスが格納されているときは、次のワードデータのアドレスを示し、またビットデータアドレスが格納されているときは、次のビットデータのアドレスを示します。

「演算フラグについて（CR2009～CR2012）」（1-8ページ）
詳細については、『命令語リファレンスマニュアル』「間接指定命令」を参照してください。

●書式例

ADRINC (bEnable, udiAddress);

ポイント

- **アドレス格納対象は、連続した2ワードを必ず占有します。**
- **ADRINC関数を使用する前には、必ずADRSET関数で間接指定アドレスをアドレス格納対象にセットしてください。**
- **アドレスの生成、および操作は、必ずADRSET, ADRINC, ADRDEC, ADRADD, ADRSUB関数を使用しておこなってください。**
- **次のいずれかの条件が成立した場合、関数は実行されません。**
  - **InOutに格納されているアドレスが不適切な場合。**
  - **加算後のアドレスが不適切な場合。**

ADRDEC　格納アドレスから1を減算

アドレスデクリメント
ADRDEC ([EN], InOut);

引数/戻り値	説明	認識する型								定数
		.U	.S	.D	.L	.F	.DF	.B	.T	
EN	実行条件。省略可能。省略時、常時（毎スキャン）実行。	—	—	—	—	—	—	.B	—	—
InOut※1※2	間接指定のアドレスの格納先を指定。	—	—	.D	.D	—	—	—	—	—
R	戻り値はなし。	—	—	—	—	—	—	—	—	—

※1　連続する2ワードを使用。
※2　CMは指定不可。

●動作説明

実行条件が成り立つ(TRUE)とき、InOutに格納されている間接指定アドレスを1つ前のアドレスにします。ワードデータアドレスが格納されているときは、前のワードデータのアドレスを示し、またビットデータアドレスが格納されているときは、前のビットデータのアドレスを示します。

「演算フラグについて（CR2009～CR2012）」（1-8ページ）
詳細については、『命令語リファレンスマニュアル』「間接指定命令」を参照してください。

●書式例

ADRDEC (bEnable, udiAddress);

ポイント
- アドレス格納対象は、連続した2ワードを必ず占有します。
- ADRDEC関数を使用する前には、必ずADRSET関数で間接指定アドレスをアドレス格納対象にセットしてください。
- アドレスの生成、および操作は、必ずADRSET, ADRINC, ADRDEC, ADRADD, ADRSUB関数を使用しておこなってください。
- 次のいずれかの条件が成立した場合、関数は実行されません。
  - InOutに格納されているアドレスが不適切な場合。
  - 減算後のアドレスが不適切な場合。

ADRADD 格納アドレスを加算

ADRADD (**[EN]**, **Offset**, **InOut**);
（アドレスアッド）

引数/戻り値	説明	認識する型								定数
		.U	.S	.D	.L	.F	.DF	.B	.T	
EN	実行条件。省略可能。省略時、常時（毎スキャン）実行。	—	—	—	—	—	—	.B	—	—
Offset	アドレス加算データ、または格納先を指定。	.S	.S	.L	.L	—	—	—	—	○
InOut※1	間接指定のアドレスの格納先を指定。	—	—	.D	.D	—	—	—	—	—
R	戻り値はなし。	—	—	—	—	—	—	—	—	—

※1　CMは指定不可。

●動作説明

実行条件が成り立つ(TRUE)とき、InOutに格納されている間接指定アドレスをOffsetで指定された値だけ加算したアドレスにします。ワードデータのアドレスが格納されているときは、アドレス加算値だけ加算したワードデータのアドレスを示し、またビットデータのアドレスが格納されているときは、アドレス加算値だけ加算したビットデータを示します。

「演算フラグについて（CR2009～CR2012）」（1-8ページ）
詳細については、『命令語リファレンスマニュアル』「間接指定命令」を参照してください。

●書式例

ADRADD (bEnable,INT#10, udiAddress);

ポイント
- アドレス格納対象は、連続した2ワードを必ず占有します。
- ADRADD関数を使用する前には、必ずADRSET関数で間接指定アドレスをアドレス格納対象にセットしてください。
- アドレスの生成、および操作は、必ずADRSET, ADRINC, ADRDEC, ADRADD, ADRSUB関数を使用しておこなってください。
- 次のいずれかの条件が成立した場合、関数は実行されません。
  - InOutに格納されているアドレスが不適切な場合。
  - 加算後のアドレスが不適切な場合。

ADRSUB 格納アドレスを減算

アドレスサブ
ADRSUB ([EN], Offset, InOut);

引数/戻り値	説明	認識する型								定数
		.U	.S	.D	.L	.F	.DF	.B	.T	
EN	実行条件。省略可能。省略時、常時(毎スキャン)実行。	—	—	—	—	—	—	.B	—	—
Offset	アドレス減算データ、または格納先を指定。	.S	.S	.L	.L	—	—	—	—	○
InOut※1	間接指定のアドレスの格納先を指定。	—	—	.D	.D	—	—	—	—	—
R	戻り値はなし。	—	—	—	—	—	—	—	—	—

※1　CMは指定不可。

●動作説明

実行条件が成り立つ(TRUE)とき、InOutに格納されている間接指定アドレスをOffsetで指定された値だけ減算したアドレスにします。ワードデータのアドレスが格納されているときは、アドレス減算値だけ減算したワードデータのアドレスを示し、またビットデータのアドレスが格納されているときは、アドレス減算値だけ減算したビットデータを示します。

「演算フラグについて(CR2009～CR2012)」(1-8ページ)
詳細については、『命令語リファレンスマニュアル』「間接指定命令」を参照してください。

●書式例

ADRSUB (bEnable,INT#10, udiAddress);

ポイント
- アドレス格納対象は、連続した2ワードを必ず占有します。
- ADRSUB関数を使用する前には、必ずADRSET関数で間接指定アドレスをアドレス格納対象にセットしてください。
- アドレスの生成、および操作は、必ずADRSET, ADRINC, ADRDEC, ADRADD, ADRSUB関数を使用しておこなってください。
- 次のいずれかの条件が成立した場合、関数は実行されません。
  - InOutに格納されているアドレスが不適切な場合。
  - 減算後のアドレスが不適切な場合。

ファイルレジスタ関数

FRSET　ファイルレジスタのバンク切り換え

ファイルレジスタバンク切換
FRSET ([EN], In);

引数/戻り値	説明	認識する型								定数
		.U	.S	.D	.L	.F	.DF	.B	.T	
EN	実行条件。省略可能。省略時、常時(毎スキャン)実行。	—	—	—	—	—	—	.B	—	—
In	切り換え後のカレントバンク番号を指定。(0～15)	.U	.U	.U	.U	—	—	—	—	—
R	戻り値はなし。	—	—	—	—	—	—	—	—	—

●動作説明

実行条件が成り立つ(TRUE)とき、ファイルレジスタのカレントバンク番号をInで指定した番号に切り換えます。

「演算フラグについて(CR2009～CR2012)」(1-8ページ)
詳細については、『命令語リファレンスマニュアル』「ファイルレジスタ命令」を参照してください。

●書式例

FRSET (bEnable, INT#2);

FRSTM　ファイルレジスタの一括保存

ファイルレジスター括保存
[R:=] FRSTM (Drive, Bank, File, Result);

引数/戻り値	説明	認識する型								定数
		.U	.S	.D	.L	.F	.DF	.B	.T	
Drive	保存するドライブ番号を指定。(0～1)	.U	.U	.U	.U	—	—	—	—	○
Bank	一括保存するファイルレジスタのバンク番号を指定。(0～15)	.U	.U	.U	.U	—	—	—	—	○
File	データを保存するときのファイル番号を指定。(0～999)	.U	.U	.U	.U	—	—	—	—	○
Result※1	保存結果を通知するビットデータの先頭を指定。	—	—	—	—	—	—	.B	—	—
R	戻り値。関数の起動に成功したらON、失敗のときはOFF。省略可能。省略時は、演算結果は内部レジスタに格納。	—	—	—	—	—	—	.B	—	—

※1　Resultにビットデータを指定した場合、連続する2ビットが通知ビットとして使用され、ワードデータを指定した場合は、連続する2ワードの各々最下位ビットが通知ビットデータとして使用。

●動作説明

DriveにBankのファイルレジスタのデータを、Fileで指定した番号のファイル名で、メモリカードにバイナリ形式で一括保存します。
Resultに書き込み完了通知と異常終了アラーム通知が格納されます。
FRSTM関数が正常に起動できた場合、戻り値をONし、失敗した場合はOFFします。

参考
- メモリカードまたはCPUメモリに保存されるファイル名は、「¥FR¥FR(ファイル番号3桁).FRU」となります。
  (例) **File**に10を指定した場合、「¥FR¥FR010.FRU」がメモリカードまたはCPUメモリに作成されます。
- **Result**の通知ビットは、以下の意味を持ちます。
  (例)MR2000を指定した場合

通知ビット		ON (1、TRUE)	OFF (0、FALSE)
MR2000	完了通知	完了	未完了
MR2001	異常通知	異常終了	正常終了

「演算フラグについて（CR2009～CR2012）」（1-8ページ）
詳細については、『命令語リファレンスマニュアル』「ファイルレジスタ命令」を参照してください。

●書式例

bFlag := **FRSTM** (INT#1, INT#10, INT#2, abFlag[0]);

FRLDM　ファイルレジスタの一括読み出し

ファイルレジスター括読出
[R:=] FRLDM (Drive, Bank, File, Result);

引数/戻り値	説明	認識する型								定数
		.U	.S	.D	.L	.F	.DF	.B	.T	
Drive	一括読み出しするドライブ番号を指定。（0～1）	.U	.U	.U	.U	–	–	–	–	○
Bank	一括読み出しするファイルレジスタのバンク番号を指定。（0～15）	.U	.U	.U	.U	–	–	–	–	○
File	データを読み出しするときのファイル番号を指定。（0～999）	.U	.U	.U	.U	–	–	–	–	○
Result※1	読出結果を通知するビットデータの先頭を指定。	–	–	–	–	–	–	.B	–	–
R	戻り値。省略可能。省略時は、演算結果は内部レジスタに格納。 関数の起動に成功したらON、失敗のときはOFF。	–	–	–	–	–	–	.B	–	–

※1　**Result**にビットデータを指定した場合、連続する2ビットが通知ビットとして使用、ワードデータを指定した場合は、連続する2ワードの各々最下位ビットが通知ビットデータとして使用。

●動作説明

Driveに保存されている、**File**で指定したデータ(¥FR¥FR(ファイル番号3桁).FRB)を読み出し、**Bank**のファイルレジスタに格納します。
Resultに書き込み完了通知と異常終了アラーム通知が格納されます。
FRLDM関数が正常に起動できた場合、戻り値をONし、失敗した場合はOFFします。

参考
- メモリカードまたはCPUメモリから読み出すファイル名は、「¥FR¥FR(ファイル番号3 桁).FRU」となります。
  (例)**File**に10を指定した場合、「¥FR¥FR010.FRU」がメモリカードまたはCPUメモリから読み出します。
- **Result**読出結果格納対象の通知ビットは、以下の意味を持ちます。
  (例)MR2000を指定した場合

通知ビット		ON(1、TRUE)	OFF(1、FALSE)
MR2000	完了通知	完了	未完了
MR2001	異常通知	正常終了	異常終了

●書式例

bFlag := **FRLDM** (INT#1, INT#10, INT#2, abFlag[0]);

ユニットプログラム関数

UPSTRT　ユニットプログラム開始

ユニットプログラム開始
UPSTRT (EN, Unit, Detail, Slot, Result);

引数/戻り値	説明	認識する型								定数
		.U	.S	.D	.L	.F	.DF	.B	.T	
EN	実行条件に使用するビットデータを指定。	—	—	—	—	—	—	.B	—	—
Unit※1	ユニット番号、または格納先を指定。(1～48)	.U	.U	.U	.U	—	—	—	—	○
Detail	実行を要求するユニットプログラムの詳細を格納した先頭を指定。	.U	.U	.U	.U	—	—	—	—	—
Slot	開始したスロット番号の格納先を指定。	.U	.U	.U	.U	—	—	—	—	—
Result	結果を格納する先頭を指定。	.U	.U	.U	.U	—	—	—	—	—
R	戻り値はなし。	—	—	—	—	—	—	—	—	—

※1　16#（16進数指定）は使用不可。

●動作説明

ENの立ち上がりで、**Unit**番目のユニットの**Detail**で指定したユニットプログラムを**Detail**+(2)で指定したパラメータに従い開始します。
Detail+(3)の0ビット目がOFFの場合、自動的に割り当てられる空いているスロットを使用します。
Detail+(3)の0ビット目がONの場合、**Detail**+(1)で指定したスロットを使用します。
関数の完了時、使用するスロットの番号を**Slot**に、実行結果を**Result**に格納します。

Detail ～：要求詳細

ワードオフセット			内容
Detail	プログラム番号		開始するユニットプログラム番号を指定します。
Detail＋(1)	スロット番号		Detail+(3)の0ビット目がONの場合、使用するスロット番号を指定します。
Detail＋(2)	パラメータ		開始するユニットプログラムのパラメータを指定します。 フローの場合：最初に実行するブロック番号を指定します。
Detail＋(3)	0ビット	スロット指定方法	ON　：Detail+(1)で指定したスロットを使用します。 OFF　：自動で割り当てられる空いているスロットを使用します。

Result ～：実行結果

ワードオフセット			内容
Result	0ビット	完了ビット	関数実行時にOFF、完了時にONします。関数が正常完了、異常完了のどちらでもONします。
	1ビット	実行失敗ビット	関数の実行に失敗した場合、完了ビットと同時にONします。関数実行時にOFFします。
Result＋(1)	完了コード		関数完了時に完了コードを格納します。 0　：正常に開始しました。 1　：指定したユニットプログラムが存在しません。 2　：空いているスロットがありません。 3　：動作許可がOFFしています。 4　：ユニットプログラムエラーが発生中です。 5　：指定したスロットが存在しません。 6　：指定したスロットが使用中です。 7　：すでに実行中のユニットプログラムです。 8　：指定したブロックが存在しません。 9　：すでに実行停止中のユニットプログラムです。 10　：指定したユニットプログラムがフローではありません。 100　：ユニットでエラーが発生しました。 1000　：指定したユニットが存在しません。 1001　：指定したユニットが不正です。

ユニットプログラム開始関数は、実行条件の立ち上がりで関数が実行されると、次のスキャンで入力条件がOFFであっても処理が継続されます。
結果格納先の完了ビット、実行失敗ビットは、完了時にONし、関数実行時にOFFします。
詳細については、『命令語リファレンスマニュアル』「ユニットプログラム命令」を参照してください。

●書式例

UPSTRT (bEnable,uiUnitNo,auiValue[0],uiValue, auiFlag[0]);

UPSTOP　ユニットプログラム強制終了

ユニットプログラム強制終了
UPSTOP (EN, Unit, Slot, Result);

引数/戻り値	説明	認識する型								定数
		.U	.S	.D	.L	.F	.DF	.B	.T	
EN	実行条件に使用するビットデータを指定。	—	—	—	—	—	—	.B	—	—
Unit※1	ユニット番号、または格納先を指定。(1～48)	.U	.U	.U	.U	—	—	—	—	○
Slot	強制終了するスロット番号の格納先を指定。	.U	.U	.U	.U	—	—	—	—	○
Result	結果を格納する先頭を指定。	—	—	—	—	—	—	.B	—	—
R	戻り値はなし。	—	—	—	—	—	—	—	—	—

※1　16#（16進数指定）は使用不可。

●動作説明

ENの立ち上がりで、**Unit**番目のユニットの**Slot**で指定したスロットで実行しているユニットプログラムを強制終了し、スロットを空き状態にします。
関数完了時、実行結果を**Result**に格納します。

参考　「UPSTRT関数」(6-15ページ)

詳細については、『命令語リファレンスマニュアル』「ユニットプログラム命令」を参照してください。

Result ～：実行結果格納先

ワードオフセット			内容
Result	0ビット	完了ビット	関数実行時にOFF、完了時にONします。関数が正常完了、異常完了のどちらでもONします。
	1ビット	実行失敗ビット	関数の実行に失敗した場合、完了ビットと同時にONします。関数実行時にOFFします。

ワードオフセット		内容
Result＋(1)	完了コード	関数完了時に完了コードを格納します。 0　　:正常に開始しました。 1　　:指定したユニットプログラムが存在しません。 2　　:空いているスロットがありません。 3　　:動作許可がOFFしています。 4　　:ユニットプログラムエラーが発生中です。 5　　:指定したスロットが存在しません。 6　　:指定したスロットが使用中です。 7　　:すでに実行中のユニットプログラムです。 8　　:指定したブロックが存在しません。 9　　:すでに実行停止中のユニットプログラムです。 10　　:指定したユニットプログラムがフローではありません。 100　:ユニットでエラーが発生しました。 1000 :指定したユニットが存在しません。 1001 :指定したユニットが不正です。

●書式例

UPSTOP (bEnable, uiUnitNo, INT#1, auResult);

UFSUS　フロー一時停止

フロー一時停止
UFSUS (EN, Unit, Slot, Detail, Result);

引数/戻り値	説明	認識する型								定数
		.U	.S	.D	.L	.F	.DF	.B	.T	
EN	実行条件に使用するビットデータを指定。	–	–	–	–	–	–	.B	–	–
Unit※1	ユニット番号、または格納先を指定。(1～48)	.U	.U	.U	.U	–	–	–	–	○
Slot	一時停止するスロット番号の格納先を指定。	.U	.U	.U	.U	–	–	–	–	○
Detail	一時停止方法を格納した対象を指定。	.U	.U	.U	.U	–	–	–	–	○
Result	結果を格納する先頭を指定。	.U	.U	.U	.U	–	–	–	–	–
R	戻り値はなし。	–	–	–	–	–	–	–	–	–

※1　16#（16進数指定）は使用不可。

●動作説明

ENの立ち上がりで、**Unit**番目のユニットの**Slot**で指定したスロットで実行しているフローを**Detail**で指定した方法に従い一時停止します。
関数完了時、実行結果を**Result**に格納します。

Result ～：実行結果格納先

ワードオフセット			内容
Detail	0ビット	完了ビット	関数実行時にOFF、完了時にONします。関数が正常完了、異常完了のどちらでもONします。
	1ビット	実行失敗ビット	関数の実行に失敗した場合、完了ビットと同時にONします。関数実行時にOFFします。

ワードオフセット		内容
Result＋(1)	完了コード	関数完了時に完了コードを格納します。 0　　:正常に開始しました。 1　　:指定したユニットプログラムが存在しません。 2　　:空いているスロットがありません。 3　　:動作許可がOFFしています。 4　　:ユニットプログラムエラーが発生中です。 5　　:指定したスロットが存在しません。 6　　:指定したスロットが使用中です。 7　　:すでに実行中のユニットプログラムです。 8　　:指定したブロックが存在しません。 9　　:すでに実行停止中のユニットプログラムです。 10　　:指定したユニットプログラムがフローではありません。 100　:ユニットでエラーが発生しました。 1000 :指定したユニットが存在しません。 1001 :指定したユニットが不正です。

Detail：一時停止方法

ワードオフセット			内容
Detail	0ビット	一時停止方法	ON　:実行中のブロックがあれば強制終了して、フローを一時停止します。 OFF :実行中のブロックが完了したタイミングでフローを一時停止します。

詳細については、『命令語リファレンスマニュアル』「ユニットプログラム命令」を参照してください。

●書式例

UFSUS (bEnable, uiUnitNo, uiValue1,uiValue2, auiValue[0]);

UFRSM　フロー再開

フロー再開
UFRSM (EN, Unit, Slot, Result);

引数/戻り値	説明	認識する型								定数
		.U	.S	.D	.L	.F	.DF	.B	.T	
EN	実行条件に使用するビットデータを指定。	—	—	—	—	—	—	.B	—	—
Unit※1	ユニット番号、または格納先を指定。(1～48)	.U	.U	.U	.U	—	—	—	—	○
Slot	再開するスロット番号の格納先を指定。	.U	.U	.U	.U	—	—	—	—	○
Result	結果を格納する先頭を指定。	.U	.U	.U	.U	—	—	—	—	—
R	戻り値はなし。	—	—	—	—	—	—	—	—	—

※1　16#（16進数指定）は使用不可。

●動作説明

ENの立ち上がりで、**Unit**番目のユニットの**Slot**で指定したスロットの一時停止しているフローを再開します。
命令完了時、実行結果を**Result**に格納します。

参考　「UFSUS関数」(6-17ページ)

詳細については、『命令語リファレンスマニュアル』「ユニットプログラム命令」を参照してください。

Result ～：実行結果格納先

ワードオフセット			内容
Result	0ビット	完了ビット	関数実行時にOFF、完了時にONします。関数が正常完了、異常完了のどちらでもONします。
	1ビット	実行失敗ビット	関数の実行に失敗した場合、完了ビットと同時にONします。関数実行時にOFFします。
Result＋(1)	完了コード		関数完了時に完了コードを格納します。 0　:正常に開始しました。 1　:指定したユニットプログラムが存在しません。 2　:空いているスロットがありません。 3　:動作許可がOFFしています。 4　:ユニットプログラムエラーが発生中です。 5　:指定したスロットが存在しません。 6　:指定したスロットが使用中です。 7　:すでに実行中のユニットプログラムです。 8　:指定したブロックが存在しません。 9　:すでに実行停止中のユニットプログラムです。 10　:指定したユニットプログラムがフローではありません。 100　:ユニットでエラーが発生しました。 1000　:指定したユニットが存在しません。 1001　:指定したユニットが不正です。

●書式例

UFRSM (bEnable, uiUnitNo, uiValue, auiValue[0]);

UMALLOC バッファメモリ確保

バッファメモリ確保
UMALLOC (EN, Unit, Count, Result, Start);

引数/戻り値	説明	認識する型								定数
		.U	.S	.D	.L	.F	.DF	.B	.T	
EN	実行条件に使用するビットデータを指定。	—	—	—	—	—	—	.B	—	—
Unit※1	ユニット番号、または格納先を指定。(1～48)	.U	.U	.U	.U	—	—	—	—	○
Count	確保するワード数の格納先を指定。	.U	.U	.U	.U	—	—	—	—	○
Result	結果を格納する先頭を指定。	.U	.U	.U	.U	—	—	—	—	○
Start	確保した領域の先頭アドレスの格納先を指定。	.U	.U	.U	.U	—	—	—	—	—
R	戻り値はなし。	—	—	—	—	—	—	—	—	—

※1　16#（16進数指定）は使用不可。

●動作説明

ENの立ち上がりで、**Unit**番目のユニットのバッファメモリから**Count**で指定したワード数の領域を確保します。確保した領域の先頭アドレスを**Start**に、実行結果を**Result**に格納します。
確保に失敗した場合、**Start**には0が格納されます。

参考
- 領域は自動で割り当てられます。
- 確保した領域は、自由に値を読み書きできます。
- ユニットの動作許可リレーOFFで、確保した領域は解放されます

Result：実行結果

ワードオフセット			内容
Result	0ビット	完了ビット	関数実行時にOFF、完了時にONします。関数が正常完了、異常完了のどちらでもONします。
	1ビット	実行失敗ビット	領域の確保に失敗した場合、完了ビットと同時にONします。関数実行時にOFFします。

詳細については、『命令語リファレンスマニュアル』「ユニットプログラム命令」を参照してください。

●書式例

UMALLOC (bEnable, uiUnitNo, uiSize, uiResult, uiAddress);

バッファメモリ関数

UREAD　バッファメモリ読み出し

バッファメモリ読み出し
UREAD ([EN], Unit, Src, Dst, Count);

引数/戻り値	説明	認識する型								定数
		.U	.S	.D	.L	.F	.DF	.B	.T	
EN	実行条件。省略可能。省略時、常時(毎スキャン)実行。	–	–	–	–	–	–	.B	–	–
Unit※1	バッファメモリデータを読み出す拡張特殊ユニット･CPUユニットのユニット番号を指定。(0～48)	.U	.U	.U	.U	–	–	–	–	○
Src	データを読み出すバッファメモリの先頭アドレスを指定。(0～59999)	.U	.U	.U	.U	–	–	–	–	○
Dst※2	読み出したデータを格納する先頭を指定。	.U	.S	.D	.L	.F	–	–	.T	–
Count	バッファメモリから読み出すデータ数を指定。(1～60000)	.U	.U	.U	.U	–	–	–	–	○
R	戻り値はなし。	–	–	–	–	–	–	–	–	–

※1　16# (16進数指定)では指定不可。
※2　Zは指定不可。

●動作説明

実行条件が成り立つ(TRUE)とき、CPUユニットに接続されたUnitの拡張特殊ユニット･CPUユニットのバッファメモリを、Srcで指定されたアドレスから、Countで指定された個数分のデータ(16ビット型ワードデータ)を読み出し、Dstに格納します。

「演算フラグについて(CR2009～CR2012)」(1-8ページ)
詳細については、『命令語リファレンスマニュアル』「バッファメモリ命令」を参照してください。

●書式例

UREAD (bEnable, uiUnitNo, uiAddress, auiResult, uiCount);

UWRIT　バッファメモリ書き込み

バッファメモリ書き込み
UWRIT ([EN], Unit, Dst, Src, Count);

引数/戻り値	説明	認識する型								定数
		.U	.S	.D	.L	.F	.DF	.B	.T	
EN	実行条件。省略可能。省略時、常時(毎スキャン)実行。	–	–	–	–	–	–	.B	–	–
Unit※1	バッファメモリに書き込む拡張特殊ユニット･CPUユニットのユニット番号を指定。(0～48)	.U	.U	.U	.U	–	–	–	–	○
Dst※2	データを書き込むバッファメモリの先頭アドレスを指定。(0～59999)	.U	.U	.U	.U	–	–	–	–	○
Src	書き込むデータの先頭を指定。文字列演算の結果を指定することも可能。文字列演算以外の演算(DM0+DM2など)は記述不可。	.U	.S	.D	.L	.F	–	–	.T	○
Count	バッファメモリに書き込むデータ数を指定。(1～60000)	.U	.U	.U	.U	–	–	–	–	○
R	戻り値はなし。	–	–	–	–	–	–	–	–	–

※1　16# (16進数指定)では指定不可。
※2　Zは指定不可。

●動作説明

実行条件が成り立つ(TRUE)とき、CPUユニットに接続された**Unit**の拡張特殊ユニット･CPUユニットのバッファメモリの、**Dst**で指定されたアドレスに、**Src**に格納されている**Count**で指定された個数分のデータを書き込みます。

参考
- **Src**に定数を指定した場合、UWRIT関数はUFILL関数と同じ動作をおこないます。
  「UFILL関数」(6-22ページ)

「演算フラグについて(CR2009～CR2012)」(1-8ページ)
詳細については、『命令語リファレンスマニュアル』「バッファメモリ命令」を参照してください。

●書式例

UWRIT (bEnable, uiUnitNo, uiAddress, auiData[0], uiCount);

UFILL　バッファメモリのデータ列に同じ値を挿入

バッファメモリフィル
UFILL ([EN], Unit, Dst, In, Count);

引数/戻り値	説明	認識する型								定数
		.U	.S	.D	.L	.F	.DF	.B	.T	
EN	実行条件。省略可能。省略時、常時(毎スキャン)実行。	–	–	–	–	–	–	.B	–	–
Unit※1	バッファメモリに書き込む拡張特殊ユニット･CPUユニットのユニット番号を指定。(0～48)	.U	.U	.U	.U	–	–	–	–	○
Dst	データを書き込むバッファメモリの先頭アドレスを指定。(0～59999)	.U	.U	.U	.U	–	–	–	–	○
In	書き込むデータまたは、データの格納先を指定。	.U	.S	.D	.L	.F	–	–	.T	○
Count	バッファメモリに書き込むデータ数を指定。(1～60000)	.U	.U	.U	.U	–	–	–	–	○
R	戻り値はなし。	–	–	–	–	–	–	–	–	–

※1　16# (16進数指定)では指定不可。
※2　Zは指定不可。

●動作説明

実行条件が成り立つ(TRUE)とき、CPUユニットに接続された**Unit**の拡張特殊ユニット･CPUユニットのバッファメモリの、**Dst**で指定されたアドレスから**Count**で指定された個数分の領域に、**In**に格納されているデータを一括で書き込みます。

「演算フラグについて(CR2009～CR2012)」(1-8ページ)
詳細については、『命令語リファレンスマニュアル』「バッファメモリ命令」を参照してください。

●書式例

UFILL (bEnable, uiUnitNo, uiAddress, uiValue, uiCount);

ユニット種別判定関数

IS_□□□　ユニット種別判定

ユニット種別判定
IS_□□□ ([EN], Unit, Result);

引数/戻り値	説明	認識する型								定数
		.U	.S	.D	.L	.F	.DF	.B	.T	
EN	実行条件。省略可能。省略時、常時(毎スキャン)実行。	—	—	—	—	—	—	.B	—	—
Unit※1	ユニット番号、または格納先を指定。(1～48)	.U	.U	.U	.U	—	—	—	—	○
Result※2	結果通知の格納先を指定。	.B	.B	.B	.B	—	—	—	—	—
R	戻り値はなし。	—	—	—	—	—	—	—	—	—

※1　16# (16進数指定)では指定不可。
※2　ワードデータを指定した場合、最下位ビットを通知ビットデータとして使用。最下位ビット以外は無変換。

●動作説明

実行条件の立ち上がりで、**Unit**番目のユニットのユニット種別を確認し、命令語に対応したユニット種別と一致したら**Result**をONします。不一致または未接続の場合、**Result**をOFFします。

命令語とユニット種別
命令語の「IS_」を「KV-」に置き換えると、対応するユニット種別になります。
例)
(命令語)　　(ユニット種別)
IS_XH16ML　→　KV-XH16ML

●書式例

IS_XH16ML (bEnable, INT#1, uiFlag);

配列要素数取得関数

ACNT　配列要素数取得

配列要素数取得
[R:=] ACNT (In, Dim);

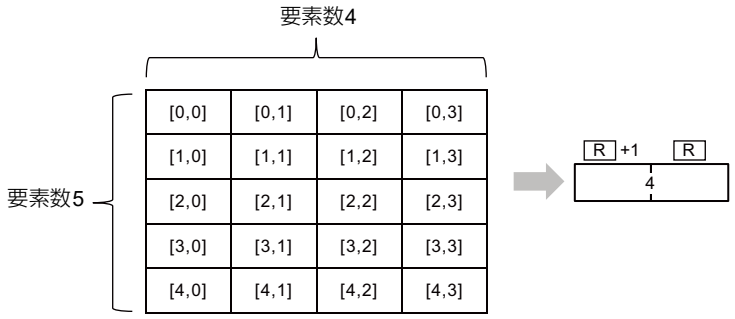
引数/戻り値	説明	認識する型								定数
		.U	.S	.D	.L	.F	.DF	.B	.T	
In	要素数を取得する配列型変数を指定。※1	—	—	—	—	—	—	—	—	—
Dim	次元数を指定。	.U	—	—	—	—	—	—	—	○
R	ACNT関数は、.Dの型を使用※2	.U	.S	.D	.L	.F	.DF※3	—	—	—

※1　配列型以外の変数は指定不可。また、可変長配列は指定不可。
※2　戻り値に.D以外の型を指定した場合、変換時に自動的に型変換。
※3　デバイスZを指定した場合、「.DF」サフィックスは指定不可。

●動作説明

InのDim次元目の要素数を数え、結果を32ビットバイナリデータとしてRに格納します。

例　In=Array_A(5×4の2次元配列)、Dim=2の場合



「演算フラグについて（CR2009～CR2012）」（1-8ページ）
詳細については、『命令語リファレンスマニュアル』「配列要素数取得」を参照してください。

ポイント　**ACNT命令で取得するのはワードサイズではなく要素数です。例えば、UDINT(2ワード符号なし整数)が4個の次元を指定した場合、Rには4が格納されます。**

●書式例

udiSizeOfDimension := **ACNT** (auiData, INT#2);

ワードサイズ取得関数

WSIZE　ワードサイズ取得

ワードサイズ取得
[R:=] WSIZE (In);

引数/戻り値	説明	認識する型								定数
		.U	.S	.D	.L	.F	.DF	.B	.T	
In	ワードサイズを取得する変数を指定。※1	–	–	–	–	–	–	–	–	–
R※2	戻り値。省略可能。省略時は、変換結果は内部レジスタに格納。	.U	.S	.D	.L	.F	.DF※3	–	–	–

※1　以下の変数は指定不可。
- ·TIMER型変数
- ·COUNTER型変数
- ·ファンクションブロック型変数
- ·可変長配列
- ·可変長文字列

その他の配列は指定可能。

※2　戻り値に.D以外の型を指定した場合、変換時に自動的に型変換。

※3　デバイスZを指定した場合、「.DF」サフィックスは指定不可。

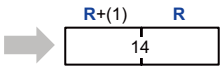
●動作説明

Inのワードサイズを32ビットバイナリデータとしてRに格納します。

例

In=以下の構造体型変数を指定した場合

メンバ名	データ型
member1	UINT
member2	STRING[20]
member3	REAL



→STRING[20]のワード数は(20+1)/2を切り上げて11ワードなので、1 + 11 + 2 = 14となります。

「演算フラグについて（CR2009～CR2012）」（1-8ページ）
詳細については、『命令語リファレンスマニュアル』「ワードサイズ取得」を参照してください。

ポイント
- **WSIZE命令で取得するのはワードサイズです。例えば、UDINT型のデータ(2ワード符号なし整数)を4つ持つ配列型変数や構造体型変数を指定した場合、2ワード×4=8ワードなので、Rには8が格納されます。要素数やメンバの個数が格納されるわけではありません。**
- **Rにはオフセットも含めたワードサイズが格納されます。オフセットの詳細は使用しているCPUユニットのユーザーズマニュアルを参照してください。**
- **Rにはワードサイズに切り上げた結果が格納されます。例えば、BOOL[33]の場合は3が格納されます。**

●書式例

udiSizeOfTag := **WSIZE** (tagAxis_A);

7

演算関数

データ転送関数 7-2
算術関数 7-6
論理演算関数 7-9
データシフト関数 7-14
データ制御関数 7-22
データ変換関数 7-29
浮動小数点関数 7-44
文字処理関数 7-59

データ転送関数

BMOV　ワードブロックの一括データ転送

ブロックムーブ
BMOV ([EN], Src, Dst, Count);

引数/戻り値	説明	認識する型								定数
		.U	.S	.D	.L	.F	.DF	.B	.T	
EN	実行条件。省略可能。省略時、常時(毎スキャン)実行。	–	–	–	–	–	–	.B	–	–
Src※1	転送元データブロックの先頭を指定。	.U	.U	.D	.D	–	–	–	–	–
Dst※1	転送先データブロックの先頭を指定。	型はSrcと同じ。					–	–	–	–
Count	転送データ数、または転送するデータ数の格納先を指定。	.U	.U	.U	.U	–	–	–	–	○
R	戻り値はなし。	–	–	–	–	–	–	–	–	–

※1　デバイスT,Cを指定すると、転送先/転送元は現在値。
※2　ビットデータを指定した場合、連続した16/32ビットで演算。R002などチャンネルの先頭以外を指定した場合、次のチャンネルにまたがって16/32ビットで演算。

●動作説明

実行条件が成り立つ(TRUE)とき、Srcを先頭とするCount x(データ型のワード数)分のワードブロックの内容をDstを先頭とするCount x(データ型別のワード数)分のワードブロックへブロック転送を実行。

「演算フラグについて(CR2009～CR2012)」(1-8ページ)
詳細については、『命令語リファレンスマニュアル』「データ転送命令」を参照してください。

●書式例

BMOV (bEnable, auiSendSrc[0], auiSendDst[0], INT#10);

ポイント　**Count=0の場合、関数は実行されません。**

参考
- 転送元と転送先のブロック (ビット) がチャンネルを越えて重複していても、正しく転送されます。

FMOV　データ列に同じ値を代入

フィルムーブ
FMOV ([EN], Src, Dst, Count);

引数/戻り値	説明	認識する型								定数
		.U	.S	.D	.L	.F	.DF	.B	.T	
EN	実行条件。省略可能。省略時、常時(毎スキャン)実行。	–	–	–	–	–	–	.B	–	–
Src※1	転送元データ、または転送元データの格納先を指定。	.U	.U	.D	.D	.D	.DF※2	–	–	○
Dst※1	転送先データブロックの先頭を指定。	型はSrcと同じ。					–	–	–	–
Count	転送データ数、または転送するデータ数の格納先を指定。	.U	.U	.U	.U	–	–	–	–	○
R	戻り値はなし。	–	–	–	–	–	–	–	–	–

※1　デバイスT,Cを指定すると、転送先/転送元は現在値。
※2　デバイスZを指定した場合、「.DF」サフィックスは指定不可。

●動作説明

実行条件が成立時(TRUE)、**Src**に格納されたデータを**Dst**を先頭とする**Count** ×(データ型別のワード数)分のワードブロックへブロック転送を実行。

「演算フラグについて（CR2009～CR2012）」（1-8ページ）
詳細については、『命令語リファレンスマニュアル』「データ転送命令」を参照してください。

●書式例

FMOV (bEnable, uiSendSrc, auiSendDst[0], INT#10);

ポイント　**Count=0の場合、関数は実行されません。**

参考
- 転送元と転送先のブロック (ビット) がチャンネルを越えて重複していても、正しく転送されます。

PMOV　連続ビット情報を指定位置に転送

バーシャルムーブ
PMOV ([EN], Src, SrcPos, Dst, DstPos, Count);

引数/戻り値	説明	認識する型								定数
		.U	.S	.D	.L	.F	.DF	.B	.T	
EN	実行条件。省略可能。省略時、常時（毎スキャン）実行。	–	–	–	–	–	–	.B	–	–
Src※1※2	転送元データの格納先を指定。	.U	.U	.D	.D	.D	–	–	–	○
SrcPos※3※4	転送元の先頭ビット位置の格納先、または先頭ビット位置を指定。	.U	.U	.U	.U	–	–	–	–	○
Dst※1※2	転送先を指定。	型はSrcと同じ。					–	–	–	–
DstPos※3※4	転送先の先頭ビット位置の格納先、または先頭ビット位置を指定。	.U	.U	.U	.U	–	–	–	–	○
Count	転送するデータ数の格納先、または転送データ数を指定。	.U	.U	.U	.U	–	–	–	–	○
R	戻り値はなし。	–	–	–	–	–	–	–	–	–

※1　ビットデータを指定した場合、連続する16ビット(.Uのとき)/32ビット(.Dのとき)を占有。
※2　**Src**と**Dst**で同じ対象を指定可能。
※3　例えば、サフィックスが .Uで、**SrcPos**(**DstPos**)にDM1000を指定、その値が20(2進数：0000 0000 0001 0100)の場合、下位4ビットの値は4(0100)なので**SrcPos**(**DstPos**)の値として4を使用。
※4　**Src**転送元が16ビット(.U/.S)の場合は0～15を、32ビット(.D/.L)の場合は0～31を指定。

●動作説明

実行条件が成り立つ(TRUE)とき、**Src**に格納された16ビット/32ビットデータの**SrcPos**で指定されたビット位置以降のビットを、**Dst**の**DstPos**で指定されたビット位置以降のビットに、**Count**で指定されたビット数分ブロック(ビット)転送します。

「演算フラグについて（CR2009～CR2012）」（1-8ページ）
詳細については、『命令語リファレンスマニュアル』「データ転送命令」を参照してください。

参考　転送されていない部分(転送対象外)のビットは、元の値が保持されます。

●書式例

PMOV (bEnable, uiSendData, uiBitPos, uiRcvData, INT#3, INT#10);

ポイント　次のいずれかの条件が成立した場合、関数は実行されません。
- **Src**のサフィックスが.Uで、**Count**=0または16<**SrcPos**＋**Count**または16<**DstPos**＋**Count**の場合。
- **Src**のサフィックスが.Dで、**Count**=0または32<**SrcPos**＋**Count**または32<**DstPos**＋**Count**の場合。

BYBMOV　連続バイト列情報を指定位置に転送(H→L)

バイBムーブ
BYBMOV ([EN], Src, SrcOffset, Dst, DstOffset, Count);

引数/戻り値	説明	認識する型								定数
		.U	.S	.D	.L	.F	.DF	.B	.T	
EN	実行条件。省略可能。省略時、常時(毎スキャン)実行。	—	—	—	—	—	—	.B	—	—
Src※1	転送元バイト列データを格納している先頭を指定。	.U	.S	.D	.L	.F	.DF※2	.B	.T	—
SrcOffset	転送元バイト列先頭からの位置の格納先、または位置を指定。(0～65535)。	.U	.U	.U	.U	—	—	—	—	○
Dst※1	転送先バイト列の先頭を指定。	.U	.S	.D	.L	.F	.DF※2	.B	.T	—
DstOffset	転送先バイト列先頭からの位置の格納先、または位置を指定。(0～65535)	.U	.U	.U	.U	—	—	—	—	○
Count	転送するデータ数の格納先、または転送データ数を指定。(1～65535)	.U	.U	.U	.U	—	—	—	—	○
R	戻り値はなし。	—	—	—	—	—	—	—	—	—

※1　・ビットデータではチャンネルの先頭を指定。
　　・**Src**と**SrcOffset**で同じ対象を指定可能。
※2　デバイスZを指定した場合、「.DF」サフィックスは指定不可。

●動作説明

実行条件が成り立つ(TRUE)とき、**Src**の(上位バイトから数えて)**SrcOffset**バイト目から**Count**分を、**Dst**の(上位バイトから数えて)**DstOffset**バイト目から(**DstOffset**+**Count**－(1))バイト目に転送します。転送がおこなわれない部分のバイトは元の値が保存されます。

「演算フラグについて(CR2009～CR2012)」(1-8ページ)
詳細については、『命令語リファレンスマニュアル』「データ転送命令」を参照してください。

●書式例

BYBMOV (bEnable, auiData[0], uiOffset1,auiDstData[0],uiOffset2,uiCount);

ポイント　**Count**=0の場合、関数は実行されません。

BYLMOV　連続バイト列情報を指定位置に転送(L→H)

バイLムーブ
BYLMOV ([EN], Src, SrcOffset, Dst, DstOffset, Count);

引数/戻り値	説明	認識する型								定数
		.U	.S	.D	.L	.F	.DF	.B	.T	
EN	実行条件。省略可能。省略時、常時(毎スキャン)実行。	—	—	—	—	—	—	.B	—	—
Src※1	転送元バイト列データを格納している先頭を指定。	.U	.S	.D	.L	.F	.DF※2	.B	.T	—
SrcOffset	転送元バイト列先頭からの位置の格納先、または位置を指定。(0～65535)。	.U	.U	.U	.U	—	—	—	—	○
Dst※1	転送先バイト列の先頭を指定。	.U	.S	.D	.L	.F	.DF※2	.B	.T	—
DstOffset	転送先バイト列先頭からの位置の格納先、または位置を指定。(0～65535)	.U	.U	.U	.U	—	—	—	—	○
Count	転送するデータ数の格納先、または転送データ数を指定。(1～65535)	.U	.U	.U	.U	—	—	—	—	○
R	戻り値はなし。	—	—	—	—	—	—	—	—	—

※1　・ビットデータではチャンネルの先頭を指定。
　　・**Src**と**SrcOffset**で同じ対象を指定可能。
※2　デバイスZを指定した場合、「.DF」サフィックスは指定不可。

●動作説明

実行条件が成り立つ(TRUE)とき、**Src**の(下位バイトから数えて)**SrcOffset**バイト目から**Count**分を、**Dst**の(下位バイトから数えて)**DstOffset**バイト目から(**DstOffset**+**Count**－1)バイト目に転送します。転送がおこなわれない部分のバイトは元の値が保存されます。

「演算フラグについて(CR2009～CR2012)」(1-8ページ)
詳細については、『命令語リファレンスマニュアル』「データ転送命令」を参照してください。

●書式例

BYLMOV (auiSrc[0], uiSrcOffset, auiDst[0], uiDstOffset, uiCount);

ポイント　**Count=0の場合、関数は実行されません。**

算術関数

INC　1を加算

インクリメントメモリ
INC ([EN], InOut);

引数/戻り値	説明	認識する型								定数
		.U	.S	.D	.L	.F	.DF	.B	.T	
EN	実行条件。省略可能。省略時、常時(毎スキャン)実行。	—	—	—	—	—	—	.B	—	—
InOut※1	インクリメント(1を加算)する対象を指定。	.U	.S	.D	.L	—	—	—	—	—
R	戻り値はなし。	—	—	—	—	—	—	—	—	—

※1　ビットデータを指定した場合、連続した16/32ビットで演算。
R002などチャンネルの先頭以外を指定した場合、次のチャンネルにまたがって16/32ビットで演算。

●動作説明

InOutのデータに1を加算し、結果をInOutに格納します。

- **InOut:加算対象が16ビット符号無しデータ(.U)の場合**
  演算結果が65535より大きくなるとオーバーフローが発生します。
  (演算前が65535の場合、演算後は0となり、オーバーフローになります)
- **InOut:加算対象が16ビット符号付きデータ(.S)の場合**
  演算結果が＋32767より大きくなるとオーバーフローが発生します。
  (演算前が＋32767の場合、演算後は–32768となり、オーバーフローになります)
- **InOut:加算対象が32ビット符号無しデータ(.D)の場合**
  演算結果が4294967295より大きくなるとオーバーフローが発生します。
  (演算前が4294967295の場合、演算後は0となり、オーバーフローになります)
- **InOut:加算対象が32ビット符号付きデータ(.L)の場合**
  演算結果が＋2147483647より大きくなるとオーバーフローが発生します。
  (演算前が＋2147483647の場合、演算後は–2147483648となり、オーバーフローになります)

ポイント　**オーバーフローが発生しても正しく演算されます。**

「演算フラグについて(CR2009～CR2012)」(1-8ページ)
詳細については、『命令語リファレンスマニュアル』「算術/比較命令」を参照してください。

●書式例

INC (bEnable, uiData);

DEC　1を減算

デクリメントメモリ
DEC ([EN], InOut);

引数/戻り値	説明	認識する型								定数
		.U	.S	.D	.L	.F	.DF	.B	.T	
EN	実行条件。省略可能。省略時、常時(毎スキャン)実行。	—	—	—	—	—	—	.B	—	—
InOut※1	デクリメント(1を減算)する対象を指定。	.U	.S	.D	.L	—	—	—	—	—
R	戻り値はなし。	—	—	—	—	—	—	—	—	—

※1　ビットデータを指定した場合、連続した16/32ビットで演算。
R002などチャンネルの先頭以外を指定した場合、次のチャンネルにまたがって16/32ビットで演算。

●動作説明

InOutのデータから1を減算し、結果をInOutに格納します。

- **InOut:減算対象が16ビット符号無しデータ(.U)の場合**
  演算結果が0より小さくなるとアンダーフローが発生します。
  (演算前が0の場合、演算後は65535となり、アンダーフローになります)
- **InOut:減算対象が16ビット符号付きデータ(.S)の場合**
  演算結果が–32768より小さくなるとアンダーフローが発生します。
  (演算前が–32768の場合、演算後は＋32767となり、アンダーフローになります)
- **InOut:減算対象が32ビット符号無しデータ(.D)の場合**
  演算結果が0より小さくなるとアンダーフローが発生します。
  (演算前が0の場合、演算後は4294967295となり、アンダーフローになります)
- **InOut:減算対象が32ビット符号付きデータ(.L)の場合**
  演算結果が–2147483648より小さくなるとアンダーフローが発生します。
  (演算前が–2147483648の場合、演算後は＋2147483647となり、アンダーフローになります)

ポイント　**アンダーフローが発生しても正しく演算されます。**

「演算フラグについて(CR2009～CR2012)」(1-8ページ)
詳細については、『命令語リファレンスマニュアル』「算術/比較命令」を参照してください。

●書式例

DEC (bEnable, uiData);

ROOT　平方根

ルート
[R:=] ROOT (In);

引数/戻り値	説明	認識する型								定数
		.U	.S	.D	.L	.F	.DF	.B	.T	
In	演算対象、または演算対象データを指定。	.U	.U	.D	.L	.F	.DF※2	—	—	○
R※1	演算結果が格納される戻り値。省略可能。省略時、内部レジスタに格納。	.U	.S	.D	.L	.F	.DF※2	—	—	—

※1　Rに演算対象と異なる型を指定した場合、変換時に自動的に型変換を実行。
※2　デバイスZを指定した場合、「.DF」サフィックスは指定不可。

●動作説明

Inに格納されたデータの平方根を求め、結果を**R**に格納します。

●書式例

udiResult := **ROOT** (diXCoord ** 2+ diYCoord ** 2);

diXCoordの2乗とdiYCoordの2乗の和の平方根を求め、結果をudiResultに格納します。

参 考　演算対象データに整数値を指定した場合:
- 変換対象データの16ビット/32ビットバイナリデータは、それぞれデータ拡張された結果の32ビット/64ビットバイナリデータとして平方根が求められ、結果はそれぞれ再度16ビット/32ビットバイナリデータとして格納されます。
- 演算結果の小数点以下の値は切り捨てられます。

「演算フラグについて(CR2009～CR2012)」(1-8ページ)
詳細については、『命令語リファレンスマニュアル』「データ転送命令」を参照してください。

SQRT　平方根

平方根
[R:=] SQRT (In);

引数/戻り値	説明	認識する型								定数
		.U	.S	.D	.L	.F	.DF	.B	.T	
In	演算対象、または演算対象データを指定。	.F	.F	.F	.F	.F	.DF※2	—	—	○
R※1	演算結果が格納される戻り値。省略可能。省略時、内部レジスタに格納。	.U	.S	.D	.L	.F	.DF※2	—	—	—

※1　**R**に演算対象と異なる型を指定した場合、変換時に自動的に型変換を実行。
※2　デバイスZを指定した場合、「.DF」サフィックスは指定不可。

●動作説明

Inに格納されたデータの平方根を求め、結果を**R**に格納します。

●書式例

rResult := **SQRT** (rXCoord ** 2+ rYCoord ** 2);

rXCoordの2乗とrYCoordの2乗の和の平方根を求め、結果をrResultに格納します。

「演算フラグについて(CR2009～CR2012)」(1-8ページ)
詳細については、『命令語リファレンスマニュアル』「算術/比較命令」を参照してください。

論理演算関数

ANDA　論理積演算

[R:=] ANDA (In1, In2);
アンドA

引数/戻り値	説明	認識する型								定数
		.U	.S	.D	.L	.F	.DF	.B	.T	
In1※1	演算対象、または演算対象データを指定。	.U	.U	.D	.D	–	–	–	–	○
In2※1	論理積データの格納先、または論理積データを指定。	型はIn1と同じ。				–	–	–	–	○
R※2	戻り値。省略可能。省略時、内部レジスタに格納。	.U	.S	.D	.L	.F	.DF※3	–	–	–

※1　ビットデータを指定した場合、連続した16/32ビットで演算。R002などチャンネルの先頭以外を指定した場合、次のチャンネルにまたがって16/32ビットで演算。
※2　Rに演算対象と異なる型を指定した場合、変換時に自動的に型変換を実行。
※3　デバイスZを指定した場合、「.DF」サフィックスは指定不可。

●動作説明

In1に格納されたデータと、In2に格納されたバイナリデータで論理積演算をおこないます。

●書式例

- **In1：演算対象が16ビット(.U/.S)の場合**

uiValue := **ANDA** (uiValue1, uiValue2);

- **In1：演算対象が32ビット(.D/.L)の場合**

udiValue := **ANDA** (udiValue1, udiValue2);

「演算フラグについて（CR2009～CR2012）」（1-8ページ）
詳細については、『命令語リファレンスマニュアル』「論理演算命令」を参照してください。

●論理積演算真理値表

演算対象	0	0	1	1
演算データ	0	1	0	1
戻り値	0	0	0	1

ORA　論理和演算

[R:=] ORA (In1, In2);
オアA

引数/戻り値	説明	認識する型								定数
		.U	.S	.D	.L	.F	.DF	.B	.T	
In1※1	演算対象、または演算対象データを指定。	.U	.U	.D	.D	–	–	–	–	○
In2※1	論理和データの格納先、または論理和データを指定。	型はIn1と同じ。				–	–	–	–	○
R※2	戻り値。省略可能。省略時、内部レジスタに格納。	.U	.S	.D	.L	.F	.DF※3	–	–	–

※1　ビットデータを指定した場合、連続した16/32ビットで演算。R002などチャンネルの先頭以外を指定した場合、次のチャンネルにまたがって16/32ビットで演算。

※2　Rに演算対象と異なる型を指定した場合、変換時に自動的に型変換を実行。
※3　デバイスZを指定した場合、「.DF」サフィックスは指定不可。

●動作説明

In1に格納されたデータと、In2に格納されたバイナリデータで論理和演算をおこないます。

●書式例

- **In1：演算対象が16ビット(.U/.S)の場合**

uiValue := **ORA** (uiValue1, uiValue2);

- **In1：演算対象が32ビット(.D/.L)の場合**

udiValue := **ORA** (udiValue1, udiValue2);

「演算フラグについて（CR2009～CR2012）」（1-8ページ）
詳細については、『命令語リファレンスマニュアル』「論理演算命令」を参照してください。

●論理和演算真理値表

演算対象	0	0	1	1
演算データ	0	1	0	1
戻り値	0	1	1	1

EORA　排他的論理和演算

エクスクルーシブオアA
[R:=] EORA (In1, In2);

引数/戻り値	説明	認識する型								定数
		.U	.S	.D	.L	.F	.DF	.B	.T	
In1※1	演算対象、または演算対象データを指定。	.U	.U	.D	.D	–	–	–	–	○
In2※1	排他的論理和データの格納先、または排他的論理和データを指定。	型はIn1と同じ。				–	–	–	–	○
R※2	戻り値。省略可能。省略時、内部レジスタに格納。	.U	.S	.D	.L	.F	.DF※3	–	–	–

※1　ビットデータを指定した場合、連続した16/32ビットで演算。R002などチャンネルの先頭以外を指定した場合、次のチャンネルにまたがって16/32ビットで演算。
※2　Rに演算対象と異なる型を指定した場合、変換時に自動的に型変換を実行。
※3　デバイスZを指定した場合、「.DF」サフィックスは指定不可。

●動作説明

In1に格納されたデータと、In2に格納されたバイナリデータで排他的論理和演算をおこないます。

●書式例

- **In1：演算対象が16ビット(.U/.S)の場合**

uiValue := **EORA** (uiValue1, uiValue2);

- **In1：演算対象が32ビット(.D/.L)の場合**

udiValue := **EORA** (udiValue1, udiValue2);

「演算フラグについて（CR2009～CR2012）」（1-8ページ）

詳細については、『命令語リファレンスマニュアル』「論理演算命令」を参照してください。

●排他的論理和演算真理値表

演算対象	0	0	1	1
演算データ	0	1	0	1
戻り値	0	1	1	0

ENRA　排他的論理和否定演算

エクスクルーシブノアA
[R:=] ENRA (In1, In2);

引数/戻り値	説明	認識する型								定数
		.U	.S	.D	.L	.F	.DF	.B	.T	
In1※1	演算対象、または演算対象データを指定。	.U	.U	.D	.D	–	–	–	–	○
In2※1	排他的論理和否定データの格納先、または排他的論理和否定データを指定。	型はIn1と同じ。				–	–	–	–	○
R※2	戻り値。省略可能。省略時、内部レジスタに格納。	.U	.S	.D	.L	.F	.DF ※3	–	–	–

※1　ビットデータを指定した場合、連続した16/32ビットで演算。R002などチャンネルの先頭以外を指定した場合、次のチャンネルにまたがって16/32ビットで演算。
※2　Rに演算対象と異なる型を指定した場合、変換時に自動的に型変換を実行。
※3　デバイスZを指定した場合、「.DF」サフィックスは指定不可。

●動作説明

In1に格納されたデータと、In2に格納されたバイナリデータで排他的論理和否定演算をおこない、結果をRに格納します。

●書式例

- **In1：演算対象が16ビット(.U/.S)の場合**

uiValue := **ENRA** (uiValue1, uiValue2);

- **In1：演算対象が32ビット(.D/.L)の場合**

udiValue := **ENRA** (udiValue1, udiValue2);

「演算フラグについて（CR2009～CR2012）」（1-8ページ）
詳細については、『命令語リファレンスマニュアル』「論理演算命令」を参照してください。

●排他的論理和否定演算真理値表

演算対象	0	0	1	1
演算データ	0	1	0	1
戻り値	1	0	0	1

COM　ビット反転

[R:=] COM (In);
（コンプリメント）

引数/戻り値	説明	認識する型								定数
		.U	.S	.D	.L	.F	.DF	.B	.T	
In	演算対象、または演算対象データを指定。	.U	.U	.D	.D	—	—	—	—	○
R※2	戻り値。省略可能。省略時、内部レジスタに格納。	.U	.S	.D	.L	.F	.DF※2	—	—	—

※1　ビットデータを指定した場合、連続した16/32ビットで演算。R002などチャンネルの先頭以外を指定した場合、次のチャンネルにまたがって16/32ビットで演算。
※2　Rに演算対象と異なる型を指定した場合、変換時に自動的に型変換を実行。
※3　デバイスZを指定した場合、「.DF」サフィックスは指定不可。

●動作説明

Inに格納されたバイナリデータの、各ビットを反転します。

「演算フラグについて（CR2009～CR2012）」（1-8ページ）
詳細については、『命令語リファレンスマニュアル』「論理演算命令」を参照してください。

●書式例

uiValue := **COM** (uiValue1);

NEG　符号反転

[R:=] NEG (In);
（ネガティブ）

引数/戻り値	説明	認識する型								定数
		.U	.S	.D	.L	.F	.DF	.B	.T	
In	演算対象、または演算対象データを指定。	.U	.S	.D	.L	.F	.DF※2	—	—	○
R※1	戻り値。省略可能。省略時、内部レジスタに格納。	.U	.S	.D	.L	.F	.DF※2	—	—	—

※1　Rに演算対象と異なる型を指定した場合、変換時に自動的に型変換を実行。
※2　デバイスZを指定した場合、「.DF」サフィックスは指定不可。

●動作説明

Inに格納されたバイナリデータの、正負の符号を反転(2の補数演算)します。

詳細については、『命令語リファレンスマニュアル』「論理演算命令」を参照してください。

●書式例

uiValue := **NEG** (uiValue1);

ABS　絶対値

アブソリュート
[R:=] ABS (In);

引数/戻り値	説明	認識する型								定数
		.U	.S	.D	.L	.F	.DF	.B	.T	
In	演算対象、または演算対象データを指定。	.U	.S	.D	.L	.F	.DF※2	—	—	○
R※1	演算結果が格納される戻り値。省略可能。省略時、内部レジスタに格納。	.U	.S	.D	.L	.F	.DF※2	—	—	—

※1　Rに演算対象と異なる型を指定した場合、変換時に自動的に型変換を実行。
※2　デバイスZを指定した場合、「.DF」サフィックスは指定不可。

●動作説明

Inに格納されたバイナリデータの、絶対値をRに格納します。

●書式例

uiValue := **ABS** (iADConvData);

ポイント

以下のデータを指定すると正しく動作しません。

- INTの正の整数範囲を超える数(-32768)
- DINTの正の整数範囲を超える数(-2147483648)

例：udiVar := ABS(iVar);　// iVar:= -32768 の時、udiVarには-32768が格納されます。
　lrVar := ABS(udiVar);　// udiVar:=-2147486348の時、lrVarには　-2147486348が格納されます。

データシフト関数

SHR　右シフト(ビット単位)

[R:=] SHR (In, Num);
（シフトライト）

引数/戻り値	説明	認識する型								定数
		.U	.S	.D	.L	.F	.DF	.B	.T	
In	演算対象、または演算対象データを指定。	.U	.U	.D	.D	–	–	–	–	○
Num	右シフト数の格納先、または右シフト数を指定。	.U	.U	.U	.U	–	–	–	–	○
R※1	演算結果が格納される戻り値。省略可能。省略時、内部レジスタに格納。	.U	.S	.D	.L	.F	.DF※2	–	–	–

※1　Rに演算対象と異なる型を指定した場合、変換時に自動的に型変換を実行。
※2　デバイスZを指定した場合、「.DF」サフィックスは指定不可。

●動作説明

Inに格納されたデータを、Numで指定したビット数分右へシフトし、上位ビットからの右シフトしたビット数分には0を格納します。

「演算フラグについて（CR2009～CR2012）」（1-8ページ）
詳細については、『命令語リファレンスマニュアル』「データシフト命令」を参照してください。

●書式例

uiValue := **SHR** (uiValue1, INT#4);

ポイント
- **Numが0の場合、RにはInの値が格納されます。**
- **Inが.U(UINT)でNum≧17、Inが.D(UDINT)でNum≧33の場合、演算結果は0になります。**

SHL　左シフト(ビット単位)

[R:=] SHL (In, Num);
（シフトレフト）

引数/戻り値	説明	認識する型								定数
		.U	.S	.D	.L	.F	.DF	.B	.T	
In	演算対象、または演算対象データを指定。	.U	.U	.D	.D	–	–	–	–	○
Num	左シフト数の格納先、または左シフト数を指定。	.U	.U	.U	.U	–	–	–	–	○
R※1	演算結果が格納される戻り値。省略可能。省略時、内部レジスタに格納。	.U	.S	.D	.L	.F	.DF※2	–	–	–

※1　Rに演算対象と異なる型を指定した場合、変換時に自動的に型変換を実行。
※2　デバイスZを指定した場合、「.DF」サフィックスは指定不可。

●動作説明

Inに格納されたデータを、Numで指定したビット数分左へシフトし、下位ビットからの左シフトしたビット数分には0を格納します。

「演算フラグについて（CR2009～CR2012）」（1-8ページ）
詳細については、『命令語リファレンスマニュアル』「データシフト命令」を参照してください。

●書式例

uiValue := **SHL** (uiValue1, INT#5);

ポイント
- **Num**が0の場合、**R**には**In**の値が格納されます。
- **In**が.U(UINT)で**Num**≧17、**In**が.D(UDINT)で**Num**≧33の場合、演算結果は0になります。

ASRA　算術右シフト(ビット単位)

[**R**:=] **ASRA** (**In**, **Offset**);
（算術シフトライト）

引数/戻り値	説明	認識する型								定数
		.U	.S	.D	.L	.F	.DF	.B	.T	
In	演算対象、または演算対象データを指定。	.U	.S	.D	.L	–	–	–	–	○
Offset	右シフト数の格納先、または右シフト数を指定。	.U	.U	.U	.U	–	–	–	–	○
R※1	演算結果が格納される戻り値。省略可能。省略時、内部レジスタに格納。	.U	.S	.D	.L	.F	.DF※2	–	–	–

※1　**R**に演算対象と異なる型を指定した場合、変換時に自動的に型変換を実行。
※2　デバイスZを指定した場合、「.DF」サフィックスは指定不可。

●動作説明

Inに格納されたデータを、**Offset**で指定したビット数分右へシフトし、上位ビットからの右シフトしたビット数分には**In**≧0のとき「0」が格納され、**In**＜0のとき「1」が格納されます。

「演算フラグについて（CR2009～CR2012）」（1-8ページ）
詳細については、『命令語リファレンスマニュアル』「データシフト命令」を参照してください。

●書式例

uiValue := **ASRA** (uiValue1, INT#5);

ポイント
- **Offset**が0の場合、**R**には**In**の値が格納されます。
- **In**が.U(UINT)で**Offset**≧17、**In**が.D(UDINT)で**Offset**≧33の場合、演算結果は0になります。

ASLA　算術左シフト(ビット単位)

算術シフトレフト
[R:=] ASLA (In, Offset);

引数/戻り値	説明	認識する型								定数
		.U	.S	.D	.L	.F	.DF	.B	.T	
In	演算対象、または演算対象データを指定。	.U	.S	.D	.L	–	–	–	–	○
Offset	左シフト数の格納先、または左シフト数を指定。	.U	.U	.U	.U	–	–	–	–	○
R※1	演算結果が格納される戻り値。省略可能。省略時、内部レジスタに格納。	.U	.S	.D	.L	.F	.DF※2	–	–	–

※1　Rに演算対象と異なる型を指定した場合、変換時に自動的に型変換を実行。
※2　デバイスZを指定した場合、「.DF」サフィックスは指定不可。

●動作説明

Inに格納されたデータを、Offsetで指定したビット数分左へシフトし、下位ビットからの左シフトしたビット数分には0が格納されます。

「演算フラグについて（CR2009～CR2012）」（1-8ページ）
詳細については、『命令語リファレンスマニュアル』「データシフト命令」を参照してください。

●書式例

uiValue := **ASLA** (uiValue1, INT#5);

ポイント
- **Offsetが0の場合、RにはInの値が格納されます。**
- **Inが.U(UINT)でOffset≧17、Inが.D(UDINT)でOffset≧33の場合、演算結果は0になります。**

RRA　右ローテート(ビット単位キャリ付)

ローテートライト
[R:=] RRA (In, Count);

引数/戻り値	説明	認識する型								定数
		.U	.S	.D	.L	.F	.DF	.B	.T	
In	演算対象、または演算対象データを指定。	.U	.U	.D	.D	–	–	–	–	○
Count	右ローテート数の格納先、または右ローテート数を指定。	.U	.U	.U	.U	–	–	–	–	○
R※1	演算結果が格納される戻り値。省略可能。省略時、内部レジスタに格納。RRA関数は、型は第1引数と同じ。※1	.U	.S	.D	.L	.F	.DF※2	–	–	–

※1　Rに演算対象と異なる型を指定した場合、変換時に自動的に型変換を実行。
※2　デバイスZを指定した場合、「.DF」サフィックスは指定不可。

●動作説明

Inに格納されたデータを、Countで指定したビット数分キャリーフラグ(CR2009)を含めて右へローテートします。

「演算フラグについて（CR2009～CR2012）」（1-8ページ）
詳細については、『命令語リファレンスマニュアル』「データシフト命令」を参照してください。

●書式例

uiValue := **RRA** (uiValue1, INT#5);

ポイント　右へローテートしたとき、最下位ビットはキャリーフラグ(CR2009)に格納されます。

RLA　左ローテート(ビット単位キャリ付)

ローテートレフト
[R:=] RLA (In, Count);

引数/戻り値	説明	認識する型								定数
		.U	.S	.D	.L	.F	.DF	.B	.T	
In	演算対象、または演算対象データを指定。	.U	.U	.D	.D	–	–	–	–	○
Count	左ローテート数の格納先、または左ローテート数を指定。	.U	.U	.U	.U	–	–	–	–	○
R※1	演算結果が格納される戻り値。省略可能。省略時、内部レジスタに格納。	.U	.S	.D	.L	.F	.DF※2	–	–	–

※1　Rに演算対象と異なる型を指定した場合、変換時に自動的に型変換を実行。
※2　デバイスZを指定した場合、「.DF」サフィックスは指定不可。

●動作説明

Inに格納されたデータを、Countで指定したビット数分キャリーフラグ(CR2009)を含めて左へローテートします。

「演算フラグについて(CR2009～CR2012)」(1-8ページ)
詳細については、『命令語リファレンスマニュアル』「データシフト命令」を参照してください。

●書式例

uiValue := **RLA** (uiValue1, INT#5);

ポイント　左へローテートしたとき、最上位ビットはキャリーフラグ(CR2009)に格納されます。

ROR　右ローテート(ビット単位キャリ無)

キャリーなしローテートライト
[R:=] ROR (In, Num);

引数/戻り値	説明	認識する型								定数
		.U	.S	.D	.L	.F	.DF	.B	.T	
In	演算対象の格納先を指定。	.U	.U	.D	.D	–	–	–	–	○
Num※1	右ローテート数、またはその格納先を指定。	.U	.U	.U	.U	–	–	–	–	○
R	演算結果が格納される戻り値。省略可能。省略時、内部レジスタに格納。ROR関数は、型は第1引数と同じ。※2	.U	.S	.D	.L	.F	.DF※3	–	–	–

※1　Inに対象を指定した場合、Numの値が16以上(演算対象が.Uのとき)/32以上(演算対象が.Dのとき)のときは以下の通り：

.Uのとき：**Num**の値を16で割った余りの値はローテートするビット数。
.Dのとき：**Num**の値を32で割った余りの値はローテートするビット数。
※2　**R**に演算対象と異なる型を指定した場合、変換時に自動的に型変換を実行。
※3　デバイスZを指定した場合、「.DF」サフィックスは指定不可。

●動作説明

Inに格納されたデータを、**Num**で指定したビット数分、キャリーを含まずに右へローテートし、結果を**R**に格納します。

「演算フラグについて（CR2009～CR2012）」（1-8ページ）
詳細については、『命令語リファレンスマニュアル』「データシフト命令」を参照してください。

参考　ローテート結果の最下位ビットの状態がキャリーフラグ(CR2009)に格納されます。

●書式例

uiValue := **ROR** (uiValue1, INT#5);

ROL　左ローテート(ビット単位キャリ無)

キャリーなしローテートレフト
[R:=] ROL (In, Num);

引数/戻り値	説明	認識する型								定数
		.U	.S	.D	.L	.F	.DF	.B	.T	
In	演算対象の格納先を指定。	.U	.U	.D	.D	–	–	–	–	○
Offset※1	左ローテート数、またはその格納先を指定。	.U	.U	.U	.U	–	–	–	–	○
R※2	演算結果が格納される戻り値。省略可能。省略時、内部レジスタに格納。	.U	.S	.D	.L	.F	.DF※3	–	–	–

※1　**In**に対象を指定した場合、**Num**の値が16以上(演算対象が.Uのとき)/32以上(演算対象が.Dのとき)のときは以下の通り：
.Uのとき：**Num**の値を16で割った余りの値はローテートするビット数。
.Dのとき：**Num**の値を32で割った余りの値はローテートするビット数。
※2　**R**に演算対象と異なる型を指定した場合、変換時に自動的に型変換を実行。
※3　デバイスZを指定した場合、「.DF」サフィックスは指定不可。

●動作説明

Inに格納されたデータを、**Num**で指定したビット数分、キャリーを含まずに左へローテートし、結果を**R**に格納します。

「演算フラグについて（CR2009～CR2012）」（1-8ページ）
詳細については、『命令語リファレンスマニュアル』「データシフト命令」を参照してください。

参考　ローテート結果の最上位ビットの状態がキャリーフラグ(CR2009)に格納されます。

●書式例

uiValue := **ROL** (uiValue1, INT#5);

WSR　ワードデバイス右シフト

ワードシフトライト
WSR ([EN], InOut, Count, Offset);

引数/戻り値	説明	認識する型								定数
		.U	.S	.D	.L	.F	.DF	.B	.T	
EN	実行条件。省略可能。省略時、常時(毎スキャン)実行。	—	—	—	—	—	—	.B	—	—
InOut※1	演算対象、または演算対象データを指定。	.U	.U	.D	.D	.D	—	—	—	—
Count※1	右シフトする範囲のデータ数の格納先、または右シフト範囲データ数を指定。	.U	.U	.U	.U	—	—	—	—	○
Offset	右シフト数の格納先、または右シフト数を指定。	.U	.U	.U	.U	—	—	—	—	○
R	戻り値はなし。	—	—	—	—	—	—	—	—	—

※1　ビットデータを指定した場合、連続した16/32ビットで演算。
R002などチャンネルの先頭以外を指定した場合、次のチャンネルにまたがって16/32ビットで演算。

●動作説明

実行条件が成り立つ(TRUE)とき、**InOut**を先頭として、**Count**で指定されたデータ数の範囲に格納されたデータ(16ビット/32ビット)を、**Offset**で指定されたデータ数分だけ右へ(デバイス番号/配列の添え字の大きい方へ)シフトします。
シフト元の範囲の内、シフト先の範囲と重ならない部分には0が格納されます。

「演算フラグについて(CR2009～CR2012)」(1-8ページ)

詳細については、『命令語リファレンスマニュアル』「データシフト命令」を参照してください。

参考　演算対象が32ビットデータのとき、演算範囲データ数、およびシフト数は指定されたデータ数×2ワードとなります。

●書式例

WSR (bFlag, uiValue, INT#32, INT#10);

ポイント
- **Count＝0の場合、関数は実行されません。**
- **Offset＝0の場合、演算結果は演算前と変わりません。**

WSL　ワードデバイス左シフト

ワードシフトレフト
WSL ([EN], InOut, Count, Offset);

引数/戻り値	説明	認識する型								定数
		.U	.S	.D	.L	.F	.DF	.B	.T	
EN	実行条件。省略可能。省略時、常時(毎スキャン)実行。	—	—	—	—	—	—	.B	—	—
InOut※1	左シフトする範囲の先頭を指定。	.U	.U	.D	.D	.D	—	—	—	—
Count※1	左シフトする範囲のデータ数の格納先、または左シフト範囲データ数を指定。	.U	.U	.U	.U	—	—	—	—	○
Offset	左シフト数の格納先、または左シフト数を指定。	.U	.U	.U	.U	—	—	—	—	○
R	戻り値はなし。	—	—	—	—	—	—	—	—	—

※1　ビットデータを指定した場合、連続した16/32ビットで演算。
R002などチャンネルの先頭以外を指定した場合、次のチャンネルにまたがって16/32ビットで演算。

●動作説明

実行条件が成り立つ(TRUE)とき、**InOut**を先頭として、**Count**で指定されたデータ数の範囲に格納されたデータ(16ビット/32ビット)を、**Offset**で指定されたデータ数分だけ左へ(デバイス番号/配列の添え字の小さい方へ)シフトします。
シフト元の範囲の内、シフト先の範囲と重ならない部分には0が格納されます。

「演算フラグについて（CR2009～CR2012）」（1-8ページ）
詳細については、『命令語リファレンスマニュアル』「データシフト命令」を参照してください。

参考　演算対象が32ビットデータのとき、演算範囲データ数、およびシフト数は指定されたデータ数×2ワードとなります。

●書式例

WSL (bFlag, uiValue, INT#32, INT#10);

ポイント
- **Count＝0の場合、関数は実行されません。**
- **Offset＝0の場合、演算結果は演算前と変わりません。**

BSR　ビットデバイス右シフト

ビットシフトライト
BSR ([EN], InOut, Count, Offset);

引数/戻り値	説明	認識する型								定数
		.U	.S	.D	.L	.F	.DF	.B	.T	
EN	実行条件。省略可能。省略時、常時（毎スキャン）実行。	–	–	–	–	–	–	.B	–	–
InOut※1※2	右シフトする範囲の先頭を指定。	–	–	–	–	–	–	.B	–	–
Count	右シフトする範囲の対象数の格納先、または右シフト範囲対象数を指定。	.U	.U	.U	.U	–	–	–	–	○
Offset	右シフト数の格納先、または右シフト数を指定。	.U	.U	.U	.U	–	–	–	–	○
R	戻り値はなし。	–	–	–	–	–	–	–	–	–

※1　ビットデータを指定した場合、チャンネルをまたがっても正しく動作。
ワードデータを指定した場合、シフトする対象は**Count**個のワードデータの最下位ビット。最下位ビット以外は変化なし。
※2　T, C, CR, Z, CMは使用不可。

●動作説明

実行条件が成り立つ(TRUE)とき、**InOut**を先頭とする、**Count**で指定された個数の状態を、**Offset**で指定された回数分だけ右へ(デバイス番号/配列の添え字の大きい方へ)シフトします。**InOut**がワードデータのとき、各ワードデータの最下位ビットのみがシフト対象となります。(最下位ビット以外は変化しません)
シフト元の範囲の内、シフト先の範囲と重ならない部分(ワードデータのときは最下位ビット)にはOFFが格納されます。

●書式例

BSR (bFlag, abFlag[0], INT#3, INT#4);

「演算フラグについて（CR2009～CR2012）」（1-8ページ）
詳細については、『命令語リファレンスマニュアル』「データシフト命令」を参照してください。

ポイント
- **Offset＝0の場合、演算結果は演算前と変わりません。**
- **次のいずれかの条件が成立した場合、関数は実行されません。**
  - **シフト先がデバイスや変数の範囲を超える場合**
  - **Countが0の場合。**

BSL　ビットデバイス左シフト

ビットシフトレフト
BSL ([EN], InOut, Count, Offset);

引数/戻り値	説明	認識する型								定数
		.U	.S	.D	.L	.F	.DF	.B	.T	
EN	実行条件。省略可能。省略時、常時（毎スキャン）実行。	—	—	—	—	—	—	.B	—	—
InOut※1※2	左シフトする範囲の先頭を指定。	—	—	—	—	—	—	.B	—	—
Count	左シフトする範囲の対象数の格納先、または左シフト範囲対象数を指定。	.U	.U	.U	.U	—	—	—	—	○
Offset	左シフト数の格納先、または左シフト数を指定。	.U	.U	.U	.U	—	—	—	—	○
R	戻り値はなし。	—	—	—	—	—	—	—	—	—

※1　ビットデータを指定した場合、チャンネルをまたがっても正しく動作。
　　ワードデータを指定した場合、シフトする対象は**Count**個のワードデータの最下位ビット。最下位ビット以外は変化なし。
※2　T, C, CR, Z, CMは使用不可。

●動作説明

実行条件が成り立つ(TRUE)とき、**InOut**を先頭とする、**Count**で指定された個数の状態を、**Offset**で指定された回数分だけ左へ(デバイス番号/配列の添え字の小さい方へ)シフトします。**InOut**がワードデータのとき、各ワードデータの最下位ビットのみがシフト対象となります。(最下位ビット以外は変化しません)
シフト元の範囲の内、シフト先の範囲と重ならない部分(ワードデータのときは最下位ビット)にはOFFが格納されます。
シフト元の範囲の内、シフト先の範囲と重ならない部分には0が格納されます。

●書式例

BSL (bFlag, abFlag[0], INT#3, INT#4);

「演算フラグについて（CR2009～CR2012）」（1-8ページ）
詳細については、『命令語リファレンスマニュアル』「データシフト命令」を参照してください。

ポイント
- **Offset＝0の場合、演算結果は演算前と変わりません。**
- **次のいずれかの条件が成立した場合、関数は実行されません。**
  - **InOut－Count＞0となる場合。**
  - **Countが0の場合。**

データ制御関数

LIMIT　上下限リミット制御

[R:=] LIMIT (MN, In, MX);
リミット

引数/戻り値	説明	認識する型								定数
		.U	.S	.D	.L	.F	.DF	.B	.T	
MN	下限値を指定。	.U	.S	.D	.L	.F	.DF※2	—	—	○
In	制御対象を指定。	.U	.S	.U	.L	.F	.DF※2	—	—	○
MX	上限値を指定。	.U	.S	.U	.L	.F	.DF※2	—	—	○
R※1	演算結果が格納される戻り値。省略可能。省略時、内部レジスタに格納。	.U	.S	.D	.L	.F	.DF※2	—	—	—

※1　Rに演算対象と異なる型を指定した場合、変換時に自動的に型変換を実行。
※2　デバイスZを指定した場合、「.DF」サフィックスは指定不可。

●動作説明

MN/MXによってInを制限し、結果をRに格納します。

In の値	R の値
In < MN	MN
MN ≦ In ≦ MX	In
MX < In	MX

●書式例

rValue := **LIMIT** (rMinValue, rInput, rMaxValue);

「演算フラグについて（CR2009～CR2012）」（1-8ページ）
詳細については、『命令語リファレンスマニュアル』「データ制御命令」を参照してください。

BANDC　不感帯制御

[R:=] BANDC (LoLim, HiLim, In);
不感帯制御

引数/戻り値	説明	認識する型								定数
		.U	.S	.D	.L	.F	.DF	.B	.T	
LoLim※1	下限値、または下限値の格納先を指定。	.S	.S	.L	.L	—	—	—	—	○
HiLim※1	上限値、または上限値の格納先を指定。	型はLoLimと同じ。				—	—	—	—	○
In※1	不感帯制御の入力値、または入力値の格納先を指定する。	型はLoLimと同じ。				—	—	—	—	○
R※2	戻り値。省略可能。省略時、内部レジスタに格納。	.U	.S	.D	.L	.F	.DF※3	—	—	—

※1　ビットデータを指定した場合、連続した16/32ビットで演算。R002などチャンネルの先頭以外を指定した場合、次のチャンネルにまたがって16/32ビットで演算。
※2　Rに演算対象と異なる型を指定した場合、変換時に自動的に型変換を実行。
※3　デバイスZを指定した場合、「.DF」サフィックスは指定不可。

●動作説明

HiLim/LoLimにInを制限し、結果をRに格納します。

In の値	R の値
In ＜ LoLim	LoLim
LoLim ≦ In ≦ HiLim	In
HiLim ＜ In	HiLim

●書式例

uiValue := **BANDC** (uiValue1, uiValue2, uiValue3);

「演算フラグについて（CR2009～CR2012）」（1-8ページ）
詳細については、『命令語リファレンスマニュアル』「データ制御命令」を参照してください。

ZONE　ゾーン制御

[R:=] ZONE (NegBias, PosBias, In);

引数/戻り値	説明	認識する型								定数
		.U	.S	.D	.L	.F	.DF	.B	.T	
NegBias※1	入力値に加算する負のバイアス値、または負のバイアス値の格納先を指定。	.S	.S	.L	.L	－	－	－	－	○
PosBias※1	入力値に加算する正のバイアス値、または正のバイアス値の格納先を指定。	型はNegBiasと同じ。				－	－	－	－	○
In※1	ゾーン制御の入力値、または入力値の格納先を指定。	型はNegBiasと同じ。				－	－	－	－	○
R	戻り値。省略可能。省略時、内部レジスタに格納。ZONE関数は、型は第１引数と同じ。※2	.U	.S	.D	.L	.F	.DF※3	－	－	－

※1　ビットデータを指定した場合、連続した16/32ビットで演算。R002などチャンネルの先頭以外を指定した場合、次のチャンネルにまたがって16/32ビットで演算。
※2　Rに演算対象と異なる型を指定した場合、変換時に自動的に型変換を実行。
※3　デバイスZを指定した場合、「.DF」サフィックスは指定不可。

●動作説明

PosBias/NegBiasにInを制限し、結果をRに格納します。

In の値	R の値
In ＜ NegBias	NegBias
NegBias ≦ In ≦ PosBias	In
PosBias ＜ In	PosBias

●書式例

uiValue := **ZONE** (uiValue1, uiValue2, uiValue3);

「演算フラグについて（CR2009～CR2012）」（1-8ページ）
詳細については、『命令語リファレンスマニュアル』「データ制御命令」を参照してください。

APR　折れ線近似(スケーリング)

[R:=] APR (Src, Table, In);
(折れ線近似)

引数/戻り値	説明	認識する型								定数
		.U	.S	.D	.L	.F	.DF	.B	.T	
Src※1	折れ線近似の入力値、または入力値の格納先を指定。	.U	.S	.D	.L	.F	.DF※4	–	–	○
Table※2	折れ線データテーブルを格納している先頭を指定。	型はSrcと同じ。						–	–	–
In	折れ線データレコード数、またはレコード数の格納先を指定。	.U	.U	.U	.U	–	–	–	–	○
R※3	戻り値。省略可能。省略時、内部レジスタに格納。	.U	.S	.D	.L	.F	.DF※4	–	–	–

※1　•ビットデータを指定した場合、連続した16/32/64ビットで演算。R002などチャンネルの先頭以外を指定した場合、次のチャンネルにまたがって16/32/64ビットで演算。
•T,Cを指定した場合、現在値で演算。
※2　Srcの型が.Fまたは.DFの場合、T,Cは指定不可。
※3　Rに演算対象と異なる型を指定した場合、変換時に自動的に型変換を実行。
※4　デバイスZを指定した場合、「.DF」サフィックスは指定不可。

●動作説明

- **Src：入力値が16ビット(.U/.S)の場合**
  Srcに格納されている折れ線近似の「入力値」に対し、Tableを先頭にInレコード分格納されている「折れ線データテーブル」にしたがって近似計算をおこないます。
  折れ線近似計算は、Srcが、折れ線データテーブルのどの範囲内にあるのかをサーチし、前後の2点で折れ線近似をおこないます。計算結果は、Rに格納されます。
  折れ線データテーブルのXの値は、昇順に設定してください。($X_0 < X_1 \ldots < X_k \ldots < X_n$)
  折れ線データテーブルは、(X_0, Y_0)、(X_1, Y_1)…(X_{n-1}, Y_{n-1})、(X_n, Y_n)の順にTableから格納してください。
  Inには、折れ線データテーブルのレコード数を格納してください。また、Inは2以上の値としてください。

  Src(入力値)<X_0の場合は、RにはY_0が格納されます。
  Src(入力値)>X_nの場合は、RにはY_nが格納されます。
  Src(入力値)=X_kの場合は、RにはY_kが格納されます。

- **Src：入力値が32ビット(.D/.L)の場合**
  [Src・Src+(1)]に格納されている折れ線近似の「入力値」に対し、[Table・Table+(1)]を先頭に[In]レコード分格納されている「折れ線データテーブル」にしたがって近似計算をおこないます。
  折れ線近似計算は、[Src・Src+(1)]が、折れ線データテーブルのどの範囲内にあるのかをサーチし、前後の2点で折れ線近似をおこないます。
  計算結果は、[R・R+(1)]に格納されます。
  折れ線データテーブルは、(X_0, Y_0)、(X_1, Y_1)…(X_{n-1}, Y_{n-1})、(X_n, Y_n)の順に[Table・Table+(1)]から格納してください。
  [In](レコード数)には、折れ線データテーブルのレコード数を格納してください。
  また、[In]は2個以上としてください。

  [Src・Src+(1)]<X_0の場合は、[R・R+(1)]にはY_0が格納されます。
  [Src・Src+(1)]>X_nの場合は、[R・R+(1)]にはY_nが格納されます。
  [Src・Src+(1)]=X_kの場合は、[R・R+(1)]にはY_kが格納されます(k=1～n)。

- **Src：入力値が32ビット(.F)の場合**
  [Src・Src+(1)]に格納されている折れ線近似の「入力値」に対し、[Table・Table+(1)]を先頭に[In]レコード分格納されている「折れ線データテーブル」にしたがって近似計算(単精度浮動小数点型)をおこないます。
  折れ線近似計算は、[Src・Src+(1)]が、折れ線データテーブルのどの範囲内にあるのかをサーチし、前後の2点で折れ線近似をおこないます。
  計算結果は、[R・R+(1)]に格納されます。

折れ線データテーブルは、(X_0, Y_0)、(X_1, Y_1)…(X_{n-1}, Y_{n-1})、(X_n, Y_n)の順に[Table・Table+(1)]から格納してください。

折れ線データテーブルのXの値は、昇順に設定してください。$(X_0 < X_1 \ldots < X_k \ldots < X_n)$
[In]には、折れ線データテーブルのレコード数を格納してください。
また、[In]は2個以上としてください。

[Src・Src+(1)]<X_0の場合は、[R・R+(1)]にはY_0が格納されます。
[Src・Src+(1)]>X_nの場合は、[R・R+(1)]にはY_nが格納されます。
[Src・Src+(1)]=X_kの場合は、[R・R+(1)]にはY_kが格納されます(k=1～n)。

- **Src：入力値が64ビット(.DF)の場合**
  [Src・Src+(1)・Src+(2)・Src+(3)]に格納されている折れ線近似の「入力値」に対し、[Table+(1)・Table+(2)・Table+(3)]を先頭に[In]レコード分格納されている「折れ線データテーブル」にしたがって近似計算(倍精度浮動小数点型)をおこないます。

  折れ線近似計算は、[Src・Src+(1)・Src+(2)・Src+(3)]が、折れ線データテーブルのどの範囲内にあるのかをサーチし、前後の2点で折れ線近似をおこないます。
  計算結果は、[R・R+(1)・R+(2)・R+(3)]に格納されます。
  折れ線データテーブルは、(X_0, Y_0)、(X_1, Y_1)…(X_{n-1}, Y_{n-1})、(X_n, Y_n)の順に[Table・Table+(1)・Table+(2)・Table+(3)]から格納してください。
  折れ線データテーブルのXの値は、昇順に設定してください。$(X_0 < X_1 \ldots < X_k \ldots < X_n)$

  [In]には、折れ線データテーブルのレコード数を格納してください。
  また、[In]は2個以上としてください。

  [Src・Src+(1)・Src+(2)・Src+(3)]<X_0の場合は、[R・R+(1)・R+(2)・R+(3)]にはY_0が格納されます。
  [Src・Src+(1)・Src+(2)・Src+(3)]>X_nの場合は、[R・R+(1)・R+(2)・R+(3)]にはY_nが格納されます。
  [Src・Src+(1)・Src+(2)・Src+(3)]=X_kの場合は、[R・R+(1)・R+(2)・R+(3)]にはY_kが格納されます(k=1～n)。

「演算フラグについて（CR2009～CR2012）」（1-8ページ）
詳細については、『命令語リファレンスマニュアル』「データ制御命令」を参照してください。

●書式例

uiValue := **APR** (uiValue1, auiData[0],uiCount);

RAMP　傾斜信号

傾斜信号
RAMP ([EN], In, Ratio, TimeUnit, InOut);

引数/戻り値	説明	認識する型								定数
		.U	.S	.D	.L	.F	.DF	.B	.T	
EN	実行条件。省略可能。省略時、常時（毎スキャン）実行。	–	–	–	–	–	–	.B	–	–
In※1	入力値、または入力値の格納先を指定。（－32768～32767）	.S	.S	.S	.S	–	–	–	–	○
Ratio※1	時間単位あたりの変化率、または変化率の格納先を指定。（0～65535）	.U	.U	.U	.U	–	–	–	–	○
TimeUnit※1※2	時間単位を指定。（0～2）	.U	.U	.U	.U	–	–	–	–	○
InOut※1	指定した間隔で、入力値への傾斜量を加算した結果を格納。	.S	.S	.S	.S	–	–	–	–	○
R	戻り値はなし。	–	–	–	–	–	–	–	–	–

※1　・ビットデータを指定した場合、連続した16ビットで演算。R002などチャンネルの先頭以外を指定した場合、次のチャンネルにまたがって16ビットで演算。

・T, C, Zは下位16ビットで演算。
※2　時間単位設定は、0～2の範囲で指定。

TimeUnit時間単位設定の説明

値	時間単位
0	1秒単位
1	1分単位
2	1時間単位

●動作説明

実行条件が成り立つ(TRUE)とき、Inに一致するまで、指定の傾き(Ratio/TimeUnit)でInOutの値を近づけていきます。

「演算フラグについて（CR2009～CR2012）」（1-8ページ）
詳細については、『命令語リファレンスマニュアル』「データ制御命令」を参照してください。

●書式例

RAMP (bFlag, uiValue1, uiValue2, INT#1, uiValue);

ポイント　**TimeUnit≧3の場合、関数は実行されません。**

TPOUT　時分割比例出力

時分割比例出力
TPOUT ([EN], Duty, Cycle, InOut);

引数/戻り値	説明	認識する型								定数
		.U	.S	.D	.L	.F	.DF	.B	.T	
EN	実行条件。省略可能。省略時、常時（毎スキャン）実行。	—	—	—	—	—	—	.B	—	—
Duty※1	出力リレーをONにする時間をms単位で指定。（0～65535）	.U	.U	.U	.U	—	—	—	—	○
Cycle※1	出力リレーのON/OFF周期をms単位で指定。（1～65535）	.U	.U	.U	.U	—	—	—	—	○
InOut※1	ON/OFFさせる対象を指定。	—	—	—	—	—	—	.B	—	—
R	戻り値はなし。	—	—	—	—	—	—	—	—	—

※1　・ビットデータを指定した場合、連続した16ビットで演算。R002などチャンネルの先頭以外を指定した場合、次のチャンネルにまたがって16ビットで演算。
・T, C, Zは下位16ビットで演算。
※2　・ワードデータを指定した場合、結果の格納先は最下位ビット。(最下位ビット以外は変化なし)
・R1002等チャンネルの先頭以外を指定した場合でも、次のチャンネルにまたがって正常に動作。
・Zは指定不可。

●動作説明

実行条件が成り立つ(TRUE)とき、Cycle(ms)でInOutをDuty(ms)だけONにします。

「演算フラグについて（CR2009～CR2012）」（1-8ページ）
詳細については、『命令語リファレンスマニュアル』「データ制御命令」を参照してください。

●書式例

TPOUT (bFlag, uiValue1, uiValue2, bControl);

ポイント

- 次のいずれかの条件が成立した場合、関数は実行されません。
  - Cycle＜Dutyの場合
  - Cycle=0の場合
- ±10μs+1スキャンタイム以内の誤差が生じます。
- Dutyに0を指定した場合、1スキャンもONしません。
- RUN中書き込みをおこなった場合、周期(Cycle)が初期化されてRUN中書き込み終了時からON時間(Duty)を計測します。
- 割り込みプログラム内および初期化モジュールでは、使用できません。
- サブルーチンプログラム内、STP～STE命令間、STG～JMP/ENDS命令間、CJ(NCJ)～LABEL命令間で使用するときは注意が必要です。
  使用しているCPUユニットのユーザーズマニュアルの「タイマ命令使用時の注意」を参照してください。
- RUN中書き込みをおこなう場合は、注意が必要です。
  使用しているCPUユニットのユーザーズマニュアルの「RUN中書き込みについて」を参照してください。

LLFLT　進み遅れフィルタ

進み遅れフィルタ
LLFLT ([EN], In, Param, InOut);

引数/戻り値	説明	認識する型								定数
		.U	.S	.D	.L	.F	.DF	.B	.T	
EN	実行条件。省略可能。省略時、常時(毎スキャン)実行。	—	—	—	—	—	—	.B	—	—
In※1※2	入フィルタの入力値、または入力値の格納先を指定。(－32768～32767)	.S	.S	.S	.S	—	—	—	—	○
Param※1※3	パラメータを格納している先頭を指定。	.U	.U	.U	.U	—	—	—	—	—
InOut※1	出力値の格納先を格納 (－32768～32767)。	.S	.S	.S	.S	—	—	—	—	—
R	戻り値はなし。	—	—	—	—	—	—	—	—	—

※1　ビットデータを指定した場合、連続した16ビットで演算。R002などチャンネルの先頭以外を指定した場合、次のチャンネルにまたがって16ビットで演算。
※2　T, C, Zは下位16ビットで演算。
※3　•ビットデータを指定した場合、連続する48ビットを占有。
•ビットデータでは、チャンネルの先頭を指定。
•ワードデータを指定した場合、連続する3ワードを占有。

●動作説明

実行条件が成り立つ(TRUE)とき、**In**に対して**Param**ごとに進み遅れ演算を実行して、**InOut**に格納します。
実行条件の立ち上がり時、**InOut**は**In**の値で初期化されます。

Param:パラメータの内容

ワードオフセット		説明
Param	サンプリング周期(T_S)	命令を実行する周期時間を指定します。(1～60000ms) 遅れ時間、進み時間よりも少ない値を指定してください。
Param+(1)	遅れ時間(T1)	遅れ要素の時定数を指定します。(0～65535ms) 0を指定した場合、遅れがなくなります。
Param+(2)	進み時間(T2)	進み要素の時定数を指定します。(0～65535ms) 0を指定した場合、進みがなくなります。

進み遅れ演算の詳細については、『命令語リファレンスマニュアル』「LLFLT命令」を参照してください。
「演算フラグについて(CR2009～CR2012)」(1-8ページ)

●書式例

LLFLT (bFlag, uiValue, auiData[0], iControl);

ポイント
- **RUN中書き込みをおこなった場合、サンプリング周期(Param+(1))が初期化され、出力値(InOut)は入力値(In)の値で初期化されます。**
- **割り込みプログラム内および初期化モジュールでは、使用できません。**
- **サブルーチンプログラム内、STP～STE命令間、STG～JMP/ENDS命令間、CJ(NCJ)～LABEL命令間で使用するときは注意が必要です。**
  使用しているCPUユニットのユーザーズマニュアルの「タイマ命令使用時の注意」を参照してください。
- **RUN中書き込みをおこなう場合は、注意が必要です。**
  使用しているCPUユニットのユーザーズマニュアルの「RUN中書き込みについて」を参照してください。

データ変換関数

TBCD　BINデータ→BCDデータ変換

トランスファBCD
[R:=] TBCD (In);

引数/戻り値	説明	認識する型								定数
		.U	.S	.D	.L	.F	.DF	.B	.T	
In	変換対象、または変換対象データを指定。	.U	.U	.D	.D	–	–	–	–	○
R※1	演算結果が格納される戻り値。省略可能。省略時、内部レジスタに格納。	.U	.S	.D	.L	.F	.DF※2	–	–	–

※1　Rに演算対象と異なる型を指定した場合、変換時に自動的に型変換を実行。
※2　デバイスZを指定した場合、「.DF」サフィックスは指定不可。

●動作説明

Inに格納された16ビットのバイナリデータを4桁BCD(16ビット)データに、あるいは32ビットのバイナリデータを8桁BCD(32ビット)データに変換し、Rに格納します。

「演算フラグについて（CR2009～CR2012）」（1-8ページ）
詳細については、『命令語リファレンスマニュアル』「データ変換命令」を参照してください。

●書式例

uiValueBCD := **TBCD** (uiValue);

ポイント　**Inが16ビットの時は0～9999、32ビットの時は0～99999999の範囲を超えた場合は、RにInの値が格納されます。**

TBIN　BCDデータ→BINデータ変換

トランスファBIN
[R:=] TBIN (In);

引数/戻り値	説明	認識する型								定数
		.U	.S	.D	.L	.F	.DF	.B	.T	
In	変換対象、または変換対象データを指定。	.U	.U	.D	.D	–	–	–	–	○
R※1	演算結果が格納される戻り値。省略可能。省略時、内部レジスタに格納。	.U	.S	.D	.L	.F	.DF※2	–	–	–

※1　Rに演算対象と異なる型を指定した場合、変換時に自動的に型変換を実行。
※2　デバイスZを指定した場合、「.DF」サフィックスは指定不可。

●動作説明

Inに格納された4桁BCD(16ビット)データを16ビットのバイナリデータに、あるいは8桁BCD(32ビット)データを32ビットのバイナリデータに変換し、Rに格納します。

「演算フラグについて（CR2009～CR2012）」（1-8ページ）
詳細については、『命令語リファレンスマニュアル』「データ変換命令」を参照してください。

●書式例

uiValue := **TBIN** (uiValueBCD);

ポイント　**In**がBCDデータでないとき、**R**には**In**の値が格納されます。

MPX　4ビットデータデコード

[**R**:=] **MPX** (**In**, **Pos**);
（マルチプレクス）

引数/戻り値	説明	認識する型								定数
		.U	.S	.D	.L	.F	.DF	.B	.T	
In※2※3	変換対象、または変換対象データを指定。	.U	.U	.D	.D	—	—	—	—	—
Pos※1	変換桁位置を指定。（0～3）	—	—	—	—	—	—	—	—	○
R※4	演算結果が格納される戻り値。省略可能。省略時、内部レジスタに格納。	.U	.S	.D	.L	.F	.DF※5	—	—	—

※1　**Pos**には、0～3(4ビット/1桁)の桁数を定数で指定。(対象は指定不可)
※2　**In**に32ビットデータが指定された場合、変換対象(有効変換桁)は下位16ビット。
※3　R1002等変換対象にビットデータのチャンネルの先頭以外を指定した場合でも、次のチャンネルにまたがって正常に動作。
※4　**R**に.U以外の型を指定した場合、変換時に自動的に型変換を実行。
※5　デバイスZを指定した場合、「.DF」サフィックスは指定不可。

●動作説明

Inに格納された16ビットデータの、**Pos**で指定された桁数の4ビット(0桁～3桁)を数値化(0～15)します。数値化された値が示すビットのみをONにした結果を**R**に格納します。

詳細については、『命令語リファレンスマニュアル』「データ変換命令」を参照してください。

●書式例

uiValueB := **MPX** (uiValue, INT#2);

DMX　4ビットデータエンコード

[**R**:=] **DMX** (**In**);
（デマルチプレクス）

引数/戻り値	説明	認識する型								定数
		.U	.S	.D	.L	.F	.DF	.B	.T	
In※1※2	変換対象、または演算対象データを指定。	.U	.U	.U	.U	—	—	—	—	○
R※3	演算結果が格納される戻り値。省略可能。省略時、内部レジスタに格納。	.U	.S	.D	.L	.F	.DF※4	—	—	—

※1　**In**に32ビットデータが指定された場合、変換対象(有効変換桁)は下位16ビット。
※2　R1002等変換対象にビットデータのチャンネルの先頭以外を指定した場合でも、次のチャンネルにまたがって正常に動作。
※3　**R**に.U以外の型を指定した場合、変換時に自動的に型変換を実行。

※4　デバイスZを指定した場合、「.DF」サフィックスは指定不可。

●動作説明

Inに格納された16ビットデータの、ONしているビットの内で最上位のビット位置(0～15)を、16ビット(下位4ビットに格納)データとして**R**に格納します。

「演算フラグについて（CR2009～CR2012）」（1-8ページ）
詳細については、『命令語リファレンスマニュアル』「データ変換命令」を参照してください。

●書式例

uiCount := **DMX** (uiValue);

ポイント　**Inの値が0の場合、Rには0が格納されます。**

GRY　BINデータ→グレイコード変換

グレイコード変換
[R:=] GRY (In);

引数/戻り値	説明	認識する型								定数
		.U	.S	.D	.L	.F	.DF	.B	.T	
In	変換対象、または変換対象データを指定。	.U	.U	.D	.D	–	–	–	–	○
R※1	演算結果が格納される戻り値。省略可能。省略時、内部レジスタに格納。	.U	.S	.D	.L	.F	.DF※2	–	–	–

※1　**R**に演算対象と異なる型を指定した場合、変換時に自動的に型変換を実行。
※2　デバイスZを指定した場合、「.DF」サフィックスは指定不可。

●動作説明

Inに格納された16ビット、あるいは32ビットのバイナリデータを、グレイコードに変換し、**R**に格納します。
詳細については、『命令語リファレンスマニュアル』「データ変換命令」を参照してください。

●書式例

uiValueGRY := **GRY** (uiValue);

グレイコード表

10進	bit															
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1
3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
4	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0
≀																
31	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
32	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0
≀																
255	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
256	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0
≀																
65535	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

RGRY　グレイコード→BINデータ変換

グレイコード逆変換
[R:=] RGRY (In);

引数/戻り値	説明	認識する型								定数
		.U	.S	.D	.L	.F	.DF	.B	.T	
In	変換対象、または演算対象データを指定。	.U	.U	.D	.D	—	—	—	—	○
R※1	演算結果が格納される戻り値。省略可能。省略時、内部レジスタに格納。	.U	.S	.D	.L	.F	.DF※2	—	—	—

※1　Rに演算対象と異なる型を指定した場合、変換時に自動的に型変換を実行。
※2　デバイスZを指定した場合、「.DF」サフィックスは指定不可。

●動作説明

Inに格納されたグレイコードを、16ビット、あるいは32ビットのバイナリデータに変換し、Rに格納します

詳細については、『命令語リファレンスマニュアル』「データ変換命令」を参照してください。

●書式例

uiValue := **RGRY** (uiValueGRY);

グレイコード表

10進	bit															
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1
3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
4	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0
≀																
31	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
32	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0
≀																
255	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
256	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0
≀																
65535	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DISN　ニブル分離

ディスN
DISN ([EN], In, Dst, Count);

引数/戻り値	説明	認識する型								定数
		.U	.S	.D	.L	.F	.DF	.B	.T	
EN	実行条件。省略可能。省略時、常時（毎スキャン）実行。	—	—	—	—	—	—	.B	—	—
In※1※3	ニブル分離するデータの格納先を指定。	.U	.U	.D	.D	—	—	.B	—	○
Dst※1※3	ニブル分離したデータを格納する先頭を指定。	型はInと同じ。					—	—	—	○
Count※2	分離するニブルデータの個数を指定。	.U	.U	.U	.U	—	—	—	—	○
R	戻り値はなし。	—	—	—	—	—	—	—	—	—

※1　ビットデータを指定した場合、**In**は連続した16/32ビット、**Dst**は連続した64/128ビットで演算。R002などチャンネルの先頭以外を指定した場合、次のチャンネルにまたがって実行。
ワードデータを指定した場合、**In**は連続した1/2ワードを扱います。**Dst**は4/8ワードで演算。
※2　**In**で指定したデータのサフィックスが.Uもしくは.Bの場合0～4、.Dの場合0～8の範囲で指定。
※3　Zは指定不可。

●動作説明

実行条件が成り立つ(TRUE)とき、**In**で指定されたデータを**Count**で指定された個数にニブル(4ビット)単位で分離し、**Dst**へ格納します。
4ビットより上位には、0が格納されます。

「演算フラグについて（CR2009～CR2012）」（1-8ページ）
詳細については、『命令語リファレンスマニュアル』「データ変換命令」を参照してください。

●書式例

DISN (uiValue, auiDst[0], uiCount);

ポイント　**Countが0の場合、関数は動作しません。**

UNIN　ニブル結合

ユニN
UNIN ([EN], Src, InOut, Count);

引数/戻り値	説明	認識する型								定数
		.U	.S	.D	.L	.F	.DF	.B	.T	
EN	実行条件。省略可能。省略時、常時（毎スキャン）実行。	—	—	—	—	—	—	.B	—	—
Src※1※3	ニブル結合するデータの格納先を指定。	.U	.U	.D	.D	—	—	.B	—	—
InOut※1※3	ニブル結合したデータを格納する先頭を指定。	型は**Src**と同じ。					—	—	—	—
Count※2	結合するニブルデータの個数を指定。	.U	.U	.U	.U	—	—	—	—	—
R	戻り値はなし。	—	—	—	—	—	—	—	—	—

※1　ビットデータを指定した場合、**Src**は連続した64/128ビット、**InOut**は連続した16/32ビットで演算。R002などチャンネルの先頭以外を指定した場合、次のチャンネルにまたがって実行。
ワードデータを指定した場合、**Src**は連続した4/8ワードを扱います。**InOut**は1/2ワードで演算。
※2　**Src**で指定したデータのサフィックスが.Uもしくは.Bの場合0～4、.Dの場合0～8の範囲で指定。
※3　Zは指定不可。

●動作説明

実行条件が成り立つ(TRUE)とき、**Src**を先頭とする結合データの下位4ビットを**Count**だけBINデータに結合し、**InOut**へ格納します。

「演算フラグについて（CR2009～CR2012）」（1-8ページ）
詳細については、『命令語リファレンスマニュアル』「データ変換命令」を参照してください。

●書式例

UNIN (auiValue[0], uiDst, uiCount);

ポイント　**Countが0の場合、関数は動作しません。**

DISB　バイト分離

ディスB
DISB ([EN], In, InOut, Count);

引数/戻り値	説明	認識する型								定数
		.U	.S	.D	.L	.F	.DF	.B	.T	
EN	実行条件。省略可能。省略時、常時(毎スキャン)実行。	—	—	—	—	—	—	.B	—	—
In※1※3	バイト分離するデータの格納先を指定。	.U	.U	.D	.D	—	—	.B	—	—
InOut※1※3	バイト分離したデータを格納する先頭を指定。	型はInと同じ。					—	—	—	—
Count※2	分離するバイトデータの個数を指定。	.U	.U	.U	.U	—	—	—	—	—
R	戻り値はなし。	—	—	—	—	—	—	—	—	—

※1　ビットデータを指定した場合、Inは連続した16/32ビット、InOutは連続した32/64ビットで演算。R002などチャンネルの先頭以外を指定した場合、次のチャンネルにまたがって実行。
ワードデータを指定した場合、Inは連続した1/2ワードを扱います。InOutは2/4ワードで演算。
※2　Inで指定したデータのサフィックスが.Uもしくは.Bの場合0～2、.Dの場合0～4の範囲で指定。
※3　Zは指定不可。

●動作説明

実行条件が成り立つ(TRUE)とき、Inで指定されたデータをCountで指定された個数にバイト(8ビット)単位で分離し、InOutへ格納します。
8ビットより上位には、0が格納されます。
「演算フラグについて(CR2009～CR2012)」(1-8ページ)
詳細については、『命令語リファレンスマニュアル』「データ変換命令」を参照してください。

●書式例

DISB (uiValue, auiDst[0], uiCount);

ポイント　**Countが0の場合、関数は動作しません。**

UNIB　バイト結合

ユニB
UNIB ([EN], Src, InOut, Count);

引数/戻り値	説明	認識する型								定数
		.U	.S	.D	.L	.F	.DF	.B	.T	
EN	実行条件。省略可能。省略時、常時(毎スキャン)実行。	—	—	—	—	—	—	.B	—	—
Src※1※3	バイト結合するデータの格納先を指定。	.U	.U	.D	.D	—	—	.B	—	—
InOut※1※3	バイト結合したデータを格納する先頭を指定。	型はSrcと同じ。					—	—	—	—
Count※2	結合するバイトデータの個数を指定。	.U	.U	.U	.U	—	—	—	—	—
R	戻り値はなし。	—	—	—	—	—	—	—	—	—

※1　ビットデータを指定した場合、Srcは連続した32/64ビット、InOutは連続した16/32ビットで演算。R002などチャンネルの先頭以外を指定した場合、次のチャンネルにまたがって実行。
ワードデータを指定した場合、Srcは連続した2/4ワードを扱います。InOutは1/2ワードで演算。
※2　Srcで指定したデータのサフィックスが.Uもしくは.Bの場合0～2、.Dの場合0～4の範囲で指定。
※3　Zは指定不可。

●動作説明

実行条件が成り立つ(TRUE)とき、**Src**を先頭とする結合データの下位8ビットを**Count**だけBINデータに結合し、**InOut**へ格納します。

「演算フラグについて（CR2009～CR2012）」（1-8ページ）
詳細については、『命令語リファレンスマニュアル』「データ変換命令」を参照してください。

●書式例

UNIB (auiValue[0], uiDst, uiCount);

ポイント　**Countが0の場合、関数は動作しません。**

SWAP　上位・下位バイト入れ替え

上位・下位バイト入れ替え
[R:=] SWAP (InOut);

引数/戻り値	説明	認識する型								定数
		.U	.S	.D	.L	.F	.DF	.B	.T	
InOut	入替対象、または入替対象データを指定。	.U	.U	.D	.D	–	–	–	–	○
R※1	演算結果が格納される戻り値。省略可能。省略時、内部レジスタに格納。	.U	.S	.D	.L	.F	.DF※2	–	–	–

※1　Rに演算対象と異なる型を指定した場合、変換時に自動的に型変換を実行。
※2　デバイスZを指定した場合、「.DF」サフィックスは指定不可。

●動作説明

InOutに格納された16ビットデータの、上位バイト(上位8ビット)と下位バイト(下位8ビット)の値を入れ替え、結果を**Rに格納**します。

詳細については、『命令語リファレンスマニュアル』「データ変換命令」を参照してください。

●書式例

uiValue := **SWAP** (uiValue);

BSWAP　ブロックスワップ

ブロックスワップ
BSWAP ([EN], InOut, Count);

引数/戻り値	説明	認識する型								定数
		.U	.S	.D	.L	.F	.DF	.B	.T	
EN	実行条件。省略可能。省略時、常時（毎スキャン）実行。	–	–	–	–	–	–	.B	–	–
InOut	入れ替え対象を指定。	.U	.U	.U	.U	–	–	.B	–	–
Count	入れ替える数を指定。	.U	.U	.U	.U	–	–	–	–	○
R	戻り値はなし。	–	–	–	–	–	–	–	–	–

●動作説明

実行条件が成り立つ(TRUE)とき、**InOut**を先頭とする**Count**分のデータの上位バイト(上位8バイト)と下位バイト(下位8バイト)を入れ替えます。

詳細については、『命令語リファレンスマニュアル』「データ変換命令」を参照してください。

●書式例

BSWAP(udiValue, INT#4);

ポイント　**Count=0の場合、関数は実行されません。**

DECO　8ビットデコード

8ビットデコード
DECO ([EN], Src, Dst, Count);

引数/戻り値	説明	認識する型								定数
		.U	.S	.D	.L	.F	.DF	.B	.T	
EN	実行条件。省略可能。省略時、常時(毎スキャン)実行。	—	—	—	—	—	—	.B	—	—
Src	変換する範囲の先頭を指定。	.U	.U	.U	.U	—	—	—	—	—
Dst※1※2	デコード結果の格納先を指定。	—	—	—	—		—	.B	—	—
Count	変換する範囲数(ビット数:n)を指定(1～8)。	.U	.U	.U	.U	—	—	—	—	○
R	戻り値はなし。	—	—	—	—	—	—	—	—	—

※1　ワードデータを指定した場合、連続する2^n個のワードデータの最下位ビットはデコード結果の格納先。(最下位ビット以外は変化なし)
※2　Zは使用不可。

●動作説明

実行条件が成り立つ(TRUE)とき、**Src**で指定されたビットデータを先頭とする、**Count**で指定された個数(ビット数:n)分の状態を数値化(0～255)します。**Dst**を先頭として連続する2^n 個のビットデータのうちで、先頭番号+数値化した値(番目)のビットをONし、その他のビットはすべてOFFします。
Srcがワードデータのとき、**Count**は数値化するビット数を示し、また格納先がワードデータのときは、指定されたワードデータを先頭として連続する2^n 個のワードデータのうちで、先頭番号+数値化した値(番目)のワードデータの最下位ビットをONし、その他のワードデータの最下位ビットはすべてOFFします。(最下位ビット以外は変化しません)

「演算フラグについて(CR2009～CR2012)」(1-8ページ)
詳細については、『命令語リファレンスマニュアル』「データ変換命令」を参照してください。

●書式例

DECO(uiValue, abResult[0], uiCount);

ポイント　**Count=0の場合、関数は実行されません。**

ENCO　8ビットエンコード

8ビットエンコード
[R:=] ENCO (In, Count);

引数/戻り値	説明	認識する型								定数
		.U	.S	.D	.L	.F	.DF	.B	.T	
In※1※2	変換する範囲の先頭を指定。	–	–	–	–	–	–	.B	–	–
Count	変換する範囲数(ビット数:n)を指定。(1～256)	.U	.U	.U	.U	–	–	–	–	○
R※2	演算結果が格納される戻り値。省略可能。省略時、内部レジスタに格納。	.U	.S	.D	.L	.F	.DF※4	–	–	–

※1　ワードデータを指定した場合、エンコードする対象は連続するCount個のワードデータの最下位ビット。
※2　Zは使用不可。
※3　Rに.U以外の型を指定した場合、変換時に自動的に型変換を実行。
※4　デバイスZを指定した場合、「.DF」サフィックスは指定不可。

●動作説明

Inで指定されたビットデータを先頭とする、Countで指定された個数(ビット数:n)分の状態のうちで、ONしているビットの最小のデバイスの位置(0～255)を、16ビットデータとしてRに格納します。
Inがワードデータのとき、Countはワード数を示し、変換対象範囲数分のワードデータのうちで、最下位ビットがONしているワードの最小のビットの位置(0～255)を、16ビットデータとしてRに格納します。

「演算フラグについて(CR2009～CR2012)」(1-8ページ)
詳細については、『命令語リファレンスマニュアル』「データ変換命令」を参照してください。

●書式例

ENCO (abData[0], INT#8);

ポイント　**Count=0の場合、関数は実行されません。**

CPMSET　CIPメッセージ作成

メッセージ作成
CPMSET ([EN], Src, Param, Dst);

引数/戻り値	説明	認識する型								定数
		.U	.S	.D	.L	.F	.DF	.B	.T	
EN	実行条件。省略可能。省略時、常時(毎スキャン)実行。	–	–	–	–	–	–	.B	–	–
Src※1	追加するデータの先頭を指定。	.U	.S	.D	.L	.F	.DF	.B	.T	–
Param	追加する送信データのデータ型やデータサイズ(バイト単位)、または格納先を指定。※1	.U	.U	.U	.U	–	–	–	–	○
Dst※1	作成する送信データの先頭を指定。	.U	.S	.D	.L	.F	.DF	.B	.T	–
R	戻り値はなし。	–	–	–	–	–	–	–	–	–

※1　ビットデータを指定する場合は、チャンネルの先頭を指定。

●動作説明

実行条件が成り立つ(TRUE)とき、Srcに格納されているデータをParamの上位4ビットで指定されたデータ型で、下位12ビットで指定されたデータサイズ(バイト単位)分、Dstに格納されたデータサイズ(バイト単位)以降の領域に追加します。

最初にデータを追加する場合は、**Dst**に0を格納してから、関数を実行します。
CPMSET関数は、CIPのデータ型の文字列データ、数値データを作成するための関数です。
CIPのデータ型については、『KV EtherNet/IP機能ユーザーズマニュアル』を参照してください。

上位4ビット	下位12ビット
0:文字列以外	1～4095(001H～FFFH):データサイズを指定
1:文字列	0～1999(000H～7CFH):CIPの文字列型データに変換する文字数を指定 0を指定した場合はNUL(00H)までを変換します。 指定した文字数の途中にNUL(00H)が含まれる場合は、NUL(00H)以降の文字には、00Hを追加します。
2以上:システム予約	–

詳細については、『命令語リファレンスマニュアル』「データ変換命令」を参照してください。

●書式例

CPMSET (auiValue[0], uiParam, auiDst[0]);

CPMGET　CIPメッセージ取得

メッセージ取得
CPMGET ([EN], Src, Param, Offset, Dst);

引数/戻り値	説明	認識する型								定数
		.U	.S	.D	.L	.F	.DF	.B	.T	
EN	実行条件。省略可能。省略時、常時(毎スキャン)実行。	–	–	–	–	–	–	.B	–	–
Src※1	受信データの格納先の先頭を指定。	.U	.S	.D	.L	.F	.DF	.B	.T	–
Param	取得する受信データのデータ型やデータサイズ(バイト単位)、または格納先を指定。※1	.U	.U	.U	.U	–	–	–	–	○
Offset※1	受信データの取得位置を指定する先頭からのバイトオフセット、または格納先を指定。	.U	.S	.D	.L	–	–	–	–	–
Dst※1	取得データを格納する対象の先頭を指定。	.U	.S	.D	.L	.F	.DF	.B	.T	–
R	戻り値はなし。	–	–	–	–	–	–	–	–	–

※1　ビットデータを指定する場合は、チャンネルの先頭を指定。

●動作説明

実行条件が成り立つ(TRUE)とき、**Src**に格納されている受信データの**Offset**で指定されたオフセット位置以降のデータを、**Param**の上位4ビットで指定されたデータ型で、**Param**の下位12ビットで指定されたデータサイズ(バイト単位)分のデータを、**Dst**に格納します。
データを格納後、**Offset**が更新されます。
CPMGET関数は、受信したCIPのデータ型の文字列データ、数値データを取得するための関数です。
CIPのデータ型については、『KV EtherNet/IP機能ユーザーズマニュアル』を参照してください。

上位4ビット	下位12ビット
0:文字列以外	1～4095(001H～FFFH):データサイズを指定
1:文字列	0～1999(000H～7CFH):CIPの文字列型データに変換する文字数を指定 0を指定した場合はNUL(00H)までを変換します。 指定した文字数の途中にNUL(00H)が含まれる場合は、NUL(00H)以降の文字には、00Hを追加します。
2以上:システム予約	–

詳細については、『命令語リファレンスマニュアル』「データ変換命令」を参照してください。

●書式例

CPMGET(auiValue[0], uiParam, uiOffset, auiDst[0]);

*_TO_** データ型変換（データ型指定）

[R:=] *_TO_** (In);
データ型変換

引数/戻り値	説明	認識する型								定数
		.U	.S	.D	.L	.F	.DF	.B	.T	
In※2	変換対象データを指定。	*で指定したデータ型						—	—	○
R※1※2	演算結果が格納される戻り値。省略可能。省略時、内部レジスタに格納。	.U	.S	.D	.L	.F	.DF	—	—	—

※1　Rに**で指定したデータ型と異なる型を指定した場合、変換時に自動的に型変換を実行。
※2　デバイスZを指定した場合、「.DF」サフィックスは指定不可。

●動作説明

Inに格納されたデータ型*のデータを、データ型**に変換し、結果をRに格納します。

*、および**は、UINT/INT/UDINT/DINT/REAL/LREALのいずれか。

In \ R	UINT	INT	UDINT	DINT	REAL	LREAL
UINT		D1	A	A	A	A
INT	D1		D1	A	A	A
UDINT	D2	D2		D1	B	A
DINT	D2	D2	D1		B	A
REAL	C	C	C	C		B
LREAL	C	C	C	C	B	

凡例：
A:値をそのまま変換
B:Inに格納されたデータがデータ型**の有効桁数の範囲を超えた場合、精度が劣化します。
例）（DINT）1234567890→（REAL）1.234567E09
例）（LREAL）3.1415926535→（REAL）3.1415926
C:小数点以下の値を下記のルールに従って整数にします。

小数点以下の値	処理	例
0.5より小さい	切り捨て	REAL_TO_INT(1.4)→1 REAL_TO_INT(-1.4)→-1
0.5	最も近い偶数整数にする	REAL_TO_INT(1.5) → 2 REAL_TO_INT(2.5) → 2 REAL_TO_INT(-1.5) → -2 REAL_TO_INT(-2.5) → -2
0.5より大きい	切り上げ	REAL_TO_INT(1.6)→2 REAL_TO_INT(-1.6)→-2

変換後の値がデータ型の範囲外の場合、Rは不定値です。
D1：Inに格納されたデータがデータ型**の範囲外の場合、変換せずにそのまま値を格納します。
D2：Inに格納されたデータの下位16ビットを対象として変換します。

●書式例

diValue := **REAL_TO_DINT** (rValue);

「演算フラグについて（CR2009～CR2012）」（1-8ページ）
詳細については『命令語リファレンスマニュアル』「データ変換命令」を参照してください。

TO_** データ型変換（多重定義）

[R:=] TO_ (In);**
（データ型変換）

引数/戻り値	説明	認識する型								定数
		.U	.S	.D	.L	.F	.DF	.B	.T	
In※2	変換対象データを指定。	.U	.S	.D	.L	.F	.DF	—	—	○
R※1※2	演算結果が格納される戻り値。省略可能。省略時、内部レジスタに格納。	.U	.S	.D	.L	.F	.DF	—	—	—

※1　Rに**で指定したデータ型と異なる型を指定した場合、変換時に自動的に型変換を実行。
※2　デバイスZを指定した場合、「.DF」サフィックスは指定不可。

●動作説明

Inに格納されたデータを、データ型**に変換し、結果をRに格納します。
**は、UINT/INT/UDINT/DINT/REAL/LREALのいずれか。

変換時の動作については、「*_TO_**」（7-39ページ）

●書式例

diValue := **TO_DINT** (rValue);

「演算フラグについて（CR2009～CR2012）」（1-8ページ）
詳細については『命令語リファレンスマニュアル』「データ変換命令」を参照してください。

*_TRUNC_** 小数切捨（データ型指定）

[R:=] *_TRUNC_ (In);**
（小数切捨）

引数/戻り値	説明	認識する型								定数
		.U	.S	.D	.L	.F	.DF	.B	.T	
In※1	変換対象データを指定。	*で指定したデータ型						—	—	○
R※1※2	演算結果が格納される戻り値。省略可能。省略時、内部レジスタに格納。	.U	.S	.D	.L	.F	.DF	—	—	—

※1　Rに**で指定したデータ型と異なる型を指定した場合、変換時に自動的に型変換を実行。
※2　デバイスZを指定した場合、「.DF」サフィックスは指定不可。

●動作説明

Inに格納されたデータ型*の小数点以下を切り捨てて、データ型**の整数データに変換し、結果を**R**に格納します。
*はREAL/LREALのいずれか。
**は、UINT/INT/UDINT/DINTのいずれか。
Inに格納されたデータが浮動小数点型実数の範囲外の場合、**R**は不定値です。
Inに格納されたデータがデータ型**の範囲外の場合、変換せずにそのまま値を格納します。

●書式例

diValue := **REAL_TRUNC_DINT** (rValue);

「演算フラグについて（CR2009～CR2012）」（1-8ページ）
詳細については『命令語リファレンスマニュアル』「データ変換命令」を参照してください。

TRUNC_** 小数切捨（多重定義）

小数切捨
[R:=] TRUNC_ (In);**

引数/戻り値	説明	認識する型								定数
		.U	.S	.D	.L	.F	.DF	.B	.T	
In※1	変換対象データを指定。	.F	.F	.F	.F	.F	.DF	—	—	○
R※1※2	演算結果が格納される戻り値。省略可能。省略時、内部レジスタに格納。	.U	.S	.D	.L	.F	.DF	—	—	—

※1　**R**に**で指定したデータ型と異なる型を指定した場合、変換時に自動的に型変換を実行。
※2　デバイスZを指定した場合、「.DF」サフィックスは指定不可。

●動作説明

Inに格納されたデータ型*の小数点以下を切り捨てて、データ型**の整数データに変換し、結果を**R**に格納します。
**は、UINT/INT/UDINT/DINTのいずれか。
Inに格納されたデータが浮動小数点型実数の範囲外の場合、**R**は不定値です。
Inに格納されたデータがデータ型**の範囲外の場合、変換せずにそのまま値を格納します。

変換時の動作については、「*_TRUNC_**」（7-40ページ）

●書式例

diValue := **TRUNC_DINT** (rValue);

「演算フラグについて（CR2009～CR2012）」（1-8ページ）
詳細については『命令語リファレンスマニュアル』「データ変換命令」を参照してください。

*_TO_BCD_** BCDデータ型変換(データ型指定)

[R:=] *_TO_BCD_** (In);
データ型変換

引数/戻り値	説明	認識する型								定数
		.U	.S	.D	.L	.F	.DF	.B	.T	
In	変換対象データを指定。	*で指定したデータ型				—	—	—	—	○
R※1	演算結果が格納される戻り値。省略可能。省略時、内部レジスタに格納。	.U	.S	.D	.L	.F	.DF	—	—	—

※1　Rに**で指定したデータ型と異なる型を指定した場合、変換時に自動的に型変換を実行。

●動作説明

Inに格納されたデータ型*のデータを、データ型**のBCDデータに変換し、結果をRに格納します。
*および**は、UINT/INT/UDINT/DINTのいずれか。
Inに格納されたデータが範囲外の場合、Rは不定値です。

●書式例

diValue := **INT_TO_BCD_DINT** (iValue);

「演算フラグについて(CR2009～CR2012)」(1-8ページ)
詳細については『命令語リファレンスマニュアル』「データ変換命令」を参照してください。

TO_BCD_** BCDデータ型変換(多重定義)

[R:=] TO_BCD_** (In);
データ型変換

引数/戻り値	説明	認識する型								定数
		.U	.S	.D	.L	.F	.DF	.B	.T	
In	変換対象データを指定。	.U	.S	.D	.L	—	—	—	—	○
R※1	演算結果が格納される戻り値。省略可能。省略時、内部レジスタに格納。	.U	.S	.D	.L	.F	.DF	—	—	—

※1　Rに**で指定したデータ型と異なる型を指定した場合、変換時に自動的に型変換を実行。

●動作説明

Inに格納されたデータを、データ型**のBCDデータに変換し、結果をRに格納します。
Inのデータ型はUINT/INT/UDINT/DINTのいずれか。
**は、UINT/INT/UDINT/DINTのいずれか。

変換時の動作については、「*_TO_BCD_**関数」(7-42ページ)

●書式例

diValue := **TO_BCD_DINT** (iValue);

「演算フラグについて(CR2009～CR2012)」(1-8ページ)
詳細については『命令語リファレンスマニュアル』「データ変換命令」を参照してください。

*_BCD_TO_** データ型変換（データ型指定）

[R:=] *_BCD_TO_ (In);**
（データ型変換）

引数/戻り値	説明	認識する型								定数
		.U	.S	.D	.L	.F	.DF	.B	.T	
In	変換対象データを指定。	*で指定したデータ型				—	—	—	—	○
R※1	演算結果が格納される戻り値。省略可能。省略時、内部レジスタに格納。	.U	.S	.D	.L	.F	.DF	—	—	—

※1　Rに**で指定したデータ型と異なる型を指定した場合、変換時に自動的に型変換を実行。

●動作説明

Inに格納されたデータ型*のBCDデータを、データ型**の整数データに変換し、結果をRに格納します。
*および**は、UINT/INT/UDINT/DINTのいずれか。
Inに格納されたBCDデータが範囲外の場合、Rは不定値です。

●書式例

iValue := **DINT_BCD_TO_INT** (diValue);

「演算フラグについて（CR2009～CR2012）」（1-8ページ）
詳細については『命令語リファレンスマニュアル』「データ変換命令」を参照してください。

BCD_TO_** データ型変換（多重定義）

[R:=] BCD_TO_ (In);**
（データ型変換）

引数/戻り値	説明	認識する型								定数
		.U	.S	.D	.L	.F	.DF	.B	.T	
In	変換対象データを指定。	.U	.S	.D	.L	—	—	—	—	○
R※1	演算結果が格納される戻り値。省略可能。省略時、内部レジスタに格納。	.U	.S	.D	.L	.F	.DF	—	—	—

※1　Rに**で指定したデータ型と異なる型を指定した場合、変換時に自動的に型変換を実行。

●動作説明

Inに格納されたBCDデータを、データ型**の整数データに変換し、結果をRに格納します。
Inのデータ型はUINT/INT/UDINT/DINTのいずれか。
**は、UINT/INT/UDINT/DINTのいずれか。
変換時の動作については、「*_BCD_TO_**関数」（7-43ページ）

●書式例

diValue := **BCD_TO_DINT** (iValue);

「演算フラグについて（CR2009～CR2012）」（1-8ページ）
詳細については『命令語リファレンスマニュアル』「データ変換命令」を参照してください。

浮動小数点関数

FLOAT　BINデータ→単精度浮動小数点型実数データ変換

単精度浮動小数変換
[R:=] FLOAT (In);

引数/戻り値	説明	認識する型								定数
		.U	.S	.D	.L	.F	.DF	.B	.T	
In	変換対象、または変換対象データを指定。	.U	.S	.D	.L	—	.DF※2	—	—	○
R※1	演算結果が格納される戻り値。省略可能。省略時、内部レジスタに格納。	.U	.S	.D	.L	.F	.DF※2	—	—	—

※1　Rに.F以外の型を指定した場合、変換時に自動的に型変換を実行。
※2　デバイスZを指定した場合、「.DF」サフィックスは指定不可。

●動作説明

Inに格納されたバイナリデータ(16ビット/32ビット(符号無し/符号付き)、倍精度浮動小数点型実数)を、単精度浮動小数点型実数データに変換し、結果をRに格納します。

詳細については、『命令語リファレンスマニュアル』「浮動小数命令」を参照してください。

●書式例

rValue := **FLOAT** (uiValue);

ポイント　**Inの整数値が－16777216～16777215(24ビットBIN)の範囲を超えた場合、25ビット目以降は丸められます。**

DFLOAT　BINデータ→倍精度浮動小数点実数データ変換

倍精度浮動小数変換
[R:=] DFLOAT (In);

引数/戻り値	説明	認識する型								定数
		.U	.S	.D	.L	.F	.DF	.B	.T	
In	変換対象のデータを格納した対象、または変換データを指定。	.U	.S	.D	.L	.F	—	—	—	○
R※1	演算結果が格納される戻り値。省略可能。省略時、64ビット内部レジスタに格納。	.U	.S	.D	.L	.F	.DF※2	—	—	—

※1　Rに.DF以外の型を指定した場合、変換時に自動的に型変換を実行。
※2　デバイスZを指定した場合、「.DF」サフィックスは指定不可。

●動作説明

Inに格納されたバイナリデータ(16ビット/32ビット(符号無し/符号付き)、単精度浮動小数点型実数)を、倍精度浮動小数点型実数データに変換し、結果をRに格納します。

詳細については、『命令語リファレンスマニュアル』「浮動小数命令」を参照してください。

●書式例

lValue := **DFLOAT** (uiValue);

INTG　単精度/倍精度浮動小数点型実数データ→BINデータ変換

[R:=] INTG (In);
整数変換

引数/戻り値	説明	認識する型								定数
		.U	.S	.D	.L	.F	.DF	.B	.T	
In	変換対象、または変換対象データを指定。	—	—	—	—	.F	.DF※3	—	—	○
R※1※2	演算結果が格納される戻り値。省略可能。省略時、内部レジスタに格納。	.U	.S	.D	.L	.F	.DF※3	—	—	—

※1　C, T, Zは使用不可。
※2　Rに.L以外の型を指定した場合、変換時に自動的に型変換を実行。
※3　デバイスZを指定した場合、「.DF」サフィックスは指定不可。

ポイント
- 単精度浮動小数点型実数の範囲は、以下のとおりです。

  −3.4E+38 ≦ N ≦ −1.4E−45
  N = 0
  1.4E−45 ≦ N ≦ 3.4E+38
  （有効桁数:約7桁）

- 倍精度浮動小数点型実数の範囲は、以下のとおりです。

  −1.79E+308 ≦ N ≦ −2.23E−308
  N = 0
  2.23E−308 ≦ N ≦ 1.79E+308

  (有効桁数：約16桁)

●動作説明

Inに格納された単精度/倍精度浮動小数点型実数データを、32ビットバイナリデータ(符号付き)に変換し、結果をRに格納します。

詳細については、『命令語リファレンスマニュアル』「浮動小数命令」を参照してください。

●書式例

uiValue := **INTG** (rValue);

ポイント
- Inが単精度浮動小数点型実数の範囲外の場合、また、変換したデータが16/32ビット符号付きバイナリの範囲外の場合、Inのビットパターンが R に格納されます。
- Inが倍精度浮動小数点型実数の範囲外の場合、また、変換したデータが16/32ビット符号付きバイナリの範囲外の場合、Inの下位32ビットのビットパターンがRに格納されます。

INT　単精度/倍精度浮動小数点型実数データ→BINデータ変換

整数変換
[R:=] INT (In);

引数/戻り値	説明	認識する型								定数
		.U	.S	.D	.L	.F	.DF	.B	.T	
In	変換対象、または変換対象データを指定。	—	—	—	—	.F	.DF※3	—	—	○
R※1※2	演算結果が格納される戻り値。省略可能。省略時、内部レジスタに格納。	.U	.S	.D	.L	.F	.DF※3	—	—	—

※1　C, T, Zは使用不可。
※2　Rに.L以外の型を指定した場合、変換時に自動的に型変換を実行。
※3　デバイスZを指定した場合、「.DF」サフィックスは指定不可。

ポイント
- **単精度浮動小数点型実数の範囲、および倍精度浮動小数点型実数の範囲は、「INTG関数」(7-45ページ)を参照。**

●動作説明

本関数は「INTG関数」と関数名が異なるだけで、動作等に違いはありません。

DISF　単精度浮動小数点型実数の仮数・指数分離

ディスF
DISF ([EN], In, InOut1, InOut2);

引数/戻り値	説明	認識する型								定数
		.U	.S	.D	.L	.F	.DF	.B	.T	
EN	実行条件。省略可能。省略時、常時(毎スキャン)実行。	—	—	—	—	—	—	.B	—	—
In※1※3	仮数・指数分離する単精度浮動小数点型実数の対象を指定。	—	—	—	—	.F	—	—	—	○※2
InOut1※1	分離した仮数部を格納する対象を指定。	—	—	.L	.L	—	—	—	—	—
InOut2	分離した指数部を格納する対象を指定。	.S	.S	—	—	—	—	—	—	—
R	戻り値はなし。	—	—	—	—	—	—	—	—	—

※1　ビットデータを指定した場合、連続する32ビットで演算。R002などチャンネルの先頭以外を指定した場合、次のチャンネルにまたがって実行。
※2　単精度浮動小数点型実数の範囲で定数を設定。
※3　C, T, Zは使用不可。

ポイント
単精度浮動小数点型実数の範囲は、「INTG関数」(7-45ページ)を参照。

●動作説明

実行条件が成り立つ(TRUE)とき、Inで指定された32ビット単精度浮動小数点型実数の仮数部と指数部を分離して、仮数部を指定されたInOut1へ32ビットバイナリデータ(符号付き)として格納し、指数部を指定されたInOut2へ16ビットバイナリデータ(符号付き)として格納します。

「演算フラグについて(CR2009～CR2012)」(1-8ページ)
詳細については、『命令語リファレンスマニュアル』「浮動小数命令」を参照してください。

●書式例

DISF (bEnable, rValue, diValue, iValue);

ポイント
- **DISF関数では、単精度浮動小数点型実数(2進数)から、仮数部(10進数)と指数部(10進数)に変換しているため、誤差が生じる場合があります。**
- **分離するデータが単精度浮動小数点型実数の範囲外の場合、関数は実行されません。**

UNIF　仮数・指数の単精度浮動小数点型実数への結合

[R:=] UNIF (In1, In2);

引数/戻り値	説明	認識する型								定数
		.U	.S	.D	.L	.F	.DF	.B	.T	
In1※1	結合する仮数部を格納した対象を指定。	—	—	.L	.L	—	—	—	—	—
In2	結合する指数部を格納した対象を指定。	.S	.S	.S	.S	—	—	—	—	—
R※2	演算結果が格納される戻り値。省略可能。省略時、内部レジスタに格納。	.U	.S	.D	.L	.F	.DF※3	—	—	—

※1　ビットデータを指定した場合、連続する32ビットで演算。R002などチャンネルの先頭以外を指定した場合、次のチャンネルにまたがって実行。
※2　Rに.F以外の型を指定した場合、変換時に自動的に型変換を実行。
※3　デバイスZを指定した場合、「.DF」サフィックスは指定不可。

●動作説明

In1で指定された32ビットバイナリデータ(符号付き)を仮数とし、In2で指定された16ビットバイナリデータ(符号付き)を指数として結合した、32ビット単精度浮動小数点型実数値をRに格納します。

「演算フラグについて（CR2009～CR2012）」（1-8ページ）
詳細については、『命令語リファレンスマニュアル』「浮動小数命令」を参照してください。

●書式例

rValue := **UNIF** (diValue, iValue);

ポイント
- **UNIF関数では、仮数部(10進数)と指数部(10進数)から、単精度浮動小数点型実数(2進数)に変換しているため、誤差が生じる場合があります。**
- **演算結果が単精度浮動小数点型実数の範囲外の場合、Rには0が格納されます。**

EXP　指数関数演算

エクスポーネント
[R:=] EXP (In);

引数/戻り値	説明	認識する型								定数
		.U	.S	.D	.L	.F	.DF	.B	.T	
In	演算対象、または演算対象データを指定。	.F	.F	.F	.F	.F	.DF※2	—	—	○
R※1	演算結果が格納される戻り値。省略可能。省略時、内部レジスタに格納。	.U	.S	.D	.L	.F	.DF※2	—	—	—

※1　EXP関数は、引数がU/S/D/L/Fの場合.Fの型を使用、DFの場合.DFの型を使用。Rに関数が持つ型と異なる型を指定した場合、変換時に自動的に型変換を実行。
※2　デバイスZを指定した場合、「.DF」サフィックスは指定不可。

ポイント
- **単精度浮動小数点型実数の範囲、および倍精度浮動小数点型実数の範囲は、「INTG関数」(7-45ページ)を参照。**

●動作説明

- **In:変換対象のサフィックスが.U/.S/.D/.L/.Fの場合**
  [In]に格納された単精度浮動小数点型実数を指数とした指数関数値を求めます。
  結果は32ビット単精度浮動小数点型実数値となり、[R]に格納されます。
- **In:変換対象のサフィックスが.DFの場合**
  [In]に格納された倍精度浮動小数点型実数を指数とした指数関数値を求めます。
  結果は64ビット倍精度浮動小数点型実数値となり、[R]に格納されます。

「演算フラグについて(CR2009～CR2012)」(1-8ページ)
詳細については、『命令語リファレンスマニュアル』「浮動小数命令」を参照してください。

●書式例

rValue := **EXP** (rValue1);

ポイント
演算する値、または演算結果が単精度浮動小数点型実数の範囲外の場合、RにはInの値が格納されます。

EXPT　べき乗

べき乗
[R:=] EXPT (In, Pwr);

引数/戻り値	説明	認識する型								定数
		.U	.S	.D	.L	.F	.DF	.B	.T	
In	演算対象、または演算対象データを指定。	.F	.F	.F	.F	.F	.DF※2	—	—	○
Pwr	演算対象、または演算対象データを指定。	.F	.F	.F	.F	.F	.DF※2	—	—	○
R※1	演算結果が格納される戻り値。省略可能。省略時、内部レジスタに格納。	.U	.S	.D	.L	.F	.DF※2	—	—	—

※1　EXPT関数は、引数がU/S/D/L/Fの場合.Fの型を使用、DFの場合.DFの型を使用。Rに関数が持つ型と異なる型を指定した場合、変換時に自動的に型変換を実行。
※2　デバイスZを指定した場合、「.DF」サフィックスは指定不可。

●動作説明

Inに格納されたデータに**Pwr**のべき乗を実施し、結果を**R**に格納します。

ポイント
- **In= 0、Pwr= 0に設定した場合、Rに1を格納します。**

●書式例

rValue1 := **EXPT** (rValue2, rValue3);

「演算フラグについて（CR2009～CR2012）」（1-8ページ）
詳細については『命令語リファレンスマニュアル』「浮動小数点命令」を参照してください。

LN　自然対数演算

[R:=] LN (In);　（$\log_e$）

引数/戻り値	説明	認識する型								定数
		.U	.S	.D	.L	.F	.DF	.B	.T	
In	演算対象、または演算対象データを指定。	.F	.F	.F	.F	.F	.DF※2	—	—	○
R※1	演算結果が格納される戻り値。省略可能。省略時、内部レジスタに格納。	.U	.S	.D	.L	.F	.DF※2	—	—	—

※1　LN関数は、引数がU/S/D/L/Fの場合.Fの型を使用、DFの場合.DFの型を使用。**R**に関数が持つ型と異なる型を指定した場合、変換時に自動的に型変換を実行。
※2　デバイスZを指定した場合、「.DF」サフィックスは指定不可。

ポイント
- **単精度浮動小数点型実数の範囲、および倍精度浮動小数点型実数の範囲は、「INTG関数」（7-45ページ）を参照。**

●動作説明

- **In：変換対象のサフィックスが.U/.S/.D/.L/.Fの場合**
  Inに格納された単精度浮動小数点型実数の自然対数を求めます。
  結果は32ビット単精度浮動小数点型実数値となり、**R**に格納されます。
- **In：変換対象のサフィックスが.DFの場合**
  Inに格納された倍精度浮動小数点型実数の自然対数を求めます。
  結果は64ビット倍精度浮動小数点型実数値となり、**R**に格納されます。

「演算フラグについて（CR2009～CR2012）」（1-8ページ）
詳細については、『命令語リファレンスマニュアル』「浮動小数命令」を参照してください。

●書式例

rValue := **LN** (rValue1);

ポイント
演算する値が0か負、または演算結果が浮動小数点型実数の範囲外の場合、RにはInの値が格納されます。

LOG　常用対数演算

[R:=] LOG (In); (log10)

引数/戻り値	説明	認識する型								定数
		.U	.S	.D	.L	.F	.DF	.B	.T	
In	演算対象、または演算対象データを指定。	.F	.F	.F	.F	.F	.DF	—	—	○
R※1	演算結果が格納される戻り値。省略可能。省略時、内部レジスタに格納。	.U	.S	.D	.L	.F	.DF※2	—	—	—

※1　LOG関数は、引数がU/S/D/L/Fの場合.Fの型を使用、DFの場合.DFの型を使用。Rに関数が持つ型と異なる型を指定した場合、変換時に自動的に型変換を実行。
※2　デバイスZを指定した場合、「.DF」サフィックスは指定不可。

ポイント
- **単精度浮動小数点型実数の範囲、および倍精度浮動小数点型実数の範囲は、「INTG関数」(7-45ページ)を参照。**

●動作説明

- **In：変換対象のサフィックスが.U/.S/.D/.L/.Fの場合**
  In格納された単精度浮動小数点型実数の常用対数を求めます。
  結果は32ビット単精度浮動小数点型実数値となり、R格納されます。
- **In：変換対象のサフィックスが.DFの場合**
  Inに格納された倍精度浮動小数点型実数の常用対数を求めます。
  結果は64ビット倍精度浮動小数点型実数値となり、Rに格納されます。

「演算フラグについて(CR2009～CR2012)」(1-8ページ)
詳細については、『命令語リファレンスマニュアル』「浮動小数命令」を参照してください。

●書式例

```
rValue  := LOG(rValue1);
```

RAD　度(°)→ラジアン単位変換

[R:=] RAD (In); (ラジアン)

引数/戻り値	説明	認識する型								定数
		.U	.S	.D	.L	.F	.DF	.B	.T	
In	変換対象、または変換対象データを指定。	.F	.F	.F	.F	.F	.DF※2	—	—	○
R※1	演算結果が格納される戻り値。省略可能。省略時、内部レジスタに格納。	.U	.S	.D	.L	.F	.DF※2	—	—	—

※1　RAD関数は、引数がU/S/D/L/Fの場合.Fの型を使用、DFの場合.DFの型を使用。Rに関数が持つ型と異なる型を指定した場合、変換時に自動的に型変換を実行。
※2　デバイスZを指定した場合、「.DF」サフィックスは指定不可。

ポイント
- **単精度浮動小数点型実数の範囲、および倍精度浮動小数点型実数の範囲は、「INTG関数」(7-45ページ)を参照。**

●動作説明

- **In:変換対象のサフィックスが.U/.S/.D/.L/.Fの場合**
  Inに格納された単精度浮動小数点型実数の角度(°単位)をラジアン単位に変換します。
  結果は32ビット単精度浮動小数点型実数値となり、Rに格納されます。
- **In:変換対象のサフィックスが.DFの場合**
  Inに格納された倍精度浮動小数点型実数の角度(°単位)をラジアン単位に変換します。
  結果は64ビット倍精度浮動小数点型実数値となり、Rに格納されます。

「演算フラグについて(CR2009～CR2012)」(1-8ページ)
詳細については、『命令語リファレンスマニュアル』「浮動小数命令」を参照してください。

参考　角度(°)と、(rad)は次のように計算されます。

$$1^\circ = \frac{\pi}{180} \fallingdotseq 0.01745\ (\mathrm{rad}),\quad 1\ (\mathrm{rad}) = \frac{180}{\pi} \fallingdotseq 57.23^\circ$$ です。

●書式例

rRadian := **RAD** (uiDegree);

ポイント　**演算する値が単精度浮動小数点型実数の範囲外の場合、RにはInの値が格納されます。**

DEG　ラジアン→度(°)単位変換

[R:=] DEG (In);
度(°)

引数/戻り値	説明	認識する型								定数
		.U	.S	.D	.L	.F	.DF	.B	.T	
In	変換対象、または変換対象データを指定。	.F	.F	.F	.F	.F	.DF※2	—	—	○
R※1	演算結果が格納される戻り値。省略可能。省略時、内部レジスタに格納。	.U	.S	.D	.L	.F	.DF※2	—	—	—

※1　DEG関数は、引数がU/S/D/L/Fの場合.Fの型を使用、DFの場合.DFの型を使用。Rに関数が持つ型と異なる型を指定した場合、変換時に自動的に型変換を実行。
※2　デバイスZを指定した場合、「.DF」サフィックスは指定不可。

ポイント
- **単精度浮動小数点型実数の範囲、および倍精度浮動小数点型実数の範囲は、「INTG関数」(7-45ページ)を参照。**

●動作説明

- **In:変換対象のサフィックスが.U/.S/.D/.L/.Fの場合**
  Inに格納された単精度浮動小数点型実数の角度(ラジアン単位)を度(°)単位に変換します。
  結果は32ビット単精度浮動小数点型実数値となり、Rに格納されます。
- **In:変換対象のサフィックスが.DFの場合**
  Inに格納された倍精度浮動小数点型実数の角度(ラジアン単位)を度(°)単位に変換します。
  結果は64ビット倍精度浮動小数点型実数値となり、Rに格納されます。

「演算フラグについて(CR2009～CR2012)」(1-8ページ)
詳細については、『命令語リファレンスマニュアル』「浮動小数命令」を参照してください。

参考　角度(°)と、(rad)は次のように計算されます。

$$1^\circ = \frac{\pi}{180} \fallingdotseq 0.01745\ (\mathrm{rad}),\quad 1\ (\mathrm{rad}) = \frac{180}{\pi} \fallingdotseq 57.23^\circ$$ です。

●書式例

iDegree := **DEG** (rRadian);

ポイント　**演算する値が単精度浮動小数点型実数の範囲外の場合、Rにはlnの値が格納されます。**

SIN　角度(ラジアン)→正弦値

[R:=] SIN (In);（サイン）

引数/戻り値	説明	認識する型								定数
		.U	.S	.D	.L	.F	.DF	.B	.T	
In	演算対象、または演算対象データを指定。	.F	.F	.F	.F	.F	.DF※2	—	—	○
R※1	演算結果が格納される戻り値。省略可能。省略時、内部レジスタに格納。	.U	.S	.D	.L	.F	.DF※2	—	—	—

※1　SIN関数は、引数がU/S/D/L/Fの場合.Fの型を使用、DFの場合.DFの型を使用。Rに関数が持つ型と異なる型を指定した場合、変換時に自動的に型変換を実行。
※2　デバイスZを指定した場合、「.DF」サフィックスは指定不可。

ポイント
- **この関数で扱える単精度浮動小数点型実数の範囲は、**

  −1.6777216E+7 ≦ N ≦ −1.4E−45
  N ＝ 0
  1.4E−45 ≦ N ≦ 1.6777216E+7

  (有効桁数：約7桁)です。
- **この関数で扱える倍精度浮動小数点型実数の範囲は、**

  −9.007199254740992E+15 ≦ N ≦ −2.23E−308
  N ＝ 0
  +2.23E−308 ≦ N ≦ +9.007199254740992E+15

  (有効桁数：約16桁)です。

●動作説明

- **In：変換対象のサフィックスが.U/.S/.D/.L/.Fの場合**
  nに格納された単精度浮動小数点型実数の角度(ラジアン単位)の正弦(SIN)値を求めます。

  結果は32ビット単精度浮動小数点型実数値となり、Rに格納されます。
- **In：変換対象のサフィックスが.DFの場合**
  Inに格納された倍精度浮動小数点型実数の角度(ラジアン単位)の正弦(SIN)値を求めます。
  結果は64ビット倍精度浮動小数点型実数値となり、Rに格納されます。

「演算フラグについて（CR2009～CR2012）」（1-8ページ）
詳細については、『命令語リファレンスマニュアル』「浮動小数命令」を参照してください。

参考　角度(°)と、(rad)は次のように計算されます。

$1^\circ = \frac{\pi}{180} \fallingdotseq 0.01745$ (rad)、 1 (rad) $= \frac{180}{\pi} \fallingdotseq 57.23^\circ$ です。

●書式例

rValue := **SIN** (rRadian);

ポイント　**演算する値がSIN関数で扱える単精度浮動小数点型実数の範囲外の場合、Rには In の値が格納されます。**

COS　角度(ラジアン)→余弦値

[R:=] COS (In);（コサイン）

引数/戻り値	説明	認識する型								定数
		.U	.S	.D	.L	.F	.DF	.B	.T	
In	演算対象、または演算対象データを指定。	.F	.F	.F	.F	.F	.DF※2	—	—	○
R※1	演算結果が格納される戻り値。省略可能。省略時、内部レジスタに格納。	.U	.S	.D	.L	.F	.DF※2	—	—	—

※1　COS関数は、引数がU/S/D/L/Fの場合.Fの型を使用、DFの場合.DFの型を使用。Rに関数が持つ型と異なる型を指定した場合、変換時に自動的に型変換を実行。
※2　デバイスZを指定した場合、「.DF」サフィックスは指定不可。

ポイント
- **この関数で扱える単精度浮動小数点型実数の範囲は、**

  −1.6777216E+7 ≦ N ≦ −1.4E−45
  N ＝ 0
  1.4E−45 ≦ N ≦ 1.6777216E+7

  (有効桁数：約7桁)です。
- **この関数で扱える倍精度浮動小数点型実数の範囲は、**

  −9.007199254740992E+15 ≦ N ≦ −2.23E−308
  N ＝ 0
  +2.23E−308 ≦ N ≦ +9.007199254740992E+15

  (有効桁数：約16桁)です。

●動作説明

- **In：変換対象のサフィックスが.U/.S/.D/.L/.Fの場合**
  In に格納された単精度浮動小数点型実数の角度(ラジアン単位)の余弦(COS)値を求めます。
  結果は32ビット単精度浮動小数点型実数値となり、R に格納されます。
- **In：変換対象のサフィックスが.DFの場合**
  In に格納された倍精度浮動小数点型実数の角度(ラジアン単位)の余弦(COS)値を求めます。
  結果は64ビット倍精度浮動小数点型実数値となり、R に格納されます。

「演算フラグについて（CR2009～CR2012）」（1-8ページ）
詳細については、『命令語リファレンスマニュアル』「浮動小数命令」を参照してください。

参考　角度(°)と、(rad)は次のように計算されます。

$1^\circ = \frac{\pi}{180} \fallingdotseq 0.01745$ (rad)、 1 (rad) $= \frac{180}{\pi} \fallingdotseq 57.23^\circ$ です。

●書式例

rValue := **COS** (rRadian);

ポイント　演算する値がCOS関数で扱える単精度浮動小数点型実数の範囲外の場合、RにはInの値が格納されます。

TAN　角度(ラジアン)→正接値

タンジェント
[R:=] TAN (In);

引数/戻り値	説明	認識する型								定数
		.U	.S	.D	.L	.F	.DF	.B	.T	
In	演算対象、または演算対象データを指定。	.F	.F	.F	.F	.F	.DF※2	—	—	○
R※1	演算結果が格納される戻り値。省略可能。省略時、内部レジスタに格納。	.U	.S	.D	.L	.F	.DF※2	—	—	—

※1　TAN関数は、引数がU/S/D/L/Fの場合.Fの型を使用、DFの場合.DFの型を使用。Rに関数が持つ型と異なる型を指定した場合、変換時に自動的に型変換を実行。
※2　デバイスZを指定した場合、「.DF」サフィックスは指定不可。

ポイント
- この関数で扱える単精度浮動小数点型実数の範囲は、

  −1.6777216E+7 ≦ N ≦ −1.4E−45
  N ＝ 0
  1.4E−45 ≦ N ≦ 1.6777216E+7

  (有効桁数：約7桁)です。
- この関数で扱える倍精度浮動小数点型実数の範囲は、

  −9.007199254740992E+15 ≦ N ≦ −2.23E−308
  N ＝ 0
  +2.23E−308 ≦ N ≦ +9.007199254740992E+15

  (有効桁数：約16桁)です。

●動作説明

- **In：変換対象のサフィックスが.U/.S/.D/.L/.Fの場合**
  Inに格納された単精度浮動小数点型実数の角度(ラジアン単位)の正接(TAN)値を求めます。
  結果は32ビット単精度浮動小数点型実数値となり、Rに格納されます。
- **In：変換対象のサフィックスが.DFの場合**
  Inに格納された倍精度浮動小数点型実数の角度(ラジアン単位)の正接(TAN)値を求めます。
  結果は64ビット倍精度浮動小数点型実数値となり、Rに格納されます。

「演算フラグについて（CR2009～CR2012）」（1-8ページ）
詳細については、『命令語リファレンスマニュアル』「浮動小数命令」を参照してください。

参考　角度(°)と、(rad)は次のように計算されます。

$1^\circ = \frac{\pi}{180} \fallingdotseq 0.01745$ (rad)、 1 (rad) $= \frac{180}{\pi} \fallingdotseq 57.23^\circ$ です。

●書式例

rValue := **TAN** (rRadian);

ポイント　演算する値がTAN関数で扱える単精度浮動小数点型実数の範囲外の場合、RにはInの値が格納されます。

ASIN　正弦値→角度(ラジアン)

[R:=] ASIN (In);
（アークサイン）

引数/戻り値	説明	認識する型								定数
		.U	.S	.D	.L	.F	.DF	.B	.T	
In	演算対象、または演算対象データを指定。	.F	.F	.F	.F	.F	.DF※2	—	—	○
R※1	演算結果が格納される戻り値。省略可能。省略時、内部レジスタに格納。	.U	.S	.D	.L	.F	.DF※2	—	—	—

※1　ASIN関数は、引数がU/S/D/L/Fの場合.Fの型を使用、DFの場合.DFの型を使用。Rに関数が持つ型と異なる型を指定した場合、変換時に自動的に型変換を実行。
※2　デバイスZを指定した場合、「.DF」サフィックスは指定不可。

ポイント
- **単精度浮動小数点型実数の範囲、および倍精度浮動小数点型実数の範囲は、「INTG関数」(7-45ページ)を参照。**

●動作説明

- **In：変換対象のサフィックスが.U/.S/.D/.L/.Fの場合**
  Inに格納された単精度浮動小数点型実数の正弦(SIN)値から角度(ラジアン単位)を求めます。
  結果は32ビット単精度浮動小数点型実数値となり、Rに格納されます。
- **In：変換対象のサフィックスが.DFの場合**
  Inに格納された倍精度浮動小数点型実数の正弦(SIN)値から角度(ラジアン単位)を求めます。
  結果は64ビット倍精度浮動小数点型実数値となり、Rに格納されます。

「演算フラグについて（CR2009～CR2012）」（1-8ページ）
詳細については、『命令語リファレンスマニュアル』「浮動小数命令」を参照してください。

参考　角度(°)と、(rad)は次のように計算されます。

$$1^\circ = \frac{\pi}{180} \fallingdotseq 0.01745\ (\mathrm{rad}),\quad 1\ (\mathrm{rad}) = \frac{180}{\pi} \fallingdotseq 57.23^\circ$$ です。

●書式例

rRadian := **ASIN** (rValue);

ポイント
演算する値が－1.0～1.0の範囲外の場合、RにはInの値が格納されます。

ACOS　余弦値→角度(ラジアン)

[R:=] ACOS (In);
（アークコサイン）

引数/戻り値	説明	認識する型								定数
		.U	.S	.D	.L	.F	.DF	.B	.T	
In	演算対象、または演算対象データを指定。	.F	.F	.F	.F	.F	.DF※2	—	—	○
R※1	演算結果が格納される戻り値。省略可能。省略時、内部レジスタに格納。	.U	.S	.D	.L	.F	.DF※2	—	—	—

※1　ACOS関数は、引数がU/S/D/L/Fの場合.Fの型を使用、DFの場合.DFの型を使用。Rに関数が持つ型と異なる型を指定した場合、変換時に自動的に型変換を実行。
※2　デバイスZを指定した場合、「.DF」サフィックスは指定不可。

●動作説明

- **In:変換対象のサフィックスが.U/.S/.D/.L/.Fの場合**
  Inに格納された単精度浮動小数点型実数の余弦(COS)値から角度(ラジアン単位)を求めます。
  結果は32ビット単精度浮動小数点型実数値となり、Rに格納されます。
- **In:変換対象のサフィックスが.DFの場合**
  Inに格納された倍精度浮動小数点型実数の余弦(COS)値から角度(ラジアン単位)を求めます。
  結果は64ビット倍精度浮動小数点型実数値となり、Rに格納されます。

「演算フラグについて(CR2009～CR2012)」(1-8ページ)
詳細については、『命令語リファレンスマニュアル』「浮動小数命令」を参照してください。

参考　角度(°)と、(rad)は次のように計算されます。

$1^\circ = \frac{\pi}{180} \fallingdotseq 0.01745$ (rad)、1 (rad) $= \frac{180}{\pi} \fallingdotseq 57.23^\circ$ です。

●書式例

rRadian := **ACOS** (rValue);

ポイント　**演算する値が－1.0～1.0の範囲外の場合、RにはInの値が格納されます。**

ATAN　正接値→角度(ラジアン)

[R:=] ATAN (In);
(アークタンジェント)

引数/戻り値	説明	認識する型								定数
		.U	.S	.D	.L	.F	.DF	.B	.T	
In	演算対象、または演算対象データを指定。	.F	.F	.F	.F	.F	.DF※2	—	—	○
R※1	演算結果が格納される戻り値。省略可能。省略時、内部レジスタに格納。	.U	.S	.D	.L	.F	.DF※2	—	—	—

※1　ATAN関数は、引数がU/S/D/L/Fの場合.Fの型を使用、DFの場合.DFの型を使用。Rに関数が持つ型と異なる型を指定した場合、変換時に自動的に型変換を実行。
※2　デバイスZを指定した場合、「.DF」サフィックスは指定不可。

●動作説明

- **In:変換対象のサフィックスが.U/.S/.D/.L/.Fの場合**
  Inに格納された単精度浮動小数点型実数の正接(TAN)値から角度(ラジアン単位)を求めます。
  結果は32ビット単精度浮動小数点型実数値となり、Rに格納されます。

- **In:変換対象のサフィックスが.DFの場合**
  Inに格納された倍精度浮動小数点型実数の正接(TAN)値から角度(ラジアン単位)を求めます。
  結果は64ビット倍精度浮動小数点型実数値となり、Rに格納されます。

「演算フラグについて(CR2009～CR2012)」(1-8ページ)
詳細については、『命令語リファレンスマニュアル』「浮動小数命令」を参照してください。

参考　角度(°)と、(rad)は次のように計算されます。

$$1^\circ = \frac{\pi}{180} \fallingdotseq 0.01745\ (\mathrm{rad}),\quad 1\ (\mathrm{rad}) = \frac{180}{\pi} \fallingdotseq 57.23^\circ$$ です。

●書式例

rRadian := **ATAN** (rValue);

ポイント　**演算する値が単精度浮動小数点型実数の範囲外の場合、Rには In の値が格納されます。**

ATAN2　座標→角度(ラジアン)

アークタンジェント2
[R:=] ATAN2 (Y, X);

引数/戻り値	説明	認識する型								定数
		.U	.S	.D	.L	.F	.DF	.B	.T	
Y	演算対象1、または演算対象データ1を指定。	.F	.F	.F	.F	.F	.DF※2	—	—	○
X	演算対象2、または演算対象データ2を指定。	.F	.F	.F	.F	.F	.DF※2	—	—	○
R※1	演算結果が格納される戻り値。省略可能。省略時、内部レジスタに格納。	.U	.S	.D	.L	.F	.DF※2	—	—	—

※1　ATAN2関数は、引数がU/S/D/L/Fの場合.Fの型を使用、DFの場合.DFの型を使用。Rに関数が持つ型と異なる型を指定した場合、変換時に自動的に型変換を実行。
※2　デバイスZを指定した場合、「.DF」サフィックスは指定不可。

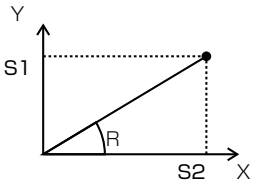
●動作説明

- **Y:変換対象1、X:変換対象2のサフィックスがU/S/D/L/Fの場合**
  Yで指定した値を単精度浮動小数点型実数に変換したY座標とXで指定した値を単精度浮動小数点型実数に変換したX座標から、単精度浮動小数点型実数のX軸に対する角度(ラジアン単位)を求めます。
  結果は32ビット単精度浮動小数点型実数値となり、Rに格納されます。
- **Y:変換対象1、X:変換対象2のサフィックスがDFの場合**
  Yで指定した値を倍精度浮動小数点型実数に変換したY座標とXで指定した値を倍精度浮動小数点型実数に変換したX座標から、倍精度浮動小数点型実数のX軸に対する角度(ラジアン単位)を求めます。
  結果は64ビット倍精度浮動小数点型実数値となり、Rに格納されます。

「演算フラグについて(CR2009～CR2012)」(1-8ページ)
詳細については、『命令語リファレンスマニュアル』「浮動小数命令」を参照してください。

参 考

ATAN 命令では、正接(TAN)値を使用することで、-π/2(rad) < **R** < π/2(rad)までの角度を求められたのに対し、ATAN2命令では、座標(**X**, **Y**)を使用することで、-π(rad) < **R** ≦ π(rad)までの角度を求められます。



$1^\circ = \frac{\pi}{180} \fallingdotseq 0.01745$ (rad) 、 1 (rad) $= \frac{180}{\pi} \fallingdotseq 57.23^\circ$ です。

S1=0、S2≧0の場合、R(戻り値)=0°=0(rad)となります。

S1=0、S2<0の場合、R(戻り値)=180°=π(rad)となります。

S1>0、S2=0の場合、R(戻り値)=90°=π/2(rad)となります。

S1<0、S2=0の場合、R(戻り値)=-90°=-π/2(rad)となります。

●書式例

rRadian := **ATAN2** (rValueY, rValueX);

ポイント

演算する値が単精度浮動小数点型実数の範囲外の場合、Rの値は変更されません。

文字処理関数

ASC　BINデータ→16進アスキー文字コード変換

16進アスキーコード変換
[R:=] ASC (In);

引数/戻り値	説明	認識する型								定数
		.U	.S	.D	.L	.F	.DF	.B	.T	
In※1	変換対象の16進アスキー文字コードを格納した対象、または変換対象データを指定。	.U	.U	.U	.U	–	–	–	–	○
R※2	変換結果が格納される戻り値。省略可能。省略時、内部レジスタに格納。	.U	.S	.D	.L	.F	.DF※3	–	–	–

※1　R1002等変換対象にビットデータのチャンネルの先頭以外を指定した場合でも、次のチャンネルにまたがって正常に動作。
※2　Rに.U以外の型を指定した場合、変換時に自動的に型変換を実行。
※3　デバイスZを指定した場合、「.DF」サフィックスは指定不可。

●動作説明

Inに格納されたバイナリデータの下位8ビットの値を、16進表記のアスキー文字コードに変換し、結果の16ビットデータをRに格納します。

「演算フラグについて（CR2009～CR2012）」（1-8ページ）
詳細については、『命令語リファレンスマニュアル』「文字処理命令」を参照してください。

●書式例

uiValue := **ASC** (uiData);

RASC　16進アスキー文字コード→BINデータ変換

16進アスキーコード逆変換
[R:=] RASC (In);

引数/戻り値	説明	認識する型								定数
		.U	.S	.D	.L	.F	.DF	.B	.T	
In※1※2	変換対象のバイナリデータを格納した対象、または変換対象データを指定。	.U	.U	.U	.U	–	–	–	–	○
R※3	変換結果が格納される戻り値。省略可能。省略時、内部レジスタに格納。	.U	.S	.D	.L	.F	.DF※4	–	–	–

※1　Inには、32ビットデータも使用可能だが、変換対象は下位16ビットのみ。
※2　R1002等Inにビットデータのチャンネルの先頭以外を指定した場合でも、次のチャンネルにまたがって正常に動作。
※3　Rに.U以外の型を指定した場合、変換時に自動的に型変換を実行。
※4　デバイスZを指定した場合、「.DF」サフィックスは指定不可。

●動作説明

Inに格納された16ビットの16進表記のアスキー文字コード(2文字分)を、バイナリデータに変換し、結果の16ビットデータ(下位8ビットが変換値、上位8ビットは0)をRに格納します。

「演算フラグについて（CR2009～CR2012）」（1-8ページ）
「ASCIIコード表」（付-9ページ）

詳細については、『命令語リファレンスマニュアル』「文字処理命令」を参照してください。

ポイント
- **Inには上位8ビット、下位8ビットとも0～9を示すアスキーコード(30H～39H)または、A～F(41H～46H)のとき、および、上位8ビットがNUL(00H)で下位8ビットが0～9(30H～39H)またはA～F(41H～46H)の値(アスキーコード)が格納されている必要があります。**
  上位8ビット、下位8ビットのどちらかがこの範囲外のとき、関数は実行しません。
- **扱えないアスキー文字コードがある場合、RにはInの値が格納されます。**

●書式例

uiValue := **RASC** (uiData);

DASC　BINデータ→10進アスキー文字列変換

10進アスキーコード変換
[R:=] DASC (In, Param);

引数/戻り値	説明	認識する型								定数
		.U	.S	.D	.L	.F	.DF	.B	.T	
In※1※2	変換対象のバイナリデータを格納した対象、または変換対象データを指定。	.U	.S	.D	.L	—	—	—	—	○※3
Param※4	変換パラメータを格納した対象、またはパラメータを指定。(0～3)	.U	.U	.U	.U	—	—	—	—	○
R※2※5	変換結果が格納される戻り値。.Tの型を使用。省略可能。省略時、内部レジスタに格納。	—	—	—	—	—	—	—	.T	—

※1　ビットデータを指定した場合、連続する16/32ビットで演算。
R002などチャンネルの先頭以外を指定した場合、次のチャンネルにまたがって16/32ビットで演算。
※2　ビットデータではチャンネルの先頭を指定。
※3　定数は16# (16進数指定) で指定不可。
※4　変換パラメータは0～3の範囲で指定。
Param変換パラメータの説明

値	内　容	CR2814	CR2815
0	ゼロサプレスOFF、符号省略OFF	0	0
1	ゼロサプレスON、符号省略OFF	1	0
2	ゼロサプレスOFF、符号省略ON	0	1
3	ゼロサプレスON、符号省略ON	1	1

ポイント　**変換パラメータに範囲外の値を指定した場合、正しく動作しない可能性あり。**

※5　T, C, CR, Zは使用不可。

●動作説明

Inに格納されたバイナリデータを、Paramで指定された変換パラメータに従い、10進アスキーコード文字列に変換して、結果をRに格納します。

「演算フラグについて(CR2009～CR2012)」(1-8ページ)
詳細については、『命令語リファレンスマニュアル』「文字処理命令」を参照してください。

●書式例

strString := **DASC** (uiValue, INT#3);

ポイント
- 変換対象データに定数を使用する場合、16# (16進数指定)での指定はできません。
- 変換対象データには半角文字(ASCIIコード)のみ指定できます。
- 割り込みプログラムでCR2814・CR2815を変更したり参照する場合は注意が必要です。
- 次のいずれかの条件が成立した場合、Rには終端コード(NUL(00H))が格納されます。
  - Rにビットデータを指定したときに、チャンネルの先頭以外を指定した場合。
  - Rに間接指定でタイマ・カウンタを指定した場合。

STR　数値データ→アスキー文字列変換

10進アスキーコード変換
[R:=] STR (In);

引数/戻り値	説明	認識する型								定数
		.U	.S	.D	.L	.F	.DF	.B	.T	
In※1※3	変換対象のバイナリデータを格納した対象、または変換対象データを指定。	.U	.S	.D	.L	.F	—	—	—	○
R※2※4	変換結果が格納される戻り値。.Tの型を使用。省略可能。省略時、内部レジスタに格納。	—	—	—	—	—	—	—	.T	—

※1　ビットデータを指定した場合、連続する16/32ビットで演算。
R002などチャンネルの先頭以外を指定した場合、次のチャンネルにまたがって16/32ビットで演算。
※2　ビットデータではチャンネルの先頭を指定。
※3　定数は16# (16進数指定)で指定不可。
※4　T, C, CR, Zは使用不可。

●動作説明

Inで指定された引数の型により、10進アスキー文字列変換または単精度浮動小数点文字列変換をおこないます。

動作の詳細については、「DASC関数」(7-60ページ)を参照してください。
「演算フラグについて(CR2009～CR2012)」(1-8ページ)

ポイント
- 変換対象データに定数を使用する場合、16# (16進数指定)での指定はできません。
- 次のいずれかの条件が成立した場合、Rには終端コード(NUL(00H))が格納されます。
  - Rにビットデータを指定したときに、チャンネルの先頭以外を指定した場合。
  - Rに間接指定でタイマ・カウンタを指定した場合。

●書式例

strString := **STR** (uiValue);

RDASC　10進アスキー文字列→BINデータ変換

10進アスキーコード逆変換
[R:=] RDASC (In);

引数/戻り値	説明	認識する型								定数
		.U	.S	.D	.L	.F	.DF	.B	.T	
In※1※2	変換対象の10進アスキー文字列を格納した対象、または10進アスキーコード文字列を指定。	–	–	–	–	–	–	–	.T	○
R※3※4	変換結果が格納される戻り値。省略可能。省略時、内部レジスタに格納。	.U	.S	.D	.L	.F	.DF※5	–	–	–

※1　ビットデータではチャンネルの先頭を指定。
※2　T, C, CR, Zは使用不可。
※3　ビットデータを指定した場合、連続する32ビットで演算。
R002などチャンネルの先頭以外を指定した場合、次のチャンネルにまたがって16/32ビットで演算。
※4　Rに.L以外の型を指定した場合、変換時に自動的に型変換を実行。
※5　デバイスZを指定した場合、「.DF」サフィックスは指定不可。

●動作説明

Inを先頭に格納されている10進アスキー文字列を、32ビットバイナリデータ(符号付き)に変換し、結果をRに格納します。

「演算フラグについて（CR2009～CR2012）」（1-8ページ）
詳細については、『命令語リファレンスマニュアル』「文字処理命令」を参照してください。

●書式例

uiValue := **RDASC** (strString);

ポイント
- **変換対象データで指定する10進アスキー文字列には、16# (16進数指定)の30～39('0'～'9')、および16# (16進数指定)の2B、2D('±'符号：符号付変換のときのみ)で構成された文字列を指定してください。**
- **Inに上記以外のアスキー文字列を指定した場合、関数は実行されません。**
- **Inで指定したアスキー文字列にNUL(00H)がない場合、関数は最大11文字の文字列を変換します。**
- **次のいずれかの条件が成立した場合、Rには0が格納されます。**
  - **Inに不適切な文字がある場合。**
  - **変換した値が32ビット符号付きバイナリデータの範囲外の場合。**
  - **Inにビットデータを指定したときに、チャンネルの先頭以外を指定した場合。**
  - **Inに間接指定でタイマ・カウンタを指定した場合。**

HASC　BINデータ→16進アスキー文字列変換

16進アスキーコード変換
[R:=] HASC (In, Count);

引数/戻り値	説明	認識する型								定数
		.U	.S	.D	.L	.F	.DF	.B	.T	
In※1	変換対象のバイナリデータを格納した対象、または変換対象データを指定。 (符号無しバイナリデータ)。	.U	.U	.D	.D	—	—	—	—	○
Count※4	変換結果を格納する数値(16進)の桁数を指定。 （0～8）	.U	.U	.U	.U	—	—	—	—	○※2
R※3※5	変換結果が格納される戻り値。省略可能。省略時、内部レジスタに格納。 HASC関数は、.Tの型を使用。	—	—	—	—	—	—	—	.T	—

※1　ビットデータを指定した場合、連続する16/32ビットで演算。
R002などチャンネルの先頭以外を指定した場合、次のチャンネルにまたがって16/32ビットで演算。
※2　定数は 16# (16進数指定)で指定。
※3　Rを格納する対象には、.Tサフィックスの指定が必要。
またRにビットデータを指定する場合は、チャンネルの先頭を指定。
※4　変換桁数には変換結果を格納する桁数(下位桁)を、0～8の範囲で指定。
(0を指定した場合は、ゼロサプレス有りで有効桁数分を格納)
※5　T, C, CR, Zは使用不可。

●動作説明

Inに格納されたバイナリデータを、終端コードNUL(00H)を付加した16進表記のアスキーコード文字列に変換し、結果を指定されたCount分だけRに格納します。(Count＝0のときInのデータの全てをゼロサプレスして変換します)

「演算フラグについて（CR2009～CR2012）」（1-8ページ）
詳細については、『命令語リファレンスマニュアル』「文字処理命令」を参照してください。

●書式例

strString := **HASC** (uiValue, INT#3);

ポイント
- **変換対象データに定数を使用する場合、10進での指定はできません。**
- **次のいずれかの条件が成立した場合、Rには終端コード(NUL(00H))が格納されます。**
  - **Rにビットデータを指定したときに、チャンネルの先頭以外を指定した場合。**
  - **Rに間接指定でタイマ・カウンタを指定した場合。**
  - **Countが.Uのとき5以上、.Dのとき9以上が指定された場合。**

RHASC　16進アスキー文字列→BINデータ変換

16進アスキーコード逆変換
[R:=] RHASC (In);

引数/戻り値	説明	認識する型								定数
		.U	.S	.D	.L	.F	.DF	.B	.T	
In※1※2	変換対象の16進アスキー文字列を格納した対象、または16進アスキーコード文字列を指定。	—	—	—	—	—	—	—	.T	○
R※3※4	変換結果が格納される戻り値。省略可能。省略時、内部レジスタに格納。	.U	.S	.D	.L	.F	.DF※5	—	—	—

※1　ビットデータではチャンネルの先頭を指定。
※2　T, C, CR, Zは使用不可。
※3　ビットデータを指定した場合、連続する32ビットで演算。
R002などチャンネルの先頭以外を指定した場合、次のチャンネルにまたがって32ビットで演算。
※4　Rに.D以外の型を指定した場合、変換時に自動的に型変換を実行。
※5　デバイスZを指定した場合、「.DF」サフィックスは指定不可。

●動作説明

Inを先頭に格納されている16進表記のアスキーコード文字列を、32ビットバイナリデータ(符号無し)に変換し、結果をRに格納します。

「演算フラグについて（CR2009～CR2012）」（1-8ページ）
詳細については、『命令語リファレンスマニュアル』「文字処理命令」を参照してください。

●書式例

udiValue := **RHASC** (strString);

ポイント
- **Inで指定する16進アスキー文字列には、16# (16進数指定)の30～39('0'～'9')、または16# (16進数指定)の41～46('A'～'F')の範囲の値が格納されている必要があります。**
- **16進アスキー文字列でA～F(41H～46H)は大文字を使用してください。小文字には対応していません。**
- **次のいずれかの条件が成立した場合、Rには0が格納されます。**
  - **Inに不適切な文字がある場合。**
  - **変換した値が32ビット符号無しバイナリデータの範囲外の場合。**
  - **Inにビットデータを指定したときに、チャンネルの先頭以外を指定した場合。**
  - **Inに間接指定でタイマ・カウンタを指定した場合。**

FASC　単精度浮動小数点型実数→文字列変換

単精度浮動小数点文字列変換
[R:=] FASC (Fmt, FullCount, SubDecCount, In, Param);

引数/戻り値	説明	認識する型								定数
		.U	.S	.D	.L	.F	.DF	.B	.T	
Fmt	表示指定形式(小数点形式、指数形式)を指定。	.U	.U	.U	.U	–	–	–	.T	○
FullCount	全体の桁数を指定。	.U	.U	.U	.U	–	–	–	–	○
SubDecCount	小数部桁数を指定。	.U	.U	.U	.U	–	–	–	–	○
In※1※2※4	変換対象の単精度浮動小数点型実数を格納した対象、または単精度浮動小数点型実数データを指定。	–	–	–	–	.F	–	–	–	○
Param	ゼロサプレスと符号省略を指定。	.U	.U	.U	.U	–	–	–	–	○
R※3※4	変換結果が格納される戻り値。.Tの型を使用。省略可能。省略時、内部レジスタに格納。	–	–	–	–	–	–	–	.T	–

※1　ビットデータを指定した場合、連続した32ビットを占有。R002などチャンネルの先頭以外を指定した場合、次のチャンネルにまたがって32ビットを占有。
※2　単精度浮動小数点型実数の範囲は、「INTG関数」(7-45ページ)を参照。
※3　ビットデータでは、チャンネルの先頭を指定。
※4　T, C, CR, Zは使用不可。

●動作説明

Inに格納された単精度浮動小数点型実数データを、Fmt～SubDecCountおよびParamに格納された変換パラメータに従って、終端コードNUL(00H)を付加した文字列に変換し、結果をRに格納します。
変換パラメータは次のように指定します。

Fmt	表示指定形式	小数点形式の文字列に変換する場合は「0」、指数形式の文字列に変換する場合は「1」を指定します。
FullCount	全体の桁数	1～24桁の範囲で指定します。
SubDecCount	小数部桁数	0～7桁の範囲で指定します。

変換パラメータで指定する内容は次のようになります。

Param値	内　容	CR2814	CR2815
0	ゼロサプレスOFF、符号省略OFF	0	0
1	ゼロサプレスON、符号省略OFF	1	0
2	ゼロサプレスOFF、符号省略ON	0	1
3	ゼロサプレスON、符号省略ON	1	1

「演算フラグについて(CR2009～CR2012)」(1-8ページ)
詳細については、『命令語リファレンスマニュアル』「文字処理命令」を参照してください。

●書式例

strString := **FASC** (INT#0, INT#10, INT#3, rValue, uiParam);

ポイント

- 変換対象データに定数を使用する場合、16# (16進数指定)での指定はできません。
- 割り込みプログラムでCR2814･CR2815を変更したり参照する場合は注意が必要です。
- 次のいずれかの条件が成立した場合、Rには終端コード(NUL(00H))が格納されます。
  - 引数の値が範囲外の場合。
  - 単精度浮動小数点型実数の範囲外を指定した場合。
  - 格納する文字列の桁数が全桁数を超えた場合。
  - Rにビットデータを指定したときに、チャンネルの先頭以外を指定した場合。
  - Rに間接指定でタイマ･カウンタを指定した場合。
  - 全桁数と、小数部桁数が次の条件を満たさない場合。

	小数点形式表示	指数形式表示
SubDecCount ＝ 0	1 ≦ FullCount ≦ 24	5 ≦ FullCount ≦ 24
0 ＜ SubDecCount ≦ 7	SubDecCount ＋ 2 ≦ FullCount 3 ≦ FullCount ≦ 24	SubDecCount ＋ 6 ≦ FullCount 7 ≦ FullCount ≦ 24

RFASC　文字列→単精度浮動小数点型実数変換

単精度浮動小数点文字列逆変換
[R:=] RFASC (In);

引数/戻り値	説明	認識する型								定数
		.U	.S	.D	.L	.F	.DF	.B	.T	
In※1※3※4	変換対象の単精度浮動小数点文字列を格納した対象、または単精度浮動小数点文字列を指定。	—	—	—	—	—	—	—	.T	○
R※2※4※5	変換結果が格納される戻り値。省略可能。省略時、内部レジスタに格納。	.U	.S	.D	.L	.F	.DF※6	—	—	—

※1　ビットデータではチャンネルの先頭を指定。
※2　ビットデータを指定した場合、連続した32ビットを占有。R002などチャンネルの先頭以外を指定した場合、次のチャンネルにまたがって32ビットを占有。
※3　変換対象で指定する単精度浮動小数点文字列には、最後にNUL(00H)が必要。
※4　T, C, CR, Zは使用不可。
※5　Rに.F以外の型を指定した場合、変換時に自動的に型変換を実行。
※6　デバイスZを指定した場合、「.DF」サフィックスは指定不可。

●動作説明

Inを先頭に格納されている単精度浮動小数点文字列を、単精度浮動小数点型実数データに変換し、結果をRに格納します。

「演算フラグについて（CR2009～CR2012）」（1-8ページ）
詳細については、『命令語リファレンスマニュアル』「文字処理命令」を参照してください。

●書式例

rValue := **RFASC** (strString);

ポイント

- 変換対象データで指定する単精度浮動小数点文字列には、16# (16進数指定)の30～39('0'～'9')、2B('+')、2C('－')、45('E')、および2E('.')の値のみが格納されている必要があります。
- 次のいずれかの条件が成立した場合、Rには0が格納されます。
  - 文字列の長さが0の場合。
  - 文字列の書式が正しくない場合。
  - 変換後の数値が単精度浮動小数点型実数の範囲を超えている場合。
  - 格納する文字列の桁数が全桁数を超えた場合。
  - Inにビットデータを指定したときに、チャンネルの先頭以外を指定した場合。

VAL　文字列→単精度浮動小数点型実数変換

単精度浮動小数点文字列逆変換
[R:=] VAL (In);

引数/戻り値	説明	認識する型								定数
		.U	.S	.D	.L	.F	.DF	.B	.T	
In※1※2※4	変換対象の単精度浮動小数点文字列を格納した対象、または単精度浮動小数点文字列を指定。	–	–	–	–	–	–	–	.T	○
R※3※5	変換結果が格納される戻り値。省略可能。省略時、内部レジスタに格納。	.U	.S	.D	.L	.F	.DF※6	–	–	–

※1　ビットデータではチャンネルの先頭を指定。
※2　変換対象で指定する単精度浮動小数点文字列には、最後にNUL(00H)が必要。
※3　ビットデータを指定した場合、連続した32ビットを占有。R002などチャンネルの先頭以外を指定した場合、次のチャンネルにまたがって32ビットを占有。
※4　T, C, CR, Zは使用不可。
※5　Rに.F以外の型を指定した場合、変換時に自動的に型変換を実行。
※6　デバイスZを指定した場合、「.DF」サフィックスは指定不可。

●動作説明

本関数は「RFASC関数」と関数名が異なるだけで、動作等に違いはありません。
動作の詳細については、「RFASC関数」(7-66ページ)を参照してください。

LEN　文字列長検出

文字列長検出
[R:=] LEN (In);

引数/戻り値	説明	認識する型								定数
		.U	.S	.D	.L	.F	.DF	.B	.T	
In※1※2※4	文字列長を求める対象の文字列を格納した対象、または対象文字列を指定。	–	–	–	–	–	–	–	.T	○※1
R※3※4	変換結果が格納される戻り値。省略可能。省略時、内部レジスタに格納。	.U	.S	.D	.L	.F	.DF※5	–	–	–

※1　定数で文字列を指定する場合は、''で囲んで入力。
※2　ビットデータではチャンネルの先頭を指定。
※3　T, C, CR, Zは使用不可。
※4　Rに.U以外の型を指定した場合、変換時に自動的に型変換を実行。
※5　デバイスZを指定した場合、「.DF」サフィックスは指定不可。

●動作説明

Inを先頭に格納されている文字列の文字数を数え、結果を16ビットバイナリデータとして**R**に格納します。

「演算フラグについて（CR2009～CR2012）」（1-8ページ）
詳細については、『命令語リファレンスマニュアル』「文字処理命令」を参照してください。

●書式例

uiValue := **LEN** (strString);

ポイント
- **全角文字は2文字として扱われます。**
- **終端コードNUL(00H)は数えません。**
- **次のいずれかの条件が成立した場合、Rには0が格納されます。**
  - **終端コード(NUL(00H))が無い場合。**
  - **終端コード(NUL(00H))を含む文字列が2000文字を超えた場合。**
  - **Inに間接指定でビットデータを指定したときに、チャンネルの先頭以外を指定した場合。**
  - **Inに間接指定でタイマ・カウンタを指定した場合。**

SRGHT　右文字列切り出し

[R:=] SRGHT (In, Count);
右文字列切り出し

引数/戻り値	説明	認識する型								定数
		.U	.S	.D	.L	.F	.DF	.B	.T	
In※1※5	切り出し対象となる文字列を格納した対象、または切り出し対象文字列を指定。	—	—	—	—	—	—	—	.T	○※2
Count※4	切り出す文字数（バイト数）を指定。（0～1999）	.U	.U	.U	.U	—	—	—	—	○※3
R※1※5	変換結果が格納される戻り値。.Tの型を使用。省略可能。省略時、内部レジスタに格納。	—	—	—	—	—	—	—	.T	—

※1　ビットデータではチャンネルの先頭を指定。
※2　定数で文字列を指定する場合は、' 'で囲んで入力。
※3　16# (16進数指定) は使用不可。
※4　**Count**には右端(終端)からの文字数(バイト数)を、0～1999の範囲で指定。
　　0を指定した場合、戻り値にはNUL(00H)が1文字分だけ格納。
※5　T, C, CR, Zは使用不可。

●動作説明

Inを先頭に格納されている文字列を、右端から、**Count**で指定された文字数(バイト数)分切り出し、終端コードNUL(00H)を付加して、結果を**R**に格納します。

「演算フラグについて（CR2009～CR2012）」（1-8ページ）
詳細については、『命令語リファレンスマニュアル』「文字処理命令」を参照してください。

●書式例

strString := **SRGHT** (strAllString, INT#40);

ポイント
- 次のいずれかの条件が成立した場合、終端コード(NUL(00H))が格納されます。
  - Inで指定した文字列に終端コード(NUL(00H))が無い場合。
  - 終端コード(NUL(00H))を含む文字列が2000文字を超えた場合。
  - R以降に格納される文字列がデバイス/変数の範囲を超えた場合。
  - In、Rにビットデータを指定したときに、チャンネルの先頭以外を指定した場合。
  - In、Rに間接指定でタイマ・カウンタを指定した場合。

RIGHT　右文字列切り出し

右文字列切り出し
[R:=] RIGHT (In, L);

引数/戻り値	説明	認識する型								定数
		.U	.S	.D	.L	.F	.DF	.B	.T	
In※1※5	切り出し対象となる文字列を格納した対象、または切り出し対象文字列を指定。	—	—	—	—	—	—	—	.T	○※2
L※4	切り出す文字数(バイト数)を指定。(0～1999)	.U	.U	.U	.U	—	—	—	—	○※3
R※1※5	変換結果が格納される戻り値。.Tの型を使用。省略可能。省略時、内部レジスタに格納。	—	—	—	—	—	—	—	.T	—

※1　ビットデータではチャンネルの先頭を指定。
※2　定数で文字列を指定する場合は、''で囲んで入力。
※3　16# (16進数指定) は使用不可。
※4　Lには右端(終端)からの文字数(バイト数)を、0～1999の範囲で指定。
　　0を指定した場合、戻り値にはNUL(00H)が1文字分だけ格納。
※5　T, C, CR, Zは使用不可。

●動作説明

本関数は「SRGHT関数」と関数名が異なるだけで、動作等に違いはありません。
動作の詳細については、「SRGHT関数」(7-68ページ)を参照してください。

SLEFT　左文字列切り出し

左文字列切り出し
[R:=] SLEFT (In, Count);

引数/戻り値	説明	認識する型								定数
		.U	.S	.D	.L	.F	.DF	.B	.T	
In※1※5	切り出し対象となる文字列を格納した対象、または切り出し対象文字列を指定。	—	—	—	—	—	—	—	.T	○※2
Count※4	切り出す文字数(バイト数)を指定。(0～1999)	.U	.U	.U	.U	—	—	—	—	○※3
R※1※5	変換結果が格納される戻り値。.Tの型を使用。省略可能。省略時、内部レジスタに格納。	—	—	—	—	—	—	—	.T	—

※1　ビットデータではチャンネルの先頭を指定。
※2　定数で文字列を指定する場合は、''で囲んで入力。
※3　16# (16進数指定) は使用不可。
※4　Countには右端(終端)からの文字数(バイト数)を、0～1999の範囲で指定。
　　0を指定した場合、戻り値にはNUL(00H)が1文字分だけ格納。

※5　T, C, CR, Zは使用不可。

●動作説明

Inを先頭に格納されている文字列を、左端(先端)から、**Count**で指定された文字数(バイト数)分切り出し、終端コードNUL(00H)を付加して、結果を**R**に格納します。

「演算フラグについて（CR2009～CR2012）」（1-8ページ）
詳細については、『命令語リファレンスマニュアル』「文字処理命令」を参照してください。

●書式例

strString := **SLEFT** (strAllString, INT#40);

ポイント

- **次のいずれかの条件が成立した場合、終端コード(NUL(00H))が格納されます。**
  - **In**で指定した文字列に終端コード(NUL(00H))が無い場合。
  - 終端コード(NUL(00H))を含む文字列が2000文字を超えた場合。
  - **R**以降に格納される文字列がデバイス/変数の範囲を超えた場合。
  - **In**、**R**にビットデータを指定したときに、チャンネルの先頭以外を指定した場合。
  - **In**、**R**に間接指定でタイマ・カウンタを指定した場合。

LEFT　左文字列切り出し

左文字列切り出し
[R:=] LEFT (In, L);

引数/戻り値	説明	認識する型								定数
		.U	.S	.D	.L	.F	.DF	.B	.T	
In※1※5	切り出し対象となる文字列を格納した対象、または切り出し対象文字列を指定。	–	–	–	–	–	–	–	.T	○※2
L※4	切り出す文字数（バイト数）を指定。（0～1999）	.U	.U	.U	.U	–	–	–	–	○※3
R※1※5	変換結果が格納される戻り値。.Tの型を使用。省略可能。省略時、内部レジスタに格納。	–	–	–	–	–	–	–	.T	–

※1　ビットデータではチャンネルの先頭を指定。
※2　定数で文字列を指定する場合は、''で囲んで入力。
※3　16# (16進数指定) は使用不可。
※4　**L**には右端(終端)からの文字数(バイト数)を、0～1999の範囲で指定。
　　0を指定した場合、戻り値にはNUL(00H)が1文字分だけ格納。
※5　T, C, CR, Zは使用不可。

●動作説明

本関数は「SLEFT関数」と関数名が異なるだけで、動作等に違いはありません。

動作の詳細については、「SLEFT関数」（7-69ページ）を参照してください。

SMID　文字列切り出し

文字列切り出し
[R:=] SMID (In, Pos, Count);

引数/戻り値	説明	認識する型								定数
		.U	.S	.D	.L	.F	.DF	.B	.T	
In※1※5	切り出し対象となる文字列を格納した対象、または切り出し対象文字列を指定。	—	—	—	—	—	—	—	.T	○※2
Pos※6	切り出し開始位置（バイト数）を指定。(0～1999)	.U	.U	.U	.U	—	—	—	—	○※3
Count※4	切り出す文字数（バイト数）を指定。（0～1999）	.U	.U	.U	.U	—	—	—	—	○※3
R※1※5	変換結果が格納される戻り値。.Tの型を使用。省略可能。省略時、内部レジスタに格納。	—	—	—	—	—	—	—	.T	—

※1　ビットデータではチャンネルの先頭を指定。
※2　定数で文字列を指定する場合は、''で囲んで入力。
※3　16# (16進数指定) は使用不可。
※4　Countには右端(終端)からの文字数(バイト数)を、0～1999の範囲で指定。
　　0を指定した場合、戻り値にはNUL(00H)が1文字分だけ格納。
※5　T, C, CR, Zは使用不可。
※6　Inで指定した文字列の先頭文字は0。

●動作説明

Inを先頭に格納されている文字列を、Posで指定された位置(バイト数)から、Countで指定された文字数(バイト数)分を切り出し、終端コードNUL(00H)を付加して、結果をRに格納します。

「演算フラグについて（CR2009～CR2012）」（1-8ページ）
詳細については、『命令語リファレンスマニュアル』「文字処理命令」を参照してください。

●書式例

strString := **SMID** (strAllString, INT#8, INT#16);

ポイント

- **次のいずれかの条件が成立した場合、終端コード(NUL(00H))が格納されます。**
  - **Inで指定した文字列に終端コード(NUL(00H))が無い場合。**
  - **終端コード(NUL(00H))を含む文字列が2000文字を超えた場合。**
  - **Inの文字列長＜Posの場合。**
  - **R以降に格納される文字列がデバイス/変数の範囲を超えた場合。**
  - **In、Rにビットデータを指定したときに、チャンネルの先頭以外を指定した場合。**
  - **In、Rに間接指定でタイマ・カウンタを指定した場合。**

MID　文字列切り出し

文字列切り出し
[R:=] MID (In, L, P);

引数/戻り値	説明	認識する型								定数
		.U	.S	.D	.L	.F	.DF	.B	.T	
In※1※4	切り出し対象文字列を指定。	—	—	—	—	—	—	—	.T	○※2
L※3	切り出す文字数（バイト数）または文字数を格納したデータを指定。（0～1999）	.U	.U	.U	.U	—	—	—	—	○
P※5	切り出しを開始する位置（バイト数）または位置を格納したデータを指定。(1～1999)	.U	.U	.U	.U	—	—	—	—	○
R※1※4	演算結果が格納される戻り値。省略可能。省略時、内部レジスタに格納。	—	—	—	—	—	—	—	.T	—

※1　ビットデータでチャンネルの先頭を指定。
※2　定数で文字列を指定する場合は、' 'で囲んで入力。
※3　Lには切り出し文字からの文字数(バイト数)を、0～1999の範囲で指定。
　　0を指定した場合、戻り値にはNUL(00H)が1文字分だけ格納。
※4　T, C, CR, Zは使用不可。
※5　Inで指定した文字列の先頭文字は1。

●動作説明

Inに格納されている文字列を、Pで指定された位置(バイト数)から、Lで指定された文字数(先頭文字を1とするバイト数)分を切り出し、終端コードNUL(00H)を付加して、結果をRに格納します。

●書式例

strResult := **MID** (strValue, uiLength, uiPosition);

SRPLC　文字列部分置換

文字列部分置換
[R:=] SRPLC (In, ByStr, Pos, Count);

引数/戻り値	説明	認識する型								定数
		.U	.S	.D	.L	.F	.DF	.B	.T	
In※1※5	置換対象となる文字列を格納した対象、または置換対象文字列を指定。	—	—	—	—	—	—	—	.T	○※2
ByStr※1※5	置換する文字列を格納した対象、または置換文字列を指定。	—	—	—	—	—	—	—	.T	○※2
Pos※6	置換開始位置（バイト数）を指定。(0～1999)	.U	.U	.U	.U	—	—	—	—	○※3
Count※4	置換する文字数（バイト数）を指定。(0～1999)	.U	.U	.U	.U	—	—	—	—	○※3
R※1※5	関数が処理した結果の値を格納。省略可能。省略時、内部レジスタに格納	—	—	—	—	—	—	—	.T	—

※1　ビットデータではチャンネルの先頭を指定。
※2　定数で文字列を指定する場合は、' 'で囲んで入力。
※3　16# (16進数指定) は使用不可。
※4　Countには置換開始位置からの文字数(バイト数)を、0～1999の範囲で指定。
※5　T, C, CR, Zは使用不可。
※6　Inで指定した文字列の先頭文字は0。

●動作説明

Inを先頭に格納されている文字列の、**Pos**で指定された位置(バイト数)から、**Count**で指定された文字数(バイト数)分の文字列を、**ByStr**を先頭に格納されている文字列で置き換え、終端コードNUL(00H)を付加して、結果を**R**に格納します。

「演算フラグについて（CR2009～CR2012）」（1-8ページ）
詳細については、『命令語リファレンスマニュアル』「文字処理命令」を参照してください。

●書式例

strString := **SRPLC** (strAllString, strReplace, INT#10, INT#20);

ポイント

- **次のいずれかの条件が成立した場合、終端コード(NUL(00H))が格納されます。**
  - **In、ByStrで指定した文字列に終端コード(NUL(00H))が無い場合。**
  - **終端コード(NUL(00H))を含む文字列が2000文字を超えた場合。**
  - **Inの文字列長 ＜Posの場合。**
  - **置換後の文字列長が終端コード(NUL(00H))を含めて2000文字を超える場合。**
  - **In、ByStr、Rにビットデータを指定したときに、チャンネルの先頭以外を指定した場合。**
  - **In、ByStr、Rに間接指定でタイマ・カウンタを指定した場合。**

REPLACE 文字列部分置換

文字列部分置換
[R:=] REPLACE (In1, In2, L, P);

引数/戻り値	説明	認識する型								定数
		.U	.S	.D	.L	.F	.DF	.B	.T	
In1※1※5	置換対象となる文字列を格納した対象、または置換対象文字列を指定。	—	—	—	—	—	—	—	.T	○※2
In2※1※5	置換する文字列を格納した対象、または置換文字列を指定。	—	—	—	—	—	—	—	.T	○※2
L※4	置換する文字数（バイト数）を指定。(0～1999)	.U	.U	.U	.U	—	—	—	—	○※3
P	置換開始位置（バイト数）を指定。(1～1999)	.U	.U	.U	.U	—	—	—	—	○※3
R※1※5	関数が処理した結果の値が格納される戻り値。省略可能。省略時、内部レジスタに格納。	—	—	—	—	—	—	—	.T	—

※1　ビットデータではチャンネルの先頭を指定。
※2　定数で文字列を指定する場合は、''で囲んで入力。
※3　16# (16進数指定) は使用不可。
※4　Lには置換開始位置からの文字数(バイト数)を、0～1999の範囲で指定。
※5　T, C, CR, Zは使用不可。

●動作説明

In1に格納されている文字列の**P**で指定された位置(先頭文字を1とするバイト数)から、**L**で指定された文字数(バイト数)分を、置換文字列**In2**で置換し、終端コードNUL(00H)を付加した結果を**R**に格納します。

本関数は「SRPLC関数」と、引数順番とPが取りうる値が異なりますが、関数の動作は同一です。
動作の詳細については、「SRPLC関数」（7-72ページ）を参照してください。

●書式例

strString := **REPLACE** (strAllString, strReplace, uiLength, uiPosition);

SINS　文字列挿入

文字列挿入
[R:=] SINS (In, ByStr, Pos);

引数/戻り値	説明	認識する型								定数
		.U	.S	.D	.L	.F	.DF	.B	.T	
In※1※5	挿入対象となる文字列を格納した対象、または挿入対象文字列を指定。	—	—	—	—	—	—	—	.T	○※2
ByStr※1※5	挿入する文字列を格納した対象、または挿入文字列を指定。	—	—	—	—	—	—	—	.T	○※2
Pos※6	挿入開始位置（バイト数）を指定。(0～1999)	.U	.U	.U	.U	—	—	—	—	○※3
R※3※5	変換結果が格納される戻り値。.Tの型を使用。省略可能。省略時、内部レジスタに格納。	—	—	—	—	—	—	—	.T	—

※1　ビットデータではチャンネルの先頭を指定。
※2　定数で文字列を指定する場合は、' 'で囲んで入力。
※3　16# (16進数指定) は使用不可。
※4　Posには挿入する位置として、左端(先端)からの文字数(バイト数)を、0～1999の範囲で指定。
※5　T, C, CR, Zは使用不可。
※6　Inで指定した文字列の先頭文字は0。

●動作説明

Inを先頭に格納されている文字列の、Posで指定された位置(バイト数)へ、ByStrを先頭に格納されている文字列を挿入し、終端コードNUL(00H)を付加して、結果をRに格納します。

「演算フラグについて（CR2009～CR2012）」（1-8ページ）
詳細については、『命令語リファレンスマニュアル』「文字処理命令」を参照してください。

●書式例

strString := **SINS** (strAllString, strInsert, INT#10);

ポイント
- **次のいずれかの条件が成立した場合、終端コード(NUL(00H))が格納されます。**
  - **In、ByStrで指定した文字列に終端コード(NUL(00H))が無い場合。**
  - **終端コード(NUL(00H))を含む文字列が2000文字を超えた場合。**
  - **Inの文字列長＜Posの場合。**
  - **挿入後の文字列長が終端コード(NUL(00H))を含めて2000文字を超える場合。**
  - **In、ByStr、Rにビットデータを指定したときに、チャンネルの先頭以外を指定した場合。**
  - **In、ByStr、Rに間接指定でタイマ・カウンタを指定した場合。**

INSERT　文字列挿入

[R:=] INSERT (In1, In2, P);
文字列挿入

引数/戻り値	説明	認識する型								定数
		.U	.S	.D	.L	.F	.DF	.B	.T	
In1※1※5	挿入対象となる文字列を格納した対象、または挿入対象文字列を指定。	—	—	—	—	—	—	—	.T	○※2
In2※1※5	挿入する文字列を格納した対象、または挿入文字列を指定。	—	—	—	—	—	—	—	.T	○※2
P※6	挿入開始位置（バイト数）を指定。(0～1999)	.U	.U	.U	.U	—	—	—	—	○※3
R※1※5	変換結果が格納される戻り値。.Tの型を使用。省略可能。省略時、内部レジスタに格納。	—	—	—	—	—	—	—	.T	—

※1　ビットデータではチャンネルの先頭を指定。
※2　定数で文字列を指定する場合は、' 'で囲んで入力。
※3　16# (16進数指定) は使用不可。
※4　Pには挿入する位置として、左端(先端)からの文字数(バイト数)を、0～1999の範囲で指定。
※5　T, C, CR, Zは使用不可。
※6　In1で指定した文字列の先頭文字は0。

●動作説明

本関数は「SINS関数」と関数名が異なるだけで、動作等に違いはありません。
動作の詳細については、「SINS関数」（7-74ページ）を参照してください。

SDEL　文字列の部分削除

[R:=] SDEL (In, Pos, Count);
文字列部分削除

引数/戻り値	説明	認識する型								定数
		.U	.S	.D	.L	.F	.DF	.B	.T	
In※1※6	削除対象となる文字列を格納した対象、または削除対象文字列を指定。	—	—	—	—	—	—	—	.T	○※2
Pos※3※4	削除開始位置（バイト数）を指定。(0～1999)	.U	.U	.U	.U	—	—	—	—	○
Count※3※5	削除する文字数（バイト数））を指定。(0～1999)	.U	.U	.U	.U	—	—	—	—	○
R※1※6	変換結果が格納される戻り値。.Tの型を使用。省略可能。省略時、内部レジスタに格納。	—	—	—	—	—	—	—	.T	—

※1　ビットデータではチャンネルの先頭を指定。
※2　定数で文字列を指定する場合は、' 'で囲んで入力。
※3　16# (16進数指定) は使用不可。
※4　Posには削除する位置として、左端(先端)からの文字数(バイト数)を、0～1999の範囲で指定。
※5　CountにはPosからの文字数(バイト数)を、0～1999の範囲で指定。
※6　T, C, CR, Zは使用不可。

●動作説明

Inを先頭に格納されている文字列の、Posで指定された位置(先頭文字を0とするバイト数)からCountで指定された文字数(バイト数)分の文字列を削除し、終端コードNUL(00H)を付加して、結果をRに格納します。

「演算フラグについて（CR2009～CR2012）」（1-8ページ）
詳細については、『命令語リファレンスマニュアル』「文字処理命令」を参照してください。

●書式例

strString := **SDEL** (strAllString, INT#10, INT#20);

ポイント
- 次のいずれかの条件が成立した場合、終端コード(NUL(00H))が格納されます。
  - Inで指定した文字列に終端コード(NUL(00H))が無い場合。
  - 終端コード(NUL(00H))を含む文字列が2000文字を超えた場合。
  - Inの文字列長＜Posの場合。
  - R以降に格納される文字列がデバイス/変数の範囲を超えた場合。
  - In、Rにビットデータを指定したときに、チャンネルの先頭以外を指定した場合。
  - In、Rに間接指定でタイマ・カウンタを指定した場合。

DELETE　文字列の部分削除

文字列部分削除
[R:=] DELETE (In, L, P);

引数/戻り値	説明	認識する型								定数
		.U	.S	.D	.L	.F	.DF	.B	.T	
In※1※5	削除対象となる文字列を格納した対象、または削除対象文字列を指定。	—	—	—	—	—	—	—	.T	○※2
L※3※4	削除する文字数（バイト数））を指定。(1～1999)	.U	.U	.U	.U	—	—	—	—	○
P※3	削除開始位置（バイト数）を指定。(1～1999)	.U	.U	.U	.U	—	—	—	—	○
R※1※5	変換結果が格納される戻り値。.Tの型を使用。省略可能。省略時、内部レジスタに格納。	—	—	—	—	—	—	—	.T	○

※1　ビットデータではチャンネルの先頭を指定。
※2　定数で文字列を指定する場合は、' 'で囲んで入力。
※3　16# (16進数指定) は使用不可。
※4　Lには削除開始位置からの文字数(バイト数)を、0～1999の範囲で指定。
※5　T, C, Zは使用不可。

●動作説明

Inに格納されている文字列を、Pで指定された位置(先頭文字を1とするバイト数)から、Lで指定された文字数(バイト数)分削除し、終端コードNUL(00H)を付加して、結果をRに格納します。

「演算フラグについて（CR2009～CR2012）」（1-8ページ）
詳細については、『命令語リファレンスマニュアル』「文字処理命令」を参照してください。

●書式例

strString := **DELETE** (strAllString, INT#10, INT#20);

ポイント
- 次のいずれかの条件が成立した場合、終端コード(NUL(00H))が格納されます。
  - Inで指定した文字列に終端コード(NUL(00H))が無い場合。
  - 終端コード(NUL(00H))を含む文字列が2000文字を超えた場合。
  - Inの文字列長＜Lの場合。
  - R以降に格納される文字列がデバイス/変数の範囲を超えた場合。
  - In、Rにビットデータを指定したときに、チャンネルの先頭以外を指定した場合。
  - In、Rに間接指定でタイマ・カウンタを指定した場合。

TRIM　文字列両端の半角スペース・タブ削除

[R:=] TRIM (In);
（トリム）

引数/戻り値	説明	認識する型								定数
		.U	.S	.D	.L	.F	.DF	.B	.T	
In※1	トリム対象となる文字列を格納した対象、またはトリム対象文字列を指定。	–	–	–	–	–	–	–	.T	○※2
R※1※3	トリム後の文字列が格納される戻り値。省略可能。省略時、変換結果は内部レジスタに格納。	–	–	–	–	–	–	–	.T	–

※1　ビットデータではチャンネルの先頭を指定。
※2　定数で文字列を指定する場合は、' 'で囲んで入力。
※3　T, C, Zは使用不可。

●動作説明

Inを先頭に格納されている文字列の両端(文字列の先頭と末尾)にある半角スペース(20H)とタブ(09H)をすべて削除し、終端コードNUL(00H)を付加して、結果をRに格納します。

「演算フラグについて（CR2009～CR2012）」（1-8ページ）
詳細については、『命令語リファレンスマニュアル』「文字処理命令」を参照してください。

●書式例

strString := **TRIM** (strAllString);

LTRIM　文字列左端の半角スペース・タブ削除

[R:=] LTRIM (In);（左トリム）

引数/戻り値	説明	認識する型								定数
		.U	.S	.D	.L	.F	.DF	.B	.T	
In※1	トリム対象となる文字列を格納した対象、またはトリム対象文字列を指定。	–	–	–	–	–	–	–	.T	○※2
R※1※3	トリム後の文字列が格納される戻り値。省略可能。省略時、変換結果は内部レジスタに格納。	–	–	–	–	–	–	–	.T	–

※1　ビットデータではチャンネルの先頭を指定。
※2　定数で文字列を指定する場合は、''で囲んで入力。
※3　T, C, Zは使用不可。

●動作説明

Inを先頭に格納されている文字列の左端(文字列の先頭)にある半角スペース(20H)とタブ(09H)をすべて削除し、終端コードNUL(00H)を付加して、結果をRに格納します。

「演算フラグについて（CR2009～CR2012）」（1-8ページ）
詳細については、『命令語リファレンスマニュアル』「文字処理命令」を参照してください。

●書式例

strString := **LTRIM** (strAllString);

RTRIM　文字列右端の半角スペース・タブ削除

[R:=] RTRIM (In);（右トリム）

引数/戻り値	説明	認識する型								定数
		.U	.S	.D	.L	.F	.DF	.B	.T	
In※1	トリム対象となる文字列を格納した対象、またはトリム対象文字列を指定。	–	–	–	–	–	–	–	.T	○※2
R※1※3	トリム後の文字列が格納される戻り値。省略可能。省略時、変換結果は内部レジスタに格納。	–	–	–	–	–	–	–	.T	–

※1　ビットデータではチャンネルの先頭を指定。
※2　定数で文字列を指定する場合は、''で囲んで入力。
※3　T, C, Zは使用不可。

●動作説明

Inを先頭に格納されている文字列の右端(文字列の末尾)にある半角スペース(20H)とタブ(09H)をすべて削除し、終端コードNUL(00H)を付加して、結果をRに格納します。

「演算フラグについて（CR2009～CR2012）」（1-8ページ）
詳細については、『命令語リファレンスマニュアル』「文字処理命令」を参照してください。

●書式例

strString := **RTRIM** (strAllString);

STRIM　文字列指定位置の半角スペース・タブ削除

[R:=] STRIM (In, Param);
（指定位置トリム）

引数/戻り値	説明	認識する型								定数
		.U	.S	.D	.L	.F	.DF	.B	.T	
In※1	トリム対象となる文字列を格納した対象、またはトリム対象文字列を指定。	–	–	–	–	–	–	–	.T	○※2
Param	トリム対象となる位置を指定。（1～3）	.U	.U	.U	.U	–	–	–	–	○
R※1※3	トリム後の文字列が格納される戻り値。省略可能。省略時、変換結果は内部レジスタに格納。	–	–	–	–	–	–	–	.T	–

※1　ビットデータではチャンネルの先頭を指定。
※2　定数で文字列を指定する場合は、''で囲んで入力。
※3　T, C, Zは使用不可。

●動作説明

Inを先頭に格納されている文字列から、**Param**で指定した位置にある半角スペースおよびタブを全て削除し、終端コードNUL(00H)を付加して、結果を**R**に格納します。
削除位置指定パラメータで指定する内容は次のようになります。

Param値	内容
1	文字列左側の半角スペースおよびタブをすべて削除します。(LTRIM関数と同じ動作)
2	文字列右側の半角スペースおよびタブをすべて削除します。(RTRIM関数と同じ動作)
3	文字列両端の半角スペースおよびタブをすべて削除します。(TRIM関数と同じ動作)

ポイント　**上記以外のパラメータを指定した場合は、関数を実行せずにInがRに格納されます。**

「演算フラグについて（CR2009～CR2012）」（1-8ページ）
詳細については、『命令語リファレンスマニュアル』「文字処理命令」を参照してください。

●書式例

strString := **STRIM** (strAllString, INT#3);

SPLIT　文字列分離

[R:=] SPLIT (Src, Delimiter, InOut1, InOut2, InOutN);

（スプリット）

引数/戻り値	説明	認識する型								定数
		.U	.S	.D	.L	.F	.DF	.B	.T	
Src※1	分離元となる文字列を格納した対象、または分離対象文字列を指定。	—	—	—	—	—	—	—	.T	○※2
Delimiter※1	区切り文字列を格納した対象、または区切り対象文字列を指定。	—	—	—	—	—	—	—	.T	○※2
InOut1※1	分離後の文字列1を格納する対象を指定。	—	—	—	—	—	—	—	.T	—
InOut2※1	分離後の文字列2を格納する対象を指定。	—	—	—	—	—	—	—	.T	—
・ ・ ・	・ ・ ・	—	—	—	—	—	—	—	.T	—
InOutN※1※3	分離後の文字列nを格納する対象を指定する。	—	—	—	—	—	—	—	.T	—
R※1※4※5	分離数が格納される戻り値。省略可能。省略時、変換結果は内部レジスタに格納。	.U	.S	.D	.L	.F	.DF	—	—	—

※1　ビットデータではチャンネルの先頭を指定。
※2　定数で文字列を指定する場合は、' 'で囲んで入力。
※3　引数の個数nは、1～10の範囲で指定。
※4　T, C, Zは使用不可。
※5　Rに.U以外の型を指定した場合、変換時に自動的に型変換を実行。

●動作説明

Srcを Delimiterで分離し、結果をInOut1～InOutN分離後文字列に格納します。
Rには、分離した文字列の数が格納されます。
Src：'18,1A,35,E0,C3,EB'、Delimiter：','(カンマ)のとき

参 考
- R分離数=0の場合、関数は実行されません。
- 0＜R分離数＜nの場合、余分な引数に対しては終端コード(NUL)のみ格納します。
- R分離数=nの場合、InOutNに区切り文字を含んでいる(まだ分離できる余地が残る)場合があります。
  分離できる余地があるか判断するだけであれば、SFIND関数でInOutN内の区切り文字の有無を調べます。
  分離できる余地があった場合、InOutNをSrcに指定して再度SPLIT関数を実行することで、分離できます。

「演算フラグについて（CR2009～CR2012）」（1-8ページ）

●書式例

uiValue := **SPLIT** (strAllString, ',', strString1, strString2, strString3);

SFIND　文字列検索

[R:=] SFIND (In, Pattern, Pos);
文字列検索

引数/戻り値	説明	認識する型								定数
		.U	.S	.D	.L	.F	.DF	.B	.T	
In※1※5	検索対象となる文字列を格納した対象、または検索対象文字列を指定。	—	—	—	—	—	—	—	.T	○※2
Pattern※1※5	検索する文字列を格納した対象、または検索文字列を指定。	—	—	—	—	—	—	—	.T	○※2
Pos※3※4	検索開始位置(バイト数)を指定。 (0～1999)	.U	.U	.U	.U	—	—	—	—	○※3
R※6	変換結果が格納される戻り値。省略可能。省略時、変換結果は内部レジスタに格納。	.U	.S	.D	.L	.F	.DF※7	—	—	—

※1　ビットデータではチャンネルの先頭を指定。
※2　定数で文字列を指定する場合は、' 'で囲んで入力。
※3　16# (16進数指定) は使用不可。
※4　Posには検索を開始する位置として、Inの先頭からの文字数(バイト数)を、0～1999の範囲で指定。
※5　T, C, Zは使用不可。
※6　Rに.U以外の型を指定した場合、変換時に自動的に型変換を実行。
※7　デバイスZを指定した場合、「.DF」サフィックスは指定不可。

●動作説明

実行条件が成り立つ(TRUE)とき、Inを先頭に格納されている検索対象文字列の、Posで指定された位置から、Patternを先頭に格納されている検索文字列を検索します。
検索結果として、Pos以降で初めて見つかった検索文字列の先頭文字位置(Inからのバイト数)が、Rに格納されます。
またはPatternの文字列長が0の場合で指定した文字列が見つからなかった場合、またはPatternの文字列長が0の場合、Rには16# (16進数指定) FFFFが格納されます。

「演算フラグについて(CR2009～CR2012)」(1-8ページ)
詳細については、『命令語リファレンスマニュアル』「文字処理命令」を参照してください。

●書式例

uiValue := **SFIND** (strAllString, strString, INT#8);

ポイント

次のいずれかの条件が成立した場合、Rには0が格納されます。

- **In、Patternで指定した文字列に終端コード(NUL(00H))が無い場合。**
- **終端コード(NUL(00H))を含む文字列が2000文字を超えた場合。**
- **(Inの文字列長)＜(Pos+Patternの文字列長)の場合(終端コード(NUL(00H))は文字列長には含みません)。**
- **In、Patternにビットデータを指定したときに、チャンネルの先頭以外を指定した場合。**
- **In、Patternに間接指定でタイマ・カウンタを指定した場合。**

SFINDN　文字列検索

[R:=] SFINDN (In, Pattern, InLen, Pos);
文字列検索N

引数/戻り値	説明	認識する型								定数
		.U	.S	.D	.L	.F	.DF	.B	.T	
In※1※3	検索対象となる文字列を格納した対象、または検索対象文字列を指定。	—	—	—	—	—	—	—	.T	○※2
Pattern※1※3	検索する文字列を格納した対象、または検索文字列を指定。	—	—	—	—	—	—	—	.T	○※2
InLen	検索対象の文字列長（バイト数）を指定。	.U	.U	.U	.U	—	—	—	—	○
Pos	検索開始バイト位置を指定。	.U	.U	.U	.U	—	—	—	—	○
R※4	変換結果が格納される戻り値。省略可能。省略時、変換結果は内部レジスタに格納。	.U	.S	.D	.L	.F	.DF※5	—	.T	—

※1　ビットデータではチャンネルの先頭を指定。
※2　定数で文字列を指定する場合は、' 'で囲む。
※3　T, C, Zは使用不可。
※4　Rに.U以外の型を指定した場合、変換時に自動的に型変換を実行。
※5　デバイスZを指定した場合、「.DF」サフィックスは指定不可。

●動作説明

実行条件が成り立つ(TRUE)とき、Inを先頭に、検索対象バイト数の文字列のPosから、Patternを先頭に格納されている文字列を検索します。
Rには、InLenで指定した文字列中のPosで指定された位置以降で、はじめて見つかったPatternで指定した文字列の開始位置(バイト数)が格納されます。
Patternで指定した文字列が見つからなかった場合、またはPatternの文字列長が0の場合、
Rには16# (16進数指定) FFFFが格納されます。

「演算フラグについて（CR2009～CR2012）」（1-8ページ）
詳細については、『命令語リファレンスマニュアル』「文字処理命令」を参照してください。

●書式例

uiValue := **SFINDN** (strAllString, strString, INT#50 , INT#4);

INSTR　文字列検索

[R:=] INSTR (Pos, In, Pattern);
文字列検索N

引数/戻り値	説明	認識する型								定数
		.U	.S	.D	.L	.F	.DF	.B	.T	
Pos※4	検索開始位置（バイト数）を指定。 (0～1999)	.U	.U	.U	.U	—	—	—	—	○※3
In※1※5	検索対象となる文字列を格納した対象、または検索対象文字列を指定。	—	—	—	—	—	—	—	.T	○※2
Pattern※1※5	検索する文字列を格納した対象、または検索文字列を指定。	—	—	—	—	—	—	—	.T	○※2
R※6	演算結果が格納される戻り値。省略可能。省略時、変換結果は内部レジスタに格納。	.U	.S	.D	.L	.F	.DF※7	—	—	—

※1　ビットデータではチャンネルの先頭を指定。
※2　定数で文字列を指定する場合は、' 'で囲んで入力。

※3　16# (16進数指定) は使用不可。
※4　Posには検索を開始する位置として、Inの先頭からの文字数(バイト数)を、0～1999の範囲で指定。
※5　T, C, Zは使用不可。
※6　Rに.U以外の型を指定した場合、変換時に自動的に型変換を実行。
※7　デバイスZを指定した場合、「.DF」サフィックスは指定不可。

●動作説明

実行条件が成り立つ(TRUE)とき、Inを先頭に格納されている検索対象文字列の、Posで指定された位置(バイト数)から、Patternを先頭に格納されている検索文字列を検索します。
検索結果として、Pos以降で初めて見つかった検索文字列の先頭文字位置(Inバイト数)が、Rに格納されます。
Patternで指定した文字列が見つからなかった場合、またはPatternの文字列長が0の場合、
Rには16# (16進数指定) FFFFが格納されます。

「演算フラグについて（CR2009～CR2012）」（1-8ページ）
詳細については、『命令語リファレンスマニュアル』「文字処理命令」を参照してください。

●書式例

uiValue := **INSTR** (INT#8, strAllString, strString);

ポイント

次のいずれかの条件が成立した場合、Rには0が格納されます。
- **In、Patternで指定した文字列に終端コード(NUL(00H))が無い場合。**
- **終端コード(NUL(00H))を含む文字列が2000文字を超えた場合。**
- **(Inの文字列長)＜(Pos+Patternの文字列長)の場合(終端コード(NUL(00H))は文字列長には含みません)。**
- **In、Patternにビットデータを指定したときに、チャンネルの先頭以外を指定した場合。**
- **In、Patternに間接指定でタイマ・カウンタを指定した場合。**

FIND　文字列検索

[R:=] FIND (In1, In2);
文字列検索

引数/戻り値	説明	認識する型								定数
		.U	.S	.D	.L	.F	.DF	.B	.T	
In1※1※3	検索対象となる文字列を格納した対象、または検索対象文字列を指定。	—	—	—	—	—	—	—	.T	○※2
In2※1※3	検索する文字列を格納した対象、または検索文字列を指定。	—	—	—	—	—	—	—	.T	○※2
R※4	演算結果が格納される戻り値。省略可能。省略時、変換結果は内部レジスタに格納。	.U	.S	.D	.L	.F	.DF※5	—	—	—

※1　ビットデータではチャンネルの先頭を指定。
※2　定数で文字列を指定する場合は、’ ’で囲み(半角31文字まで)。
※3　T, C, Zは使用不可。
※4　Rに.U以外の型を指定した場合、変換時に自動的に型変換を実行。

●動作説明

In1に格納されている文字列から、In2で指定された文字列に一致する分を先頭から順に検索し、一致個所の先頭位置(元の文字列の先頭文字を1とするバイト数)をRに格納します。

「演算フラグについて（CR2009～CR2012）」（1-8ページ）
詳細については、『命令語リファレンスマニュアル』「文字処理命令」を参照してください。

●書式例

uiValue := **FIND**(strString, strFind);

ポイント

次のいずれかの条件が成立した場合、**R**には**0**が格納されます。

- **In1**、**In2**で指定した文字列に終端コード(NUL(00H))が無い場合。
- 終端コード(NUL(00H))を含む文字列が2000文字を超えた場合。
- (**In1**の文字列長)＜(**In2**の文字列長)の場合。
- **In1**、**In2**にビットデータを指定したときに、チャンネルの先頭以外を指定した場合。
- **In1**、**In2**に間接指定でタイマ・カウンタを指定した場合。
- **In2**の文字列長が0の場合、または文字列がみつからなかった場合は、65535が**R**に入ります。

DISS　文字列バイト分離

文字列分離
DISS ([**EN**], **Src**, **Dst**);

引数/戻り値	説明	認識する型								定数
		.U	.S	.D	.L	.F	.DF	.B	.T	
EN	実行条件。省略可能。省略時、常時(毎スキャン)実行。	—	—	—	—	—	—	.B	—	—
Src※1※3	分離対象となる文字列を格納した対象、または分離対象文字列を指定。	—	—	—	—	—	—	—	.T	○※2
Dst※1※4	分離した文字列(ワード単位)を格納する対象を指定。	.U	.U	.U	.U	—	—	—	—	
R	戻り値はなし。	—	—	—	—	—	—	—	—	

※1　ビットデータではチャンネルの先頭を指定。
※2　定数で文字列を指定する場合は、' 'で囲んで入力。
※3　T, C, CR, Zは使用不可。
※4　T, C, Zは使用不可。

●動作説明

実行条件が成り立つ(TRUE)とき、**Src**を先頭に格納されているバイト単位の文字列を、**Dst**を先頭にしてワード単位に分離して格納します。

「演算フラグについて(CR2009～CR2012)」(1-8ページ)
詳細については、『命令語リファレンスマニュアル』「文字処理命令」を参照してください。

●書式例

DISS (bEnable , strString, auiData[0]);

ポイント

次のいずれかの条件が成立した場合、**Dst**には**0**が格納されます。

- **Src**で指定した文字列に終端コード(NUL(00H))が無い場合。
- 終端コード(NUL(00H))を含む文字列が2000文字を超えた場合。
- **Src**、**Dst**にビットデータを指定したときに、チャンネルの先頭以外を指定した場合。
- **Src**、**Dst**に間接指定でタイマ・カウンタを指定した場合。

UNIS　バイト文字列結合

文字列結合
UNIS ([EN], Src, Dst, Count);

引数/戻り値	説明	認識する型								定数
		.U	.S	.D	.L	.F	.DF	.B	.T	
EN	実行条件。省略可能。省略時、常時(毎スキャン)実行。	—	—	—	—	—	—	.B	—	—
Src※1※4	結合対象となる文字列(ワード単位)を格納した対象、または結合対象文字列(ワード単位)を指定。	.U	.U	.U	.U	—	—	—	—	—
Dst※1※4	結合した文字列を格納する対象を指定。	—	—	—	—	—	—	—	.T	—
Count※1※4	結合する文字数(バイト数(ワード単位数))を指定。(1～1999)	.U	.U	.U	.U	—	—	—	—	○※2
R	戻り値はなし。	—	—	—	—	—	—	—	—	—

※1　ビットデータではチャンネルの先頭を指定。
※2　16# (16進数指定) は使用不可。
※3　Countには結合する文字(ワード単位)数を、1～1999の範囲で指定。
※4　T, C, Zは使用不可。
※5　T, C, CR, Zは使用不可。

●動作説明

実行条件が成り立つ(TRUE)とき、Srcを先頭に格納されているワード単位の文字列を、Countで指定された文字数(ワード単位数)分だけバイト単位に結合し、Dstを先頭にして、終端コードNUL(00H)を付加して格納します。

「演算フラグについて(CR2009～CR2012)」(1-8ページ)
詳細については、『命令語リファレンスマニュアル』「文字処理命令」を参照してください。

●書式例

UNIS (bEnable, auiData[0], strString, uiValue);

ポイント

次のいずれかの条件が成立した場合、関数は実行されません。

- **Countの値が0または2000以上の場合。**
- **終端コード(NUL(00H))を含む文字列が2000文字を超えた場合。**
- **Src、Dstにビットデータを指定したときに、チャンネルの先頭以外を指定した場合。**
- **Src、Dstに間接指定でタイマ・カウンタを指定した場合。**

CHR　16進アスキー文字コード→文字変換

キャラクタ変換
[R:=] CHR (In);

引数/戻り値	説明	認識する型								定数
		.U	.S	.D	.L	.F	.DF	.B	.T	
In	変換対象のASCIIコードを格納した対象、またはASCIIコードデータを指定。(16ビットデータ)	.U	.U	.U	.U	—	—	—	—	○※1
R※2	演算結果が格納される戻り値。.Tの型を使用。省略可能。省略時、変換結果は内部レジスタに格納。	—	—	—	—	—	—	—	.T	—

※1　定数の値は0～255の範囲。対象を指定し、その値が0～255以外のとき上位8ビットを切り捨てた値を使用。

※2　T, C, CR, Zは使用不可。

●動作説明

Inに格納されたASCIIコードデータを、終端コードNUL(00H)を付加した1バイトの文字列に変換し、結果を**R**に格納します。

●書式例

strString := **CHR** (uiValue);

CPSASC　CIP文字列変換

CIP文字列変換
[R:=] CPSASC (In);

引数/戻り値	説明	認識する型								定数
		.U	.S	.D	.L	.F	.DF	.B	.T	
In	変換前のCIPの文字列型データが格納されている対象の先頭を指定。	.U	.S	.D	.L	.F	.DF	.B	.T	—
R※1	変換結果が格納される戻り値。.Tの型を使用。省略可能。省略時、変換結果は内部レジスタに格納。	—	—	—	—	—	—	—	.T	—

※1　ビットデータを指定する場合は、チャンネルの先頭を指定。

●動作説明

Inを先頭に格納されているCIPの文字列型データを文字列に変換し、結果を**R**に格納します。
CPSASC関数は、CIPの文字列型データを変換し、文字列として処理するための関数です。

CIPのデータ型については、『KV EtherNet/IP機能ユーザーズマニュアル』を参照してください。
CPSASC関数を実行すると、CIPの文字列型データを変換して戻り値に格納します。
詳細については、『命令語リファレンスマニュアル』「文字処理命令」を参照してください。

●書式例

strValue := **CPSASC**(auiValue[0]);

RCPSASC　CIP文字列逆変換

CIP文字列逆変換
[R:=] RCPSASC (Src, Dst);

引数/戻り値	説明	認識する型								定数
		.U	.S	.D	.L	.F	.DF	.B	.T	
Src	変換前の文字列が格納されている対象の先頭を指定。	—	—	—	—	—	—	—	.T	○※2
Dst	変換後のCIPの文字列型データを格納する対象の先頭を指定。	.U	.S	.D	.L	.F	.DF	.B	.T	—
R※1	Dstに格納された文字数（バイト単位）が格納される戻り値。省略可能。省略時、変換結果は内部レジスタに格納。	.U	.S	.D	.L	.F	.DF	—	—	—

※1　Rに.U以外の型を指定した場合、変換時に自動的に型変換を実行。
※2　定数で文字列を指定する場合は、' 'で囲んで入力。

●動作説明

Srcに格納されている文字列をCIPの文字列型データに変換し、結果を**Dst**に格納します。変換後に**Dst**に格納された文字数(バイト単位)を**R**に格納します。
RCPSASC関数は、文字列をCIPの文字列型データに変換するための関数です。
CIPのデータ型については、『KV EtherNet/IP機能ユーザーズマニュアル』を参照してください。
RCPSASC関数を実行すると、文字列をCIPの文字列型データに変換します。

詳細については、『命令語リファレンスマニュアル』「文字処理命令」を参照してください。

●書式例

uiValue := **RCPSASC** (strValue, auiDst[0]);

CONCAT 文字列連結

[**R**:=] **CONCAT** (**In1**, **In2**, … ,**InN**);
文字列連結

引数/戻り値	説明	認識する型								定数
		.U	.S	.D	.L	.F	.DF	.B	.T	
In1	連結対象文字列を指定。	–	–	–	–	–	–	–	.T	○
In2	連結対象文字列を指定。	–	–	–	–	–	–	–	.T	○
:	:									
InN	連結対象文字列を指定。	–	–	–	–	–	–	–	.T	○
R※1	演算結果が格納される戻り値。省略可能。省略時、内部レジスタに格納。	–	–	–	–	–	–	–	.T	–

※1　**R**に演算対象と異なる型を指定した場合、変換時に自動的に型変換を実行。
※2　デバイスZを指定した場合、「.DF」サフィックスは指定不可。

●動作説明

In1から**InN**に格納された文字列を順番に連結した結果を**R**に格納します。
Nは2～10を指定可能です。

●書式例

strAllString := **CONCAT** (strString1, strString2, strString3);

「演算フラグについて（CR2009～CR2012）」（1-8ページ）
詳細については『命令語リファレンスマニュアル』「文字列処理命令」を参照してください。

8

拡張関数

データ処理関数 8-2
テーブル処理関数 8-17
時計処理関数 8-22
高速処理関数 8-25
PID関数 8-28
ログ関数 8-29
ストレージ関数 8-32
アクセスウィンドウ関数 8-45
フリーランカウンタ関数 8-47
センサ設定関数 8-48
サイクリック通信リフレッシュ関数 8-52

データ処理関数

SEG　4ビットデータの7セグメントデコード

7セグメントデコード
[R:=] SEG (In, Pos);

引数/戻り値	説明	認識する型								定数
		.U	.S	.D	.L	.F	.DF	.B	.T	
In※1	デコード対象のデータを格納した対象、またはデコード対象データを指定。	.U	.U	.D	.D	–	–	–	–	–
Pos※2	デコードする桁位置を格納した対象、またはデコード桁位置を指定。(.Uの場合0～3、.Dの場合0～7)	.U	.U	.U	.U	–	–	–	–	–
R※3	演算結果が格納される戻り値。省略可能。省略時、内部レジスタに格納。	.U	.S	.D	.L	.F	.DF※4	–	–	–

※1　**In**デコード対象にビットデータを指定した場合、連続した16/32ビットで演算。R002などチャンネルの先頭以外を指定した場合、次のチャンネルにまたがって16/32ビットで演算。
※2　**Pos**対象桁位置に対象を指定した場合で、格納された値が0～3または0～7を超えた時：
デコード対象のサフィックスが.Uの場合：指定した対象の下位2ビットの値を使用。
デコード対象のサフィックスが.Dの場合：指定した対象の下位3ビットの値を使用。
例えば、**In**デコード対象のサフィックスが.Uで、**Pos**にDM01000が指定され、その値が23(2進数：0000 0000 0001 0111)の場合、下位2ビットの値は3なので**Pos**の値として3を使用。
※3　SEG関数は、.Uの型を使用。**R**に.U以外の型を指定した場合、変換時に自動的に型変換を実行。
※4　デバイスZを指定した場合、「.DF」サフィックスは指定不可。

●動作説明

Inに格納された16ビット/32ビットデータの、**Pos**で指定された桁数の4ビット(1桁)を、7セグメント表示用データ(8ビット)にデコードして、結果を**R**に格納します。

「演算フラグについて（CR2009～CR2012）」（1-8ページ）
詳細については、『命令語リファレンスマニュアル』「データ処理命令」を参照してください。

●書式例

- **In：デコード対象が16ビット(.U/.S)の場合**
  uiValue := **SEG** (uiData, INT#2);

- **In：デコード対象が32ビット(.D/.L)の場合**
  uiValue := **SEG** (udiData, INT#5);

BCNT　対象のONビットカウント

[R:=] BCNT (In);
（ビットカウント）

引数/戻り値	説明	認識する型								定数
		.U	.S	.D	.L	.F	.DF	.B	.T	
In	演算対象、または演算対象データを指定。	.U	.U	.D	.D	—	—	—	—	—
R※1	演算結果が格納される戻り値。省略可能。省略時、内部レジスタに格納。	.U	.S	.D	.L	.F	.DF※2	—	—	—

※1　Rに.U以外の型を指定した場合、変換時に自動的に型変換を実行。
※2　デバイスZを指定した場合、「.DF」サフィックスは指定不可。

●動作説明

Inに格納された(16ビット/32ビット)バイナリデータの、ONしているビットの数を数え、結果を戻り値に格納します。

「演算フラグについて（CR2009～CR2012）」（1-8ページ）
詳細については、『命令語リファレンスマニュアル』「データ処理命令」を参照してください。

●書式例

uiValue := **BCNT** (uiData);

DCNT　指定ブロック内の検出データカウント

[R:=] DCNT (Pattern, In, Count);
（データカウント）

引数/戻り値	説明	認識する型								定数
		.U	.S	.D	.L	.F	.DF	.B	.T	
Pattern※1	検出する値を格納した対象、または検出値を指定。	.U	.U	.D	.D	.D	—	—	—	○
In※1	検出するデータブロックの先頭を指定。	型はPatternと同じ。					—	—	—	—
Count	検出するブロックのデータ数の格納先、または検出データ数を指定。	.U	.U	.U	.U	—	—	—	—	○
R※2	演算結果が格納される戻り値。省略可能。省略時、内部レジスタに格納。	.U	.S	.D	.L	.F	.DF※3	—	—	—

※1　ビットデータを指定した場合、連続した16/32ビットで演算。R002などチャンネルの先頭以外を指定した場合、次のチャンネルにまたがって16/32ビットで演算。
※2　Rに.U以外の型を指定した場合、変換時に自動的に型変換を実行。
※3　デバイスZを指定した場合、「.DF」サフィックスは指定不可。

●動作説明

Inを先頭とする、Countで指定された個数のブロックデータを、Patternで指定されたデータとそれぞれ比較し、一致したデータの数をR戻り値に格納します。

「演算フラグについて（CR2009～CR2012）」（1-8ページ）
詳細については、『命令語リファレンスマニュアル』「データ処理命令」を参照してください。

●書式例

uiValue := **DCNT** (uiData , auiValue[0], uiCount);

ポイント　Count検出データ数の値が0の場合、Pattern検出対象の値が、R戻り値に格納されます。

SER　指定ブロック内のデータ検出

[R:=] SER (Pattern, In, Count);

引数/戻り値	説明	認識する型								定数
		.U	.S	.D	.L	.F	.DF	.B	.T	
Pattern※1	検出する値を格納した対象、または検出値を指定。	.U	.U	.D	.D	.D	–	–	–	○
In※1	検出するデータブロックの先頭を指定。	型はPatternと同じ。					–	–	–	–
Count	検出するブロックのデータ数の格納先、または検出データ数を指定。	.U	.U	.U	.U	–	–	–	–	○
R※2	演算結果が格納される戻り値。省略可能。省略時、内部レジスタに格納。	.U	.S	.D	.L	.F	.DF※3	–	–	–

※1　ビットデータを指定した場合、連続した16/32ビットで演算。R002などチャンネルの先頭以外を指定した場合、次のチャンネルにまたがって16/32ビットで演算。
※2　Rに.U以外の型を指定した場合、変換時に自動的に型変換を実行。
※3　デバイスZを指定した場合、「.DF」サフィックスは指定不可。

●動作説明

Inを先頭とする、Countで指定された個数のブロックデータを、Patternで指定されたデータとそれぞれ比較し、一致したデータが格納されているデバイス番号をRに格納します。
一致するデータが複数個検出された場合は、一番小さい番号が格納されます。

「演算フラグについて（CR2009～CR2012）」（1-8ページ）
詳細については、『命令語リファレンスマニュアル』「データ処理命令」を参照してください。

●書式例

uiValue := **SER** (uiData, auiValue[0], INT#40);

ポイント
- データがみつからなかった場合や、指定したCount検出データ数が0だった場合は、Pattern検出対象の値がR戻り値に格納されます。
- 検出範囲先頭にビットデータを指定した場合、検索結果(戻り値)には16ビット単位ではなくR(MR、LR)0000を基点とした10進単位に変換された値が格納されます。
  (例)検索結果がMR82312の場合：82312→823×16+12＝13180が格納されます。

DSER　指定範囲内のデータ検索

[R:=] DSER (Pattern, Start, Count, Result);

データ検索

引数/戻り値	説明	認識する型								定数
		.U	.S	.D	.L	.F	.DF	.B	.T	
Pattern※1	検索データ、または検索データの格納先を指定。	.U	.U	.D	.D	.D	–	–	–	○
Start※1	検索する範囲の先頭を指定。	型はPatternと同じ。					–	–	–	–
Count※1	検索する対象の個数、または個数の格納先を指定。	.U	.U	.U	.U	–	–	–	–	○
Result※1	検出結果（検出個数）を格納する対象を指定。	.U	.U	.U	.U	–	–	–	–	–
R※2	演算結果が格納される戻り値。検出位置（検出範囲先頭からのオフセット）を格納する対象を指定。省略可能。省略時、内部レジスタに格納。	.U	.S	.D	.L	.F	.DF※3	–	–	–

※1　ビットデータを指定した場合、連続した16/32ビットで演算。R002などチャンネルの先頭以外を指定した場合、次のチャンネルにまたがって16/32ビットで演算。
※2　Rに.U以外の型を指定した場合、変換時に自動的に型変換を実行。
※3　デバイスZを指定した場合、「.DF」サフィックスは指定不可。

●動作説明

Patternに格納されたバイナリデータをキーワードとして、Startから始まるCountのワードデータの範囲を検索します。

- Patternと一致したワードデータの個数が、Resultに格納されます。
- 最初に一致したワードデータの位置(Patternからの相対位置)をRに格納します。
- 一致するデータが見つからなかった場合、ResultおよびRには0を格納します。

●書式例

uiValue := **DSER** (uiData, auiValue[0], uiCount, uiResult);

MAX　最大

[R:=] MAX (In1, In2, …, InN);

最大

引数/戻り値	説明	認識する型								定数
		.U	.S	.D	.L	.F	.DF	.B	.T	
In1※3※4	選択対象を指定。	.U	.S	.D	.L	.F	.DF※2	–	–	○
In2※3※4	選択対象を指定。	.U	.S	.D	.L	.F	.DF※2	–	–	○
:	:									
InN※3※4	選択対象を指定。（Nの最大値は10）	.U	.S	.D	.L	.F	.DF※2	–	–	○
R※1	演算結果が格納される戻り値。省略可能。省略時、内部レジスタに格納。	.U	.S	.D	.L	.F	.DF※2	–	–	-

※1　Rに演算対象と異なる型を指定した場合、変換時に自動的に型変換を実行。
※2　デバイスZを指定した場合、「.DF」サフィックスは指定不可。
※3　ビットデータを指定した場合、連続した16/32/64ビットで演算。R002などチャンネルの先頭以外を指定した場合、次のチャンネルにまたがって16/32/64ビットで演算。
※4　データ型が.Fまたは.DFの場合、T、Cは指定不可。

●動作説明

In1から**InN**のうち最大の結果を検索し、最大値が**R**に格納されます。Nは2~10を指定可能です。

●書式例

rValue := **MAX** (rValue1, rValue2, rValue3);

MAX_RANGE　指定ブロック内の最大値検索

[R:=] MAX_RANGE (In, Count);
最大値検索

引数/戻り値	説明	認識する型								定数
		.U	.S	.D	.L	.F	.DF	.B	.T	
In※1※2※3	検索する範囲の先頭を指定。	.U	.S	.D	.L	.F	.DF※5	—	—	—
Count※1	検索する範囲のデータ数の格納先、または検索範囲データ数を指定。	.U	.U	.U	.U	—	—	—	—	○
R※4	演算結果が格納される戻り値。省略可能。省略時、内部レジスタに格納。	.U	.S	.D	.L	.F	.DF※5	—	—	—

※1　サフィックスにより扱える数値の範囲が異なる。
※2　ビットデータを指定した場合、連続した16/32/64ビットで演算。R002などチャンネルの先頭以外を指定した場合、次のチャンネルにまたがって16/32/64で演算。
※3　**In**の型が.Fまたは.DFの場合、T,Cは指定不可。
※4　**R**に演算対象と異なる型を指定した場合、変換時に自動的に型変換を実行。
※5　デバイスZを指定した場合、「.DF」サフィックスは指定不可。

●動作説明

Inを先頭として、**Count**で指定されたワード数の範囲に格納されたバイナリデータ(16ビット/32ビット/64ビット)の中で最大値を検索します。(検索対象が32ビットデータのとき、検索範囲は指定されたデータ数×2ワード、64ビットデータのとき、検索範囲は指定されたデータ数×4ワード)
検索の結果、最大値が**R**に格納されます。

「演算フラグについて（CR2009～CR2012）」（1-8ページ）
詳細については、『命令語リファレンスマニュアル』「データ処理命令」を参照してください。

●書式例

uiValue := **MAX_RANGE** (auiValue[0], INT#32);

ポイント　**Countの値が0の場合、Rには0が格納されます。**

MIN　最小

[R:=] MIN (In1, In2, …, InN);
最小

引数/戻り値	説明	認識する型								定数
		.U	.S	.D	.L	.F	.DF	.B	.T	
In1※3※4	選択対象を指定。	.U	.S	.D	.L	.F	.DF※2	—	—	○
In2※3※4	選択対象を指定。	.U	.S	.D	.L	.F	.DF※2	—	—	○
:	:									
InN※3※4	選択対象を指定。(Nの最大値は10)	.U	.S	.D	.L	.F	.DF※2	—	—	○
R※1	演算結果が格納される戻り値。省略可能。省略時、内部レジスタに格納。	.U	.S	.D	.L	.F	.DF※2	—	—	-

※1　Rに演算対象と異なる型を指定した場合、変換時に自動的に型変換を実行。
※2　デバイスZを指定した場合、「.DF」サフィックスは指定不可。
※3　ビットデータを指定した場合、連続した16/32/64ビットで演算。R002などチャンネルの先頭以外を指定した場合、次のチャンネルにまたがって16/32/64ビットで演算。
※4　データ型が.Fまたは.DFの場合、T、Cは指定不可。

●動作説明

In1からInNのうち最小の結果を検索し、最小値がRに格納されます。Nは2~10を指定可能です。

●書式例

rValue := **MIN** (rValue1, rValue2, rValue3);

MIN_RANGE　指定ブロック内の最小値検索

[R:=] MIN_RANGE (In, Count);
最小値検索

引数/戻り値	説明	認識する型								定数
		.U	.S	.D	.L	.F	.DF	.B	.T	
In※1※2※3	検索する範囲の先頭を指定。	.U	.U	.D	.D	.F	.DF※5	—	—	—
Count※1	検索する範囲のデータ数の格納先、または検索範囲データ数を指定。	.U	.U	.U	.U	—	—	—	—	○
R※4	演算結果が格納される戻り値。省略可能。省略時、内部レジスタに格納。	.U	.S	.D	.L	.F	.DF※5	—	—	—

※1　サフィックスにより扱える数値の範囲が異なる。
※2　ビットデータを指定した場合、連続した16/32/64ビットで演算。R002などチャンネルの先頭以外を指定した場合、次のチャンネルにまたがって16/32/64で演算。
※3　Inの型が.Fまたは.DFの場合、T,Cは指定不可。
※4　Rに演算対象と異なる型を指定した場合、変換時に自動的に型変換を実行。
※5　デバイスZを指定した場合、「.DF」サフィックスは指定不可。

●動作説明

Inを先頭として、Countで指定されたワード数の範囲に格納されたバイナリデータ(16ビット/32ビット/64ビット)の中で最小値を検索します。(検索対象が32ビットデータのとき、検索範囲は指定されたデータ数×2ワード、64ビットデータのとき、検索範囲は指定されたデータ数×4ワード)検索の結果、最小値がRに格納されます。

「演算フラグについて（CR2009～CR2012）」（1-8ページ）
詳細については、『命令語リファレンスマニュアル』「データ処理命令」を参照してください。

●書式例

uiValue := **MIN_RANGE** (auiValue[0], INT#32);

ポイント　**Countの値が0の場合、Rには0が格納されます。**

SEL　選択

[R:=] SEL (G, In0, In1);

引数/戻り値	説明	認識する型								定数
		.U	.S	.D	.L	.F	.DF	.B	.T	
G	ゲートを指定	—	—	—	—	—	—	.B	-	○
In0	選択対象を指定	.U	.S	.D	.L	.F	.DF※2	.B	.T	○
In1	選択対象を指定。	.U	.S	.D	.L	.F	.DF※2	.B	.T	○
R※1	演算結果が格納される戻り値。省略可能。省略時、内部レジスタに格納。	.U	.S	.D	.L	.F	.DF※2	.B	.T	—

※1　RにIn0、In1と異なる型を指定した場合、変換時に自動的に型変換を実行。
※2　デバイスZを指定した場合、「.DF」サフィックスは指定不可。

●動作説明

GがFALSEならIn0を、GがTRUEならIn1を、Rに格納します。

●書式例

rValue := **SEL** (bFlag, rValue1, rValue2);

MUX　マルチプレクサ

[R:=] MUX (K, In0, In1, ... ,InN);

引数/戻り値	説明	認識する型								定数
		.U	.S	.D	.L	.F	.DF	.B	.T	
K	セレクタを指定。(0～N)	.U	.U	.U	.U	—	—	—	—	○
In0	選択対象を指定。	.U	.S	.D	.L	.F	.DF※2	.B	.T	○
In1	選択対象を指定。	.U	.S	.D	.L	.F	.DF※2	.B	.T	○
:	:									
InN	選択対象を指定。（Nの最大値は9）	.U	.S	.D	.L	.F	.DF※2	.B	.T	○
R※1	演算結果が格納される戻り値。省略可能。省略時、内部レジスタに格納。	.U	.S	.D	.L	.F	.DF※2	.B	.T	—

※1　Rに In0、In1、…、InN と異なる型を指定した場合、変換時に自動的に型変換を実行。
※2　デバイスZを指定した場合、「.DF」サフィックスは指定不可。
※3　ビットデータを指定した場合、連続した16/32/64ビットで演算。R002などチャンネルの先頭以外を指定した場合、次のチャンネルにまたがって16/32/64ビットで演算。
※4　InNの型が.Fまたは.DFの場合、T,Cは指定不可。

●動作説明

Kで指定した値(k)に合わせてInNの値をRに格納します。

●書式例

rValue := **MUX** (udiSelector, rValue1, rValue2, rValue3);

ポイント　**Kに 1 ～N(Nは2~10)以外の値を指定した場合。Rは不定値です。**

AVG　指定ブロックの平均値

[R:=] AVG (In, Count);
(アベレージ)

引数/戻り値	説明	認識する型								定数
		.U	.S	.D	.L	.F	.DF	.B	.T	
In※1※2※3	平均する範囲の先頭を指定。	.U	.U	.D	.D	.F	.DF※5	—	—	—
Count※1	平均する範囲のデータ数、または格納先を指定。	.U	.U	.U	.U	—	—	—	—	○
R※4	演算結果が格納される戻り値。省略可能。省略時、内部レジスタに格納。	.U	.S	.D	.L	.F	.DF※5	—	—	—

※1　サフィックスにより扱える数値の範囲が異なる。
※2　ビットデータを指定した場合、連続した16/32/64ビットで演算。R002などチャンネルの先頭以外を指定した場合、次のチャンネルにまたがって16/32/64で演算。
※3　Inの型が.Fまたは.DFの場合、T,Cは指定不可。
※4　Rに演算対象と異なる型を指定した場合、変換時に自動的に型変換を実行。
※5　デバイスZを指定した場合、「.DF」サフィックスは指定不可。

●動作説明

Inを先頭として、Countで指定されたワード数の範囲に格納されたバイナリデータ(16ビット/32ビット(符号無し/符号付き)/64ビット)全ての平均値を求めます。(演算対象が32ビットデータのとき、演算範囲は指定されたデータ数×2ワード、64ビットデータのとき、演算範囲は指定されたデータ数×4ワード)
演算の結果、平均値をRに格納します。

「演算フラグについて(CR2009～CR2012)」(1-8ページ)
詳細については、『命令語リファレンスマニュアル』「データ処理命令」を参照してください。

●書式例

uiValue := **AVG** (auiValue[0], ACNT(auiValue, INT#1));

ポイント　**Countの値が0の場合、Rには0が格納されます。**

WSUM　指定ブロックの合計値

[R:=] WSUM (In, Count);
ワードサム

引数/戻り値	説明	認識する型								定数
		.U	.S	.D	.L	.F	.DF	.B	.T	
In※1※2※3	合計する範囲の先頭を指定。	.U	.U	.D	.D	.F	.DF※5	—	—	—
Count※1	合計する範囲のデータ数の格納先、または合計範囲データを指定。	.U	.U	.U	.U	—	—	—	—	○
R※4	演算結果が格納される戻り値。省略可能。省略時、内部レジスタに格納。	.U	.S	.D	.L	.F	.DF※5	—	—	—

※1　サフィックスにより扱える数値の範囲が異なる。
※2　ビットデータを指定した場合、連続した16/32/64ビットで演算。R002などチャンネルの先頭以外を指定した場合、次のチャンネルにまたがって16/32/64で演算。
※3　Inの型が.Fまたは.DFの場合、T,Cは指定不可。
※4　Inのデータ型が.U/.S/.D/.LでRに.L以外の型を指定した場合、変換時に自動的に型変換を実行。Inのデータ型が.FでRに.F以外の型を指定した場合、変換時に自動的に型変換を実行。Inのデータ型が.DFでRに.DF以外の型を指定した場合、変換時に自動的に型変換を実行。
※5　デバイスZを指定した場合、「.DF」サフィックスは指定不可。

●動作説明

Inを先頭として、Countで指定されたワード数の範囲に格納されたバイナリデータ(16ビット/32ビット(符号無し)/64ビット)全ての合計値を求めます。(演算対象が32ビットデータのとき、演算範囲は指定されたデータ数×2ワード、64ビットデータのとき、演算範囲は指定されたデータ数×4ワード)演算の結果、合計値をRに格納します。

「演算フラグについて(CR2009～CR2012)」(1-8ページ)
詳細については、『命令語リファレンスマニュアル』「データ処理命令」を参照してください。

●書式例

uiValue := **WSUM** (auiValue[0], INT#24);

参考　Inのデータ型が.U/.S/.D/.Lで、演算結果が符号付き32ビットデータの範囲(－2147483648～+2147483647)を超える場合、戻り値には下位32ビットデータが格納され、上位32ビットデータはTM001、TM000に格納されます。

ポイント　**Countの値が0の場合、Rには0が格納されます。**

BSUM　指定ブロックのバイト単位の合計値

[R:=] BSUM (In, Count);
バイトサム

引数/戻り値	説明	認識する型								定数
		.U	.S	.D	.L	.F	.DF	.B	.T	
In※1※2※3	合計値を求める範囲の先頭を指定。	.U	.U	.U	.U	—	—	—	.U	—
Count※1	合計値を求めるデータ数(バイト単位)を指定。	.U	.U	.U	.U	—	—	—	—	○
R※4	演算結果が格納される戻り値。省略可能。省略時、内部レジスタに格納。	.U	.S	.D	.L	.F	.DF※5	—	—	—

※1　サフィックスにより扱える数値の範囲が異なる。

※2　ビットデータを指定した場合、連続した16/32ビットで演算。R002などチャンネルの先頭以外を指定した場合、次のチャンネルにまたがって16/32ビットで演算。
※3　T, C, Zは使用不可。
※4　Rに.D以外の型を指定した場合、変換時に自動的に型変換を実行。
※5　デバイスZを指定した場合、「.DF」サフィックスは指定不可。

●動作説明

Inから始まる指定範囲内に格納されているバイナリデータ上位･下位それぞれ8ビットの合計値を求めRに格納します。Countが奇数の場合は最後のワードデータの上位8ビットまでが計算されます。計算は上位→下位の順におこないCountが奇数のときは最後のワードデータの下位8ビットは無視されます。

「演算フラグについて（CR2009～CR2012）」（1-8ページ）
詳細については、『命令語リファレンスマニュアル』「データ処理命令」を参照してください。

●書式例

uiValue := **BSUM** (auiValue[0], uiCount);

参考　演算結果が符号付き32ビットデータの範囲を超える場合、戻り値には下位32ビットデータが格納され、上位32ビットデータはTM1、TM0に格納されます。

ポイント　**次のいずれかの条件が成立した場合、Rには0が格納されます。**
- **Inにタイマ･カウンタを指定した場合。**
- **Inにビットデータを指定したときに、チャンネルの先頭以外を指定した場合。**
- **Countの値が0の場合。**

CRC　指定ブロック内のCRC値算出

[R:=] CRC (Init, CalcType, CacOrder, BitOrder, In, Count);

引数/戻り値	説明	認識する型								定数
		.U	.S	.D	.L	.F	.DF	.B	.T	
Init※4	動作モードを指定。※1	.U	.U	.U	.U	–	–	–	–	○
CalcType※4		.U	.U	.U	.U	–	–	–	–	○
CalcOrder※4		.U	.U	.U	.U	–	–	–	–	○
BitOrder※4		.U	.U	.U	.U	–	–	–	–	○
In※2※4	算出したいデータブロックの先頭を指定。	.U	.U	.U	.U	–	–	–	–	–
Count※3	データの個数（1～65535）を指定。	.U	.U	.U	.U	–	–	–	–	○
R※4	演算結果が格納される戻り値。省略可能。省略時、内部レジスタに格納。	.U	.S	.D	.L	.F	.DF※6	–	–	–

※1　指定した対象または式の値が0、1以外の場合、1を使用。
※2　ビットデータを指定するときは、チャンネルの先頭を指定。
※3　CalcOrderが「0」のときは1ワード(16bit)単位で個数を指定。
　　CalcOrderが「1(0以外)」のときは1バイト(8bit)単位で個数を指定。
※4　T, C, Zは使用不可。
※5　Rに.U以外の型を指定した場合、変換時に自動的に型変換を実行。
※6　デバイスZを指定した場合、「.DF」サフィックスは指定不可。

●動作説明

Init～**BitOrder**で指定した動作モードで、**In**から始まる**Count**個のデータのCRC値を算出し、**R**に格納します。

●書式例

uiValue := **CRC** (INT#1, INT#0, INT#0, INT#0, auiValue[0], uiCount);

「演算フラグについて（CR2009～CR2012）」（1-8ページ）
詳細については、『命令語リファレンスマニュアル』「データ処理命令」CRC命令を参照してください。

ポイント
次のいずれかの条件が成立した場合、Rには Initの値が格納されます。
- **Init**にワードデータまたは、タイマ・カウンタを指定した場合。
- **CalcType**に間接指定でタイマ・カウンタまたは、ビットデータを指定したとき、チャンネルの先頭以外を指定した場合。
- **Count**の値が0の場合。

ZRES　指定ブロックのリセット(0を格納)

ゾーンリセット
ZRES ([EN], Start, Last);

引数/戻り値	説明	認識する型								定数
		.U	.S	.D	.L	.F	.DF	.B	.T	
EN	実行条件。省略可能。省略時、常時（毎スキャン）実行。	–	–	–	–	–	–	.B	–	–
Start	リセット（0を格納）する範囲の先頭を指定。	.U	.U	.U	.U	.U	.U※1	.B	.T	–
Last	リセット（0を格納）する範囲の終端を指定。	型は**Start**と同じ。								–
R	戻り値はなし。	–	–	–	–	–	–	–	–	–

※1　デバイスZを指定した場合、「.DF」サフィックスは指定不可。

●動作説明

実行条件が成り立つ(TRUE)とき、**Start**と**Last**で指定された範囲を、すべてリセット(0を格納)します。
※サフィックスに.D .L .F .DFを指定してもすべて.Uとして扱われるので、終端ワードは指定した1ワードのみ0が格納されます。

参考
ワードデータが指定された場合はワード単位で0に、ビットデータが指定された場合はビット単位でOFFになります。

「演算フラグについて（CR2009～CR2012）」（1-8ページ）
詳細については、『命令語リファレンスマニュアル』「データ処理命令」を参照してください。

●書式例

ZRES (bEnable, auiValue[0], auiValue[10]);

ポイント
Startと**Last**に指定されたデバイスの種類が異なる場合、ZRES関数を実行しません。

BCMP　指定ブロック間データ比較

ブロック間データ比較
[R:=] BCMP (In1, In2, InOut, Count);

引数/戻り値	説明	認識する型								定数
		.U	.S	.D	.L	.F	.DF	.B	.T	
In1※1	比較元のデータブロックの先頭を指定。	.U	.U	.D	.D	.D	–	–	–	–
In2※1	比較先のデータブロックの先頭を指定。	型はIn1と同じ。					–	–	–	–
InOut※1※2	比較結果を格納する先頭を指定。	–	–	–	–	–	–	.B	–	–
Count	比較するブロックのデータ数の格納先、または比較データ数を指定。	.U	.U	.U	.U	–	–	–	–	○
R	ビット情報が格納される戻り値。全データが一致していればON、それ以外はOFF。省略可能。省略時、内部レジスタに格納。	–	–	–	–	–	–	.B	–	–

※1　ビットデータを指定した場合、連続した16/32ビットで演算。R002などチャンネルの先頭以外を指定した場合、次のチャンネルにまたがって16/32ビットで演算。
※2　ビットデータを指定した場合、チャンネルをまたがっても正しく動作。
InOutにワードデータを指定した場合、結果の格納先はCount個の16ビットワードデータの最下位ビット。最下位ビット以外は変化なし。

●動作説明

実行条件が成り立つ(TRUE)とき、In1を先頭とする、Countで指定された個数のブロックデータと、In2を先頭とする、Count分のブロックデータをそれぞれ比較します。
比較結果は、一致した場合はON、一致しなかった場合はOFFとして、InOutを先頭とする、比較データ数分のビットデータへ格納されます。
InOutが.B指定したワードデータのとき、比較結果の格納は16ビットワードデータ単位での扱いとなり、各ワードデータの最下位ビットのみが対象となります。(最下位ビット以外は変化しません)
比較の結果、全データが一致していればRをONし、そうでなければOFFします。

「演算フラグについて（CR2009～CR2012）」（1-8ページ）
詳細については、『命令語リファレンスマニュアル』「データ処理命令」を参照してください。

●書式例

bFlag := **BCMP** (auiValue[0], auiData[0], abFlag[0], uiCount);

ポイント　**Countの値が0の場合、RはONします。**

BCMPI　ブロックと値のデータ比較

[R:=] BCMPI (In1, In2, InOut, Count);
ブロック間データ比較

引数/戻り値	説明	認識する型								定数
		.U	.S	.D	.L	.F	.DF	.B	.T	
In1※1	比較するデータブロックの先頭を指定。	.U	.U	.D	.D	.D	–	–	–	–
In2※1	比較する値を格納した対象、または比較を指定。	型はIn1と同じ。					–	–	–	–
InOut※1※2	比較結果を格納する先頭を指定。	–	–	–	–	–	–	.B	–	–
Count	比較するブロックのデータ数の格納先、または比較データ数を指定。	.U	.U	.U	.U	–	–	–	–	○
R	演算結果が格納される戻り値。省略可能。全データが一致していればON、それ以外はOFF。省略時、内部レジスタに格納。	–	–	–	–	–	–	.B	–	–

※1　ビットデータを指定した場合、連続した16/32ビットで演算。R002などチャンネルの先頭以外を指定した場合、次のチャンネルにまたがって16/32ビットで演算。
※2　ビットデータを指定した場合、チャンネルをまたがっても正しく動作。
InOutにワードデータを指定した場合、結果の格納先はCount個の16ビットワードデータの最下位ビット。最下位ビット以外は変化なし。

●動作説明

実行条件が成り立つ(TRUE)とき、In1を先頭とする、Countで指定された個数のブロックデータを、In2で指定されたデータとそれぞれ比較します。
比較結果は、一致した場合はON、一致しなかった場合はOFFとして、InOutを先頭とする、Count分の対象へ格納されます。
InOutがワードデータのとき、比較結果の格納は16ビットワードデータ単位での扱いとなり、各ワードデータの最下位ビットのみが対象となります。(最下位ビット以外は変化しません)
比較の結果、全データが一致していればRをONし、そうでなければOFFします。

「演算フラグについて（CR2009～CR2012）」（1-8ページ）
詳細については、『命令語リファレンスマニュアル』「データ処理命令」を参照してください。

●書式例

bFlag := **BCMPI** (auiValue[0], uiValue, abFlag[0], uiCount);

ポイント　**Countの値が0の場合、RはONします。**

RND　乱数発生

[R:=] RND (In);
ランダム

引数/戻り値	説明	認識する型								定数
		.U	.S	.D	.L	.F	.DF	.B	.T	
In※1	乱数発生の元となる、演算対象または演算対象データを指定。	.U	.U	.U	.U	–	–	–	–	–
R※2	演算結果が格納される戻り値。省略可能。省略時、内部レジスタに格納。	.U	.S	.D	.L	.F	.DF※3	–	–	–

※1　In(変換対象)にビットデータのチャンネルの先頭以外(R1002, R1005等)を指定した場合でも、次のチャンネルにまたがって正常に動作。
※2　Rに.U以外の型を指定した場合、変換時に自動的に型変換を実行。
※3　デバイスZを指定した場合、「.DF」サフィックスは指定不可。

●動作説明

In(演算対象)から乱数を発生させ、乱数を**R**に格納します。
発生する乱数は1～65535の疑似乱数です。**In**=0のときは、**R**に1を格納します。

「演算フラグについて(CR2009～CR2012)」(1-8ページ)
詳細については、『命令語リファレンスマニュアル』「算術/比較命令」を参照してください。

ポイント
- **乱数発生のアルゴリズムには「M系列」を用いています。**
- **RND関数を使って生成される値には周期性がある(例えば、1の後には必ず2が生成される)ため、In乱数の種の値が変化しない場合、生成される乱数の値も変化しません。**

●書式例

uiValue1 := **RND** (uiValue2);

SORT　バイナリデータソート

ソート
SORT ([EN], Order, InOut, Count);

引数/戻り値	説明	認識する型								定数
		.U	.S	.D	.L	.F	.DF	.B	.T	
EN	実行条件。省略可能。省略時、常時(毎スキャン)実行。	–	–	–	–	–	–	.B	–	○
Order	ソート順序を指定。(0:昇順、1:降順)	.U	.U	.U	.U	–	–	–	–	○
InOut※1	検索するバイト列の先頭を指定。	.U	.S	.D	.L	–	–	–	–	–
Count※2	ソートするデータ数を指定。	.U	.U	.U	.U	–	–	–	–	○
R	戻り値はなし。	–	–	–	–	–	–	–	–	–

※1　ビットデータを指定した場合、**InOut**の型が.U/.Sの時**Count**チャンネル、
.D/.Lの時2×**Count**チャンネルを占有。
ワードデータを指定した場合、**InOut**の型が.U/.Sの時**Count**ワード、.D/.Lの時2×**Count**ワードを占有。
※2　ビットデータを指定した場合、連続する16ビットを占有。ワードデータを指定した場合、1ワードを占有

●動作説明

InOutで指定したデータテーブルから**Count**のバイナリデータを、**Order**で指定したソート順序でソートします。

ポイント
SORT関数は、関数実行時に全データをソート処理します。そのため、Orderに大きな値を指定すると、スキャンタイムが大きく伸びることがあります。
スキャンタイムへの影響を調整する必要がある場合は、SORTN関数で複数スキャンに分割してソートをおこなってください。
「SORTN関数」(8-16ページ)

「演算フラグについて(CR2009～CR2012)」(1-8ページ)
詳細については、『命令語リファレンスマニュアル』「データ処理命令」を参照してください。

●書式例

SORT (bEnable, INT#0 , auiValue[0], uiCount);

SORTN　バイナリデータ分割ソート

[R:=] SORTN (EN, Order, CmpCount, InOut, Count);
（分割ソート）

引数/戻り値	説明	認識する型								定数
		.U	.S	.D	.L	.F	.DF	.B	.T	
EN	実行条件がONの時ソート処理を実行。	—	—	—	—	—	—	.B	—	—
Order	ソート順序を指定。 （0：昇順、1：降順）	.U	.U	.U	.U	—	—	—	—	○
CmpCount※2	1スキャンでの比較回数を指定。 （1～65535）	.U	.U	.U	.U	—	—	—	—	○
InOut※1	検索するバイト列の先頭を指定。	.U	.S	.D	.L	—	—	—	—	—
Count※2	ソートするデータ数を指定。	.U	.U	.U	.U	—	—	—	—	○
R	検索結果が格納される戻り値。省略可能。ソート中はOFF、ソート完了時ON。省略時、内部レジスタに格納。	—	—	—	—	—	—	.B	—	—

※1　ビットデータを指定した場合、InOutの型が.U/.Sの時Countチャンネル、.D/.Lの時2×Countチャンネルを占有。
　　ワードデータを指定した場合、InOutの型が.U/.Sの時Countワード、.D/.Lの時2×Countワードを占有。
※2　ビットデータを指定した場合、連続する16ビットを占有。ワードデータを指定した場合、1ワードを占有。

●動作説明

InOutで指定したデータテーブルからCountのバイナリデータを、Orderで指定したソート順序でソートします。
1スキャンあたりの比較回数は、CmpCount比較回数で指定します。

「演算フラグについて（CR2009～CR2012）」（1-8ページ）
詳細については、『命令語リファレンスマニュアル』「データ処理命令」を参照してください。

●書式例

bFlag := **SORTN** (bEnable, INT#0, uiValue, auiValue[0], uiCount);

テーブル処理関数

FIFOW　FIFOテーブルへ書き込み

FIFO書き込み
FIFOW ([EN], In, Dst);

引数/戻り値	説明	認識する型								定数
		.U	.S	.D	.L	.F	.DF	.B	.T	
EN	実行条件。省略可能。省略時、常時(毎スキャン)実行。	–	–	–	–	–	–	.B	–	–
In※1	FIFOデータブロックに書き込むデータの格納先を指定。	型はDstと同じ。					–	–	–	–
Dst	FIFOデータテーブルの先を指定。	.U	.U	.D	.D	.D	–	–	–	–
R	戻り値はなし。	–	–	–	–	–	–	–	–	–

※1　ビットデータを指定した場合、連続した16/32ビットで演算。R002などチャンネルの先頭以外を指定した場合、次のチャンネルにまたがって16/32ビットで演算。

●動作説明

実行条件が成り立つ(TRUE)とき、Inに格納されたデータ(16/32ビット)を、Dstで指定されたデータブロックの最後に書き込みます。

「演算フラグについて(CR2009～CR2012)」(1-8ページ)
詳細については、『命令語リファレンスマニュアル』「テーブル処理命令」を参照してください。

●書式例

FIFOW (bEnable, uiValue, auiValue[0]);

ポイント　**次のいずれかの条件が成立した場合、関数は実行されません。**
- 最大格納データ数が0の場合。
- データ数nが最大格納データ数maxを超えた場合。
- データテーブルがデバイス/変数範囲を超えるような最大格納データ数maxの場合。

参考　32ビットデータテーブルのとき、書き込みデータは、単精度浮動小数点型実数データも使用できます。

FIFOR　FIFOテーブルから読み出し

FIFO読み出し
FIFOR ([EN], In, Dst);

引数/戻り値	説明	認識する型								定数
		.U	.S	.D	.L	.F	.DF	.B	.T	
EN	実行条件。省略可能。省略時、常時(毎スキャン)実行。	–	–	–	–	–	–	.B	–	–
In※1	FIFOデータテーブルの先頭を指定。	.U	.U	.D	.D	.D	–	–	–	–
Dst※1	FIFOデータブロックから読み出したデータを格納する対象を指定。	型はInと同じ。					–	–	–	–
R	戻り値はなし。	–	–	–	–	–	–	–	–	–

※1　ビットデータを指定した場合、連続した16/32ビットで演算。R002などチャンネルの先頭以外を指定した場合、次のチャンネルにまたがって16/32ビットで演算。

●動作説明

実行条件が成り立つ(TRUE)とき、**In**で指定されたデータブロックに格納されている先頭のデータを読み出して、**Dst**へ格納します。
格納後、データブロックの内容が1データずつ前詰めされ、格納データ数が減算(–1)されます。

「演算フラグについて（CR2009～CR2012）」（1-8ページ）
詳細については、『命令語リファレンスマニュアル』「テーブル処理命令」を参照してください。

●書式例

FIFOR (bEnable, auiValue[0] , uiValue);

ポイント

次のいずれかの条件が成立した場合、関数は実行されません。
- データテーブルにデータが無い場合。
- データ数nが最大格納データ数maxより多い場合。
- データテーブルがデバイス/変数範囲を超えるような最大格納データ数maxの場合。

LIFOW　LIFOテーブルへ書き込み

LIFO書き込み
LIFOW ([EN], In, Dst);

引数/戻り値	説明	認識する型								定数
		.U	.S	.D	.L	.F	.DF	.B	.T	
EN	実行条件。省略可能。省略時、常時（毎スキャン）実行。	–	–	–	–	–	–	.B	–	–
In※1	LIFOデータブロックに書き込むデータの格納先を指定。	型は**Dst**と同じ。					–	–	–	–
Dst	LIFOデータテーブルの先頭を指定。	.U	.U	.D	.D	.D	–	–	–	–
R	戻り値はなし。	–	–	–	–	–	–	–	–	–

※1　ビットデータを指定した場合、連続した16/32ビットで演算。R002などチャンネルの先頭以外を指定した場合、次のチャンネルにまたがって16/32ビットで演算。

●動作説明

実行条件が成り立つ(TRUE)とき、**In**に格納されたデータ(16/32ビット)を、**Dst**で指定されたデータブロックの最後に書き込みます。

「演算フラグについて（CR2009～CR2012）」（1-8ページ）
詳細については、『命令語リファレンスマニュアル』「テーブル処理命令」を参照してください。

●書式例

LIFOW (bEnable, uiValue, auiValue[0]);

ポイント

次のいずれかの条件が成立した場合、関数は実行されません。
- 最大格納データ数が0の場合。
- データ数nが最大格納データ数maxを超えた場合。
- データテーブルがデバイス/変数範囲を超えるような最大格納データ数maxの場合。

LIFOR　LIFOテーブルから読み出し

LIFO読み出し
LIFOR ([EN], In, Dst);

引数/戻り値	説明	認識する型								定数
		.U	.S	.D	.L	.F	.DF	.B	.T	
EN	実行条件。省略可能。省略時、常時(毎スキャン)実行。	—	—	—	—	—	—	.B	—	—
In※1	LIFOデータテーブルの先頭を指定。	.U	.U	.D	.D	.D	—	—	—	—
Dst	LIFOデータブロックから読み出したデータを格納する対象を指定。	型はInと同じ。					—	—	—	—
R	戻り値はなし。	—	—	—	—	—	—	—	—	—

※1　ビットデータを指定した場合、連続した16/32ビットで演算。R002などチャンネルの先頭以外を指定した場合、次のチャンネルにまたがって16/32ビットで演算。

●動作説明

実行条件が成り立つ(TRUE)とき、Inで指定されたデータブロックに格納されている最終のデータを読み出して、Dstへ格納します。
格納後、データブロックの内容が1データずつ前詰めされ、格納データ数が減算(-1)されます。
(読み出した最終データ部分には、0が格納されます)

「演算フラグについて(CR2009～CR2012)」(1-8ページ)
詳細については、『命令語リファレンスマニュアル』「テーブル処理命令」を参照してください。

●書式例

LIFOR (bEnable, auiValue[0], uiValue);

ポイント
次のいずれかの条件が成立した場合、関数は実行されません。
- データテーブルにデータが無い場合。
- データ数nが最大格納データ数maxより多い場合。
- データテーブルがデバイス/変数範囲を超えるような最大格納データ数maxの場合。

FWRIT　データテーブル指定位置の上書き

データテーブルの上書き
FWRIT ([EN], In, Dst, Pos);

引数/戻り値	説明	認識する型								定数
		.U	.S	.D	.L	.F	.DF	.B	.T	
EN	実行条件。省略可能。省略時、常時(毎スキャン)実行。	—	—	—	—	—	—	.B	—	—
In※1	上書きするデータの格納先、または上書きデータを指定。	型はDstと同じ。					—	—	—	—
Dst※1	テーブルの先頭を指定。	.U	.U	.D	.D	.D	—	—	—	—
Pos	上書きするデータブロックの位置の格納先、または上書位置を指定。	.U	.U	.U	.U	—	—	—	—	○
R	戻り値はなし。	—	—	—	—	—	—	—	—	—

※1　ビットデータを指定した場合、連続した16/32ビットで演算。R002などチャンネルの先頭以外を指定した場合、次のチャンネルにまたがって16/32ビットで演算。

●動作説明

実行条件が成り立つ(TRUE)とき、**In**に格納されたデータ(16/32ビット)を、**Dst**で指定されたデータブロックの、**Pos**で指定された位置(番目)に上書きします。

「演算フラグについて（CR2009～CR2012）」（1-8ページ）
詳細については、『命令語リファレンスマニュアル』「テーブル処理命令」を参照してください。

●書式例

FWRIT (bEnable, uiValue , auiValue[0] , uiCount);

ポイント

次のいずれかの条件が成立した場合、関数は実行されません。

- **Pos**の値が0または、データ数Nよりも大きい場合。
- 最大格納データ数maxが0の場合。
- データ数Nが最大格納データ数maxより大きい場合。
- デバイス/変数範囲を超えるような最大格納データ数maxの場合。

FINS　データテーブル指定位置への挿入

データテーブルの挿入
FINS ([EN], In, Dst, Pos);

引数/戻り値	説明	認識する型								定数
		.U	.S	.D	.L	.F	.DF	.B	.T	
EN	実行条件。省略可能。省略時、常時（毎スキャン）実行。	—	—	—	—	—	—	.B	—	—
In※1	上書きするデータの格納先、または上書きデータを指定。	型は**Dst**と同じ。					—	—	—	—
Dst※1	テーブルの先頭を指定。	.U	.U	.D	.D	.D	—	—	—	—
Pos	挿入するデータブロックの格納先、または挿入位置を指定。	.U	.U	.U	.U	—	—	—	—	○
R	戻り値はなし。	—	—	—	—	—	—	—	—	—

※1　ビットデータを指定した場合、連続した16/32ビットで演算。R002などチャンネルの先頭以外を指定した場合、次のチャンネルにまたがって16/32ビットで演算。

●動作説明

実行条件が成り立つ(TRUE)とき、**In**に格納されたデータ(16/32ビット)を、**Dst**で指定されたデータブロックの、**Pos**で指定された位置(番目)に挿入します。

「演算フラグについて（CR2009～CR2012）」（1-8ページ）
詳細については、『命令語リファレンスマニュアル』「テーブル処理命令」を参照してください。

●書式例

FINS (bEnable, uiValue , auiValue[0] , uiCount);

ポイント

次のいずれかの条件が成立した場合、関数は実行されません。

- **Pos**の値が0または、データ数Nよりも大きい場合。
- 最大格納データ数maxが0の場合。
- データ数Nが最大格納データ数maxより大きい場合。
- デバイス/変数範囲を超えるような最大格納データ数maxの場合。

FDEL　データテーブル指定位置の削除

データテーブルの削除
FDEL ([EN], Dst, Pos);

引数/戻り値	説明	認識する型								定数
		.U	.S	.D	.L	.F	.DF	.B	.T	
EN	実行条件。省略可能。省略時、常時(毎スキャン)実行。	—	—	—	—	—	—	.B	—	—
Dst※1	テーブルの先頭を指定。	.U	.U	.D	.D	.D	—	—	—	—
Pos	削除するデータブロックの格納先、または削除位置を指定。	.U	.U	.U	.U	—	—	—	—	○
R	戻り値はなし。	—	—	—	—	—	—	—	—	—

※1　ビットデータを指定した場合、連続した16/32ビットで演算。R002などチャンネルの先頭以外を指定した場合、次のチャンネルにまたがって16/32ビットで演算。

●動作説明

実行条件が成り立つ(TRUE)とき、**Dst**で指定されたデータブロックの、**Pos**で指定された位置(番目)のデータを削除します。
削除後、データブロックの内容が1データずつ前詰めされ、格納データ数が減算(-1)されます。

「演算フラグについて(CR2009～CR2012)」(1-8ページ)
詳細については、『命令語リファレンスマニュアル』「テーブル処理命令」を参照してください。

●書式例

FDEL (bEnable ,auiValue[0], uiValue);

ポイント

次のいずれかの条件が成立した場合、関数は実行されません。

- **Pos**の値が0または、データ数Nよりも大きい場合。
- 最大格納データ数maxが0の場合。
- データ数Nが最大格納データ数maxより大きい場合。
- デバイス/変数範囲を超えるような最大格納データ数maxの場合。

時計処理関数

WTIME　カレンダタイマに時刻データ書き込み

時刻書き込み
WTIME ([EN], Year, Month, Day, Hour, Min, Sec, DayOfWeek);

引数/戻り値	説明	認識する型								定数
		.U	.S	.D	.L	.F	.DF	.B	.T	
EN	実行条件。省略可能。省略時、常時(毎スキャン)実行。	—	—	—	—	—	—	.B	—	—
Year	年データ(0～99)	.U	.U	.U	.U	—	—	—	—	○
Month	月データ(1～12)	.U	.U	.U	.U	—	—	—	—	○
Day	日データ(1～31)	.U	.U	.U	.U	—	—	—	—	○
Hour	時データ(0～23)	.U	.U	.U	.U	—	—	—	—	○
Min	分データ(0～59)	.U	.U	.U	.U	—	—	—	—	○
Sec	秒データ(0～59)	.U	.U	.U	.U	—	—	—	—	○
DayOfWeek	曜日データ(0～6)	.U	.U	.U	.U	—	—	—	—	○
R	戻り値はなし。	—	—	—	—	—	—	—	—	—

参考　Year～Secにワードデータを指定した場合、すべて16ビット符号無しデータで演算。(指定したサフィックスを無視)

●動作説明

実行条件が成り立つ(TRUE)とき、Year～Secに格納されている時刻データを内蔵カレンダタイマに書き込みます。

「演算フラグについて(CR2009～CR2012)」(1-8ページ)
詳細については、『命令語リファレンスマニュアル』「時計処理命令」を参照してください。

●書式例

WTIME (bEnable, uiYear, uiMonth, uiDay, uiHour, uiMin, uiSec, uiDayOfWeek);

ポイント
- 時刻データを個別に直接コントロールメモリに書き込むことはできません。
- WTIME関数を使用する場合、年・月・日・時・分・秒・曜日を全て正確に指定してください。
  年・月・日・時・分・秒に対する曜日のチェックはおこなわれません。
- 書き込むデータが時刻データとして不適切な場合、WTIME関数は実行しません。

SEC　日時データを秒形式に変換

日時→秒変換
[R:=] SEC (In);

引数/戻り値	説明	認識する型								定数
		.U	.S	.D	.L	.F	.DF	.B	.T	
In※1※2	変換元の日時を格納した先頭を指定。	.U	.U	.U	.U	.F	—	—	—	—
R※3	演算結果が格納される戻り値。省略可能。省略時、変換結果は内部レジスタに格納。	.U	.S	.D	.L	.F	.DF ※4	—	—	—

※1　ビットデータを指定した場合、日時データは連続した96ビット、秒データは連続した32ビットで演算。R002などチャンネルの先頭以外を指定した場合でも可能。
※2　T, C, Zは使用不可。
※3　Rに.D以外の型を指定した場合、変換時に自動的に型変換を実行。
※4　デバイスZを指定した場合、「.DF」サフィックスは指定不可。

●動作説明

Inを先頭に格納されている日時データの6ワード(96ビット)を、2000年1月1日0時0分0秒を基準日時とした通算秒に換算し、換算結果を32ビットバイナリデータ(符号無し)としてRに格納します。

参考　日時データは、全て16ビットバイナリデータで指定し、符号無し/符号付きのどちらでも使用できます。

「演算フラグについて(CR2009～CR2012)」(1-8ページ)
詳細については、『命令語リファレンスマニュアル』「時計処理命令」を参照してください。

●書式例

udiValue := **SEC** (auiValue[0]);

ポイント　**次のいずれかの条件が成立した場合、Rには0が格納されます。**
- 変換する日時データが不適切な場合。
- In、Rに間接指定でタイマ・カウンタを指定した場合。

RSEC　秒形式を日時データに変換

秒→日時変換
RSEC ([EN], In, InOut);

引数/戻り値	説明	認識する型								定数
		.U	.S	.D	.L	.F	.DF	.B	.T	
EN	実行条件。省略可能。省略時、常時(毎スキャン)実行。	—	—	—	—	—	—	.B	—	—
In※1	変換元の秒(通算秒)の格納先を指定。	—	—	.D	.D	—	—	—	—	—
InOut	変換した日時を格納する先頭を指定。	.U	.U	.U	.U	—	—	—	—	—
R	戻り値はなし。	—	—	—	—	—	—	—	—	—

※1　ビットデータを指定した場合、日時データは連続した96ビット、秒データは連続した32ビットで演算。R002などチャンネルの先頭以外を指定した場合でも可能。
※2　タイマ、カウンタは使用不可。

●動作説明

Inで指定された、2000年1月1日0時0分0秒を基準日時とした通算秒(32ビットバイナリデータ(符号無し))を日時データに変換し、**InOut**を先頭とする6ワード(96ビット)に格納します。

参考　日時データは、全て16ビットバイナリデータで格納されます。

「演算フラグについて(CR2009～CR2012)」(1-8ページ)
詳細については、『命令語リファレンスマニュアル』「時計処理命令」を参照してください。

●書式例

RSEC (bEnable, udiValue, auiValue[0]);

ポイント　**次のいずれかの条件が成立した場合、関数は実行されません。**
- 変換する秒データが不適切な場合。
- **In**、**InOut**に間接指定でタイマ・カウンタを指定した場合。

AJST　カレンダタイマ補正

アジャスト
AJST ([EN]);

引数/戻り値	説明	認識する型								定数
		.U	.S	.D	.L	.F	.DF	.B	.T	
EN	実行条件。省略可能。省略時、常時(毎スキャン)実行。	–	–	–	–	–	–	.B	–	–
R	戻り値はなし。	–	–	–	–	–	–	–	–	–

●動作説明

実行条件が成り立つ(TRUE)とき、内蔵カレンダタイマの現在時刻の「秒」が0～29のときは0にし、「秒」が30～59のときは「分」を桁上げして「秒」を0にして、分単位(0秒)で近い方に補正します。

実行前		実行後
12時59分15秒	→	12時59分00秒
12時59分30秒	→	13時00分00秒

詳細については、『命令語リファレンスマニュアル』「時計処理命令」を参照してください。

●書式例

AJST (bEnable);

高速処理関数

DI	割り込み禁止
EI	割り込み許可

割り込み禁止
DI ([EN]);
割り込み許可
EI ([EN]);

引数/戻り値	説明	認識する型								定数
		.U	.S	.D	.L	.F	.DF	.B	.T	
EN	実行条件。省略可能。省略時、常時(毎スキャン)実行。	—	—	—	—	—	—	.B	—	—
R	戻り値はなし。	—	—	—	—	—	—	—	—	—

●動作説明

DI関数は、実行条件が成り立つ(TRUE)とき、割り込みプログラムの実行を禁止します。
EI関数は、実行条件が成り立つ(TRUE)とき、割り込みプログラムの実行を許可します。
DI関数で一時的に禁止した割り込みプログラムの実行を、EI関数で許可(禁止の解除)します。

- 一度EI関数を実行すると、DI関数を実行するまでは割り込み許可となります。
- 運転開始時は常に割り込み禁止状態になっています。
- DI関数を実行中(割り込み禁止中)に割り込みが発生したときは、16重まで記憶されます。
  EI関数が実行されると記憶した割り込みが実行されます。
- 割り込みプログラムを実行中にさらに割り込みプログラムを実行したい場合は、割り込みプログラム中でEI関数を実行することにより16重まで受け付け可能となります。

割り込み処理の詳細については、「命令語リファレンスマニュアル」「INT/RETI命令」「高速処理命令」を参照してください。

●書式例

DI ();

EI ();

ポイント　**運転開始時は常に割り込み禁止状態で、一度EI関数を実行すると、DI関数を実行するまで割り込み許可状態になります。**

参考　割り込みプログラムが実行されると、他の割り込みはその割り込みプログラムが終了するまで禁止状態になります。他の割り込みを許可したいときは、割り込みプログラムの実行中にEI関数を実行する必要があります。EI関数を実行すると、実行した割り込み要因より高い優先順位の割り込みプログラムを許可(多重割り込み)できます。ただし同じ要因の割り込みはできません。

DIC　割り込み禁止範囲の設定

割り込み禁止
DIC ([EN], In);

引数/戻り値	説明	認識する型								定数
		.U	.S	.D	.L	.F	.DF	.B	.T	
EN	実行条件。省略可能。省略時、常時(毎スキャン)実行。	—	—	—	—	—	—	.B	—	—
In※1	命令実行前の割り込み禁止状態を格納する対象を指定。	—	—	—	—	—	—	.B	—	—
R	戻り値はなし。	—	—	—	—	—	—	—	—	—

※1　割込状態格納デバイスがワードデータのとき、割込状態の格納対象は最下位ビットのみ。(最下位ビット以外は変化なし)

●動作説明

実行条件が成り立つ(TRUE)とき、割り込みプログラムの実行を禁止します。
同時にInを、関数実行前が割り込み許可状態であればセットし、割り込み禁止状態であればリセットします。

詳細については、『命令語リファレンスマニュアル』「高速処理命令」を参照してください。

●書式例

DIC (bEnable, bFlag);

IEDGE　割り込み条件設定

割込条件設定
IEDGE ([EN], Unit, Input, In);

引数/戻り値	説明	認識する型								定数
		.U	.S	.D	.L	.F	.DF	.B	.T	
EN	実行条件。省略可能。省略時、常時(毎スキャン)実行。	—	—	—	—	—	—	.B	—	—
Unit	入力割り込み検出条件を書き込む対象の拡張ユニット番号を指定。(0～48)	.U	.U	.U	.U	—	—	—	—	—
Input	割り込み要因番号を指定。(0～31)	.U	.U	.U	.U	—	—	—	—	—
In	検出条件(立ち上がり:0もしくは1、立ち下がり:2、立ち上がり・下がり両方:3)を指定。	.U	.U	.U	.U	—	—	—	—	—
R	戻り値はなし。	—	—	—	—	—	—	—	—	—

●動作説明

実行条件が成り立つ(TRUE)とき、Inで指定した1ワードのデータをUnit番目のユニットのInput番目の要因の割り込みの検出条件に設定します。

Inの値により、検出条件は以下のようになります。

In	検出条件
0～1	立ち上がり
2	立ち下がり
3	立ち上がり・下がり両方

ポイント
- **Inに4～65535を指定した場合は演算エラーにはなりませんが、検出条件は変更されません。**
- **本命令が実行完了したタイミングで、上記検出条件の設定が変更されます。**

「演算フラグについて（CR2009～CR2012）」（1-8ページ）
詳細については、『命令語リファレンスマニュアル』「IEDGE命令」を参照してください。

●書式例

IEDGE (bEnable, uiUnit, uiValue, INT#3);

PID関数

PIDAT　オートチューニング付きPID制御

AT付きPID制御
PIDAT ([EN], In, Param, InOut);

引数/戻り値	説明	認識する型								定数
		.U	.S	.D	.L	.F	.DF	.B	.T	
EN	実行条件。省略可能。省略時、常時(毎スキャン)実行。	—	—	—	—	—	—	.B	—	—
In	制御対象から観測された値を指定。	.S	.S	.S	.S	—	—	—	—	—
Param※1	パラメータを格納している先頭を指定。	.U	.U	.U	.U	—	—	—	—	—
InOut	PID演算の結果を格納する対象を指定。	.S	.S	.S	.S	—	—	—	—	—
R	戻り値はなし。	—	—	—	—	—	—	—	—	—

※1　Paramパラメータは、連続する52ワードを占有。パラメータの詳細はPIDAT命令語の説明を参照。

●動作説明

InとParamから、InOut(PID演算結果)を取得します。
初回呼び出し時(初期化処理を実行する場合)は、実行条件ONかつParamのワークエリアを0クリアしてPIDAT関数を呼び出します。
演算を継続する場合は、実行条件ONを継続してPIDAT関数を呼び出します。

参考　PIDAT関数を実行条件OFFで呼び出した後、実行条件ONで呼び出すと初期化処理を実行します。

演算フラグやParamの詳細については、『命令語リファレンスマニュアル』「PIDAT命令」を参照してください。

●書式例

PIDAT (bEnable, iPV , auiPIDParam, iMV);

ログ関数

LOGE　指定IDのロギング許可
LOGD　指定IDのロギング禁止

ロギング許可
LOGE ([EN], In);
ロギング不許可
LOGD ([EN], In);

引数/戻り値	説明	認識する型								定数
		.U	.S	.D	.L	.F	.DF	.B	.T	
EN	実行条件。省略可能。省略時、常時(毎スキャン)実行。	—	—	—	—	—	—	.B	—	—
In	ロギングの許可/不許可をおこなうロギングIDを格納した対象、またはをロギングIDを指定します。(0～9)	.U	.U	.U	.U	—	—	—	—	○※1
R	戻り値はなし。	—	—	—	—	—	—	—	—	—

※1　16# (16進数指定)での指定は不可。

●動作説明

LOGE関数は、実行条件が成り立つ(TRUE)とき、Inで指定したロギングIDのロギングを実行許可状態にします。関数実行後、ロギングIDごとのロギング実行リレーがセットされます。
LOGD関数は、実行条件が成り立つ(TRUE)とき、Inで指定したロギングIDのロギングを実行不許可状態にします。関数実行後、ロギングIDごとのロギング実行リレーがリセットされます。

ロギングID	ロギング実行リレー
0	CR0000
1	CR0100
2	CR0200
3	CR0300
4	CR0400
5	CR0500
6	CR0600
7	CR0700
8	CR0800
9	CR0900

「演算フラグについて(CR2009～CR2012)」(1-8ページ)
詳細については、使用しているCPUユニットのユーザーズマニュアルを参照してください。

●書式例

LOGE (bEnable, uiValue);

LOGD (bEnable, uiValue);

TRGD　ロギングデータ取得

ロギングデータ取得
TRGD ([EN], In);

引数/戻り値	説明	認識する型								定数
		.U	.S	.D	.L	.F	.DF	.B	.T	
EN	実行条件。省略可能。省略時、常時(毎スキャン)実行。	—	—	—	—	—	—	.B	—	—
In	ロギングIDを指定。(0～10)※1	.U	.U	.U	.U	—	—	—	—	○
R	戻り値はなし。	—	—	—	—	—	—	—	—	—

※1　ロギングIDは、0～10を指定。リアルタイムチャートモニタでトリガ命令を使用する場合は、ロギングIDに10を指定。

●動作説明

実行条件が成り立つ(TRUE)とき、Inで指定したロギングIDのデータを取得します。
通常、ロギング・トレース・リアルタイムチャートモニタはEND処理のタイミングでデータを取得しますが、TRGD命令を使用することで、ラダーの任意のタイミングでデータが取得できます。

ポイント
- **TRGD関数(命令)は、1つのプロジェクトにつき各IDで8個までしか使用できません。**
- **TRGD関数(命令)でデータを取得するには、KV STUDIOのロギング・トレース・リアルタイムチャートモニタの設定で、トリガ種類を「トリガ命令」に設定する必要があります。「トリガ命令」に設定していない場合、TRGD関数(命令)を実行してもデータは取得されません。**

「演算フラグについて(CR2009～CR2012)」(1-8ページ)
詳細については、使用しているCPUユニットのユーザーズマニュアルを参照してください。

●書式例

TRGD (bEnable, INT#1);

TRGR　デバイス/変数データ取得

デバイス/変数データ取得
TRGR ([EN], In);

引数/戻り値	説明	認識する型								定数
		.U	.S	.D	.L	.F	.DF	.B	.T	
EN	実行条件。省略可能。省略時、常時(毎スキャン)実行。	—	—	—	—	—	—	.B	—	—
In	運転記録IDを指定。(0～3)	.U	.U	.U	.U	—	—	—	—	○
R	戻り値はなし。	—	—	—	—	—	—	—	—	—

●動作説明

実行条件が成り立つ(TRUE)とき、Inで指定した運転記録IDのデータを取得します。
通常はEND処理のタイミングでデータを取得しますが、TRGR命令を使用することで、ラダーの任意のタイミングでデータが取得できます。

ポイント
- **TRGR命令はKV STUDIO Ver.10.0以降で対応機種がKV-8000のときのみ使用できます。**
- **TRGR関数(命令)でデータを取得するには、KV STUDIOの運転記録設定でサンプリング周期を「トリガ命令」に設定する必要があります。「トリガ命令」に設定していない場合、TRGR関数(命令)を実行してもデータは取得されません。**

「演算フラグについて（CR2009～CR2012）」（1-8ページ）
詳細については、『KV-8000シリーズユーザーズマニュアル』「運転記録機能」を参照してください。

●書式例

TRGR (bEnable, INT#1);

ストレージ関数

MWRIT　ストレージ(メモリカード/CPUメモリ)書き込み

ストレージ書込
[R:=] MWRIT (Path, Src, Count, Param, Result);

引数/戻り値	説明	認識する型								定数
		.U	.S	.D	.L	.F	.DF	.B	.T	
Path※6	データを保存するファイル名の格納先、または保存ファイル名を指定。(1～247文字(終端コードNUL、フォルダ名を含む)の範囲の文字列を指定)	—	—	—	—	—	—	—	.T	○※2
Src※3※4	保存するデータを格納している先頭を指定。	.U	.U	.U	.U	—	—	—	—	—
Count	保存する範囲のデータ数の格納先、または保存データ数を指定。	.U	.U	.U	.U	—	—	—	—	—
Param※4※5	保存形式等に関するパラメータ、およびデータの保存状態を格納する先頭を指定。	.U	.U	.U	.U	—	—	—	—	—
Result※2	保存結果を通知するビットデータを指定。	—	—	—	—	—	—	.B	—	—
R	演算結果が格納される戻り値。省略可能。関数の起動に成功したらON、失敗のときOFF。省略時、省略時、内部レジスタに格納。	—	—	—	—	—	—	.B	—	—

※1　Resultにビットデータを指定した場合、通知ビットは連続する2ビットを使用、ワードデータを指定した場合は、通知ビットデータは連続する2ワードの各々最下位ビットを使用。
※2　定数で文字列を指定する場合は、' 'で囲んで入力。
※3　インデックスレジスタZおよびアクセスウィンドウで設定したデジタルトリマ値を保存する場合は、一度データメモリDM等にコピーして、そのデータメモリDMを設定。
※4　ビットデータを使用する場合はチャンネルの先頭を指定。
※5　Paramにビットデータを指定した場合、連続する96ビットを占有。ワードデータを指定した場合、連続する6ワードを占有。
※6　T, C, CR, Zは使用不可。

●動作説明

Pathで指定されたファイル名(文字列)で、Srcを先頭として、Countで指定された範囲に格納されたデータをメモリカードまたはCPUメモリに書き込みます。
書き込み処理は、Paramで指定された書き込みパラメータに従っておこなわれ、Resultに書き込み完了通知と異常終了アラーム通知が格納されます。
MWRIT関数が正常に起動できた場合、RをONし、失敗した場合はOFFします。

Paramに指定する保存形式等に関するパラメータの詳細は、『命令語リファレンスマニュアル』「MWRIT命令」を参照してください。

参考　Resultの保存結果格納対象の通知ビットは次のような意味を持ちます。
(例) MR2000を指定した場合

通知ビット		ON(1、TRUE)	OFF(0、FALSE)
MR2000	完了通知	完了	未完了
MR2001	異常通知	異常終了	正常終了

「演算フラグについて(CR2009～CR2012)」(1-8ページ)
詳細については、『命令語リファレンスマニュアル』「ストレージ命令」を参照してください。

●書式例

bFlag := **MWRIT** (strPath, auiValue[0], uiValue, auiMwritParam, abFlag[0]);

ポイント

次のいずれかの条件が成立した場合、関数は実行されません。

- **Path**で指定した文字列に終端コード(NUL(00H))が無い場合。
- **Path**で指定した文字列が247文字(終端コード(NUL(00H))含む)を超えた場合。
- **Path**に間接指定でタイマ・カウンタを指定した場合。
- **Path**、**Src**、**Param**にビットデータを指定したとき、チャンネルの先頭以外を指定した場合。
- 引数に範囲外の値を指定した場合。
- 引数の予約に値を指定した場合。

MREAD　ストレージ(メモリカード/CPUメモリ)読み出し

ストレージ読出

[R:=] MREAD (Path, Src, Count, Param, Result);

引数/戻り値	説明	認識する型								定数
		.U	.S	.D	.L	.F	.DF	.B	.T	
Path※3※6	データを読み出すファイル名の格納先、または読出ファイル名を指定。(1～247文字(終端コードNUL、フォルダ名を含む)の範囲の文字列を指定)	–	–	–	–	–	–	–	.T	○※2
Src※3	読み出したデータを格納している先頭を指定。	.U	.U	.U	.U	–	–	–	–	–
Count※4	読み出す範囲のデータ数の格納先、または読み出しデータ数を指定。	.U	.U	.U	.U	–	–	–	–	○
Param※5	読み出す形式等に関するパラメータ、およびデータの読み出し状態を格納する先頭を指定。	.U	.U	.U	.U	–	–	–	–	–
Result※1※3	読み出し結果を通知するビットデータを指定。	–	–	–	–	–	–	.B	–	–
R	演算結果が格納される戻り値。関数の起動に成功したらON、失敗のときOFF。省略可能。省略時、内部レジスタに格納。	–	–	–	–	–	–	.B	–	–

※1　**Result**にビットデータを指定した場合、通知ビットは連続する2ビットを使用、ワードデータを指定した場合は、通知ビットデータは連続する2ワードの各々最下位ビットを使用。
※2　定数で文字列を指定する場合は、' 'で囲んで入力。
※3　ビットデータを使用する場合は、チャンネルの先頭を指定。
※4　ASCII文字列データを読み出す場合、文字列の最大文字数(バイト数)を指定。
※5　**Param**にビットデータを指定した場合、連続する96ビットを占有。ワードデータを指定した場合、連続する6ワードを占有。
※6　T, C, CR, Zは使用不可。

●動作説明

ストレージから**Path**で指定されたファイル名(文字列)のデータを、**Count**を先頭として、**Src**で指定された範囲に読み出し、格納します。
読み出し処理は、**Param**で指定された読み出しパラメータに従っておこなわれ、**Result**に読み出し完了通知と異常終了アラーム通知が格納されます。
MREAD関数が正常に起動できた場合、**R**をONし、失敗した場合はOFFします。

Paramに指定する読み出す形式等に関するパラメータの詳細は、『命令語リファレンスマニュアル』「MREAD命令」を参照してください。

参考

Result読出結果格納対象の通知ビットは次のような意味を持ちます。
(例) MR2000を指定した場合

通知ビット		ON	OFF
MR2000	完了通知	完了	未完了
MR2001	異常通知	異常終了	正常終了

「演算フラグについて(CR2009～CR2012)」(1-8ページ)

詳細については、『命令語リファレンスマニュアル』「ストレージ命令」を参照してください。

●書式例

bFlag := **MREAD** (strPath, auiValue[0], uiValue, auiParam[0], abFlag[0]);

ポイント

次のいずれかの条件が成立した場合、関数は実行されません。

- **Path**で指定した文字列に終端コード(NUL(00H))が無い場合。
- **Path**で指定した文字列が247文字(終端コード(NUL(00H))含む)を超えた場合。
- **Path**に間接指定でタイマ･カウンタを指定した場合。
- **Path**、**Count**、**Param**にビットデータを指定したときに、チャンネルの先頭以外を指定した場合。
- 引数に範囲外の値を指定した場合。
- 引数の予約に値を指定した場合。

MFREE　ストレージ(メモリカード/CPUメモリ)の空き容量をバイト単位で取得

ストレージ容量取得

[R:=] MFREE (Drive, InOut, Result);

引数/戻り値	説明	認識する型								定数
		.U	.S	.D	.L	.F	.DF	.B	.T	
Drive※3	空き容量を確認するドライブの番号(メモリカード:0、CPUメモリ:1):を指定。	.U	.U	.U	.U	–	–	–	–	–
InOut※1	取得したメモリカード(CPUメモリ)の空き容量(バイト単位)を格納する対象を指定。(32ビットデータ)	–	–	.D	.D	–	–	–	–	–
Result※2	空き容量の取得結果を通知するビットデータを指定。	–	–	–	–	–	–	.B	–	–
R	演算結果が格納される戻り値。関数の起動に成功したらON、失敗のときOFF。省略可能。省略時、内部レジスタに格納。	–	–	–	–	–	–	.B	–	–

※1　ビットデータを指定した場合、チャンネルの先頭から連続した32ビットで演算。
※2　**Result**にビットデータを指定した場合、通知ビットは連続する2ビットを使用、ワードデータを指定した場合は、通知ビットデータは連続する2ワードの各々最下位ビットを使用。
※3　空き容量を確認するストレージを、メモリカードまたはCPUメモリから選択可能。

●動作説明

Driveで指定されたメモリカードまたはCPUメモリの空き容量(バイト単位)を取得して、**InOut**に32ビットバイナリデータ(2ワード)として格納し、**Result**に空き容量取得の完了通知と異常終了アラーム通知が格納されます。

「演算フラグについて(CR2009～CR2012)」(1-8ページ)
詳細については、『命令語リファレンスマニュアル』「ストレージ命令」を参照してください。

●書式例

bFlag := **MFREE** (uiValue, udiValue, abFlag[0]);

●通知ビットデータ(Result:完了通知、Result+(1):エラー通知)

項　目		内　容
Result	完了通知	命令実行開始時にOFF、メモリカード(またはCPUメモリ)より空き容量取得処理終了時にONします。命令が正常終了、異常終了のどちらかでもONします。
Result+(1)	エラー通知	命令が正常終了でOFF、異常終了でONします。

ポイント

- **完了通知、エラー通知は命令実行完了後END処理で結果が書き込まれます。**
- **空き容量取得には、数スキャンを要する場合があります。**
- **取得処理中にFTPサーバ機能経由でファイル操作を行っていた場合や、ロギング機能を使用していた場合、取得した容量は実際の空き容量と異なる場合があります。**
- **ストレージ命令実行中(CR3214がONしている)は、MFREE関数は実行されません。**
- **InOutにチャンネルの先頭以外のリレーを指定した場合、関数は実行されません。**
- **割り込みプログラムおよび初期化モジュール内では使用できません。**

MFREEK ストレージ(メモリカード/CPUメモリ)の空き容量をキロバイト単位で取得

ストレージ容量取得
[R:=] MFREEK (Drive, InOut, Result);

引数/戻り値	説明	認識する型								定数
		.U	.S	.D	.L	.F	.DF	.B	.T	
Drive※3	空き容量を確認するドライブの番号(メモリカード:0、CPUメモリ:1):を指定。	.U	.U	.U	.U	—	—	—	—	—
InOut※1	取得したメモリカード(CPUメモリ)の空き容量(キロバイト単位)を格納する対象を指定。(32ビットデータ)	—	—	.D	.D	—	—	—	—	—
Result※1	空き容量の取得結果を通知するビットデータを指定。	—	—	—	—	—	—	.B	—	—
R	演算結果が格納される戻り値。関数の起動に成功したらON、失敗のときOFF。省略可能。省略時、内部レジスタに格納。	—	—	—	—	—	—	.B	—	—

※1　ビットデータを指定した場合、チャンネルの先頭から連続した32ビットで演算。
※2　Resultにビットデータを指定した場合、通知ビットは連続する2ビットを使用、ワードデータを指定した場合は、通知ビットデータは連続する2ワードの各々最下位ビットを使用。
※3　空き容量を確認するストレージを、メモリカードまたはCPUメモリから選択可能。

●動作説明

Driveで指定されたメモリカードまたはCPUメモリの空き容量(キロバイト単位)を取得して、InOutに32ビットバイナリデータ(2ワード)として格納し、Resultに空き容量取得の完了通知と異常終了アラーム通知が格納されます。

「演算フラグについて(CR2009～CR2012)」(1-8ページ)
詳細については、『命令語リファレンスマニュアル』「MFREEK命令」を参照してください。

●書式例

bFlag := **MFREEK** (uiValue, udiValue, abFlag[0]);

●通知ビットデータ(Result:完了通知、Result+(1):エラー通知)

項　目		内　容
Result	完了通知	命令実行開始時にOFF、メモリカード(CPUメモリ)より空き容量取得処理終了時にONします。命令が正常終了、異常終了のどちらかでもONします。
Result+(1)	エラー通知	命令が正常終了でOFF、異常終了でONします。

ポイント
- 完了通知、エラー通知は命令実行完了後END処理で結果が書き込まれます。
- 空き容量取得には、数スキャンを要する場合があります。
- 取得処理中にFTPサーバ機能経由でファイル操作をおこなっていた場合や、ロギング機能を使用していた場合、取得した容量は実際の空き容量と異なる場合があります。
- ストレージ命令実行中(CR3214がONしている)は、MFREEK関数は実行されません。
- InOutにチャンネルの先頭以外のリレーを指定した場合、関数は実行されません。
- 割り込みプログラムおよび初期化モジュール内では使用できません。

MMKDIR ストレージ(メモリカード/CPUメモリ)フォルダ作成

ストレージフォルダ作成
[R:=] MMKDIR (Path, Result);

引数/戻り値	説明	認識する型								定数
		.U	.S	.D	.L	.F	.DF	.B	.T	
Path※2※4	作成するフォルダのパス名の格納先、または作成フォルダパス名を指定。(1～247文字(終端コードNUL含む)の範囲の文字列を指定)	—	—	—	—	—	—	—	.T	○※1
Result※3	作成結果を通知する対象を指定。	—	—	—	—	—	—	.B	—	—
R	演算結果が格納される戻り値。関数の起動に成功したらON、失敗のときOFF。省略可能。省略時、内部レジスタに格納。	—	—	—	—	—	—	.B	—	—

※1　定数で文字列を指定する場合は、' 'で囲んで入力。
※2　ビットデータを指定した場合、チャンネルの先頭を指定。
※3　ビットデータを指定した場合、連続する2ビットを占有。ワードデータを指定した場合、通知ビットデータは連続する2ワードのそれぞれ最下位ビットを使用。最下位ビット以外は変化なし。
※4　T, C, CR, Zは使用不可。

●動作説明

Pathで指定されたパス名(文字列)で、メモリカードまたはCPUメモリにフォルダを作成します。
作成結果がResultに格納されます。

「演算フラグについて(CR2009～CR2012)」(1-8ページ)
詳細については、『命令語リファレンスマニュアル』「ストレージ命令」を参照してください。

●書式例

bFlag := **MMKDIR** (strPath, abFlag[0]);

●通知ビットデータ(Result:完了通知、Result+(1):エラー通知)

項　目		内　容
Result	完了通知	関数実行開始時にOFF、フォルダ作成終了時にONします。 関数が正常終了、異常終了のどちらでもONします。
Result+(1)	エラー通知	関数が正常終了でOFF、異常終了でONします。

ポイント

- フォルダ作成には、数スキャンを要する場合があります。
- 通知ビットデータは関数実行完了後、END処理で結果が書き込まれます。
- 通知ビットデータに間接指定、インデックス修飾を指定した時の格納先は、実行条件の立ち上がり時点での格納先になります。
- ストレージ命令実行中(CR3214がONしている)は、MMKDIR関数は実行されません。
- 次のいずれかの条件が成立した場合、関数は実行されません。
  - Pathで指定した文字列に終端コード(NUL(00H))が無い場合。
  - Pathで指定した文字列が247文字(終端コード(NUL(00H))含む)を超えた場合。
  - Pathに間接指定でタイマ･カウンタを指定した場合。
  - Pathにビットデータを指定したときに、チャンネルの先頭以外を指定した場合。
- 割り込みプログラムおよび初期化モジュール内では使用できません。

MRMDIR ストレージ(メモリカード/CPUメモリ)フォルダ削除

ストレージフォルダ削除
[R:=] MRMDIR (Path, Result);

引数/戻り値	説明	認識する型								定数
		.U	.S	.D	.L	.F	.DF	.B	.T	
Path※2※4	除削するフォルダのパス名の格納先、または除削フォルダパス名を指定。(1～247文字(終端コードNUL含む)の範囲の文字列を指定)	—	—	—	—	—	—	—	.T	○※1
Result※3	除削結果を通知する対象を指定。	—	—	—	—	—	—	.B	—	—
R	演算結果が格納される戻り値。関数の起動に成功したらON、失敗のときOFF。省略可能。省略時、内部レジスタに格納。	—	—	—	—	—	—	.B	—	—

※1　定数で文字列を指定する場合は、' 'で囲んで入力。
※2　ビットデータを指定した場合、チャンネルの先頭を指定。
※3　ビットデータを指定した場合、連続する2ビットを占有。ワードデータを指定した場合、通知ビットデータは連続する2ワードのそれぞれ最下位ビットを使用。最下位ビット以外は変化なし。
※4　T, C, CR, Zは使用不可。

●動作説明

Pathで指定されたパス名(文字列)のフォルダをメモリカードまたはCPUメモリから削除します。
削除結果がResultに格納されます。

- **Pathで指定したフォルダがメモリカードに存在しない場合**
  Result+1がONし、CM2390に3(ファイルもしくはフォルダが存在しない)が格納されます。
- **Pathで指定したフォルダ内にファイルやフォルダが存在する場合**
  フォルダに存在する全てのファイル･フォルダを消去した後、指定したフォルダを消去します。

「演算フラグについて(CR2009～CR2012)」(1-8ページ)
詳細については、『命令語リファレンスマニュアル』「ストレージ命令」を参照してください。

●書式例

bFlag := **MRMDIR** (strPath, abFlag[0]);

●通知ビット格納(Result:完了通知、Result+(1):エラー通知)

項　目		内　容
Result	完了通知	関数実行開始時にOFF、フォルダ削除完了時にONします。 関数が正常終了、異常終了のどちらでもONします。
Result+(1)	エラー通知	関数が正常終了でOFF、異常終了でONします。

ポイント

- フォルダ削除には、数スキャンを要する場合があります。
- 通知ビットデータは関数実行完了後、END処理で結果が書き込まれます。
- 通知ビットデータに間接指定、インデックス修飾を指定した時の格納先は、関数実行の立ち上がり時点での格納先になります。
- ストレージ命令実行中(CR3214がONしている)は、MRMDIR関数は実行されません。
- 次のいずれかの条件が成立した場合、関数は実行されません。
  - Pathで指定した文字列に終端コード(NUL(00H))が無い場合。
  - Pathで指定した文字列が247文字(終端コード(NUL(00H))含む)を超えた場合。
  - Pathに間接指定でタイマ・カウンタを指定した場合。
  - Pathにビットデータを指定したときに、チャンネルの先頭以外を指定した場合。
- 割り込みプログラムおよび初期化モジュール内では使用できません。

MDEL　ストレージ(メモリカード/CPUメモリ)ファイル削除

ストレージファイル削除
[R:=] MDEL (Path, Result);

引数/戻り値	説明	認識する型								定数
		.U	.S	.D	.L	.F	.DF	.B	.T	
Path※2※4	削除するファイル名の格納先、または削除ファイル名を指定。(1～247文字(終端コードNUL含む)の範囲の文字列を指定)	—	—	—	—	—	—	—	.T	○※1
Result※3	除削結果を通知する対象を指定。	—	—	—	—	—	—	.B	—	—
R	演算結果が格納される戻り値。関数の起動に成功したらON、失敗のときOFF。省略可能。省略時、内部レジスタに格納。	—	—	—	—	—	—	.B	—	—

※1　定数で文字列を指定する場合は、' 'で囲んで入力。
※2　ビットデータを指定した場合、チャンネルの先頭を指定。
※3　ビットデータを指定した場合、連続する2ビットを占有。ワードデータを指定した場合、通知ビットデータは連続する2ワードのそれぞれ最下位ビットを使用。最下位ビット以外は変化なし。
※4　T, C, CR, Zは使用不可。

●動作説明

Pathで指定されたファイル名(文字列)のファイルをメモリカードまたはCPUメモリから削除します。
削除結果がResultに格納されます。

- Pathで指定したファイルがメモリカードに存在しない場合
  Result+(1)がONし、CM2390に3(ファイルもしくはフォルダが存在しない)が格納されます。

「演算フラグについて(CR2009～CR2012)」(1-8ページ)
詳細については、『命令語リファレンスマニュアル』「ストレージ命令」を参照してください。

●書式例

bFlag := **MDEL** (strpath, abFlag[0]);

●通知ビット格納(Result:完了通知、Result+(1):エラー通知)

項　目		内　容
Result	完了通知	関数実行開始時にOFF、ファイル削除完了時にONします。 関数が正常終了、異常終了のどちらでもONします。
Result+(1)	エラー通知	関数が正常終了でOFF、異常終了でONします。

ポイント

- ファイル削除には、数スキャンを要する場合があります。
- 通知ビットデータは関数実行完了後、END処理で結果が書き込まれます。
- 通知ビットデータに間接指定、インデックス修飾を指定した時の格納先は、関数実行の立ち上がり時点での格納先になります。
- ストレージ命令実行中(CR3214がONしている)は、MDEL関数は実行されません。
- 次のいずれかの条件が成立した場合、関数は実行されません。
  - Pathで指定した文字列に終端コード(NUL(00H))が無い場合。
  - Pathで指定した文字列が247文字(終端コード(NUL(00H))含む)を超えた場合。
  - Pathに間接指定でタイマ・カウンタを指定した場合。
  - Pathにビットデータを指定したときに、チャンネルの先頭以外を指定した場合。
- 割り込みプログラムおよび初期化モジュール内では使用できません。

MPRINT　ストレージ(メモリカード/CPUメモリ)のファイルにテキスト書き込み

ストレージ文字列書き込み
[R:=] MPRINT (Path, In, Result);

引数/戻り値	説明	認識する型								定数
		.U	.S	.D	.L	.F	.DF	.B	.T	
Path※2※4	データを保存するファイル名の格納先、または保存ファイル名を指定。(1～247文字(終端コードNUL含む)の範囲の文字列を指定)	.U	.S	.D	.L	.F	.DF	.B	.T	○
In※3	書き込む文字列を格納している先頭または、書き込み文字列を指定。	.U	.S	.D	.L	.F	.DF	.B	.T	○
Result※3	書き込み結果を通知するビットデータを指定。	—	—	—	—	—	—	.B	—	—
R	演算結果が格納される戻り値。関数の起動に成功したらON、失敗のときOFF。省略可能。省略時、内部レジスタに格納。	.U	.S	.D	.L	.F	.DF	—	—	—

※1　書き込み結果格納対象にビットデータを指定した場合、通知ビットは連続する2ビットを使用。
※2　定数で文字列を指定する場合は、' 'で囲んで入力。
※3　ビットデータを使用する場合はチャンネルの先頭を指定。
※4　T, C, Zは使用不可。

●動作説明

Pathで指定されたファイル名(文字列)の末尾に、Inで指定された文字列をメモリカードに書き込みます。
Path名が存在しない場合、ファイルを新規作成します。

Resultに書き込み完了通知と異常終了のエラー通知が格納されます。
MPRINT関数が正常に起動できた場合、RをONし、失敗した場合はOFFします。

参 考

Resultの通知ビットは次のような意味を持ちます。
(例) MR2000を指定した場合

通知ビット		ON(1, TRUE)	OFF(0, FALSE)
MR2000	完了通知	完了	未完了
MR2001	エラー通知	異常終了	正常終了

「演算フラグについて（CR2009～CR2012）」（1-8ページ）
詳細については、『命令語リファレンスマニュアル』「ストレージ命令」を参照してください。

●書式例

bFlag := **MPRINT** (strPath, strString, abFlag[0]);

MREADL ストレージ（メモリカード/CPUメモリ）のファイルから1行読み出し

ストレージ1行読み出し
[R:=] MREADL (Path, Dst, Param, Result);

引数/戻り値	説明	認識する型								定数
		.U	.S	.D	.L	.F	.DF	.B	.T	
Path※5	データを読み出すファイル名の格納先、または読み出しファイル名を指定。（1～247文字(終端コードNUL含む)の範囲の文字列を指定）	—	—	—	—	—	—	—	.T	○
Dst※4	読み出した文字列を格納する先頭を指定。	—	—	—	—	—	—	—	.T	○※3
Param※1	読み出しパラメータを指定。	.U	.U	.U	.U	—	—	—	—	—
Result※2	読み出し結果を通知するビットデータを指定。	—	—	—	—	—	—	.B	—	—
R	演算結果が格納される戻り値。関数の起動に成功したらON、失敗のときOFF。省略可能。省略時、内部レジスタに格納。	—	—	—	—	—	—	.B	—	—

※1　Paramは、連続する4ワードを占有。
※2　Resultは、通知ビットは連続する4ビットを使用。
※3　定数で文字列を指定する場合は、' 'で囲んで入力。
※4　ビットデータを使用する場合はチャンネルの先頭を指定。
※5　T, C, Zは使用不可。

●動作説明

Pathで指定されたファイル名(文字列)から１行読み出してDstに読出します。
(行を表す文字コードは、0DH(CR)、0AH(LF)、0D0AH(CR+LF)です)
読み出す文字の最大バイトサイズはParam+(2)(読み出し上限バイト数)で設定します。
ファイル内の1行のテキストデータ長がParam+(2)で指定した長さを超えている場合、
Result+(3)(改行文字なし通知)がONし、Param+(2)バイトまでテキストデータをDstに格納されます。

Resultに読み出し完了通知と異常終了アラーム通知が格納されます。
MREADL関数が正常に起動できた場合、RをONし、失敗した場合はOFFします。

「演算フラグについて（CR2009～CR2012）」（1-8ページ）
ParamおよびResultパラメータの詳細については、『命令語リファレンスマニュアル』「MREADL命令」を参照してください。

●書式例

bFlag := **MREADL** (strPath, strString, auiValue[0], abFlag[0]);

MCOPY　ストレージ(メモリカード/CPUメモリ)のファイルをコピー

ストレージファイルコピー
[R:=] MCOPY (Src, Dst, Param, Result);

引数/戻り値	説明	認識する型								定数
		.U	.S	.D	.L	.F	.DF	.B	.T	
Src※4※5	コピー元のファイル名･フォルダ名の格納先、またはファイル名･フォルダ名を指定。(1～247文字(終端コードNUL含む)の範囲の文字列を指定)	—	—	—	—	—	—	—	.T	○
Dst※4※5	コピー先のファイル･フォルダ名の格納先、またはファイル･フォルダ名を指定。(1～247文字(終端コードNUL、フォルダ名を含む)の範囲の文字列を指定)	—	—	—	—	—	—	—	.T	○※3
Param※1	コピー形式のパラメータを指定。	.U	.U	.U	.U	—	—	—	—	—
Result※2	コピー結果を通知するビットデータを指定。	—	—	—	—	—	—	.B	—	—
R	演算結果が格納される戻り値。関数の起動に成功したらON、失敗のときOFF。省略可能。省略時、内部レジスタに格納。	—	—	—	—	—	—	.B	—	—

※1　Paramは、1ワードを占有。
※2　Resultは、通知ビットは連続する2ビットを使用。
※3　定数で文字列を指定する場合は、' 'で囲んで入力。
※4　ビットデータを使用する場合はチャンネルの先頭を指定。
※5　T, C, Zは使用不可。

●動作説明

Srcで指定されたファイル名(文字列)をDstで指定したファイルにコピーします。
コピー先に同名のファイル･フォルダが存在する場合、Paramに従って上書きするかエラーとするかを判断します。
コピー形式パラメータ値＝0の場合: 同名ファイル存在時、同名ファイルを上書きする
コピー形式パラメータ値＝1の場合: 同名ファイル存在時、上書きしないでエラー終了する

Resultにコピー完了通知と異常終了アラーム通知が格納されます。
MCOPY関数が正常に起動できた場合、RをONし、失敗した場合はOFFします。
Src、Dstに指定できるファイル･フォルダ名の組み合わせ表

		Dst	
		ファイル名	フォルダ名
Src	ファイル名	正常動作	正常動作
	フォルダ名	エラー発生	正常動作

「演算フラグについて(CR2009～CR2012)」(1-8ページ)
Paramパラメータおよび Resultコピー結果の詳細については、『命令語リファレンスマニュアル』「MCOPY命令」を参照してください。

●書式例

bFlag := **MCOPY** (strPathSrc, strPathDst, auiValue[0], abFlag[0]);

MMOV　ストレージ(メモリカード/CPUメモリ)のファイルを移動

ストレージファイル移動
[R:=] MMOV (Src, Dst, Param, Result);

引数/戻り値	説明	認識する型								定数
		.U	.S	.D	.L	.F	.DF	.B	.T	
Src※4※5	移動元のファイル名･フォルダ名の格納先、またはファイル名･フォルダ名を指定。(1～247文字(終端コードNUL含む)の範囲の文字列を指定)	—	—	—	—	—	—	—	.T	○
Dst※4※5	移動先のファイル･フォルダ名の格納先、またはファイル･フォルダ名を指定。(1～247文字(終端コードNUL、フォルダ名を含む)の範囲の文字列を指定)	—	—	—	—	—	—	—	.T	○※3
Param※1	移動形式のパラメータを指定。	.U	.U	.U	.U	—	—	—	—	—
Result※2	移動結果を通知するビットデータを指定。	—	—	—	—	—	—	.B	—	—
R	演算結果が格納される戻り値。関数の起動に成功したらON、失敗のときOFF。省略可能。省略時、内部レジスタに格納。	—	—	—	—	—	—	.B	—	—

※1　Paramは、1ワードを占有。
※2　Resultは、通知ビットは連続する2ビットを使用。
※3　定数で文字列を指定する場合は、' 'で囲んで入力。
※4　ビットデータを使用する場合はチャンネルの先頭を指定。
※5　T, C, Zは使用不可。

●動作説明

Srcで指定されたファイル名(文字列)をDstで指定したファイルに移動します。
移動先に同名のファイル･フォルダが存在する場合、Paramに従って上書きするかエラーとするかを判断します。
　移動形式パラメータ値＝0の場合: 同名ファイル存在時、同名ファイルを上書きする
　移動形式パラメータ値＝1の場合: 同名ファイル存在時、上書きしないでエラー終了する

Resultに移動完了通知と異常終了アラーム通知が格納されます。
MMOV関数が正常に起動できた場合、RをONし、失敗した場合はOFFします。

Src、Dstに指定できるファイル･フォルダ名の組み合わせ表

		Dst	
		ファイル名	フォルダ名
Src	ファイル名	正常動作	正常動作
	フォルダ名	エラー発生	正常動作

「演算フラグについて(CR2009～CR2012)」(1-8ページ)
Paramパラメータおよび Result移動結果の詳細については、『命令語リファレンスマニュアル』「MMOV命令」を参照してください。

●書式例

bFlag := **MMOV** (strPathSrc, strPathDst, auiValue[0], abFlag[0]);

MREN　ストレージ(メモリカード/CPUメモリ)のファイル名を変更

ストレージファイル名変更
[R:=] MREN (Path, Name, Result);

引数/戻り値	説明	認識する型								定数
		.U	.S	.D	.L	.F	.DF	.B	.T	
Path※3※4	変更するフのファイル名･フォルダ名の格納先、またはファイル名･フォルダ名を指定。(1～247文字(終端コードNUL含む)の範囲の文字列を指定)	—	—	—	—	—	—	—	.T	○
Name※3※4	変更後のファイル･フォルダ名の格納先、またはファイル･フォルダ名を指定。(1～247文字(終端コードNUL、フォルダ名を含む)の範囲の文字列を指定)	—	—	—	—	—	—	—	.T	○※2
Result※1	ファイル名変更結果を通知するビットデータを指定。	—	—	—	—	—	—	.B	—	—
R	演算結果が格納される戻り値。関数の起動に成功したらON、失敗のときOFF。省略可能。省略時、内部レジスタに格納。	—	—	—	—	—	—	.B	—	—

※1　Resultは、通知ビットは連続する2ビットを使用。
※2　定数で文字列を指定する場合は、' 'で囲んで入力。
※3　ビットデータを使用する場合はチャンネルの先頭を指定。
※4　T, C, Zは使用不可。

●動作説明

Pathで指定されたファイル名･フォルダ名(文字列)をNameで指定したファイル名･フォルダ名に変更します。
Resultに変更完了通知と異常終了アラーム通知が格納されます。
MREN関数が正常に起動できた場合、RをONし、失敗した場合はOFFします。

「演算フラグについて(CR2009～CR2012)」(1-8ページ)
動作の詳細については、『命令語リファレンスマニュアル』「MREN命令」を参照してください。

●書式例

bFlag := **MREN** (strPath, strString, abFlag[0]);

MSTAT　ストレージ(メモリカード/CPUメモリ)のファイル状態取得

ストレージファイル状態取得
[R:=] MSTAT (Path, InOut, Result);

引数/戻り値	説明	認識する型								定数
		.U	.S	.D	.L	.F	.DF	.B	.T	
Path※4	状態を取得するファイル名の格納先、またはファイル名を指定。(1～247文字(終端コードNUL、フォルダ名を含む)の範囲の文字列を指定)	—	—	—	—	—	—	—	.T	○※2
InOut※3	ファイル状態を格納する対象を指定。	.U	.U	.U	.U	—	—	—	—	—
Result※1	取得結果を通知するビットデータを指定。	—	—	—	—	—	—	.B	—	—
R	演算結果が格納される戻り値。省略可能。省略時、内部レジスタに格納。 SEG関数は、.Uの型を使用。	—	—	—	—	—	—	.B	—	—

※1　Resultは、通知ビットは連続する3ビットを使用。
※2　定数で文字列を指定する場合は、' 'で囲んで入力。
※3　ファイル状態格納対象は、連続する10ワードを占有す。
※4　T, C, Zは使用不可。

●動作説明

Pathで指定されたファイルの状態を取得し、**InOut**に格納します。
Resultに取得完了通知と異常終了アラーム通知およびファイルの存在有無が格納されます。
MSTAT関数が正常に起動できた場合、**R**をONし、失敗した場合はOFFします。

「演算フラグについて（CR2009～CR2012）」（1-8ページ）
詳細については、『命令語リファレンスマニュアル』「MSTAT命令」を参照してください。

●書式例

bFlag := **MSTAT** (strPath, auiValue, abFlag[0]);

アクセスウィンドウ関数

AWSHOW ユーザメッセージ表示

ユーザメッセージ表示
AWSHOW (EN, Id);

引数/戻り値	説明	認識する型								定数
		.U	.S	.D	.L	.F	.DF	.B	.T	
EN	実行条件がOFF→ONすると、アクセスウィンドウにユーザメッセージを表示。	–	–	–	–	–	–	.B	–	○
Id	アクセスウィンドウに表示するユーザメッセージのIDを指定。(1～63)	.U	.U	.U	.U	–	–	–	–	○
R	戻り値はなし。	–	–	–	–	–	–	–	–	–

●動作説明

ENがOFF→ONすると、Idで指定したユーザメッセージを表示します。

参考　ユーザメッセージを表示解除するには、アクセスウィンドウを操作するかAWHIDE命令を実行します。

「演算フラグについて(CR2009～CR2012)」(1-8ページ)
詳細については、『命令語リファレンスマニュアル』「アクセスウィンドウ命令」または使用しているCPUユニットのユーザーズマニュアルの「ユーザメッセージ」を参照してください。

●書式例

AWSHOW (bFlag, uiValue);

ポイント
- **実行条件がON時にIdの値を変更しても変化しません。**
- **実行条件の立ち上がりで指定されたIDのユーザメッセージに対応するCRがセットされます。**
- **AWSHOW関数とAWHIDE関数が同時に実行された場合、後から実行された関数の動作に従います。**
- **AWSHOW関数がプロジェクト中に複数存在していた場合、アクセスウィンドウの状態は最初に実行したAWSHOW関数に従います。**
- **割り込みプログラムおよび初期化モジュール内では使用できません。**

AWHIDE ユーザメッセージ表示解除

ユーザメッセージ表示解除
AWHIDE (EN, ID);

引数/戻り値	説明	認識する型								定数
		.U	.S	.D	.L	.F	.DF	.B	.T	
EN	実行条件がOFF→ONすると、アクセスウィンドウのユーザメッセージを表示解除。	–	–	–	–	–	–	.B	–	–
Id	表示解除するユーザメッセージのIDを指定。(1～63)	.U	.U	.U	.U	–	–	–	–	○
R	戻り値はなし。	–	–	–	–	–	–	–	–	–

●動作説明

ENがOFF→ONすると、**Id**で指定したユーザメッセージを表示解除します。

「演算フラグについて（CR2009～CR2012）」（1-8ページ）
詳細については、『命令語リファレンスマニュアル』「アクセスウィンドウ命令」または使用しているCPUユニットのユーザーズマニュアルの「ユーザメッセージ」を参照してください。

●書式例

AWHIDE (bFlag, uiValue);

ポイント
- **実行条件がON時にIdの値を変更しても変化しません。**
- **実行条件の立ち上がりで指定されたIDのユーザメッセージに対応するCRがリセットされます。**
- **AWSHOW関数とAWHIDE関数が同時に実行された場合、後から実行された関数の動作に従います。**
- **割り込みプログラムおよび初期化モジュール内では使用できません。**

フリーランカウンタ関数

RFSFRC　フリーランカウンタリフレッシュ

フリーランカウンタリフレッシュ
RFSFRC ([EN]);

引数/戻り値	説明	認識する型								定数
		.U	.S	.D	.L	.F	.DF	.B	.T	
EN	実行条件。省略可能。省略時、常時（毎スキャン）実行。	—	—	—	—	—	—	.B	—	—
R	戻り値はなし。	—	—	—	—	—	—	—	—	—

●動作説明

実行条件が成り立つ(TRUE)とき、フリーランカウンタを最新の値にリフレッシュします。
更新されるフリーランカウンタは以下のとおりです。

CM1030～CM1031：1msカウンタ
CM1032～CM1033：10μsカウンタ
CM1034～CM1035：1μsカウンタ
CM1036～CM1037：0.1μsカウンタ

詳細については、『命令語リファレンスマニュアル』「フリーランカウンタ命令」を参照してください

●書式例

RFSFRC (bEnable);

センサ設定関数

SPRD　センサパラメータ読み出し

センサパラメータ読み出し
SPRD (EN, Unit, Node, Slot, Param, Result);

引数/戻り値	説明	認識する型								定数
		.U	.S	.D	.L	.F	.DF	.B	.T	
EN	実行条件に使用するビットデータを指定。	—	—	—	—	—	—	.B	—	—
Unit※1	ユニット番号を指定。(0～48、本体は'0')	—	—	—	—	—	—	—	—	○
Node※3	ノードアドレス(1～256)、または格納先を指定。	.U	.U	.U	.U	—	—	—	—	○
Slot※3	スロット番号、またはそれを格納した対象を指定。	.U	.U	.U	.U	—	—	—	—	○
Param※3	パラメータ番号、またはそれを格納した対象を指定。	.U	.U	.U	.U	—	—	—	—	○
Result※2※4※5	完了コードやパラメータの読出結果を格納するデバイスの対象の先頭を指定。	.U	.U	.U	.U	.F	.DF	.B	.T	—
R	戻り値はなし。	—	—	—	—	—	—	—	—	—

※1　16# (16進数指定)は使用不可。
※2　Zは指定不可。
※3　ビットデータを指定した場合、連続した16ビットで演算。R002などチャンネルの先頭以外を指定した場合、次のチャンネルにまたがって16ビットで演算。
※4　ビットデータを指定する場合は、チャンネルの先頭を指定。
※5　実行結果により格納されるワード数(4ワード以上)が変化。

●動作説明

ENがONのとき、**Unit**番目のユニットに接続されたノードアドレス**Node**、スロット番号**Slot**のセンサの**Param**で指定したパラメータを読み出し、完了時に**Result**の0ビット目に完了ビット、**Result**の1ビット目に実行失敗ビット、**Result**＋(1)に完了コード、**Result**＋(2)に詳細完了コード、**Result**＋(3)に配列要素数(配列でない場合は1)、**Result**＋(4)以降に読み出されたパラメータの内容が格納されます。

Result～：結果格納先

結果格納先			内容
Result	0ビット	完了ビット	命令実行開始時にOFF、パラメータの読み出し完了時にONします。命令が正常完了、異常完了のどちらでもONします。
	1ビット	実行失敗ビット	命令の実行に失敗した時に完了ビットと同時にONします。 命令実行開始時にOFFします。
	2～15ビット	システム予約	OFF(固定値)
Result＋(1)	完了コード		パラメータの読み出し完了時に完了コードが格納されます。 (正常時は、0)
Result＋(2)	詳細完了コード		パラメータの読み出し完了時に詳細完了コードが格納されます。
Result＋(3)	配列要素数		読み出したパラメータの配列の要素数が格納されます。 配列以外の場合は1、命令が異常完了した場合は、0が格納されます。
Result＋(4～)	読出結果		パラメータの値が格納されます。読み出すパラメータによって占有されるデータ数が変化します。

ポイント　**「EtherNet/IP機器予約」されているセンサを指定した場合、実行失敗します。**

センサパラメータ読出命令は、実行条件の立ち上がりで命令が実行されると、次のスキャンで入力条件がOFFであっても処理が継続されます。
結果格納先の完了ビット、失敗ビットは、完了時にONし、実行開始時にOFFになります。
命令が実行されるとセンサ設定命令実行数に1が加算され、実行完了時に減算されます。

詳細については、『命令語リファレンスマニュアル』「センサ設定命令」を参照してください。

●書式例

SPRD (bFlag, 1, uiNode, uiSlot, uiValue, auiValue[0]);

SPWR　センサパラメータ書き込み

センサパラメータ書き込み
SPWR (EN, Unit, Node, Slot, Param, Result);

引数/戻り値	説明	認識する型								定数
		.U	.S	.D	.L	.F	.DF	.B	.T	
EN	実行条件に使用するビットデータを指定。	—	—	—	—	—	—	.B	—	—
Unit※1	ユニット番号を指定。(0～48、本体は'0')	—	—	—	—	—	—	—	—	○
Node※3	ノードアドレス(1～256)、または格納先を指定。	.U	.U	.U	.U	—	—	—	—	○
Slot※3	スロット番号、またはそれを格納した対象を指定。	.U	.U	.U	.U	—	—	—	—	○
Param※3	パラメータ番号、またはそれを格納した対象を指定。	.U	.U	.U	.U	—	—	—	—	○
Result※2※4※5	完了コードやパラメータの読出結果を格納するデバイスの対象の先頭を指定。	.U	.U	.U	.U	.F	.DF	.B	.T	—
R	戻り値はなし。	—	—	—	—	—	—	—	—	—

※1　16# (16進数指定) は使用不可。
※2　Zは指定不可。
※3　ビットデータを指定した場合、連続した16ビットで演算。R002などチャンネルの先頭以外を指定した場合、次のチャンネルにまたがって16ビットで演算。
※4　ビットデータを指定する場合は、チャンネルの先頭を指定。
※5　実行結果により格納されるワード数(4ワード以上)が変化する。

●動作説明

ENがONのとき、**Unit**番目のユニットに接続されたノードアレス**Node**、スロット番号**Slot**のセンサの**Param**で指定したパラメータに、**Result**＋(4)以降で指定した設定値を書き込み、完了時に**Result**の0ビット目に完了ビット、**Result**の1ビット目に実行失敗ビット、**Result**＋(1)に完了コード、**Result**＋(2)に詳細完了コードが格納されます。

Result～：結果格納先

結果格納先			内容
Result	0ビット	完了ビット	命令実行開始時に**OFF**、パラメータ書き込み完了時に**ON**します。命令が正常完了、異常完了のどちらでも**ON**します。
	1ビット	実行失敗ビット	命令の実行に失敗した時に完了ビットと同時に**ON**します。 命令実行開始時に**OFF**します。
	2～15ビット	システム予約	**OFF(固定値)**
Result＋(1)	完了コード		パラメータの書き込み完了時に完了コードが格納されます。 (正常時は、**0)**
Result＋(2)	詳細完了コード		パラメータの書き込み完了時に詳細完了コードが格納されます。
Result＋(3)	システム予約		**0**(固定値)
Result＋(4～)	設定値		設定値を格納します。書き込むパラメータによって占有するデータ数が変化します。

ポイント　**「EtherNet/IP機器予約」されているセンサ(アダプタ)を指定した場合、実行失敗します。**

センサパラメータ書込命令は、実行条件の立ち上がりで命令が実行され、次のスキャンで入力条件がOFFであっても処理が継続されます。
結果格納先の完了ビット、失敗ビットは、完了時にONし、実行開始時にOFFになります。
命令が実行されるとセンサ設定命令実行数に1が加算され、実行完了時に減算されます。

詳細については、『命令語リファレンスマニュアル』「センサ設定命令」を参照してください。

●書式例

SPWR (bFlag, 1, uiNode, uiSlot, uiValue, auiValue[0]);

SSVC　センササービス実行

センササービス実行
SSVC (EN, Unit, Node, Slot, Service, Result);

引数/戻り値	説明	認識する型								定数
		.U	.S	.D	.L	.F	.DF	.B	.T	
EN	実行条件に使用するビットデータを指定。	—	—	—	—	—	—	.B	—	—
Unit※1	ユニット番号を指定。(0～48、本体は'0')	—	—	—	—	—	—	—	—	○
Node※3	ノードアドレス(1～256)、または格納先を指定。	.U	.U	.U	.U	—	—	—	—	○
Slot※3	スロット番号、またはそれを格納した対象を指定。	.U	.U	.U	.U	—	—	—	—	○
Service※3	パラメータ番号、またはそれを格納した対象を指定。	.U	.U	.U	.U	—	—	—	—	○
Result※2※4※5	完了コードやパラメータの読出結果を格納するデバイスの対象の先頭を指定。	.U	.U	.U	.U	.F	.DF	.B	.T	—
R	戻り値はなし。	—	—	—	—	—	—	—	—	—

※1　16#(16進数指定)は使用不可。
※2　Zは指定不可。
※3　ビットデータを指定した場合、連続した16ビットで演算。R002などチャンネルの先頭以外を指定した場合、次のチャンネルにまたがって16ビットで演算。
※4　ビットデータを指定する場合は、チャンネルの先頭を指定。
※5　指定するサービスにより最大6ワードを使用。

●動作説明

ENがONのとき、**Unit**番目のユニットに接続されたノードアレス**Node**、スロット番号**Slot**のセンサの**Service**で指定したサービスを実行し、完了時に**Result**の0ビット目に完了ビット、**Result**の1ビット目に実行失敗ビット、**Result**＋(1)に完了コード、**Result**＋(2)に詳細完了コードが格納されます。

Result～：結果格納先

結果格納先			内容
Result	0ビット	完了ビット	命令実行開始時に**OFF**、サービスの実行完了時に**ON**します。命令が正常完了、異常完了のどちらでも**ON**します。
	1ビット	実行失敗ビット	命令の実行に失敗した時に完了ビットと同時に**ON**します。 命令実行開始時に**OFF**します。
	2～15ビット	システム予約	**OFF**(固定値)
Result＋(1)	完了コード		サービスの実行完了時に完了コードが格納されます。(正常時は、**0**)
Result＋(2)	詳細完了コード		サービスの実行完了時に詳細完了コードが格納されます。
Result＋(3)	システム予約		**0**(固定値)
Result＋(4) Result＋(5)	送信データ		実行するサービスに合わせて送信データ(最大**2**ワード)を指定します。

ポイント　**「EtherNet/IP機器予約」されているセンサを指定した場合、実行失敗します。**

センササービス実行命令は、実行条件の立ち上がりで命令が実行され、次のスキャンで入力条件がOFFであっても処理が継続されます。結果格納先の完了ビット、失敗ビットは、完了時にONし、実行開始時にOFFになります。

命令が実行されるとセンサ設定命令実行数に1が加算され、実行完了時に減算されます。

詳細については、『命令語リファレンスマニュアル』「センサ設定命令」を参照してください。

●書式例

SSVC (bFlag, 1, uiNode, uiSlot, uiValue, auiValue[0]);

サイクリック通信リフレッシュ関数

RFSCI　サイクリック通信入力リフレッシュ

サイクリック通信入力リフレッシュ
RFSCI ([EN], Start, Count);

引数/戻り値	説明	認識する型								定数
		.U	.S	.D	.L	.F	.DF	.B	.T	
EN	実行条件。省略可能。省略時、常時(毎スキャン)実行。	—	—	—	—	—	—	.B	—	—
Start※1	サイクリック通信入力データのリフレッシュをおこなう先頭を指定。	.U	.S	.D	.L	.F	—	—	—	—
Count	リフレッシュをおこなうデータ数を指定。	—	—	—	—	—	—	—	—	○
R	戻り値はなし。	—	—	—	—	—	—	—	—	—

※1　ビットデータを指定する場合は、チャンネルの先頭を指定。
配列型変数を指定する場合、要素番号にデバイス/変数を指定することはできません。

●動作説明

実行条件がONのとき、**Start**に割り付けられているデータから、**Count**で指定したデータ数のサイクリック通信入力データをリフレッシュします。(先頭が、32ビットデータのとき、リフレッシュされるワード数は、データ数×2になります。)

詳細については、『命令語リファレンスマニュアル』「サイクリック通信リフレッシュ命令」を参照してください。

●書式例

RFSCI (bEnable, auiValue[0], 128);

RFSCO　サイクリック通信出力リフレッシュ

サイクリック通信出力リフレッシュ
RFSCO ([EN], Start, Count);

引数/戻り値	説明	認識する型								定数
		.U	.S	.D	.L	.F	.DF	.B	.T	
EN	実行条件。省略可能。省略時、常時(毎スキャン)実行。	—	—	—	—	—	—	.B	—	—
Start※1	サイクリック通信出力データのリフレッシュをおこなう先頭を指定。	.U	.S	.D	.L	.F	—	—	—	—
Count	リフレッシュをおこなうデータ数を指定。	—	—	—	—	—	—	—	—	○
R	戻り値はなし。	—	—	—	—	—	—	—	—	—

※1　ビットデータを指定する場合は、チャンネルの先頭を指定。
配列型変数を指定する場合、要素番号にデバイス/変数を指定することはできません。

●動作説明

実行条件がONのとき、**Start**に割り付けられているデータから、**Count**で指定したデータ数のサイクリック通信出力データをリフレッシュします。(先頭が、32ビットデータのとき、リフレッシュされるワード数は、データ数×2になります。)

詳細については、『命令語リファレンスマニュアル』「サイクリック通信リフレッシュ命令」を参照してください。

●書式例

RFSCO (bEnable, auiValue[0], 128);

付 録

CR/CM一覧 ····· 付-2
ASCIIコード表 ····· 付-9
使用できない文字一覧 ····· 付-10
使用できない関数一覧 ····· 付-12
索引 ····· 付-14
関数索引 ····· 付-17

CR/CM一覧

コントロールリレー CR

コントロールリレー（CR0000～CR7915）は、CPUユニットの動作状態の確認、各内蔵機能の設定やモニタなどができます。
システム予約されているコントロールリレーは使用できません。

CR	属性 R：読出専用 空欄：読書可能	機　能	
CR0000		ロギング・トレース機能（ID0）	ロギング・トレース 実行（ON：許可、OFF：禁止）
CR0001	R		ロギング・トレース 動作中
CR0002	R		ロギング・トレース ファイル書き込み終了
CR0003	R		ロギング バッファ50%未満警告
CR0004	R		ロギング バッファオーバーフロー
CR0005	R		ロギング 実行中RUN中書き込みあり
CR0006～CR0007	－		システム予約
CR0008	R		ロギング・トレース エラー
CR0009	R		ロギング・トレース メモリ容量なしエラー
CR0010	R		ロギング・トレース デバイス・トリガ設定異常
CR0011	R		トレース ファイル保存トリガモニタ
CR0012	R		トレース データ取得完了
CR0013	R		ロギング・トレース 設定書き込みあり
CR0014～CR0015	－		システム予約
CR0100～CR0915		ロギング・トレース機能（ID1～ID9）	
CR1000～CR1015		リアルタイムチャートモニタ（ID10）	
CR1100～CR2001	－	システム予約	
CR2002	R	常時ON	
CR2003	R	常時OFF	
CR2004	R	10msクロックパルス（duty50%）	
CR2005	R	100msクロックパルス（duty50%）	
CR2006	R	1sクロックパルス（duty50%）	
CR2007	R	運転開始時１スキャンのみOFF	
CR2008	R	運転開始時１スキャンのみON	
CR2009		演算結果が負またはオーバーフローのときON	
CR2010	R	演算結果がゼロのときON	
CR2011	R	演算結果が正のときON	
CR2012	R	演算実行エラーのときON	
CR2013～CR2302	－	システム予約	
CR2303		一定スキャンタイム運転（ON：有効、OFF：無効）	
CR2304	R	スキャンタイム設定（（一定スキャンタイム運転、END処理時間設定）オーバー時1スキャンのみON）	
CR2305	－	システム予約	
CR2306	R	バックアップ電池異常（ON:異常、OFF：正常）	
CR2307		スキャンタイム最小値最大値のクリア	
CR2308		END処理時間設定 （ON：有効、OFF:無効）（CR2303と同時にONした場合は、CR2303が優先）	
CR2309～CR2310	－	システム予約	
CR2311	R	RTC電圧低下エラー（ON：エラー発生中、OFF：正常）	
CR2312	－	システム予約	
CR2313		VTセンサアプリケーション画面強制切換（切換する：ON、切換しない：OFF）	
CR2314	R	プロジェクトパスワード認証禁止状態（ON：認証禁止、OFF:認証可能）	
CR2315～CR2813	－	システム予約	
CR2814		DASC、FASC命令（全サフィックス）実行時にゼロサプレスを（ON:する、OFF：しない）	
CR2815		DASC.S、DASC.L、FASC実行時に＋の符号を省略（ON:する、OFF：しない）	
CR2900～CR3005	－	システム予約	
CR3100	R	ユニット間同期リフレッシュ時間超過	
CR3101～CR3115	－	システム予約	
CR3200		CPUメモリ　保存処理要求	
CR3201	R	CPUメモリ　保存処理中	
CR3202	R	CPUメモリ　保存処理完了	
CR3203	R	CPUメモリ　保存処理エラー	
CR3204～CR3206	－	システム予約	

CR	属性 R:読出専用 空欄:読書可能	機　能
CR3207		命令語 RAMモード動作 (CPUメモリに対するファイルレジスタ命令･ストレージ命令のRAMモード有効化 (ON:有効､OFF:無効))
CR3208～CR3209	－	システム予約
CR3210	R	任意ロギング･トレース動作中
CR3211～CR3213	－	システム予約
CR3214	R	ストレージ命令実行中
CR3215	－	システム予約
CR3300～CR3415		アラームリレー
CR3500	R	アラーム動作リレー(ON : CR3300～CR3415のいずれかがON)
CR3501		アラームリレーがON→OFFのときに履歴を記録(ON : 残す､OFF : 残さない)
CR3502		OFF→ON時にアラームリレー履歴クリア
CR3503～CR3512	－	システム予約
CR3513	R	ロギング内部バッファオーバフロー
CR3514		HKEY命令　複数キー禁止
CR3515	R	HKEY命令　スキャン終了
CR3600～CR3615	R	HKEYの情報格納エリア
CR3700～CR3901	－	システム予約
CR3902	R	重度エラー履歴が最大件数
CR3903	R	軽度エラー履歴が最大件数
CR3904	R	RUN/PRG切換SW状態(ON:RUN側/OFF:PRG側)
CR3905	R	重度エラー発生中
CR3906	R	軽度エラー発生中
CR3907～CR3908	－	システム予約
CR3909		現在のエラークリア(ラダープログラム中でSETしてください)
CR3910		OFF→ON時にすべてのエラー､イベント履歴をクリア
CR3911～CR4000	－	システム予約
CR4001～CR4300	R	拡張ユニット有無(拡張ユニット1～48)
CR4301～CR4315	－	システム予約
CR4400	R	AW　上キー押下中
CR4401	R	AW　下キー押下中
CR4402	R	AW　左キー押下中
CR4403	R	AW　右キー押下中
CR4404	R	AW　MENUキー押下中
CR4405	R	AW　ENTキー押下中
CR4406～CR4414	－	システム予約
CR4415	R	AW　ユーザメッセージ表示中
CR4500		AW　1行目ブリンク
CR4501		AW　2行目ブリンク
CR4502		AW　3行目ブリンク
CR4503		AW　4行目ブリンク
CR4504～CR4600	－	システム予約
CR4601～CR4915		AW ユーザメッセージ1～63
CR5000～CR5315	－	システム予約
CR5400		メモリカードマウント要求
CR5401		メモリカードアンマウント要求
CR5402	R	メモリカード使用中
CR5403	R	メモリカード認識完了
CR5404	R	メモリカードあり
CR5405	R	メモリカードライトプロテクト
CR5406～CR5407	－	システム予約
CR5408		CPUメモリマウント要求
CR5409		CPUメモリアンマウント要求
CR5410	R	CPUメモリ使用中
CR5411	R	CPUメモリ認識完了
CR5412～CR4915	―	システム予約
CR5500		プロジェクトロード実行要求
CR5501	―	システム予約
CR5502		プロジェクトセーブ実行要求
CR5503	―	システム予約
CR5504	R	プロジェクトロード実行完了
CR5505	R	プロジェクトロード実行失敗
CR5506	R	プロジェクトセーブ実行完了

CR	属性 R:読出専用 空欄:読書可能	機　能	
CR5507	R	プロジェクトセーブ実行失敗	
R5508		レシピロード実行要求	
R5509	—	システム予約	
R5510	R	レシピロード実行完了	
R5511	R	レシピロード実行失敗	
R5512		レシピセーブ実行要求	
R5513	—	システム予約	
R5514	R	レシピセーブ実行完了	
R5515	R	レシピセーブ実行失敗	
CR5600		運転記録機能(ID0)	運転記録0 保存要求
CR5601	R		運転記録0 動作中
CR5602	R		運転記録0 保存トリガモニタ
CR5603	R		運転記録0 保存中
CR5604	R		運転記録0 収集中
CR5605	R		運転記録0 設定異常
CR5606	—		システム予約
DR5607	R		運転記録0 保存完了
CR5608	R		運転記録0 保存エラー
CR5609	R		デバイス記録0 バッファ不足
CR5610～ CR5613	—		システム予約
CR5614			運転記録0 バッファ領域全保存モード
CR5615	—		システム予約
CR5700～ CR5919		運転記録機能(ID1～ID3)	
CR6000～ CR7915	—	システム予約	

コントロールメモリ CM

コントロールメモリ（CM0000～CM7599）は、CPUユニットの動作状態の確認、各内蔵機能の設定やモニタなどができます。
システム予約されているコントロールメモリは使用できません。

CM	属性 R:読出専用 空欄:読書可能	機　能	保持 ○：保持
CM0000～CM0699	－	システム予約	
CM0700	R	RTCからの読み出し（年）	
CM0701	R	RTCからの読み出し（月）	
CM0702	R	RTCからの読み出し（日）	
CM0703	R	RTCからの読み出し（時）	
CM0704	R	RTCからの読み出し（分）	
CM0705	R	RTCからの読み出し（秒）	
CM0706	R	RTCからの読み出し（曜日）	
CM0707～CM0709	－	システム予約	
CM0710	R	ファイルレジスタのカレントバンク番号	
CM0711～CM0899	－	システム予約	
CM900	R	CPU機能バージョン	
CM901～CM906	－	システム予約	
CM907	R	プロジェクトバージョン	
CM908～CM999	－	システム予約	
CM1000～CM1001	R	スキャンタイム測定値（1μs）（32ビット）	
CM1002～CM1003	R	END処理時間測定値（1μs単位）（32ビット）	
CM1004～CM1005	R	スキャンタイム最小値（1μs単位）（32ビット）	
CM1006～CM1007	R	スキャンタイム最大値（1μs単位）（32ビット）	
CM1008～CM1009		一定スキャンタイム運転時設定値（1μs単位）（32ビット）	
CM1010～CM1011		END処理時間設定値（1μs単位）（32ビット）	
CM1012～CM1013	R	設定超過時スキャンタイム（1μs単位）（32ビット）	
CM1014～CM1015	R	END処理時間超過値（1μs単位）（32ビット）	
CM1016～CM1029	－	システム予約	
CM1030～CM1031	R	32ビット1msフリーランカウンタ	
CM1032～CM1033	R	32ビット10μsフリーランカウンタ	
CM1034～CM1035	R	32ビット1μsフリーランカウンタ	

CM	属性 R：読出専用 空欄：読書可能	機　能	保持 ○：保持
CM1036～CM1037	R	32ビット0.1μsフリーランカウンタ	
CM1038～CM1499	－	システム予約	
CM1500～CM1509	R	ロギング･トレース（ID0～ID9）ファイル保存カウンタ	
CM1510～CM1629	－	システム予約	
CM1630～CM1639	R	ロギング･トレース リングバッファの空き容量最小 （ID0～ID9）（KByte単位）	
CM1640～CM1649	－	システム予約	
CM1650	R	センサ設定命令実行数	
CM1651～CM1657	－	システム予約	
CM1658	R	プロジェクトパスワード連続認証失敗回数	○
CM1659	R	プロジェクトパスワード累積認証失敗回数	○
CM1660～CM1669	R	ロギング･トレース リングバッファの空き容量（ID0~ID9）（KByte単位）	
CM1670～CM1679	－	システム予約	
CM1680～CM1689	R	ロギング･トレース ファイル保存カウンタ （ID0～ID9）（メモリカード用）	○
CM1690～CM1699	－	システム予約	
CM1700～CM1701		デジタルトリマ0上限値（32ビット）	
CM1702～CM1703		デジタルトリマ1上限値（32ビット）	
CM1704～CM1705		デジタルトリマ2上限値（32ビット）	
CM1706～CM1707		デジタルトリマ3上限値（32ビット）	
CM1708～CM1709		デジタルトリマ4上限値（32ビット）	
CM1710～CM1711		デジタルトリマ5上限値（32ビット）	
CM1712～CM1713		デジタルトリマ6上限値（32ビット）	
CM1714～CM1715		デジタルトリマ7上限値（32ビット）	
CM1716～CM1719	－	システム予約	
CM1720	R	AW　ユーザメッセージ表示番号	
CM1721～CM1735	－	システム予約	
CM1736		AW　初期表示言語設定	
CM1737		AW　初期表示コメント番号	
CM1738		AW　操作禁止設定	
CM1739	－	システム予約	
CM1740～CM1772	R	ラダープロジェクト名（最大32文字＋終端コード）	
CM1773～CM1779	－	システム予約	
CM1780		AW　初期画面設定 有効	
CM1781		AW　初期画面設定 デバイス種別	
CM1782		AW　初期画面設定 画面ID	
CM1783		AW　初期画面設定 画面ID	
CM1784		AW　初期画面設定 デバイス番号	
CM1785		AW　初期画面設定 デバイス番号	
CM1786		AW　初期画面設定 表示形式	
CM1787		AW　初期画面設定 デバイス名表記	
CM1788		AW　初期画面設定 キーロック状態	
CM1789	－	システム予約	
CM1790		オートロード　フォルダ番号（要求）	
CM1791	R	オートロード　完了コード	
CM1792	R	オートロード　フォルダ番号（完了）	
CM1793	－	システム予約	
CM1794		ランロード　フォルダ番号（要求）	
CM1795	R	ランロード　完了コード	
CM1796	R	ランロード フォルダ番号（完了）	
CM1797～CM1868	－	システム予約	
CM1869	R	拡張ユニット接続台数	
CM1870		レシピロード　ドライブ番号	○
CM1871		レシピロード　フォルダ番号	○
CM1872	—	システム予約	
CM1873	R	レシピロード　完了コード	○
CM1874	—	システム予約	
CM1875		レシピセーブ　ドライブ番号	○
CM1876		レシピセーブ　参照フォルダ番号	○
CM1877		レシピセーブ　出力フォルダ番号	○
CM1878	R	レシピセーブ　完了コード	○
CM1879	—	システム予約	
CM1880		プロジェクトロード時ドライブ番号（要求）	
CM1881		プロジェクトロード時フォルダ番号（要求）	
CM1882	R	プロジェクトロード時ドライブ番号（完了）	
CM1883	R	プロジェクトロード時フォルダ番号（完了）	
CM1884	R	プロジェクトロード時完了コード	
CM1875～CM1877	—	システム予約	
CM1888		プロジェクトセーブ時ドライブ番号（要求）	

CM	属性 R：読出専用 空欄：読書可能	機　能		保持 ○：保持
CM1889		プロジェクトセーブ時フォルダ番号（要求）		
CM1890	R	プロジェクトセーブ時ドライブ番号（完了）		
CM1891	R	プロジェクトセーブ時フォルダ番号（完了）		
CM1892	R	プロジェクトセーブ時完了コード		
CM1893～CM1999	－	システム予約		
CM2000～CM2001	R	運転記録機能(ID0)	デバイス記録0 平均サンプリング周期	
CM2002～CM2003	R		デバイス記録0 平均変化デバイス数	
CM2004～CM2005	R		デバイス記録0 最大収集期間	
CM2006～CM2009	－		システム予約	
CM2010～CM2039		運転記録機能(ID1～ID3)		
CM2040～CM2389	－	システム予約		
CM2390	R	ストレージ命令用エラーコード		
CM2391		ストレージアクセス中電源OFF （発生時に0以外の値になる。プログラムから0にしてください。）		
CM2392		メモリカードリトライ発生回数（電源断時も保持）		
CM2393		メモリカードベリファイエラー発生回数（電源断時も保持）		
CM2394		メモリカードタイムアウト発生回数（電源断時も保持）		
CM2395		メモリカードエラー時リセット遅延時間		
CM2396～CM2399	－	システム予約		
CM2400～CM2401	R	ユニット間同期リフレッシュ開始時間(0.1μs)		
CM2402～CM2403	R	ユニット間同期リフレッシュ終了時間(0.1μs)		
CM2404～CM2405		ユニット間同期リフレッシュ開始時間最大(0.1μs) (リセットするにはプログラムから0にしてください。)		○
CM2406～CM2407		ユニット間同期リフレッシュ終了時間最大(0.1μs) (リセットするにはプログラムから0にしてください。)		○
CM2408～CM2409	R	ユニット間同期リフレッシュ実行時間(0.1μs)		
CM2410～CM5099	－	システム予約		
CM5100	R	最新重度エラー詳細	年月(10進数で下二桁が月、百の桁から上が年。例：1210→2012年10月)	○
CM5101	R		日	○
CM5102	R		時	○
CM5103	R		分	○
CM5104	R		秒	○
CM5105	R		エラー番号	○
CM5106	R		有効データ数	○
CM5107	R		詳細情報01	○
CM5108	R		詳細情報02	○
:	R			○
CM5126	R		詳細情報20	○
CM5127～CM5149	－	システム予約		
CM5150	R	最新軽度エラー詳細	年月(10進数で下二桁が月、百の桁から上が年。例：1210→2012年10月)	○
CM5151	R		日	○
CM5152	R		時	○
CM5153	R		分	○
CM5154	R		秒	○
CM5155	R		エラー番号	○
CM5156	R		有効データ数	○
CM5157	R		詳細情報01	○
CM5158	R		詳細情報02	○
:	R			○
CM5176	R		詳細情報20	○
CM5177～CM5871	－	システム予約		
CM5872～CM5873	R	CPUメモリ使用量（32ビット、Byte単位）		○
CM5874～CM5999	－	システム予約		
CM6000		運転記録機能(ID0)	運転記録0 保存完了コード	
CM6001	－		システム予約	
CM6002			運転記録0 保存カウンタ	
CM6003	－		システム予約	
CM6004	R		運転記録0 保存進捗詳細1	
CM6005	R		運転記録0 保存進捗詳細2	
CM6006	R		運転記録0 保存進捗	
CM6007～CM6009	－		システム予約	
CM6010～CM6171	R		運転記録0 保存パス	
CM6172～CM6199	－		システム予約	
CM6200～CM6799		運転記録機能(ID1～ID3)		
CM6800～CM7599	－	システム予約		

■エラーの詳細情報

- No.128（演算エラー）の場合

詳細情報n	説　明
詳細情報1 （CM0712/CM5157）	**エラーの詳細内容** 0　（未使用） 1　間接指定が不適切 2　間接指定先がT/C 3　簡易間接指定が不適切 4　オペランド範囲外 5　浮動小数点型実数オーバーフロー 6　デバイス指定が不適切 7　ROOT命令の引数が負 8　ENCO命令でONしているビットがない 9　浮動小数点型実数フォーマット異常 10　三角関数命令の引数が不適切 11　変換できないデータ 12　テーブル指定が不適切 13　時刻比較命令の引数が不適切 14　カムスイッチオペランドが不適切 15　周波数カウンタ命令の引数が不適切 16　0除算 17　ロギングID不正 18　ダイレクト処理エラー 19　ダイレクト処理できないユニット 20　ジャンプ先サブルーチンが存在しない 21　RUN中書き込み中なので命令が実行不可 22　拡張ユニット専用命令対象エラー（ユニット番号） 23　システム予約 24　PIDAT 動作設定範囲外 25　PIDAT サンプリング周期範囲外 26　PIDAT 比例定数範囲外 27　PIDAT 積分定数範囲外 28　PIDAT 微分定数範囲外 29　PIDAT 比例定数2自由度化係数範囲外 30　PIDAT 積分定数2自由度化係数範囲外 31　PIDAT 微分定数2自由度化係数範囲外 32　PIDAT 操作量上限値･下限値範囲外 33　PIDAT 操作量変化幅制限値範囲外 34　PIDAT 測定値変化幅範囲外 35　PIDAT AT状態遷移異常 36　PIDAT 動作制御フラグ範囲外 37　演算周期オーバー 38　PIDAT AT調整パラメータ範囲外 39　PIDAT ATヒステリシス範囲外 40　PIDAT ATタイムアウト時間範囲外 41　PIDAT AT算出制御方式設定範囲外 42　PIDAT ATタイムアウト 43　PIDAT AT比例定数オーバー 44　PIDAT AT積分定数オーバー 45　PIDAT AT微分定数オーバー 46　負の数の累乗根は演算できない 47　ABS命令の引数が不適切 48　演算オーバーフロー 49　UR/UM/UV ユニット未接続 50　ユニットデバイス（R）未割付 51　ユニットデバイス（DM）未割付 52　ユニット種別不一致 53　運転記録ID不正 54　スタックオーバーフロー
詳細情報2 （CM5158）	00H固定
詳細情報3 （CM5159）	エラーが発生しているステップ番号上位
詳細情報4 （CM5160）	エラーが発生しているステップ番号下位
詳細情報5～20 （CM5161～CM5176）	モジュール名をプロジェクト言語設定に従って格納します。 （モジュール名以外の部分は00H）

※ CR2012がON時にCM0712には即時、CM5150~CM5176には数スキャン後値が格納されます。

- No.50,53,55,56,58,59,63（ユニット番号に関するエラー）の場合

詳細情報n	説　明
詳細情報1 （CM5107）	ユニット番号（エラーの発生状況によってユニット番号が特定できない場合には、格納されません。）
詳細情報2～20 （CM5108～CM5126）	00H固定

- No.54（ユニットの台数に関するエラー）の場合

詳細情報n	説　明
詳細情報1 （CM5107）	ユニット設定情報でのユニットの台数
詳細情報2 （CM5108）	実際に接続されているユニットの台数
詳細情報3～20 （CM5109～ CM5126）	00H固定

- No.127（オートローディング失敗エラー）の場合

詳細情報n	説　明
詳細情報1 （CM5157）	エラーの詳細内容
詳細情報2～20 （CM5158～ CM5176）	00H固定

- No.129（ユニットエラー）の場合

拡張情報n	解　説
拡張情報1 （CM5157）	ユニット番号
拡張情報2 （CM5158）	ユニット毎のエラー番号
拡張情報3～20 （CM5159～ CM5176）	00H固定

- 上記以外のエラー番号の場合

拡張情報n	解　説
拡張情報1～20	00H固定

- ※1　Bluetooth通信に対応したCPUユニットのみで有効です。

ASCIIコード表

ASCIIコードの一覧表です。

	上位4ビット																
下位4ビット		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
	0	NUL	DLE	SP	0	@	P	`	p				ー	タ	ミ		
	1	SOH	DC1	!	1	A	Q	a	q			。	ア	チ	ム		
	2	STX	DC2	”	2	B	R	b	r			「	イ	ツ	メ		
	3	ETX	DC3	#	3	C	S	c	s			」	ウ	テ	モ		
	4	EOT	DC4	$	4	D	T	d	t			、	エ	ト	ヤ		
	5	ENQ	NAK	%	5	E	U	e	u			・	オ	ナ	ユ		
	6	ACK	SYN	&	6	F	V	f	v			ヲ	カ	ニ	ヨ		
	7	BEL	ETB	’	7	G	W	g	w			ァ	キ	ヌ	ラ		
	8	BS	CAN	(	8	H	X	h	x			ィ	ク	ネ	リ		
	9	HT	EM	)	9	I	Y	i	y			ゥ	ケ	ノ	ル		
	A	LF	SUB	*	:	J	Z	j	z			ェ	コ	ハ	レ		
	B	VT	ESC	+	;	K	[	k	{			ォ	サ	ヒ	ロ		
	C	FF	→	,	<	L	\	l	|			ャ	シ	フ	ワ		
	D	CR	←	—	=	M	]	m	}			ュ	ス	ヘ	ン		
	E	SO	↑	.	>	N	^	n	~			ョ	セ	ホ	゛		
	F	SI	↓	／	?	O	_	o	DEL			ッ	ソ	マ	゜		

使用できない文字一覧

プロジェクト名、モジュール名、マクロ名、ラベル、デバイスコメントで使用できない文字を説明します。

● プロジェクト名に使用できない文字

使用できない文字	! ″ # $ % & ‘ * , . / : ; < = > ? @ ¥ ^ ` | ~ (スペース)

● プログラム名(モジュール名、ファンクションブロック名、マクロ名)に使用できない文字

使用できない文字	! ″ # $ % & ’ ( ) * , . / : ; < = > ? @ [¥] ^ ` { | }
	} ~ (スペース)
先頭に使用できない文字	+ - 0 1 2 3 4 5 6 7 8 9 _
末尾に使用できない文字または文字列	_ _0 _1 _2 _3 _4 _5 _6 _7 _8 _9
使用できない文字列	ｸﾞﾛｰﾊﾞﾙ　グローバル　.

● 変数名に使用できない文字

使用できない文字	! ″ # $ % & ’ ( ) * + , - . / : ; < = > ? @ [¥] ^ ` {
	| } ~ (スペース)
先頭に使用できない文字※1	0 1 2 3 4 5 6 7 8 9 _ U _
使用できない文字列	ｸﾞﾛｰﾊﾞﾙ　グローバル
	(命令語と同じ名前)
	(デバイス名)
	• アルファベット1文字＋ R または M ＋数字 • アルファベット1文字＋数字 • X ＋(A ～ F) • Y ＋(A ～ F)
使用できない文字列	• (ST言語関数と同じ名前) (KVスクリプト関数と同じ名前) KVスクリプトで規定された以下のキーワード !!,#PRAGMA,',(,),*,*=,+,+=,-,-=,.B,.D,.DF,.F,.L,.S,.T,.U,/,/ =,<,<=,<>,=,>,>=,AND,AS,B,BREAK,C,CASE,CM,CR,CTC,CTH,DI M,DM,DO,DOUBLE,DR,ELSE,ELSE IF,EM,END,END IF,END SELECT,END TYPE,END WHILE,FALSE,FM,FOR,IF,IS,LR,MC,MCR,MOD,MR,NEXT,NOT,OFF,ON,OR,P,R,REM,SELECT,S ELECT CASE,STEP,STRWORK,T,THEN,TM,TO,TRUE,TYPE,UM,UNTIL,UR,UV,V,W,WHILE,XOR,Z,ZF,[,\ \,\\A,\\B,\\F,\\N,\\R,\\T,\\V,] • IEC61131-3で規定された以下のキーワード TRUE, FALSE TIME, DATE,TIME_OF_DAY, TOD, DATE_AND_TIME, DT BOOL, SINT, INT, DINT, LINT, USINT, UINT, UDINT, ULINT, REAL, LREAL, STRING, WSTRING, BYTE, WORD, DWORD, LWORD ANY, ANY_DERIVED, ANY_ELEMENTARY, ANY_MAGNITUDE, ANY_NUM, ANY_REAL, ANY_INT, ANY_BIT, ANY_STRING, ANY_DATE ACTION, END_ACTION, ARRAY, OF, AT, CASE, END_CASE, CONFIGURATION, END_CONFIGURATION, CONSTANT, EN, ENO, EXIT, F_EDGE, FOR, TO, BY, DO, END_FOR, FUNCTION, END_FUNCTION, FUNCTION_BLOCK, END_FUNCTION_BLOCK, IF, THEN, ELSIF, ELSE, END_IF,

使用できない文字列	INITIAL_STEP, END_STEP, NOT, MOD, AND, XOR, OR, PROGRAM, WITH, END_PROGRAM, R_EDGE, READ_ONLY, READ_WRITE, REPEAT, UNTIL, END_REPEAT, RESOURCE, END_RESOURCE, RETAIN, NON_RETAIN, RETURN, STEP, STRUCT, END_STRUCT, TASK, TRANSITION, FROM, END_TRANSITION, TYPE, END_TYPE, VAR,VAR_INPUT, VAR_OUTPUT, VAR_IN_OUT, VAR_TEMP, VAR_EXTERNAL, VAR_ACCESS, VAR_CONFIG,VAR_GLOBAL, END_VAR, WHILE, DO, END_WHILE

● 構造体のメンバ名に使用できない文字

使用できない文字	( ) * / + − < > = & ! ¥ ″ # $ % % & | ' ~ ` @ [ ] { : ; , , . ? ^ (半角スペース) (全角スペース)
先頭に使用できない文字	0 1 2 3 4 5 6 7 8 9
末尾に使用できない文字または文字列	_
使用できない文字列	ｸﾞﾛｰﾊﾞﾙ グローバル Global 全局 全域 글로벌 • 数値と区別できない文字列 　• K 　• K + 数値 　• K + 数値 + サフィックス(U/S/D/L/F/DF) 　• K + 数値 + E 　• K + 数値 + E + 数値 　• K + 数値 + E + 数値 + サフィックス(F/DF) 　• H + 16進数値(10進数値、もしくはABCDEFの文字列) • ST/スクリプトで規定されたのキーワード （キーワードの内容は変数/ラベル名に使用できない文字を参照してください） • IEC61131-3で規定されたキーワード （キーワードの内容は変数/ラベル名に使用できない文字を参照してください）

● デバイスコメントに使用できない文字

使用できない文字	,

使用できない関数一覧

初期化モジュール、ファンクションブロック、マクロ、割り込みプログラム内で使用できない関数を説明します。

● 初期化モジュール、割り込みプログラム

関数の種類	関数	
接点関数	LDP/LDF	ロードパルス/ロードパルフ
	LDPB/LDFB	ロードパルスバー/ロードパルフバー
タイマ･カウンタ関数	TMR/TMH/TMS/TMU	タイマ/高速タイマ/高速1msタイマ/高速10μsタイマ
	C	カウンタ
ファイルレジスタ関数	FRSTM/FRLDM	ファイルレジスター括保存/ファイレジスター括読出
ストレージ関数	MWRIT/MREAD/MFREE/ MMKDIR	ストレージライト/ストレージリード/ ストレージ容量取得/ストレージフォルダ作成
	MRMDIR/MDEL	ストレージフォルダ削除/ストレージファイル削除
	MPRINT/MREADL/MCOPY/ MMOV/MREN/MFREEK/ MSTAT	ストレージテキスト書き込み/ストレージ1行読み出し/ ストレージファイルコピー/ストレージファイル移動/ ストレージファイルリネーム/ストレージ容量取得/ ストレージ状態取得
アクセスウィンドウ関数	AWSHOW/AWHIDE	ユーザメッセージ表示/ユーザメッセージ表示解除

● FEND（ファンクションブロックエンド）関数は、ファンクションブロック内に記述します。ファンクションやモジュールなどほかの種類のプログラムでは使用できません。

● ファンクション

関数の種類	関数	
ファンクションブロック関数	FBSTRT	ファンクションブロック

● マクロ

関数の種類	関数	
ファンクションブロック関数	FBSTRT	ファンクションブロック/ファンクション

標準関数として対応している関数一覧

以下の関数はPLC OPENの標準関数として使用できます。

関数の種類	関数
論理演算関数	ABS
データシフト関数	SHR, SHL, ROR, ROL
データ制御関数	LIMIT
データ変換関数	*_TO_**,TO_**,*_TRUNC_**,TRUNC_**, *_BCD_TO_**,BCD_TO_**, *_TO_BCD_** ,TO_BCD_**
浮動小数点関数	EXP, EXPT, LOG, LN, SQRT, SIN, COS, TAN, ASIN, ACOS, ATAN
文字処理関数	LEN, RIGHT, LEFT, MID, REPLACE, INSERT, DELETE, FIND, CONCAT
データ処理関数	MIN, MUX, SEL, MAX

索引

英・数・記号

ASCII コード表 ..... 付 -9
CASE 4-11
ELSE 4-8
ELSIF 4-9
HELP 機能 2-32
IF ～ ELSE 4-8
IF ～ ELSEIF 4-9
LDP 5-3
MOD 4-4
SELECT CASE 4-11
ST 関数一覧 2-30
ST 記述 2-2
ST で演算式を記述するときの注意事項 2-23
ST の種類 1-2
ST の記述方法 2-2
ST の入力方法 2-3
　直接入力 2-3
　ウィンドウ入力 2-3
ST を記述する時の注意 4-18
ST を作成する 2-2
WHILE ～ 4-14, 4-15, 4-16

あ

アウトプットウィンドウの使いかた 2-25
アウトプットウィンドウのエラー表示 2-25

い

インデックス修飾 3-25, 3-27
　インデックス定数による修飾 3-26
　インデックスレジスタによる修飾 3-26
　サフィックスの付いたデバイスの修飾 3-27
　デバイスや式による修飾 3-27
　不正なインデックス修飾 3-27
インデックス修飾と間接指定 3-25
インデックスレジスタ　Z 3-18

う

ウィンドウ入力 2-3
ウォッチウィンドウ 2-29

え

エラーメッセージ
　アウトプットウィンドウのエラー表示 2-25
　変換エラーが発生したときは 2-24
エリア型 ST の IF 文の中でタイマ関数を使用する 4-18
エリア型 ST はこんな時に使う 2-11
エリア型 ST はこんなときに使う 2-11
エリア型 ST をコピーするには 2-14
エリア型 ST を削除するには 2-14
エリア型 ST を作成する 2-11
エリア型 ST を新規作成する 2-12
エリア型 ST を編集するには 2-13
演算子 4-4
演算子一覧 1-5, 4-4
　算術演算子 4-4
　乗除算・加減算の優先順位 4-5
　比較演算子 4-4
　文字列連結演算子 4-4
　論理演算子 4-4
演算式で扱えるデータの範囲 3-36
演算子の種類 4-4
演算代入文　+=,-=,*=,/= 4-3
演算フラグ 1-8

か

カウンタ(現在値)　C 3-18
カウンタ (接点)　C 3-12
カウンタの設定値･現在値を扱う方法 3-13
カスタマイズ
　ST ウィンドウのカスタマイズ 2-34
　ST 記述時の注意事項 2-34
型(データの型(サフィックス)) 3-31
　最大値・最小値を超える時の数値の変化 3-35
型変換によるデータの欠落 2-24
簡易間接指定　#TM 3-30
関数一覧の呼び出し 2-30
間接指定(＊)とは 3-27
間接指定デバイスの型 3-29
間接指定の記述方法 3-27
間接指定の仕組み 3-27
間接指定の使用方法 3-28
間接指定の使用例 3-29
間接指定やインデックス修飾時の注意事項 1-8

き

機能バージョンについて(KV-8000) 2

く

繰り返し制御文 4-7
繰返し制御文中のタイマ動作について 4-17
繰返し制御文と処理時間 4-16
グローバル変数とローカル変数 2-15

こ

高速カウンタコンパレータ (接点)　CTC 3-13
コントロールメモリ CM 3-19
コントロールリレー CR 3-13

さ

最下位ビットでビットデバイスとして扱う 3-20
サフィックス
　.U と .S について 3-32
　.D と .L について 3-32
　.DF について 3-33
　.B について 3-34

.T について 3-34
.F について 3-33
サフィックスの省略 1-7

し

条件分岐制御文 1-5
使用できない文字一覧
構造体のメンバ名に使用できない文字 ..... 付 -11
デバイスコメントに使用できない文字 ..... 付 -11
プログラム名（モジュール名、ファンクションブロック名、マクロ名）に使用できない文字 ..... 付 -10
プロジェクト名に使用できない文字 ..... 付 -11
処理時間について 4-16

せ

制御文一覧 4-6
制御文について
条件分岐制御文 1-5
制御文を使う時の注意 4-16
繰返し制御文中のタイマ動作について 4-17
繰返し制御文と処理時間 4-16

た

代入文について
数値代入 1-4
ビット代入 1-4
文字列代入 1-4
代入文の使用方法
右辺と左辺で型が異なる場合 4-3
演算代入文 4-3
左辺に代入できない場合 4-3
数値代入文 4-2
ビット代入文 4-2
文字列代入文 4-2
代入文を使用するときの注意 4-3
タイマ(現在値)　T 3-18
タイマ (接点)　T 3-11
タイマ関数使用時の注意 3-12
タイマ関数を使用する場合 2-9
タイマの誤差 3-11
タイマの設定値･現在値を扱う方法 3-12
タイマ関数を使用するときの注意 4-18
単精度浮動小数点型定数 3-23

ち

直接入力 2-3

て

データの型(サフィックス) 3-31
データの取り扱い 3-1
データメモリ / 拡張データメモリ　DM/EM/FM 3-13
定数 3-22
単精度浮動小数点型定数 3-23
倍精度浮動小数点型定数 3-24
固定文字列（文字列型定数） 3-24
$16 進定数 3-23
(#)10 進定数 3-22
固定文字列（文字列型定数） 3-24
デバイス 3-2
デバイス番号一覧 ..... 付 -2
デバイス一覧
定数 3-3
内部レジスタ 3-3
ビットデバイス 3-2
ファンクションブロック引数 3-3
ワードデバイス 3-2
ローカルデバイス一覧 3-4
テンポラリデータメモリ　TM 3-17

と

登録モニタ 2-28

な

内部補助リレー　R･MR･LR 3-9

に

入力 / 出力リレー　R 3-7
入力(ST の入力方法)
ウィンドウ入力 2-3
直接入力 2-3
任意のビット位置でビットデバイスとして扱う 3-20

ね

ネスティングの実動作 4-16

は

倍精度浮動小数点型定数 3-24
配 2-16

ひ

比較演算子 4-4
ビット代入 1-4
ビット代入文 4-2
ビットデバイス
B 3-9
LR（ラッチリレー） 3-10
MR 3-10
R 3-9
ビットの状態 3-7
制御したビットデータの状態保持 4-18

ふ

ブール値 3-7
ファイルレジスタ　ZF/FM 3-15
ファンクションブロックの呼び出し
ファンクションの呼び出し 2-27

へ

HELP 機能 2-32
変換エラーが発生したときは 2-24
変数 3-21
変数のサフィックス 1-7

変数の設定 2-20
変数プログラミング 2-15

ほ

ボックス ST はこんな時に使う 2-4
ボックス ST や IF 文中にタイマ関数を使用するときの注意 4-18
ボックス型 ST 2-4
ボックス型 ST でタイマ関数を指定する 4-19
ボックス型 ST の特長 2-4
ボックス型 ST をコピーする 2-8
ボックス型 ST を削除する 2-7
ボックス型 ST を作成する 2-4
ボックス型 ST を使用するときの注意 2-9
　タイマ関数を使用する場合 2-9
ボックスス型 ST を使用するときの注意
　微分実行型関数を使用する場合 2-9
ボックス型 ST を編集する 2-7

も

文字列代入 1-4
文字列代入文 4-2
文字列連結演算子 4-4

よ

予約語 2-31

ら

ラベルのサフィックス 1-7

り

リレー 3-9
　内部補助リレー　R・MR・LR 3-9
　入力 / 出力リレー　R 3-7
リンクリレー　B 3-8
リンクレジスタ　W 3-16

わ

ワークデバイス 2-35

関数索引

*_BCD_TO_** 7-43
*_TO_** 7-39
*_TO_BCD_** 7-42
*_TRUNC_** 7-40

A

ABS 7-13
ACNT 6-24
ACOS 7-55
ADRADD 6-11
ADRDEC 6-10
ADRINC 6-9
ADRSET 6-9
ADRSUB 6-12
AJST 8-24
ANDA 7-9
APR 7-24
ASC 7-59
ASIN 7-55
ASLA 7-16
ASRA 7-15
ATAN 7-56
ATAN2 7-57
AVG 8-9
AWHIDE 8-45
AWSHOW 8-45

B

BANDC 7-22
BCD_TO_** 7-43
BCMP 8-13
BCMPI 8-14
BCNT 8-3
BLD 5-5
BLDB 5-7
BMOV 7-2
BOUB 5-10
BOUT 5-10
BRES 5-12
BSET 5-11
BSL 7-21
BSR 7-20
BSUM 8-10
BSWAP 7-35
BYBMOV 7-4
BYLMOV 7-5

C

CALL 6-2
CHR 7-85
CNT 5-16
COM 7-12
CONCAT 7-87
COS 7-53
CPMGET 7-38
CPMSET 7-37
CPSASC 7-86
CRC 8-11

D

DASC 7-60
DCNT 8-3
DEC 7-7
DECO 7-36
DEG 7-51
DELETE 7-76
DFLOAT 7-44
DI 8-25
DIC 8-26
DISB 7-34
DISF 7-46
DISN 7-32
DISS 7-84
DMX 7-30
DSER 8-5

E

ECALL 6-2
EI 8-25
ENCO 7-37
ENRA 7-11
EORA 7-10
EXIT 4-15
EXP 7-48

F

FASC 7-65
FBSTRT 6-4
FDEL 8-21
FEND 6-5
FIFOR 8-17
FIFOW 8-17
FIND 7-83
FINS 8-20
FLOAT 7-44
FMOV 7-2
FRLDM 6-14
FRSET 6-13
FRSTM 6-13
FWRIT 8-19

G

GRY 7-31

H

HASC 7-63

I

IEDGE 8-26
INC 7-6
INSERT 7-75
INSTR 7-82
INT 7-46
INTG 7-45
IS_ □□□ 6-23

L

LDF 5-3
LDFB 5-5
LDP 5-3
LDPB 5-4
LEFT 7-70
LEN 7-67
LIFOR 8-19
LIFOW 8-18
LIMIT 7-22
LLFLT 7-28
LN 7-49
LOG 7-50
LOGD 8-29
LOGE 8-29
LTRIM 7-78

M

MAX 8-5
MAX_RANGE 8-6
MCOPY 8-41
MDEL 8-38
MDSTOP 6-6
MDSTRT 6-5
MFREE 8-34
MFREEK 8-35
MID 7-72
MIN 8-7
MIN_RANGE 8-7
MMKDIR 8-36
MMOV 8-42
MPRINT 8-39
MPX 7-30
MREAD 8-33
MREADL 8-40
MREN 8-43
MRMDIR 8-37
MSTAT 8-43
MUX 8-8
MWRIT 8-32

N

NEG 7-12

O

ORA 7-9

P

PIDAT 8-28
PMOV 7-3

R

RAD 7-50
RAMP 7-25
RASC 7-59
RCPSASC 7-86
RDASC 7-62
REPEAT 4-14
REPLACE 7-73
RES 5-8
RETURN 4-7
RFASC 7-66
RFSCI 8-52
RFSCO 8-52
RFSFRC 8-47
RFSX 5-18
RFSY 5-18
RGRY 7-32
RHASC 7-64
RIGHT 7-69
RLA 7-17
RND 8-14
ROL 7-18
ROOT 7-7
RRA 7-16
RRNCA 7-17
RSEC 8-23
RTRIM 7-78

S

SDEL 7-75
SEC 8-23
SEG 8-2
SEL 8-8
SER 8-4
SET 5-8
SFIND 7-81
SFINDN 7-82
SIN 7-52
SINS 7-74
SLA 7-14
SLEFT 7-69
SMID 7-71
SORT 8-15
SORTN 8-16
SPLIT 7-80
SPRD 8-48
SPWR 8-49
SQRT 7-8
SRA 7-14
SRGHT 7-68
SRPLC 7-72
SSVC 8-50
STR 7-61
STRIM 7-79
SWAP 7-35

T

TAN 7-54
TBCD 7-29
TBIN 7-29
TMH 5-14
TMR 5-13
TMS 5-15
TMU 5-16
TO_** 7-40
TO_BCD_** 7-42
TPOUT 7-26
TRGD 8-30
TRGR 8-30
TRIM 7-77
TRUNC_** 7-41

U

UFILL 6-22
UFRSM 6-18
UFSUS 6-17
UMALLOC 6-19
UNIB 7-34
UNIF 7-47
UNIN 7-33

UNIS 7-85
UPSTOP 6-16
UPSTRT 6-15
UREAD 6-21
UWRIT 6-21

V

VAL 7-67

W

WSIZE 6-25
WSL 7-19
WSR 7-19
WSUM 8-10
WTIME 8-22

Z

ZONE 7-23
ZPOP 6-8
ZPUSH 6-7
ZRES 8-12

改訂履歴

印刷年月日	版　　数	改　訂　内　容
2021年　4月	初　　版	
2021年　7月	改訂1版	
2021年　9月	2訂1版	
2022年　2月	3訂1版	

保証について

1　対象製品

以下に規定する保証は、本マニュアルに記載している製品(以下「対象製品」という)に適用します。
なお、対象製品に内蔵されているリレーや電池などの消耗品は対象外とさせていただきます。

2　保証期間

対象製品の保証期間は、貴社のご指定場所に納入後3年間とします。ただし、次に該当する場合の保証期間は1年間とさせていただきます。

① 対象製品を海外でご購入いただいた場合。
② 海外で修理をご依頼いただいた場合。

3　保証範囲

(1) 上記保証期間内に当社の責任による故障が発生した場合は、無償での代替品との交換または修理をさせていただきます。ただし、保証期間内であっても、次に該当する故障の場合は保証対象外とさせていただきます。なお、代替品との交換または修理をおこなった場合でも保証期間の起算日は対象製品の当初ご納入日とさせていただきます。

① 取扱説明書、ユーザーズマニュアル、別途取り交わした仕様書などに記載された以外の不適当な条件･環境･取り扱い･使用方法に起因した故障。
② お客様の装置または、ソフトウエアの設計内容など、対象製品以外に起因した故障。
③ 当社以外による改造、修理に起因した故障。
④ 取扱説明書、ユーザーズマニュアルなどに記載している消耗部品が正しく保守、交換されていれば、防止できたと確認できる故障。
⑤ 当社出荷時の科学･技術水準では、予見が不可能だった事由による故障。
⑥ その他、火災、地震、水害などの災害及び電圧異常など当社の責任ではない外部要因による故障。

(2) 保証範囲は上記(1)を限度とし、対象製品の故障に起因するお客様での二次損害(装置の損傷、機会損失、逸失利益等)及びいかなる損害も保証の対象外とさせていただきます。

4　適用用途

当社製品は、一般工業向けの汎用品として設計･製造されております。
従いまして、下記のような用途での使用は意図しておりませんので適用外とさせていただきます。ただし、事前に当社までご相談いただき、お客様の責任において製品の仕様をご確認のうえ、定格･性能に対してご了承いただき、必要な安全対策を講じていただく場合は適用可能とさせていただきます。なお、この場合においても保証範囲は上記と同様といたします。

① 原子力発電、航空、鉄道、船舶、車両、医療機器等の人命や財産に多大な影響が予想される設備
② 電気、ガス、水道等の公共設備
③ 屋外での使用および、それに準ずる取扱説明書などで規定していない条件･環境での使用
④ 上記①及び②に準じる安全に関して高度な配慮と注意が要求される用途

全商品、送料無料で
当日出荷

必要な時に、必要な量だけ
在庫不要でトータルコストを削減

調べたいことを、いつでも、すぐに調べられる!

▼ 本社技術スタッフ直営 ▼ FAQ(よくあるご質問)

www.keyence.co.jp/seigyofaq ← CLICK!

マニュアルと合わせてご活用ください。

■ 制御システムに関するお問合せ 06-6379-1164

月～金曜(祝日を除く)8:30～20:00

■ ユーザー登録 www.keyence.co.jp/kvuser

ユーザー登録で以下のサービスが受けられます。

KV STANDARD SERVICE

● 無償バージョンアップ
専用ホームページから最新版ソフトをダウンロードできます。

● サイトライセンス
ソフトひとつ分の費用で、事務所内のPCなら何台でも使用可能。

■ 輸出書類サービス www.keyence.co.jp/yushutsu

輸出に必要な書類をその場でダウンロードできます。

株式会社 キーエンス
本社・研究所／制御システム事業部
〒533-8555 大阪市東淀川区東中島1-3-14

仕様は改良のため予告なく変更することがあります。
記載されている会社名、製品名等は、それぞれ各社の商標または登録商標です。

制御10-2041

Copyright© 2021 KEYENCE CORPORATION.
All rights reserved.

2022-1 084699